Cat. No. I139E-EN-02

SCARA Robot
YRC Series

YRC SCARA Robot Controller

PROGRAMMING MANUAL

OMRON

Introduction

Our sincere thanks for your purchase of this OMRON YRC series robot controller.

This manual describes robot program commands and related information for using OMRON YRC series robot controllers. Be sure to read this manual carefully as well as related manuals and comply with their instructions for using the OMRON robot controllers safely and correctly.
For details on how to operate OMRON robot controllers, refer to the separate controller user's manual that comes with the OMRON robot controller.

Applicable controllers: YRC (4-axis controller)

Safety precautions

Be sure to read before using

Before using the OMRON robot controller, be sure to read this manual and related manuals, and follow their instructions to use the robot controller safely and correctly.
Warning and caution items listed in this manual relate to OMRON robot controllers.
When this robot controller is used in a robot controller system, please take appropriate safety measures as required by the user's individual system.
This manual classifies safety caution items and operating points into the following levels, along with symbols for signal words "CAUTION" and "NOTE".

CAUTION

"CAUTION" indicates a potentially hazardous situation which, if not avoided, could result in minor or moderate injury or damage to the equipment or software.

NOTE

Primarily explains function differences, etc., between software versions.

MEMO

Explains robot operation procedures in a simple and clear manner.

Note that the items classified into "CAUTION" might result in serious injury depending on the situation or environmental conditions. So always comply with CAUTION instructions since these are essential to maintain safety.
Keep this manual carefully so that the operator can refer to it when needed. Also make sure that this manual reaches the end user.

■ System design precautions

CAUTION

When the program execution stops before it is complete, the program re-executes the command that has stopped. Keep this point in mind when re-executing the program, for example, when using an arch motion with the MOVE command, a relative movement command such as the MOVEI or DRIVEI command, or a communication command such as the SEND command.

CONTENTS

Chapter 1 Writing Programs

1 The OMRON Robot Language 1-1

2 Characters 1-1

3 Program Basics 1-1

4 Program Names 1-2

5 Identifiers 1-7

6 Comment 1-7

7 Command Statement Format 1-8

Chapter 2 Constants

1 Outline 2-1

2 Numeric constants 2-1

2.1 Integer constants 2-1

2.2 Real constants 2-1

3 Character constants 2-2

Chapter 3 Variables

1 Outline 3-1

2 User Variables & System Variables 3-2

2.1 User Variables 3-2

2.2 System Variables 3-2

3 Variable Names 3-3

3.1 Dynamic Variable Names 3-3

3.2 Static Variable Names 3-3

4 Variable Types 3-4

4.1 Numeric variables 3-4

4.2 Character variables 3-4

5 Array variables 3-5

6 Value Assignments 3-5

7 Type Conversions **3-6**

8 Value Pass-Along & Reference Pass-Along **3-6**

9 System Variables **3-7**

9.1 Point data variable 3-7
9.2 Shift coordinate variable 3-8
9.3 Point element variable 3-9
9.4 Shift element variable 3-10
9.5 Parallel input variable 3-10
9.6 Parallel output variable 3-11
9.7 Internal output variable 3-12
9.8 Arm lock output variable 3-13
9.9 Timer output variable 3-14
9.10 Serial input variable 3-15
9.11 Serial output variable 3-16
9.12 Serial word input 3-17
9.13 Serial double word input 3-17
9.14 Serial word output 3-18
9.15 Serial double word output 3-18

10 Bit Settings **3-19**

11 Valid range of variables **3-20**

11.1 Valid range of dynamic variables 3-20
11.2 Valid range of static variables 3-20
11.3 Valid range of dynamic array variables 3-20

12 Clearing variables **3-21**

12.1 Clearing dynamic variables 3-21
12.2 Clearing static variables 3-21

Chapter 4 Expressions and Operations

1 Arithmetic operations **4-1**

1.1 Arithmetic operators 4-1
1.2 Relational operators 4-1
1.3 Logic operations 4-2
1.4 Priority of arithmetic operation 4-3
1.5 Data format conversion 4-3

2 Character string operations **4-4**

2.1 Character string connection 4-4
2.2 Character string comparison 4-4

3 Point data format 4-5

4 DI/DO conditional expressions 4-6

Chapter 5 Multi-tasking

1 Outline 5-1

2 Task definition 5-1

3 Task status and transition 5-2

3.1 Starting tasks 5-2
3.2 Task scheduling 5-3
3.3 Condition wait in task 5-4
3.4 Suspending tasks (SUSPEND) 5-5
3.5 Restarting tasks (RESTART) 5-5
3.6 Deleting tasks 5-6
3.7 Stopping tasks 5-7

4 Multi-task program example 5-8

5 Sharing the data 5-8

6 Cautionary Items 5-9

Chapter 6 Sequence function

1 Sequence function 6-1

2 Creating a sequence program 6-1

2.1 Programming method 6-1
2.2 Compiling 6-2

3 Executing a sequence program 6-4

3.1 Sequence program STEP execution 6-4

4 Creating a sequence program 6-5

4.1 Assignment statements 6-5
4.2 Input/output variables 6-5
4.3 Timer definition statement 6-7
4.4 Logical operators 6-7
4.5 Priority of logic operations 6-8
4.6 Sequence program specifications 6-8

CONTENTS

Chapter 7 Robot Language Lists

How to read the robot language table			**7-1**
Command list in alphabetic order			**7-3**
Function Specific			**7-7**
Functions: in alphabetic order			**7-13**
Functions: operation-specific			**7-15**
1	ABS	Acquires absolute values	7-17
2	ABSINIT	Resets the current position of a specified axis	7-18
3	ABSRPOS	Acquires a machine reference	7-20
4	ABSRST	Absolute motor axis return-to-origin operation	7-21
5	ACCEL	Specifies/acquires the acceleration coefficient parameter	7-22
6	ARCH	Specifies/acquires the acceleration coefficient parameter	7-23
7	ARMCND	Arm status acquisition	7-25
8	ARMTYPE	SCARA robot hand system	7-26
9	ATN	Acquires the arctangent of the specified value	7-27
10	ASPEED	Sets the automatic movement speed	7-28
11	AXWGHT	Sets/acquires the axis tip weight	7-29
12	CALL	Calls a sub-procedure	7-30
13	CHANGE	Switches the hand	7-31
14	CHGPRI	Changes the priority ranking of a specified task	7-32
15	CHR$	Acquires a character with the specified character code	7-33
16	COS	Acquires the cosine value of a specified value	7-34
17	CURTRQ	Acquires the current torque of the specified axis	7-34
18	CUT	Terminates another sub task which is currently being executed	7-35
19	DATE$	Acquires the date	7-36
20	DECEL	Specifies/acquires the deceleration rate parameter	7-37
21	DECLARE	Declares that a sub-routine or sub-procedure is to be used within the COMMON program	7-38
22	DEF FN	Defines functions which can be used by the user	7-40
23	DEGRAD	Angle conversion (angle → radian)	7-41
24	DELAY	Program execution waits for a specified period of time	7-42
25	DI	Acquires the input status from the parallel port	7-43
26	DIST	Acquires the distance between 2 specified points	7-44

27	DIM	Declares array variable	7-45
28	DO	Outputs to parallel port	7-46
29	DRIVE	Executes absolute movement of specified axes	7-47
30	DRIVEI	Moves the specified robot axes in a relative manner	7-55
31	END SELECT	Ends the SELECT CASE statement	7-60
32	END SUB	Ends the sub-procedure definition	7-61
33	ERR / ERL	Acquires the error code / error line No	7-62
34	EXIT FOR	Terminates the FOR to NEXT statement loop	7-63
35	EXIT SUB	Terminates the sub-procedure defined by SUB to END	7-64
36	EXIT TASK	Terminates its own task which is in progress	7-65
37	FOR to NEXT	Performs loop processing until the variable-specified value is exceeded	7-66
38	GOSUB to RETURN	Jumps to a sub-routine	7-67
39	GOTO	Executes an unconditional jump to the specified line	7-68
40	HALT	Stops the program and performs a reset	7-69
41	HAND	Defines the hand	7-70
	41.1 For SCARA Robots		7-70
42	HOLD	Temporarily stops the program	7-73
43	IF	Evaluates a conditional expression value, and executes the command in accordance with the conditions	7-74
	43.1 Simple IF statement		7-74
	43.2 Block IF statement		7-75
44	INPUT	Assigns a value to a variable specified from the programming box	7-76
45	INT	Truncates decimal fractions	7-77
46	JTOXY	Performs axis unit system conversions (pulse → mm)	7-78
47	LABEL Statement	Defines labels at program lines	7-79
48	LEFT$	Extracts character strings from the left end	7-80
49	LEFTY	Sets the SCARA robot hand system as a left-hand system	7-81
50	LEN	Acquires a character string length	7-82
51	LET	Assigns values to variables	7-83
52	LO	Arm lock output	7-86
53	LOCx	Specifies/acquires point data or shift data for a specified axis	7-87
54	LSHIFT	Left-shifts a bit	7-89
55	MCHREF	Acquires a machine reference	7-90
56	MID$	Acquires a character string from a specified position	7-91
57	MO	Outputs a specified value to the MO port (internal output)	7-92
58	MOVE	Performs absolute movement of all robot axes	7-93

59	MOVEI	Performs absolute movement of all robot axes	7-109
60	OFFLINE	Sets a specified communication port to the "offline" mode	7-114
61	ORD	Acquires a character code	7-115
62	ON ERROR GOTO	Jumps to a specified label when an error occurs	7-116
63	ON to GOSUB	Executes the subroutine specified by the <expression> value	7-117
64	ON to GOTO	Jumps to the label specified by the <expression> value	7-118
65	ONLINE	Sets the specified communication port to the "online" mode	7-119
66	ORGORD	Specifies/acquires the robot's return-to-origin sequence	7-120
67	ORIGIN	Performs an incremental mode axis return-to-origin	7-121
68	OUT	Turns ON the specified port output	7-122
69	OUTPOS	Specifies/acquires the OUT enable position parameter of the robot	7-123
70	PATH	Specifies the main robot axis PATH motion path	7-125
71	PATH END	Ends the movement path setting	7-131
72	PATH SET	Starts the movement path setting	7-132
73	PATH START	Starts the PATH motion	7-134
74	PDEF	Defines the pallet used to execute pallet movement commands	7-135
75	PMOVE	Executes a pallet movement command for the robot	7-136
76	Pn	Defines points within a program	7-140
77	PPNT	Creates pallet point data	7-142
78	PRINT	Displays the specified expression value at the programming box	7-143
79	RADDEG	Performs a unit conversion (radians → degrees)	7-144
80	REM	Inserts a comment	7-145
81	RESET	Turns OFF the bits of specified ports, or clears variables	7-146
82	RESTART	Restarts another task during a temporary stop	7-147
83	RESUME	Resumes program execution after error recovery processing	7-148
84	RETURN	Processing which was branched by GOSUB, is returned to the next line after GOSUB	7-149
85	RIGHT$	Extracts a character string from the right end of another character string	7-150
86	RIGHTY	Sets the SCARA robot hand system to "Right"	7-151
87	RSHIFT	Shifts a bit value to the right	7-152
88	Sn	Defines the shift coordinates in the program	7-153
89	SELECT CASE	Executes the specified command block in accordance with the <expression> value	7-154
90	SEND	Sends <read file> data to the <write file>	7-155
91	SERVO	Controls the servo status	7-157
92	SET	Turns the bit at the specified output port ON	7-158
93	SHARED	Enables sub-procedure referencing without passing on the variable	7-159

94	SHIFT	Sets the shift coordinates	7-160
95	SIN	Acquires the sine value for a specified value	7-161
96	SO	Outputs a specified value to the serial port	7-162
97	SPEED	Changes the program movement speed	7-163
98	START	Starts a new task	7-164
99	STR$	Converts a numeric value to a character string	7-165
100	SQR	Acquires the square root of a specified value	7-166
101	SUB to END SUB	Defines a sub-procedure	7-167
102	SUSPEND	Temporarily stops another task which is being executed	7-169
103	SWI	Switches the program being executed	7-170
104	TAN	Acquires the tangent value for a specified value	7-171
105	TCOUNTER	Timer & counter	7-172
106	TIME$	Acquires the current time	7-173
107	TIMER	Acquires the current time	7-174
108	TO	Outputs a specified value to the TO port	7-175
109	TOLE	Specifies/acquires the tolerance parameter	7-176
110	TORQUE	Specifies/acquires the maximum torque command value which can be set for a specified axis	7-177
111	TRQSTS	Acquires the status when DRIVE statement ends	7-179
112	TRQTIME	Sets/acquires the time-out period for the torque limit setting option	7-180
113	VAL	Converts character strings to numeric values	7-182
114	WAIT	Waits until the conditions of the DI/DO conditional expression are met	7-183
115	WAIT ARM	Waits until the robot axis operation is completed	7-184
116	WEIGHT	Specifies/acquires the tip weight parameter	7-185
117	WEND	Ends the WHILE statement's command block	7-186
118	WHERE	Acquires the arm's current position (pulse coordinates)	7-187
119	WHILE to WEND	Repeats an operation for as long as a condition is met	7-188
120	WHRXY	Acquires the arm's current position in Cartesian coordinates	7-189
121	XYTOJ	Converts the main group axes Cartesian coordinate data ("mm") to joint coordinate data ("pulse")	7-190
122	_SYSFLG	Axis status monitoring flag	7-190

Chapter 8 PATH Statements

1 Overview **8-1**

2 Features **8-1**

CONTENTS

3 How to use 8-1

4 Cautions when using this function 8-2

Chapter 9 Limitless motion

1 Overview 9-1

2 Operation Procedure 9-1

2.1 Parameters 9-1
2.2 Robot language 9-1
2.3 Sample program 9-2

3 Restrictions 9-3

Chapter 10 Data file description

1 Overview 10-1

1.1 Data file types 10-1
1.2 Cautions 10-1

2 Program file 10-2

2.1 All programs 10-2
2.2 One program 10-3

3 Point file 10-4

3.1 All points 10-4
3.2 One point 10-6

4 Point comment file 10-8

4.1 All point comments 10-8
4.2 One point comment 10-8

5 Parameter file 10-10

5.1 All parameters 10-10
5.2 One parameter 10-12

6 Shift coordinate definition file 10-13

6.1 All shift data 10-13
6.2 One shift definition 10-14

7 Hand definition file 10-15

7.1 All hand data 10-15
7.2 One hand definition 10-16

CONTENTS

8 Pallet definition file	**10-17**
8.1 All pallet definitions	10-17
8.2 One pallet definition	10-20
9 All file	**10-23**
9.1 All files	10-23
10 Program directory file	**10-24**
10.1 Entire program directory	10-24
10.2 One program	10-25
11 Parameter directory file	**10-26**
11.1 Entire parameter directory	10-26
12 Variable file	**10-27**
12.1 All variables	10-27
12.2 One variable	10-29
13 Constant file	**10-30**
13.1 One character string	10-30
14 Array variable file	**10-31**
14.1 All array variables	10-31
14.2 One array variable	10-32
15 DI file	**10-33**
15.1 All DI information	10-33
15.2 One DI port	10-34
16 DO file	**10-35**
16.1 All DO information	10-35
16.2 One DO port	10-36
17 MO file	**10-37**
17.1 All MO information	10-37
17.2 One MO port	10-38
18 LO file	**10-39**
18.1 All LO information	10-39
18.2 One LO port	10-40
19 TO file	**10-41**
19.1 All TO information	10-41
19.2 One TO port	10-42

20 SI file **10-43**
20.1 All SI information 10-43
20.2 One SI port 10-44

21 SO file **10-45**
21.1 All SO information 10-45
21.2 One SI port 10-46

22 Error message history file **10-47**
22.1 All error message history 10-47

23 Error Message History Details File **10-48**
23.1 General error message history details 10-48

24 Machine reference file **10-49**
24.1 All machine reference file 10-49

25 EOF file **10-50**
25.1 EOF data 10-50

26 Serial port communication file **10-51**
26.1 Serial port communication file 10-51

27 SIW file **10-52**
27.1 All SIW 10-52
27.2 One SIW data 10-53

28 SOW file **10-54**
28.1 All SIW 10-54
28.2 One SOW data 10-55

29 Ethernet port communication file **10-56**
29.1 Ethernet port communication file 10-56

Chapter 11 User program examples

1 Basic operation **11-1**
1.1 Directly writing point data in program 11-1
1.2 Using point numbers 11-2
1.3 Using shift coordinates 11-3
1.4 Palletizing 11-4
1.4.1 Calculating point coordinates 11-4
1.4.2 Utilizing pallet movement 11-6
1.5 DI/DO (digital input and output) operation 11-7

2 Application 11-8
- 2.1 Pick and place between 2 points 11-8
- 2.2 Palletizing 11-10
- 2.3 Pick and place of stacked parts 11-12
- 2.4 Parts inspection (Multi-tasking example) 11-14
- 2.5 Sealing 11-17
- 2.6 Connection to an external device through RS-232C (example 1) 11-18
- 2.7 Connection to an external device through RS-232C (example 2) 11-19

Chapter 12 Online commands

1 Online Command List 12-1
- 1.1 Online command list: Function specific 12-1
- 1.2 Online command list: In alphabetic order 12-4

2 Key operation 12-6
- 2.1 Changing the mode 12-6
- 2.2 AUTO mode operation 12-7
- 2.3 MANUAL mode operation 12-9

3 Utility operation 12-12
- 3.1 Acquiring the program execution status 12-12
- 3.2 Copy 12-12
- 3.3 Erase 12-14
- 3.4 Rename program name 12-16
- 3.5 Changing the program attribute 12-16
- 3.6 Initialize 12-17
- 3.7 Setting the display language 12-18
- 3.8 Setting the coordinates and units in MANUAL mode 12-19
- 3.9 Clearing the programming box error message 12-19
- 3.10 Setting the UTILITY mode 12-20
- 3.11 Checking and setting the date 12-22
- 3.12 Checking and setting the time 12-23

4 Data handling 12-24
- 4.1 Acquiring the display language 12-24
- 4.2 Acquiring the access level 12-24
- 4.3 Acquiring the arm status 12-25
- 4.4 Acquiring the break point status 12-25
- 4.5 Acquiring the controller configuration status 12-26
- 4.6 Acquiring the execution level 12-26
- 4.7 Acquiring the mode status 12-27
- 4.8 Acquiring the message 12-28

CONTENTS

4.9 Acquiring return-to-origin status 12-29
4.10 Acquiring the absolute reset status 12-29
4.11 Acquiring the servo status 12-30
4.12 Acquiring the sequence program execution status 12-30
4.13 Acquiring the speed setting status 12-31
4.14 Acquiring the point coordinates and units 12-31
4.15 Acquiring the version information 12-32
4.16 Acquiring the current positions 12-32
4.17 Acquiring the tasks in RUN or SUSPEND status 12-34
4.18 Acquiring the tasks operation status 12-35
4.19 Acquiring the shift status 12-35
4.20 Acquiring the hand status 12-36
4.21 Acquiring the remaining memory capacity 12-36
4.22 Acquiring the emergency stop status 12-37
4.23 Acquiring the error status by self-diagnosis 12-37
4.24 Acquiring the option slot status 12-38
4.25 Acquiring various values 12-39
4.26 Data readout processing 12-41
4.27 Data write processing 12-42
4.28 Current torque value acquisition 12-43

5 Executing the robot language independently 12-44

5.1 Switching the program 12-44
5.2 Other robot language command processing 12-45

6 Control codes 12-46

6.1 Interrupting the command execution 12-46

Chapter 13 IO commands

1 Overview 13-1

2 IO command format 13-1

3 Sending and receiving IO commands 13-2

4 IO command list 13-4

CONTENTS

5 IO command description **13-5**

5.1 MOVE command 13-5
5.2 MOVEI command 13-6
5.3 Pallet movement command 13-6
5.4 Jog movement command 13-7
5.5 Inching movement command 13-7
5.6 Point teaching command 13-8
5.7 Absolute reset movement command 13-8
5.8 Absolute reset command 13-9
5.9 Return-to-origin command 13-9
5.10 Servo command 13-10
5.11 Manual movement speed change command 13-11
5.12 Auto movement speed change command 13-11
5.13 Program speed change command 13-11
5.14 Shift designation change command 13-12
5.15 Hand designation change command 13-12
5.16 Arm designation change command 13-12
5.17 Point display unit designation command 13-12

Chapter 14 Appendix

1 Reserved word list **15-1**

2 Robot Language Lists: Command list in alphabetic order **15-3**

3 Robot Language Lists: Function Specific **7**

4 Functions: in alphabetic order **13**

5 Functions: operation-specific **15**

6 Execution Level **17**

Index

Chapter 1

Writing Programs

1 The OMRON Robot Language 1-1

2 Characters 1-1

3 Program Basics 1-1

4 Program Names 1-2

5 Identifiers 1-7

6 Comment 1-7

7 Command Statement Format 1-8

1 The OMRON Robot Language

The OMRON robot language is similar to BASIC (Beginner's All-purpose Symbolic Instruction Code) and makes even complex robot movements easy to program. This manual explains how to write robot control programs with the OMRON robot language, including actual examples on how its commands are used.

2 Characters

The characters and symbols used in the OMRON robot language are shown below.
Only 1-byte characters can be used.

- **Alphabetic characters**
  A to Z, a to z
- **Numbers**
  0 to 9
- **Symbols**
  ( ) [ ] + - * / ^ = < > & | ~ _ % ! # $: ; , . " ' @ ?
- **katakana (Japanese phonetic characters)**

MEMO

- Katakana (Japanese phonetic characters) cannot be entered from a programming box. Katakana can be used when communicating with a host computer (if it handles katakana).
- Spaces are also counted as characters (1 space = 1 character).

3 Program Basics

NOTE
- For sub-procedure details, refer to the "CALL" and "SUB ~ END SUB" items.

NOTE
- For details regarding user defined functions, refer to the "DEF FN" item.

Programs are written in a "1 line = 1 command" format, and every line must contain a command. Blank lines (lines with no command) will cause an error when the program is compiled (creation of execution objects). The program's final line, in particular, must not be blank.

To increase the program's efficiency, processes which are repeated within the program should be written as subroutines or sub-procedures which can be called from the main routine. Moreover, same processing items which occurs in multiple programs should be written as common routines within a program named [COMMON], allowing those processing items to be called from multiple programs.

User functions can be defined for specific calculations. Defined user functions are easily called, allowing even complex calculations to be easily performed.

Multi-task programs can also be used to execute multiple command statements simultaneously in a parallel processing manner.

Using the above functions allows easy creation of programs which perform complex processing.

4 Program Names

Each program to be created in the robot controller must have its own name.
Programs can be named as desired provided that the following conditions are satisfied:

- Program names may contain no more than 8 characters, comprising a combination of alphanumeric characters and underscores (_).
- Each program must have a unique name (no duplications).

The 4 program names shown below are reserved for system operations, and programs with these names have a special meaning.

A) FUNCTION
B) SEQUENCE
C) _SELECT
D) COMMON

The functions of these programs are explained below.

A) FUNCTION

Functions Pressing the USER key in "PROGRAM" mode or "MANUAL" mode allows the user function to be used. When user functions are used in the "PROGRAM" mode, commands (MOVE, GOTO, etc.) which are frequently used during program editing can be entered by function keys. When used in "MANUAL" mode, DO output is available with the function keys without running the program. The FUNCTION program defines function keys which are used to execute user functions. The desired functions can be freely assigned to the function keys.

SAMPLE

```
'FOR MANUAL MODE

*M_F1:'DO(20)ALTERNATE
    DO(20)=~DO(20) ·················· DO (20) ON/OFF highlighting occurs when the key is pressed.
*M_F2:'DO(21)ALTERNATE
    DO(21)=~DO(21) ·················· DO (21) is highlighted.
        :
*M_F6:'DO(25)MOMENTARY
    DO(25)=1 DO (25) is set to "1" when the key is pressed.
    DO(25)=0 DO (25) is set to "0" when the key is released.

*M_F7:'MOTION
    MOVE P,P1 ······················ Movement to Point 1 occurs.
    MOVE P,P2 ······················ Movement to Point 2 occurs.
        :
'FOR PROGRAM MODE
*P_F1:'MOVE P, ······················ [MOVE P,] is written to the program when the key is pressed.
*P_F6:'MOVE L, ······················ [MOVE L,] is written to the program when the key is pressed.
*P_F2:'GOTO * ······················· [GOTO *] is written to the program when the key is pressed.
        :
```

● Registering editing function keys used in the PROGRAM mode

Format
*P_F <n>: ' <character string>

Values <n> ....Denotes the No. of the function key being registered (n = 1 to 15).

<character string>......................The character string which is registered and displayed for the function key.

MEMO

- Although up to 65 characters can be entered for a <character string>, no more than 7 characters are displayed on the Menu.

SAMPLE

*P_F2:'MOVE P, ····················	Registers "MOVE P," at the [F2] key.
*P_F8:'DELAY ·····················	Registers "DELAY" at the [F8] key.

● Registering output command function keys used in the MANUAL mode

Format

```
*M_F <n>:' <character string>
<Output statement 1>
<Output statement 2>
```

Values <n> ....Denotes the No. of the function key being registered (n = 1 to 15).

<character string>......................The character string which is displayed for the function key.

<Input/output statement 1> ....Command statement which is executed when the key is pressed.

<Input/output statement 2> ....Command statement which is executed when the key is released

MEMO

- Although up to 65 characters can be entered for a <character string>, no more than 7 characters are displayed on the Menu.

SAMPLE

*M_F2:'MOMENT···················	Displays "MOMENT" at the [F2] key.
DO(20)=1 ·······················	DO (20) is turned ON when the [F2] key is pressed.
DO(20)=0 ·······················	DO (20) is turned OFF when the [F2] key is released.
*M_F14:'ALTER ····················	Displays "ALTER" at the [F14] key.
DO(20)=~DO(20) ···················	The DO(20) output status is highlighted when the [F14] key is pressed.

REFERENCE For details, refer to the relevant controller manual.

B) SEQUENCE

Functions Unlike standard robot programs, the YRC Controller allows the execution of high-speed-processing programs (sequence programs) in response to robot inputs and outputs (DI, DO, MO, LO, TO, SI, SO). Specify a program name of "SEQUENCE" to use this function, thus creating a pseudo PLC within the controller.

When the controller is in the AUTO or MANUAL mode, a SEQUENCE program can be executed in fixed cycles (regardless of the program execution status) in response to dedicated DI10 (sequence control input) input signals, with the cycle being determined by the program capacity. For details, see Chapter 7 "4.6 Sequence program specifications".
This allows sensors, push-button switches, and solenoid valves, etc., to be monitored and operated by input/output signals.
Moreover, because the sequence programs are written in robot language, they can easily be created without having to use a new and unfamiliar language.

SAMPLE

```
DO(20)=~DI(20)
DO(25)=DI(21) AND DI(22)
MO(26)=DO(26) OR DO(25)
            :
```

REFERENCE For details, see Chapter 7 "Sequence function".

C)_SELECT

Functions This function allows the user to create a program which is always selected and executed when the robot program is reset. Specify a program name of "_SELECT" to use this function. For example, if multiple programs exist, and there is a need to switch between the programs by using DI inputs, simply create a program-switching program named "_SELECT". Even if another program is running, the system always returns to this program when a reset input occurs after that program stops. The various reset types and their corresponding processing are as follows (also refer to the program example shown below):

1. When a reset is executed from the Programming Box, a query displays, asking if a change to "_SELECT" is desired. If "No" is pressed, a selection screen displays, allowing the user to select whether or not a reset is to be executed.
2. When reset by the HALT command in a program, dedicated DI (reset signal) or online command, the system switches to the "_SELECT" program.
3. The operation which occurs at power ON varies according to the "execution level". If the execution level has been selected as "execute program reset at power ON", a reset is executed at power ON, and "_SELECT" is then selected.

NOTE

- For details regarding the "execution level", refer to the controller manual.

A program is selected according to the value input from DI3( ).
When DI3( ) is 0, the system repeatedly monitors the DI input.
When DI3( ) is from 1 to 3, the matching program is selected.
When DI3( ) is other than the above cases, the system quits the program that is currently running.

SAMPLE

```
ON ERROR GOTO *ER1
*ST:
    SELECT CASE DI3( ) ················ Branching occurs based on the DI3 "( )" value.
        CASE 0
            GOTO *ST ················· If "0", a return to "*ST" occurs, and the processing is repeated.
        CASE 1
            SWI <PART1> ·············· If "1"
        CASE 2
            SWI <PART2> ·············· If "2"
        CASE 3
            SWI <PART3> ·············· If "3"
        CASE ELSE
            GOTO *FIN ················ For any other value, a jump to "*FIN" occurs, and processing ends.
    END SELECT
    GOTO *ST
*FIN:
HALT
*ER1:
    IF ERR=&H0303 THEN *NEXT_L ········· A return is executed if a "no program exists" error occurs.
    ON ERROR GOTO 0 ················· For any other error, processing ends.
*NEXT_L:
RESUME NEXT
```

NOTE

- Using an ON ERROR statement allows running the program in a loop not ending in an error even without the program name specified by a SWI statement.
- An error code issued during execution of the program is input into a variable ERR. "ERR=&0303" means "Program doesn't exist".

REFERENCE For details, refer to the command explanations given in this manual.

1 2 3 4 5 6 7

D) COMMON

Functions A separate "COMMON" program can be created to perform the same processing in multiple robot programs. The common processing routine which has been written in the COMMON program can be called and executed as required from multiple programs. This enables efficient use of the programming space.

The sample COMMON program shown below contains two processing items (obtaining the distance between 2 points (SUB *DISTANCE), and obtaining the area (*AREA)) which are written as common routines, and these are called from separate programs (SAMPLE 1 and SAMPLE 2).
When SAMPLE1 or SAMPLE2 is executed, the SUB *DISTANCE (A!,B!,C!) and the *AREA routine specified by the DECLARE statement are executed.

SAMPLE

```
Program name: SAMPLE1
    DECLARE SUB *DISTANCE(A!,B!,C!)
    DECLARE *AREA
    X!=2.5
    Y!=1.2
    CALL  *DISTANCE(X!,Y!,REF C!)
    GOSUB *AREA
    PRINT C!,Z!
    HALT

Program name: SAMPLE2
    DECLARE SUB *DISTANCE(A!,B!,C!)
    DECLARE *AREA
    X!=5.5
    Y!=0.2
    CALL  *DISTANCE(X!,Y!,REF C!)
    GOSUB *AREA
    PRINT C!,Z!
    HALT

Program name: COMMON ···················Common routine
    SUB *DISTANCE(A!,B!,C!)
        C!=SQR(A!^2+B!^2)
    END SUB
    *AREA:
        Z!=X!*Y!
    RETURN
```

REFERENCE For details, refer to the command explanations given in this manual.

5 Identifiers

"Identifiers" are a combination of characters and numerals used for label names, variable names, and procedure names. Identifiers can be named as desired provided that the following conditions are satisfied:

- Identifiers must consist only of alphanumeric characters and underscores (_). Special symbols cannot be used, and the identifier must not begin with an underscore (_).
- The identifier length must not exceed 16 characters (all characters beyond the 16th character are ignored).
- Up to 500 identifiers may be used.
- Variable names must not be the same as a reserved word, or the same as a name defined as a system variable. Moreover, variable name character strings must begin with an alphabetic character. For label names, however, the "*" mark may be immediately followed by a numeric character.

SAMPLE

```
LOOP, SUBROUTINE, GET_DATA
```

REFERENCE For details regarding reserved words, see Chapter 15 "1. Reserved word list".

6 Comment

Characters which follow REM or an apostrophe mark (" ' ") are processed as a comment. Comment statements are not executed. Moreover, comments may begin at any point in the line.

SAMPLE

```
REM *** MAIN PROGRAM ***
      (Main program)
'*** SUBROUTINE ***
      (Subroutine)
HALT   'HALT COMMAND ··············· This comment may begin at any point in the line.
```

7 Command Statement Format

Format
[<label>:] <statement> [<operand>]

One robot language command must be written on a single line and arranged in the format shown below:

- Items enclosed in [ ] can be omitted.
- Items enclosed in < > must be written in a specific format.
- Items not enclosed in < > should be written directly as shown.
- Items surrounded by | | are selectable.
- The label can be omitted. When using a label, it must always be preceded by an asterisk (*), and it must end with a colon (:) (the colon is unnecessary when a label is used as a branching destination).

For details regarding labels, refer to Chapter 8 "45. LABEL Statement".

- Operands may be unnecessary for some commands.
- Programs are executed in order from top to bottom unless a branching instruction is given.

1 line may contain no more than 75 characters.

Chapter 2

Constants

1 Outline..2-1

2 Numeric constants..2-1

3 Character constants 2-2

1 Outline

Constants can be divided into two main categories: "numeric types" and "character types". These categories are further divided as shown below.

Category	Type	Details/Range
Numeric type	Integer type	Decimal constants -1,073,741,824 to 1,073,741,823
		Binary constants &B0 to &B11111111
		Hexadecimal constants &H80000000 to &H7FFFFFFF
	Real type	Single-precision real numbers -999,999.9 to +999,999.9
		Exponential format single-precision real numbers $-1.0*10^{38}$ to $+1.0*10^{38}$
Character type	Character string	Alphabetic, numeric, special character, or katakana (Japanese) character string of 75 bytes or less.

2 Numeric constants

2.1 Integer constants

1. **Decimal constants**
   Integers from –1,073,741,824 to 1,073,741,823 may be used.
2. **Binary constants**
   Unsigned binary numbers of 8 bits or less may be used. The prefix "&B" is attached to the number to define it as a binary number.
   Range: &B0 (decimal: 0) to &B11111111 (decimal: 255)
3. **Hexadecimal constants**
   Signed hexadecimal numbers of 32 bits or less may be used. The prefix "&H" is attached to the number to define it as a hexadecimal number.
   Range: &H80000000 (decimal: -2,147,483,648) to &H7FFFFFFF (decimal: 2,147,483,647)

2.2 Real constants

1. **Single-precision real numbers**
   Real numbers from -999999.9 to +999999.9 may be used.
   • 7 digits including integers and decimals. (For example, ".0000001" may be used.)
2. **Single-precision real numbers in exponent form**
   Numbers from $-1.0*10^{38}$ to $+1.0*10^{38}$ may be used.
   • Mantissas should be 7 digits or less, including integers and decimals.

Examples:
```
-1. 23456E-12
3. 14E0
1. E5
```

MEMO

• An integer constant range of –1,073,741,824 to 1,073,741,823 is expressed in signed hexadecimal number as &HC0000000 to &H3FFFFFFF.

1
2
3
4
5
6
7

3 Character constants

Character type constants are character string data enclosed in quotation marks ("). The character string must not exceed 75 bytes in length, and it may contain upper-case alphabetic characters, numerals, special characters, or katakana (Japanese) characters.
To include a double quotation mark (") in a string, enter two double quotation marks in succession.

SAMPLE

```
"OMRON ROBOT"
"EXAMPLE OF""A"""･･････････････EXAMPLE OF "A"
PRINT "COMPLETED"
"OMRON ROBOT"
```

Chapter 3

Variables

1	Outline	3-1
2	User Variables & System Variables	3-2
3	Variable Names	3-3
4	Variable Types	3-4
5	Array variables	3-5
6	Value Assignments	3-5
7	Type Conversions	3-6
8	Value Pass-Along & Reference Pass-Along	3-6
9	System Variables	3-7
10	Bit Settings	3-19
11	Valid range of variables	3-20
12	Clearing variables	3-21

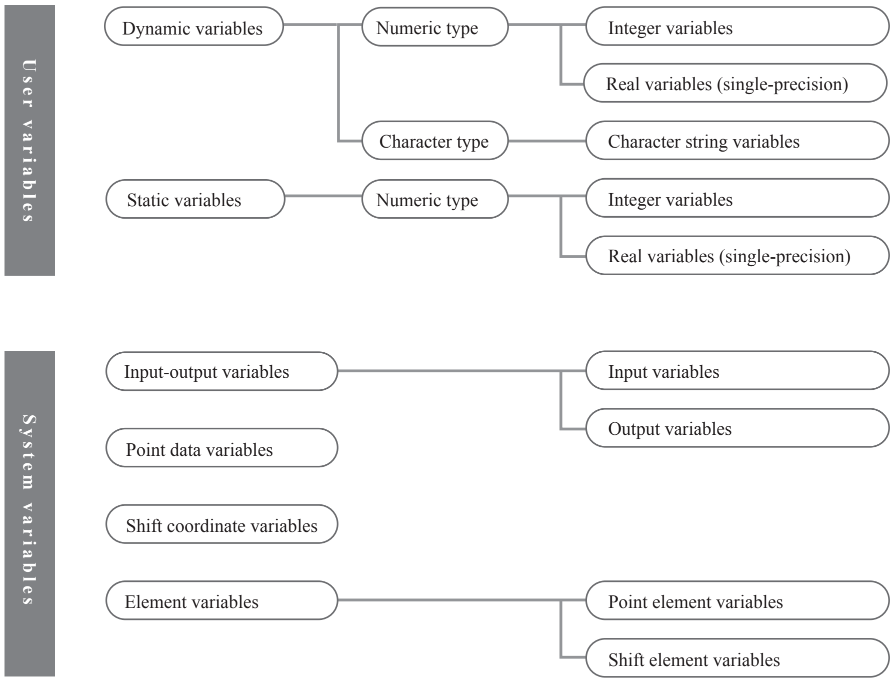
1 Outline

There are "user variables" which can be freely defined, and "system variables" which have pre-defined names and functions.

User variables consist of "dynamic variables" and "static variables". "Dynamic variables" are cleared at program editing, compiling, program resets, and program switching. "Static variables" are not cleared unless the memory is cleared. The names of dynamic variables can be freely defined, and array variables can also be used.

Variables can be used simply by specifying the variable name and type in the program. A declaration is not necessarily required. However, array variables must be pre-defined by a DIM statement.

User variables & system variables



REFERENCE For details regarding the above array, see Chapter 3 "5 Array variables".

1
2
3
4
5
6
7

2 User Variables & System Variables

2.1 User Variables

Numeric type variables consist of an "integer type" and a "real type", and these two types have different usable numeric value ranges. Moreover, each of these types has different usable variables (character string variables, array variables, etc.), and different data ranges, as shown below.

Category	Variable Type	Details/Range
Dynamic variables	Numeric type	Integer type variables -1,073,741,824 to 1,073,741,823 (Signed hexadecimal constants: &HC0000000 to &H3FFFFFFF)
		Real variables (single-precision) $-1.0*10^{38}$ to $+1.0*10^{38}$
	Character type	Character string variables Alphabetic, numeric, special character, or katakana (Japanese) character string of 75 bytes or less.
Static variables	Numeric type	Integer type variables -1,073,741,824 to 1,073,741,823
		Real variables (single-precision) $-1.0*10^{38}$ to $+1.0*10^{38}$
Array variables	Numeric type	Integer array variables -1,073,741,824 to 1,073,741,823
		Real number array variables (single-precision) $-1.0*10^{38}$ to $+1.0*10^{38}$
	Character type	Character string array variables Alphabetic, numeric, special character, or katakana (Japanese) character string of 75 bytes or less.

NOTE

- Array variables are dynamic variables.

2.2 System Variables

As shown below, system variables have pre-defined names which cannot be changed.

Category	Type	Details	Specific Examples
Input/output variables	Input variable	External signal / status inputs	DI, SI, SIW, SID
	Output variable	External signal / status outputs	DO, SO, SOW, SOD
Point variable		Handles point data	Pnnnn
Shift variable		Specifies the shift coordinate No. as a numeric constant or expression.	Sn
Element variables	Point element variable	Handles point data for each axis, hand system flag, or for the X-arm or Y-arm rotation information.	LOCx (point expression)
	Shift element variable	Handles shift data in element units.	LOCx (shift expression)

REFERENCE For details, see Section "9 System Variables".

3 Variable Names

3.1 Dynamic Variable Names

Dynamic variables can be named as desired, provided that the following conditions are satisfied:

- The name must consist only of alphanumeric characters and underscores (_). Special symbols cannot be used.
- The name must not exceed 16 characters (all characters beyond the 16th character are ignored).
- The name must begin with an alphabetic character.

SAMPLE

```
COUNT      ............................. ○ Use is permitted
COUNT123   ............................. ○ Use is permitted
2COUNT     ............................. × Use is not permitted
```

- Variable names must not be the same as a reserved word.

Variable names must not begin with characters used for system variable names (pre-defined variables). These characters include the following: FN, DIn, DOn, MOn, LOn, TOn, SIn, SOn, Pn, Sn, Hn ("n" denotes a numeric value).

SAMPLE

```
COUNT  ............................. ○ Use is permitted
ABS    ............................. × (Reserved word)
FNAME  ............................. × (FN: pre-defined variable)
S91    ............................. × (Sn: pre-defined variable)
```

REFERENCE For details regarding reserved words, see Chapter 15 "1 Reserved word list".

3.2 Static Variable Names

Static variable names are determined as shown below, and these names cannot be changed.

Variable Type	Variable Name
Integer variable	SGIn (n: 0 to 7)
Real variable	SGRn (n: 0 to 7)

Static variables are cleared only when initializing is executed by a SYSTEM mode or online command.

REFERENCE For details regarding the clearing of static variables, see Section "12 Clearing variables".

4 Variable Types

The type of variable is specified by the type declaration character attached at the end of the variable name. However, because the names of static variables are determined based on their type, no type declaration statement is required.

Type Declaration Character	Variable Type	Specific Examples
$	Character type variables	STR1$
%	Integer type variables	CONT0%, ACT%(1)
!	Real type variables	CNT1!, CNT1

MEMO

- If no type declaration character is attached, the variable is viewed as a real type.
- Variables using the same identifier are recognized to be different from each other by the type of each variable.
  - ASP_DEF%...............Integer variable
  - ASP_DEF ....Real variable

  → ASP_DEF% and ASP_DEF are different variables.
  - ASP_DEF!.................Real variable
  - ASP_DEF ....Real variable

  → ASP_DEF! and ASP_DEF are the same variables.

4.1 Numeric variables

Integer variables

NOTE

- When a real number is assigned to an integer type variable, the decimal value is rounded off to the nearest whole number. For details, refer to Chapter 4 "1.5 Data format conversion".

Integer variables and integer array elements can handle an integer from –1,073,741,824 to 1,073,741,823 (in signed hexadecimal, this range is expressed as &HC0000000 to &H3FFFFFFF).

```
Examples:    R1% = 10
             R2%(2) = R1% + 10000
```

Real variables

NOTE

- The "!" used in real variables may be omitted .

Real variables and real array elements can handle a real number from $–1.0*10^{38}$ to $1.0*10^{38}$.

```
Examples:    R1!   = 10.31
             R2!(2) = R1% + 1.98E3
```

4.2 Character variables

Character variables and character array elements can handle a character string of up to 75 characters. Character strings may include alphabetic characters, numbers, symbols and katakana (Japanese phonetic characters).

```
Examples:    R1$ = "OMRON"
             R2$(2) = R1$ + "MOTOR" "OMRON MOTOR"
```

5 Array variables

Both numeric and character type arrays can be used at dynamic variables.
Using an array allows multiple same-type continuous data to be handled together.
Each of the array elements is referenced in accordance with the parenthesized subscript which appears after each variable name. Subscripts may include integers or <expressions> in up to 3 dimensions.
In order to use an array, a DIM statement must be declared in advance, and the maximum number of elements which can be used is the declared subscripts + 1 (0 ~ number of declared subscripts).

MEMO

- Array variables are all dynamic variables (for details regarding dynamic variables, see Chapter 3 "11 Valid range of variables".)
- The length of an array variable that can be declared with the DIM statement depends on the program size.

Format

```
<variable name>[ % ](<expression>, [<expression>, [<expression>]])
                 !
                 $
```

SAMPLE

A%(1)	······················	Integer array variable
DATA!(1,10,3)	······················	Single-precision real number array variable (3-dimension array)
STRING$(10)	······················	Character array variable

6 Value Assignments

An assignment statement (LET) can also be used to assign a value to a variable.

MEMO

- "LET" directly specifies an assignment statement, and it can always be omitted.

Format

```
[LET] <variable> = <expression>
```

Write the value assignment target variable on the left side, and write the assignment value or the <expression> on the right side. The <expression> may be a constant, a variable, or an arithmetic expression, etc.

REFERENCE For details, refer to Chapter 8 "49 LET (Assignment Statement)"

1
2
3
4
5
6
7

7 Type Conversions

When different-type values are assigned to variables, the data type is converted as described below.

- When a real number is assigned to an integer type:
  The decimal value is rounded off to the nearest whole number.
- When an integer is assigned to a real type:
  The integer is assigned as it is, and is handled as a real number.
- When a numeric value is assigned to a character string type:
  The numeric value is automatically converted to a character string which is then assigned.
- When a character string is assigned to numeric type:
  This assignment is not possible, and an error will occur at the compiling operation. Use the "VAL" command to convert the character string to a numeric value, and that value is then assigned.

8 Value Pass-Along & Reference Pass-Along

A variable can be passed along when a sub-procedure is called by a CALL statement. This pass-along can occur in either of two ways: as a value pass-along, or as a reference pass-along.

Value pass-along

With this method, the variable's value is passed along to the sub-procedure. Even if this value is changed within the sub-procedure, **the content of the call source variable is not changed.**
A value pass-along occurs when the CALL statement's actual argument specifies a constant, an expression, a variable, or an array element (array name followed by (<subscript>)).

Reference pass-along

With this method, the variable's reference (address in memory) is passed along to the sub-procedure. If this value is changed within the sub-procedure, **the content of the call source variable is also changed.**
A reference pass-along occurs when the CALL statement's actual argument specifies an entire array (an array named followed by parenthetical content), or when the actual argument is preceded by "REF".

Value pass-along & reference pass-along

Value pass-along

```
X%=5
CALL *TEST( X% )
PRINT X%
HALT
' SUB ROUTINE
SUB *TEST( A% )
   A%=A%*10
END SUB
```

Execution result: the X% value remains as "5".

Reference pass-along

```
X%=5
CALL *TEST( REF X% )
PRINT X%
HALT
' SUB ROUTINE
SUB *TEST( A% )
   A%=A%*10
END SUB
```

Execution result: the X% value becomes "50".

9 System Variables

The following system variables are pre-defined, and other variable names must not begin with the characters used for these system variable names.

Variable Type	Format	Meaning
Point variable	Pnnn / P " ["<expression>"] "	Specifies a point number.
Shift variable	Sn / S " ["<expression>"] "	Specifies the shift number as a constant or as an expression.
Point element variable	LOCx (<point expression>)	Handles point data for each axis, hand system flag, or for the X-arm or Y-arm rotation information.
Shift element variable	LOCx (<shift expression>)	Handles shift data with the element range.
Parallel input variable	DI(mb), DIm(b)	Parallel input signal status.
Parallel output variable	DO(mb), DOm(b)	Parallel output signal setting and status.
Internal output variable	MO(mb), MOm(b)	Controller's internal output signal setting and status
Arm lock output variable	LO(mb), LOm(b)	Axis-specific movement prohibit.
Timer output variable	TO(mb), TOm(b)	For sequence program's timer function.
Serial input variable	SI(mb), SIm(b)	Serial input signal status.
Serial output variable	SO(mb), SOm(b)	Serial output signal setting and status.
Serial word input	SIW(m)	Serial input's word information status
Serial double-word input	SID(m)	Serial input's double-word information status.
Serial word output	SOW(m)	Serial output's word information status
Serial double-word output	SOD(m)	Serial output's double-word information status.

9.1 Point data variable

This variable specifies a point data number with a numeric constant or expression.

Format

Pnnnn or P" ["<expression>"]"

Values n: Point number .... 0 to 9
Each bracket in quotation marks ("[" "]") must be written. Brackets are not used to indicate an item that may be omitted.

Functions A point data number is expressed with a 'P' followed by a number of 4 digits or less, or an expression surrounded by brackets ("[" <expression> "]").
Point numbers from 0 to 9999 can be specified with point variables.

```
Examples:   P0
            P110
            P [A]
            P [START_POINT]
            P [A(10)]
```

9.2 Shift coordinate variable

This variable specifies a shift coordinate number with a numeric constant or expression.

Format
Sn or S "["<expression>"]"

Values n: Shift number .... 0 to 9

Each bracket in quotation marks ("[" "]") must be written. Brackets are not used to indicate an item that may be omitted.

Functions A shift number is expressed with an 'S' followed by a 1-digit number or an expression surrounded by brackets ("[" <expression> "]").

```
Examples:   S1
            S [A]
            S [BASE]
            S [A(10)]
```

MEMO

- The "shift coordinate range" for each shift number can be changed from the programming box.

9.3 Point element variable

Specifies point data for each axis, hand system flag, or for the X-arm or Y-arm rotation information.

NOTE

- Hand system flags are only valid on SCARA robots, and the point data must be specified in "mm" units.
- The hand system flag value may be 0 (no designation), 1 (right-handed system) or 2 (left-handed system).
- X-arm and Y-arm rotation information is only available in software Ver.1.66M or higher.
- X-arm and Y-arm rotation information is only available on a R6YXTW500 model robot with "mm" units point data. Attempting to use this information on any other robot model will result in the "5.37: Specification mismatch" error, and execution is stopped.
- For details regarding the X-arm and Y-arm rotation information, see Chapter 4 "3. Point data format".

Format

LOCx (<point expression>)

Values

x..X,Y,Z,R,A,B (axis setting), F (hand system flag setting), F1 (X-arm rotation information), F2 (Y-arm rotation information).

Functions Extracts the point-data-specified axis coordinates, hand system flag, X-arm rotation information, and Y-arm rotation information, or changes the value.

Examples:
```
A(1)=LOCX(P10)
  →The X-axis data of P10 is assigned to array variable A(1).
LOCZ(P[A])=100.0
  →The Z-axis data of P[A] is set to 100.0.
LOCF(P100)=1
  →Changing the P100 hand system flag to a right-handed system
  (The P100 point data must be in "mm" units)
LOCF1(P100)=1
  →Changes the P100 X-arm rotation information to 1.
  (The P100 point data must be in "mm" units)
LOCF2(P100)=1
  →Changes the P100 Y-arm rotation information to 1.
  (The P100 point data must be in "mm" units)
B=LOCX(WHERE)
  →Assigns the current X-axis motor pulse value to array variable "B".
C(3)=LOCX(WHRXY)
  →Assigns the current arm position's X-axis to array variable C(3).
D=LOCX(JTOXY(WHERE))
E=LOCX(XYTOJ(WHRXY))
```

MEMO

- Because JTOXY is a command for handling a <point expression>, a "JTOXY(LOCx(WHERE))" or "XYTOJ(LOCx(WHRXY))" command will result in an error.

9.4 Shift element variable

This variable is used with shift data for each element.

Format

LOCx (<shift expression>)

Values x: Axis setting .... X,Y,Z,R

Functions Extracts the shift-data-specified axis coordinates, or changes the value.

Examples: A(1)=LOCX(S1)
→The X data of S1 is assigned to array variable A(1).
LOCR(S[A])=45.0
→The R data of S[A] is set to 45.0º.

9.5 Parallel input variable

This variable is used to indicate the status of parallel input signals.

Format 1

DIm ([b,····················, b])

Format 2

DI (mb,····················, mb)

Values m : port number .... 0 to 7, 10 to 17, 20 to 27
b : bit definition .... 0 to 7
If the bit definition is omitted, bits 0 to 7 are all selected.

Examples: A%=DI1()
→Input status of ports DI(17) to DI(10) is assigned to variable A%.
A 0 to 255 integer can be assigned to A%.
A%=DI5(7,4,0)
→Input status of DI(57), DI(54) and DI(50) is assigned to variable A%.
(If all above signals are 1(ON), then A%=7.)
A%=DI(27,15,10)
→Input status of DI(27), DI(15) and DI(10) is assigned to variable A%.
(If all above signals except DI(10) are 1 (ON), then A%=6.)
WAIT DI(21)=1
→Waits for DI(21) to change to 1(ON).

MEMO

- When specifying multiple bits, specify them from left to right in descending order (large to small).
- A '0' is entered if there is no actual input board.

9.6 Parallel output variable

Specifies the parallel output signal or indicates the output status.

Format 1

DOm ([b,・・・・・・・・・・・・・・・・・・・・, b])

Format 2

DO (mb,・・・・・・・・・・・・・・・・・・・・, mb)

Values

m : port number .... 0 to 7, 10 to 17, 20 to 27
b : bit definition .... 0 to 7
If the bit definition is omitted, bits 0 to 7 are all selected.

Examples:

A%=DO2()
→Output status of DO(27) to DO(20) is assigned to variable A%.
A%=DO5(7,4,0)
→Output status of DO(57), DO(54) and DO(50) is assigned to variable A%.
(If all above signals are 1(ON), then A%=7.)
A%=DO(37,25,20)
→Output status of DO(37), DO(25) and DO(20) is assigned to variable A%.
(If all above signals except DO(20) are 1 (ON), then A%=6.)
DO3()=B%
→Changes to a status in which the DO(37) to DO(30) output can be indicated by B%.
For example, if B% is "123": If a binary number is used, "123" will become "01111011", DO(37) and DO(32) will become "0", and the other bits will become "1".
DO4(5,4,0)=&B101
→DO(45) and DO(40) become "1", and DO(44) becomes "0".

MEMO

- When specifying multiple bits, specify them from left to right in descending order (large to small).
- A '0' is entered if there is no actual input board.

9.7 Internal output variable

Specifies the controller's internal output signals and indicates the signal status.

Format 1

MOm ([b,••••••••••••••••••••, b])

Format 2

MO (mb,••••••••••••••••••••, mb)

Values

m : port number .... 0 to 7, 10 to 17, 20 to27
b : bit definition .... 0 to 7
• If the bit definition is omitted, bits 0 to 7 are all selected.

Functions Internal output variables which are used only in the controller, can be changed and referenced.
These variables are used for signal communications, etc., with the sequence program.
Ports 0 and 1 are for dedicated internal output variables which can only be referenced (they cannot be changed).

1. **Port 0 indicates the status of origin sensors for axes 1 to 8 (in order from bit 0).**
   Each bit sets to '1' when the origin sensor turns ON, and to '0' when OFF.
2. **Port 1 indicates the HOLD status of axes 1 to 8 (in order from bit 0).**
   Each bit sets to '1' when the axis is in HOLD status, and to '0' when not.

Bit	7	6	5	4	3	2	1	0
Port 0	Axis 8	Axis 7	Axis 6	Axis 5	Axis 4	Axis 3	Axis 2	Axis 1
	Origin sensor statuses 0: OFF / 1: ON							
Port 1	Axis 8	Axis 7	Axis 6	Axis 5	Axis 4	Axis 3	Axis 2	Axis 1
	Hold status 0: RELEASE / 1: HOLD (Axis 1 is not used)							

MEMO

- Axes where no origin sensor is connected are always ON.
- Being in HOLD status means that the axis movement is stopped and positioned within the target point tolerance while the servo is still turned ON.
- When the servo turns OFF, the HOLD status is released.
- Axes not being used are set to '1'.

Examples: A%=MO2 ()
→Internal output status of MO(27) to MO(20) is assigned to variable A%.
A%=MO5(7,4,0)
→Internal output status of MO(57), MO(54) and MO(50) is assigned to variable A%.
(If all above signals are 1 (ON), then A%=7.)
A%=MO(37,25,20)
→Internal output status of MO(37), MO(25) and MO(20) is assigned to variable A%.
(If all above signals except MO(25) are 1 (ON), then A%=5.)

MEMO

- When specifying multiple bits, specify them from left to right in descending order (large to small).

9.8 Arm lock output variable

Specifies axis-specific movement prohibit settings.

Format 1

LOm ([b,・・・・・・・・・・・・・・・・・・・・, b])

Format 2

LO (mb,・・・・・・・・・・・・・・・・・・・・, mb)

Values

m : port number 0
b : bit definition .... 0 to 7
• If the bit definition is omitted, bits 0 to 7 are all selected.

Functions

The contents of this variable can be output and referred to as needed.
There is only 1 port, and bits 0 to 7 respectively correspond to axes 1 to 8.
When this bit is ON, movement on the corresponding axis is prohibited.

Examples:
A%=LO0()
→Arm lock status of LO(07) to LO(00) is assigned to variable A%.
A%=LO0(7,4,0)
→Arm lock status of LO(07), LO(04) and LO(00) is assigned to variable A%.
(If all above signals are 1 (ON), then A%=7.)
A%=LO0(06,04,01)
→Arm lock status of LO(06), LO(04) and LO(01) is assigned to variable A%.
(If all above signals except LO(01) are 1 (ON), then A%=6.)

MEMO

- When specifying multiple bits, specify them from left to right in descending order (large to small).
- Servo OFF to ON switching is disabled if an arm lock is in effect at even 1 axis.
- When performing JOG movement in the MANUAL mode, axis movement is possible at axes where an arm lock status is not in effect, even if an arm lock status is in effect at another axis.
- When executing movement commands from the program, etc., the "12.3 XX.Arm lock" error will occur if an arm lock status is in effect at the axis in question. (XX: arm lock enabled axis. Example: M1 S1)

9.9 Timer output variable

This variable is used in the timer function of a sequence program.

Format 1

TOm ([b,‥‥‥‥‥‥‥‥‥‥, b])

Format 2

TO (mb,‥‥‥‥‥‥‥‥‥‥, mb)

Values m : port number 0
b : bit definition .... 0 to 7
• If the bit definition is omitted, bits 0 to 7 are all selected.

Functions The contents of this variable can be changed and referred to as needed.
Timer function can be used only in the sequence program. If this variable is output in a normal program, it is an internal output.

For details regarding sequence program usage examples, refer to the timer usage examples given in Chapter 7 "4.2 Input/output variables".

Examples: A%=TO0()
→Status of TO(07) to TO(00) is assigned to variable A%.
A%=TO0(7,4,0)
→Status of TO(07), TO(04) and TO(00) is assigned to variable A%.
(If all above signals are 1 (ON), then A%=7.)
A%=TO(06,04,01)
→Status of TO(06), TO(04) and TO(01) is assigned to variable A%.
(If all above signals except TO(01) are 1 (ON), then A%=6.)

MEMO

• When specifying multiple bits, specify them from left to right in descending order (large to small).

9.10 Serial input variable

This variable is used to indicate the status of serial input signals.

Format 1

```
SIm ([b,····················, b])
```

Format 2

```
SI (mb,····················, mb
```

Values

m : port number .... 0 to 7, 10 to 17, 20 to 27
b : bit definition .... 0 to 7
• If the bit definition is omitted, bits 0 to 7 are all selected.

Examples:

A%=SI1()
→Input status of ports SI(17) to SI(10) is assigned to variable A%.

A%=SI5(7,4,0)
→Input status of SI(57), SI(54) and SI(50) is assigned to variable A%.
(If all above signals are 1(ON), then A%=7.)

A%=SI(27,15,10)
→Input status of SI(27), SI(15) and SI(10) is assigned to variable A%.
(If all above signals except SI(10) are 1 (ON), then A%=6.)

WAIT SI(21)=1
→Waits until SI(21) sets to 1 (ON).

MEMO

- When specifying multiple bits, specify them from left to right in descending order (large to small).
- A '0' is entered if there is no actual serial board.

9.11 Serial output variable

This variable is used to define the serial output signals and indicate the output status.

Format 1

SOm ([b,···················, b])

Format 2

SO (mb,···················, mb)

Values

m : port number .... 0 to 7, 10 to 17, 20 to 27
b : bit definition .... 0 to 7
• If the bit definition is omitted, bits 0 to 7 are all selected.

Examples:

A%=SO2()
→Output status of SO(27) to SO(20) is assigned to variable A%.

A%=SO5(7,4,0)
→Output status of SO(57), SO(54) and SO(50) is assigned to variable A%.
(If all above signals are 1(ON), then A%=7.)

A%=SO(37,25,20)
→Output status of SO(37), SO(25) and SO(20) is assigned to variable A%.
(If all above signals except SO(25) are 1 (ON), then A%=5.)

SO3()=B%
→Changes to a status in which the DO(37) to DO(30) output can be indicated by B%.
For example, if B% is "123": If a binary number is used, "123" will become "01111011", DO(37) and DO(32) will become "0", and the other bits will become "1".

SO4(5,4,0)=&B101
→DO(45) and DO(40) become "1", and DO(44) becomes "0".

MEMO

- When specifying multiple bits, specify them from left to right in descending order (large to small).
- External output is unavailable if the serial port does not actually exist.

9.12 Serial word input

This variable indicates the status of the serial input word information.

Format

SIW(m)

Values m : Port No. 2 to 15

The acquisition range is 0 (&H0000) to 65535 (&HFFFF).

Examples: A%=SIW(2)

→The input state from SIW (2) is assigned to variable A%.

A%=SIW(15)

→The input state from SIW (15) is assigned to variable A%.

MEMO

- The information is handled as unsigned word data.
- '0' is input if the serial port does not actually exist.

9.13 Serial double word input

This variable indicates the state of the serial input word information as a double word.

Format

SID(m)

Values m : Port No. 2, 4, 6, 8, 10, 12, 14

The acquisition range is -1073741824 (&HC0000000) to 1073741823 (&H3FFFFFFF).

Examples: A%=SID(2)

→The input state from SIW (2) , SIW (3) is assigned to variable A%.

A%=SID(14)

→The input state from SIW (14), SIW (15) is assigned to variable A%.

MEMO

- The information is handled as signed double word data.
- '0' is input if the serial port does not actually exist.
- An error will occur if the value is not within the acquisition range (&H80000000 to &HBFFFFFFF, &H40000000 to &H7FFFFFFF.)
- The lower port number data is placed at the lower address.
  For example, if SIW(2) =&H2345,SIW(3) =&H0001, then SID(2) =&H000123245.

9.14 Serial word output

Outputs to the serial output word information or indicates the output status.

Format
SOW(m)

Values m : Port No. 2 to 15

The output range is 0 (&H0000) to 65535 (&HFFFF).

Note that if a negative value is output, the low-order word information will be output after being converted to hexadecimal.

Examples: A%=SOW(2)
→The output status from SOW (2) is assigned to variable A%.
SOW(15)=A%
→The contents of variable A% are assigned in SOW (15).
If the variable A% value exceeds the output range, the low-order word information will be assigned.
SOW(15)=-255
→The contents of -255 (&HFFFFFF01) are assigned to SOW (15).
-255 is a negative value, so the low-order word information (&HFF01) will be assigned.

MEMO

- The information is handled as unsigned word data.
- If a serial board does not actually exist, the information is not output externally.
- If a value exceeding the output range is assigned, the low-order 2-byte information is output.

9.15 Serial double word output

Output the status of serial output word information in a double word, or indicates the output status.

Format
SOD(m)

Values m : Port No. 2, 4, 6, 8, 10, 12, 14

The output range is -1073741824 (&HC0000000) to 1073741823 (&H3FFFFFFF).

Examples: A%=SOD(2)
→The input status from SOW (2) is assigned to variable A%.
SOD(14)=A%
→The contents of variable A% are assigned in SOD (14).

- The information is handled as signed double word data.
- If a serial board does not actually exist, the information is not output externally.
- An error will occur if the value is not within the output range (&H80000000 to &HBFFFFFFF, &H40000000 to &H7FFFFFFF.)
- The lower port number data is placed at the lower address.
  For example, if SOW(2) =&H2345,SOW(3) =&H0001, then SOD(2) =&H000123245.

10 Bit Settings

Bits can be specified for input/output variables by any of the following methods.

1. Single bit

To specify only 1 of the bits, the target port number and bit number are specified in parentheses.
The port number may also be specified outside the parentheses.

Programming example: DOm(b)DOm(b)

Example: DO(25) Specifies bit 5 of port 2.
DO2(5)

2. Same-port multiple bits

To specify multiple bits at the same port, those bit numbers are specified in parentheses (separated by commas) following the port number.
The port number may also be specified in parentheses.

Programming example: DOm(b,b,…,b) DO(mb,mb,…,mb)

Example: DO2(7,5,3) Specifies DO(27), DO(25), DO(23)
DO(27,25,23)

3. Different-port multiple bits

To specify multiple bits at different ports, the port number and the 2-digit bit number must be specified in parentheses and must be separated by commas.

Programming example: DO(mb,mb,…,mb)

Example: DO(37,25,20) Specifies DO(37), DO(25), DO(20).

4. All bits of 1 port

To specify all bits of a single port, use parentheses after the port number. Methods 2 and 3 shown above can also be used.

Programming example: DOm()

Example: DO2() Specifies all the DO(27) to DO(20) bits
→The same result can be obtained by the following:
DO(27,26,25,24,23,22,21,20)
or,
DO2(7,6,5,4,3,2,1,0)

1
2
3
4
5
6
7

11 Valid range of variables

Variable branching occurs as shown below.

11.1 Valid range of dynamic variables

Dynamic variables are divided into global variables and local variables, according to their declaration position in the program. Global and local variables have different valid ranges.

Variable Type	Explanation
Global variables	Variables are declared outside of sub-procedures (outside of program areas enclosed by a SUB statement and END SUB statement). These variables are valid throughout the entire program.
Local variables	Variables are declared within sub-procedures and are valid only in these sub-procedures.

11.2 Valid range of static variables

Static variable data is not cleared when a program reset occurs. Moreover, variable data can be changed and referenced from any program.
The variable names are determined as shown below (they cannot be named as desired).

Variable type	Variable name
Integer variable	SGIn (n: 0 to 7)
Real variable	SGRn (n: 0 to 7)

11.3 Valid range of dynamic array variables

Dynamic array variables are classified into global array variables and local array variables according to their declaration position in the program.

Variable Type	Explanation
Global variables	Variables are declared outside of sub-procedures (outside of program areas enclosed by a SUB statement and END SUB statement). These variables are valid throughout the entire program.
Local variables	Variables are declared within sub-procedures and are valid only in these sub-procedures.

MEMO

- For details regarding arrays, refer to Chapter 3 "5 Array variables".
- A variable declared at the program level can be referenced from a sub-procedure without being passed along as a dummy argument, by using the SHARED statement (for details, refer to Chapter 8 "91 SHARED").

12 Clearing variables

12.1 Clearing dynamic variables

In the cases below, numeric variables are cleared to zero, and character variables are cleared to a null string. The variable array is cleared in the same manner.

- When a program is edited.
- When program switching occurs (including SWI command execution).
- When program compiling occurs.
- When a program reset occurs.
- When dedicated input signal DI15 (program reset input) was turned on while the program was stopped in AUTO mode.
- When either of the following was initialized in SYSTEM mode.
  1. Program memory (SYSTEM>INIT>MEMORY>PROGRAM)
  2. Entire memory (SYSTEM>INIT>MEMORY>ALL)
- When any of the following online commands was executed.
  @RESET, @INIT PGM, @INIT MEM, @INIT ALL, @SWI
- When the HALT statement was executed in the program.

12.2 Clearing static variables

In the cases below, integer variables and real variables are cleared to zero.

- When the following was initialized in SYSTEM mode.
  Entire memory (SYSTEM>INIT>MEMORY>ALL)
- When any of the following online commands was executed.
  @INIT MEM, @INIT ALL

MEMO

- Static variable values are not cleared even if the program is edited.

Chapter 4

Expressions and Operations

1 Arithmetic operations..4-1

2 Character string operations 4-4

3 Point data format...4-5

4 DI/DO conditional expressions...........................4-6

1 Arithmetic operations

1.1 Arithmetic operators

Operators	Usage Example	Meaning
+	A+B	Adds A to B
-	A-B	Subtracts B from A
*	A*B	Multiplies A by B
/	A/B	Divides A by B
^	A^B	Obtains the B exponent of A (exponent operation)
-	-A	Reverses the sign of A
MOD	A MOD B	Obtains the remainder A divided by B

When a "remainder" (MOD) operation involves real numbers, **the decimal value is rounded off to the nearest whole number which is then converted to an integer** before the calculation is executed. The result represents the remainder of an integer division operation.

Examples:	A=15 MOD 2	→	A=1(15/2=7....1)
	A=17.34 MOD 5.98	→	A=2(17/5=3....2)

1.2 Relational operators

Relational operators are used to compare 2 values. If the result is "true", a "-1" is obtained. If it is "false", a "0" is obtained.

Operators	Usage Example	Meaning
=	A=B	"-1" if A and B are equal, "0" if not.
<>, ><	A<>B	"-1" if A and B are unequal, "0" if not.
<	A<B	"-1" if A is smaller than B, "0" if not.
>	A>B	"-1" if A is larger than B, "0" if not.
<=, =<	A<=B	"-1" if A is equal to or smaller than B, "0" if not.
>=, =>	A>=B	"-1" if A is equal to or larger than B, "0" if not.

Examples:	A=10>5	→	Since 10 > 5 is "true", A = -1.

MEMO

- When using equivalence relational operators with real variables and real arrays, the desired result may not be obtained due to the round-off error.
  Examples: .... A=2
  B=SQR(A!)
  IF A!=B!*B! THEN...
  → In this case, A! will be unequal to B!*B!.

1.3 Logic operations

Logic operators are used to manipulate 1 or 2 values bit by bit. For example, the status of an I/O port can be manipulated.

- Depending on the logic operation performed, the results generated are either 0 or 1.
- Logic operations with real numbers convert the values into integers before they are executed.

Operators	Functions	Meaning
NOT, ~	Logical NOT	Reverses the bits.
AND, &	Logical AND	Becomes "1" when both bits are "1".
OR, |	Logical OR	Becomes "1" when either of the bits is "1".
XOR	Exclusive OR	Becomes "1" when both bits are different.

Examples: A%=NOT 13.05 → "-14" is assigned to A% (reversed after being rounded off to 13).

Bit	7	6	5	4	3	2	1	0
13	0	0	0	0	1	1	0	1
NOT 13=-14	1	1	1	1	0	0	1	0

Examples: A%=3 AND 10 → "2" is assigned to A%

Bit	7	6	5	4	3	2	1	0
3	0	0	0	0	0	0	**1**	1
10	0	0	0	0	1	0	**1**	0
3 AND 10 = 4	0	0	0	0	0	0	**1**	0

Examples: A%=3 OR 10 → "11" is assigned to A%

Bit	7	6	5	4	3	2	1	0
3	0	0	0	0	**0**	0	**1**	**1**
10	0	0	0	0	**1**	0	**1**	**0**
3 OR 10 = 11	0	0	0	0	**1**	0	**1**	**1**

Examples: A%=3 XOR 10 → "9" is assigned to A%

Bit	7	6	5	4	3	2	1	0
3	0	0	0	0	**0**	0	1	**1**
10	0	0	0	0	**1**	0	1	**0**
3 OR 10 = 11	0	0	0	0	**1**	0	0	**1**

1.4 Priority of arithmetic operation

Operations are performed in the following order of priority. When two operations of equal priority appear in the same statement, the operations are executed in order from left to right.

Priority Rank	Arithmetic Operation
1	Expressions included in parentheses
2	Functions, variables
3	^ (exponents)
4	Independent "+" and "-" signs (monominal operators)
5	* (multiplication), / (division)
6	MOD
7	+ (addition), - (subtraction)
8	Relational operators
9	NOT, ~ (Logical NOT)
10	AND, & (logical AND)
11	OR, |, XOR (Logical OR, exclusive OR)

1.5 Data format conversion

Data format is converted in cases where two values of different formats are involved in the same operation.

1. When a real number is assigned to an integer, decimal places are rounded off.

Examples: A%=125.67 → A%=126

2. When integers and real numbers are involved in the same operation, the result becomes a real number.

Examples: A(0)=125 * 0.25 → A(0)=31.25

3. When an integer is divided by an integer, the result is an integer with the remainder discarded.

Examples: A(0)=100/3 → A(0)=33

2 Character string operations

2.1 Character string connection

Character strings may be combined by using the "+" sign.

SAMPLE

```
A$="OMRON"
B$="ROBOT"
C$="LANGUAGE"
D$="MOUNTER"
E$=A$+" "+B$+" "+C$
F$=A$+" "+D$
PRINT E$
PRINT F$

Results:   OMRON ROBOT LANGUAGE
           OMRON MOUNTER
```

2.2 Character string comparison

Characters can be compared with the same relational operators as used for numeric values. Character string comparison can be used to find out the contents of character strings, or to sort character strings into alphabetical order.

- In the case of character strings, the comparison is performed from the beginning of each string, character by character.
- If all characters match in both strings, they are considered to be equal.
- Even if only one character in the string differs from its corresponding character in the other string, then the string with the larger (higher) character code is treated as the larger string.
- When the character string lengths differ, the longer of the character strings is judged to be the greater value string.

All examples below are "true".

```
Examples:   "AA"<"AB"
            "X&">"X#"
            "DESK"<"DESKS"
```

3 Point data format

NOTE

- The XYZRAB data format is used for both the joint coordinate format and the Cartesian coordinate format.
- Plus (+) signs can be omitted.
- X-arm and Y-arm rotation information is only available in software Ver.1.66M onwards.
- X-arm and Y-arm rotation information is not available on any robot model except the R6YXTW500.

There are two types of point data formats: joint coordinate format and Cartesian coordinate format.
Point numbers are in the range of 0 to 9999.

Coordinate Format	Data Format	Explanation
Joint coordinate format	± nnnnnnn	This is a decimal integer constant of 7 digits or less with a plus or minus sign, and can be specified from –6144000 to 6144000. Unit: [pulses]
Cartesian coordinate format	± nnn.nn to ± nnnnnnn	This is a decimal fraction of a total of 7 digits including 2 or less decimal places. Unit: [mm] or [degrees]

When setting an extended hand system flag for SCARA robots, set either 1 or 2 at the end of the data. If a value other than 1 or 2 is set, or if no value is designated, 0 will be set to indicate that no hand system flag is set.

Hand System	Data Value
RIGHTY (right-handed system)	1
LEFTY (left-handed system)	2

On the R6YXTW500 model robot, the X-arm and Y-arm movement range is extended beyond 360 degrees (The movable range for both the X-arm and Y-arm is -225° to +225°).
Therefore, attempts to convert Cartesian coordinate data ("mm" units) to joint coordinate data (pulse units) will result in multiple solutions, making the position impossible to determine.
In order to obtain the correct robot position and arm posture when converting to joint coordinates, X-arm and Y-arm rotation information is added after the "mm" units point data's extended hand system flag.
The Cartesian coordinate data ("mm" units) is then converted to joint coordinate data (pulse units) according to the specified X-arm and Y-arm rotation information.

To set extended X-arm and Y-arm rotation information at the R6YXTW500 model robot, a "-1", "0", or "1" value must be specified after the hand system flag. Any other value, or no value, will be processed as "0".

Arm rotation information	Data Value
"mm" → pulse converted angle data x (*1) range: -180° < x <= 180°	0
"mm" → pulse converted angle data x (*1) range: 180° < x <= 540°	1
"mm" → pulse converted angle data x (*1) range: -540° < x <= -180°	-1

*1: The joint-coordinates-converted pulse data represents each arm's distance (converted to angular data) from its mechanical origin point.

4 DI/DO conditional expressions

DI/DO conditional expressions may be used to set conditions for WAIT statements and STOPON options in MOVE statements.
Numeric constants, variables and arithmetic operators that may be used with DI/DO conditional expressions are shown below.

- Constant
  Decimal integer constant, binary integer constant, hexadecimal integer constant
- Variables
  Global integer type, global real number type, input/output type
- Operators
  Relational operators, logic operators
- Operation priority
  1. Relational operators
  2. NOT, ~
  3. AND, &
  4. OR, |, XOR

Examples: WAIT DI(31)=1 OR DI(34)=1
→ The program waits until either DI31 or DI34 turns ON.

Chapter 5

Multi-tasking

1	Outline	5-1
2	Task definition	5-1
3	Task status and transition	5-2
4	Multi-task program example	5-8
5	Sharing the data	5-8
6	Cautionary Items	5-9

1 Outline

The multi-task function performs multiple processing simultaneously in a parallel manner, and can be used to create programs of higher complexity. Before using the multi-tasking function, read this section thoroughly and make sure that you fully understand its contents.
Multi-tasking allows executing two or more tasks in parallel. However, this does not mean that multiple tasks are executed simultaneously because the controller has only one CPU to execute the tasks. In multi-tasking, the CPU time is shared among multiple tasks by assigning a priority to each task so that they can be executed efficiently.

- A maximum of 8 tasks (task 1 to task 8) can be executed in one program.
- Tasks can be prioritized and executed in their priority order (higher priority tasks are executed first).
- The priority level of task 1 is fixed at 32, while the priority of task 2 to task 8 can be set to any level between 17 and 47.
- Smaller values have higher priority, and larger values have lower priority
  (High priority: 17 to 47: low priority).

2 Task definition

A task is a set of instructions within a program which are executed as a single sequence. As explained below, a task is defined by assigning a label to it.

1. Assign a label to the first line of the command block which is to be defined as a task.
2. At the Task 1 (main task) START statement, specify the label which was assigned at step 1 above.
   Task Nos. are then assigned, and the program starts.

The task definition may call for 2 to 8 subtasks. Task 1 (main task) is automatically defined.

MEMO
- Although all tasks are written within a single program, parallel processing occurs at each of the tasks.

SAMPLE

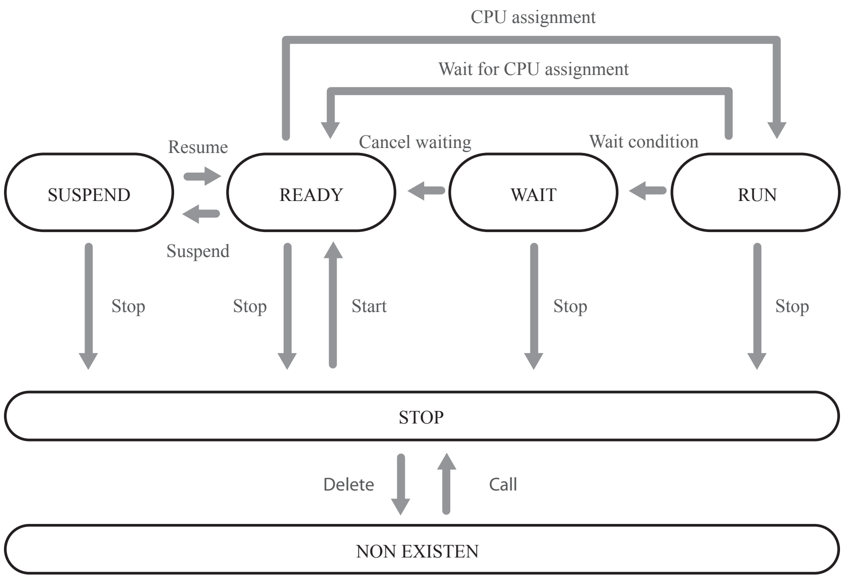
```
'MAIN TASK(TASK1)
START *IOTASK,T2 ··················*IOTASK is started as Task 2
*ST1:
MOVE P,P1,P0
    IF DI(20)= 1 THEN
        HALT
    ENDIF
GOTO *ST
HALT
'SUB TASK(TASK2)
*IOTASK: ·····························Task 2 begins from here
    IF DI(21)=1 THEN
        DO(30)=1
    ELSE
        DO(30)=0
    ENDIF
GOTO *IOTASK ························Task 2 processing ends here
EXIT TASK
```

3 Task status and transition

There are 6 types of task status:

1. **STOP status**
   A task is present but the task processing is stopped.
2. **RUN status**
   A task is present and the task processing is being executed by the CPU.
3. **READY status**
   A task is present and ready to be allocated to the CPU for task processing.
4. **WAIT status**
   A task is present and waiting for an event to begin the task processing.
5. **SUSPEND status**
   A task is present but suspended while waiting to begin the task processing.
6. **NON EXISTEN status**
   No tasks exist in the program. (The START command is used to perform a call).

Task state transition



3.1 Starting tasks

When the program is being executed in the AUTO mode, Task 1 (main task) is automatically selected and placed in a RUN status when the program begins. Therefore, the delete, forced wait, forced end commands, etc., cannot be executed for Task 1.
Other tasks (2 to 8 subtasks) will not be called simply by executing the program. The START command must be used at Task 1 in order to call, start, and place these tasks in a READY status.

MEMO

- The RESTART, SUSPEND, EXIT TASK and CUT commands cannot be executed at Task 1.

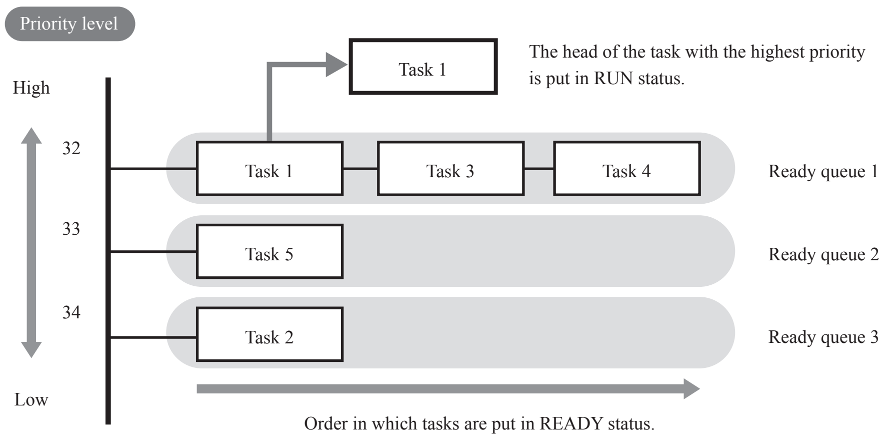
3.2 Task scheduling

Task scheduling determines the priority to be used in allocating tasks in the READY(execution enabled) status to the CPU and executing them.

When there are two or more tasks which are put in the READY status, ready queues for CPU allocation are used to determine the priority for executing the tasks. One of these READY status tasks is then selected and executed (RUN status).

Only tasks with the same priority ranking are assigned to a given ready queue. Therefore, where several tasks with differing priority rankings exist, a corresponding number of ready queues are created. Tasks within a given ready queue are handled on a first come first serve (FCFS) basis. The task where a READY status is first established has priority. The smaller the number, the higher the task priority level.

Task scheduling



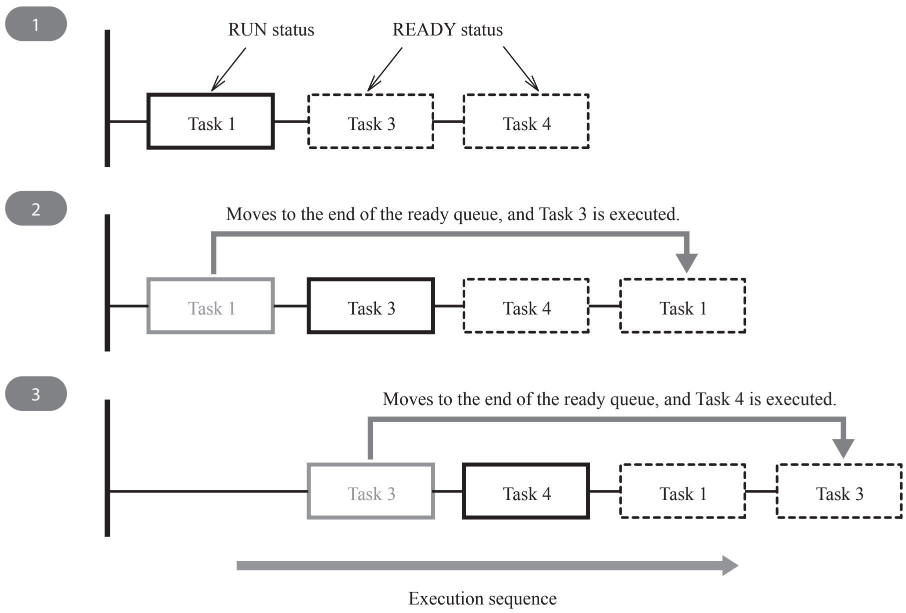
A RUN status task will be moved to the end of the ready queue if placed in a READY status by any of the following causes:

1) A WAIT status command was executed.
2) The CPU occupation time exceeds a specified time.
3) A task with a higher priority level is put in READY status.

NOTE

- When the prescribed CPU occupation time elapses, the active command is ended, and processing moves to the next task. However, if there are no other tasks of the same or higher priority (same or higher ready queue), the same task will be executed again.

Ready queue



3.3 Condition wait in task

A task is put in the WAIT status (waiting for an event) when a command causing a wait status is executed for that task. At this time, the transition to READY status does not take place until the wait condition is canceled.

1. When a command causing a wait status is executed, the following transition happens.

- Task for which a command causing a wait status is executed → WAIT status
- Task at the head of the ready queue with higher priority → RUN status

MEMO

- For example, when a MOVE statement (a command that establishes a WAIT status) is executed, the CPU sends a "MOVE" instruction to the driver, and then waits for a "MOVE COMPLETED" reply from the driver. This is a "waiting for an event" status. In this case, a WAIT status is established at the task which executed the MOVE command, and that task is moved to the end of the ready queue. A RUN status is then established at the next task.

NOTE

- If multiple tasks are in WAIT status awaiting the same condition event, or different condition events occur simultaneously, all tasks for which the waited events occur are put in READY status.

2. When an event waited by the task in the WAIT status occurs, the following status transition takes place by task scheduling.

- Task in the WAIT status for which the awaited event occurred → READY status

However, if the task put in the READY status was at the head of the ready queue with the highest priority, the following transition takes place.

1) Task that is currently in RUN status → READY status

2) Task at the head of the ready queue with higher priority → RUN status

MEMO

- In the above MOVE statement example, the task is moved to the end of the ready queue. Then, when a "MOVE COMPLETED" reply is received, this task is placed in READY status.

Tasks are put in WAIT status by the following commands.

Event		Command	
Wait for axis movement to complete	Axis movement command	MOVE DRIVE PMOVE WAIT ARM	MOVEI DRIVEI SERVO
	Parameter command	ACCEL AXWEIGHT OUTPOS ORGORD	ARCH DECEL TOLE WEIGHT
	Robot status change command	CHANGE LEFTY ASPEED	SHIFT RIGHTY SPEED
Wait for time to elapse		DELAY, SET (Time should be specified.), WAIT (Time should be specified.)	
Wait for condition to be met		WAIT	
Wait for data to send or to be received		SEND	
Wait for print buffer to become empty		PRINT	
Wait for key input		INPUT	

MEMO

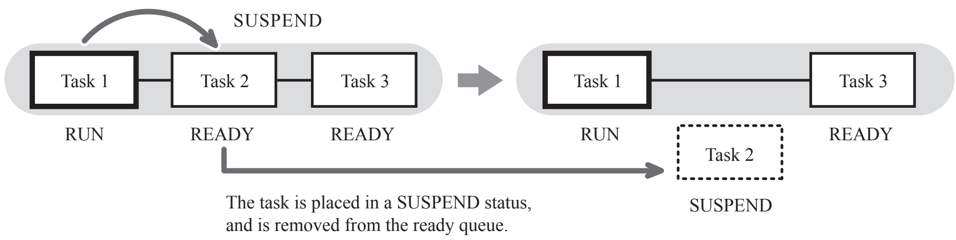
- The tasks are not put in WAIT status if the event has been established before the above commands are executed.

3.4 Suspending tasks (SUSPEND)

The SUSPEND command temporarily stops tasks other than task 1 and places them in SUSPEND status. The SUSPEND command cannot be used for task 1.
When the SUSPEND command is executed, the status transition takes place as follows.

- Task that executed the SUSPEND command → RUN status
- Specified task → SUSPEND status

Suspending tasks (SUSPEND)

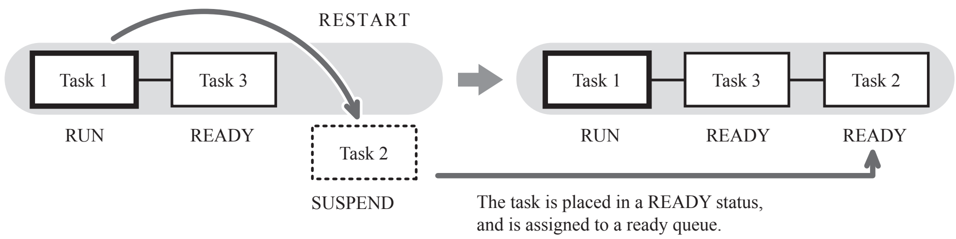


3.5 Restarting tasks (RESTART)

Tasks in the SUSPEND status can be restarted with the RESTART command. However, the RESTART command cannot be used for task 1.
When the RESTART command is executed, the status transition takes place as follows.

- Task for which the RESTART command was executed → RUN status
- Specified task → READY status

Restarting tasks (RESTART)



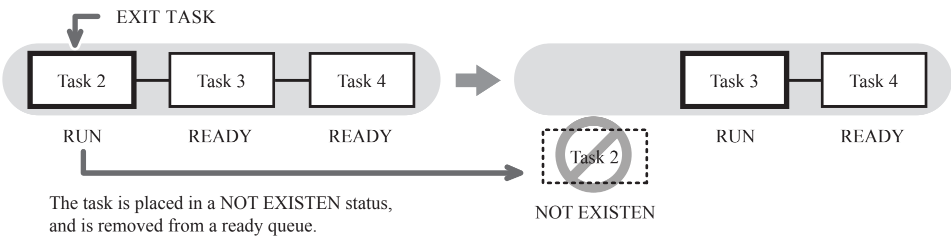
3.6 Deleting tasks

Task self-delete (EXIT TASK)

Tasks can delete themselves by using the EXIT TASK command and set to the NON EXISTEN (no task registration) status. The EXIT TASK command cannot be used for task 1.
When the EXIT TASK command is executed, the status transition takes place as follows.

- Task that executed the EXIT TASK command → NON EXISTEN status
- Task at the head of the ready queue with higher priority → RUN status

Task self-delete (EXIT TASK)

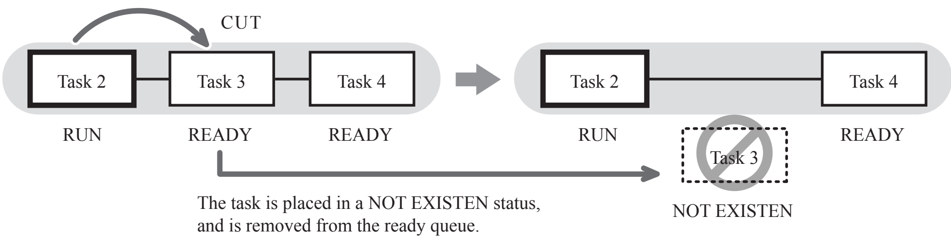


Other-task delete (CUT)

A task can also be deleted and put in the NON EXISTEN (no task registration) status by the other tasks using the CUT command. The CUT command cannot be used for task 1.
When the CUT command is executed, the status transition takes place as follows.

- Task that executed the CUT command → RUN
- Specified task → NON EXISTEN

Other-task delete (CUT)



MEMO

- If a SUSPEND command is executed for a WAIT-status task, the commands being executed by that task are ended.
- None of these commands can be executed for Task 1.

3.7 Stopping tasks

All tasks stop if any of the following cases occurs.

1. **HALT command is executed.** (stop & reset)
   The program is reset and all tasks other than task 1 are put in the NON EXISTEN status.
   Task 1 is put in the STOP status.
2. **HOLD command is executed.** (temporary stop)
   All tasks are put in the STOP status. When the program is restarted, the tasks in the STOP status set to the READY or SUSPEND status.
3. **STOP key on the programming box is pressed or the interlock signal is cut off.**
   Just as in the case where the HOLD command is executed, all tasks are put in the STOP status. When the program is restarted, the tasks in the STOP status set to the READY status (or, the task is placed in a SUSPEND status after being placed in a READY status).
4. **When the emergency stop switch on the programming box is pressed or the emergency stop signal is cut off.**
   All tasks are put in STOP status. At this point, the power to the robot is shut off and the servo sets to the non-hold state.
   After the canceling emergency stop, when the program is restarted, the tasks in STOP status are set to the READY or SUSPEND status. However, a servo ON is required in order to restart the robot power supply.

MEMO

- When the program is restarted without being reset after the tasks have been stopped by a cause other than 1., then each task is processed from the status in which the task stopped. This holds true when the power to the controller is turned off and then turned on.

1
2
3
4
5
6
7

4 Multi-task program example

Tasks are executed in their scheduled order. An example of a multi-task program is shown below.

SAMPLE

```
'TASK1
START *ST2,T2
START *ST3,T3
*ST1:
DO(20) = 1
WAIT MO(20) = 1
MOVE P,P1,P2,Z=0
IF MO(21)=1 THEN *FIN
GOTO *ST1
*FIN:
CUT T2
HALT
'TASK2
*ST2: ·····························Task 2 begins here.
IF DI(20) = 1
MO(20) = 1
DELAY 100
ELSE
MO(20) = 0
ENDIF
GOTO *ST2
EXIT TASK ····························Ends here.
'TASK3 ·····························Task 3 begins here.
*ST3:
IF DI(21) = 0 THEN *ST3
IF DI(30) = 0 THEN *ST3
IF DI(33) = 0 THEN *ST3
MO(21) = 1
EXIT TASK ····························Ends here.
```

5 Sharing the data

Point data, shift coordinate definition data, hand definition data, pallet definition data, all global variables and other variables are shared between all tasks.
Execution of each task can be controlled while using the same variables and data shared with the other tasks.

MEMO

- In this case, however, use sufficient caution when rewriting the variable and data because improper changes may cause trouble in the task processing.

6 Cautionary Items

A silence stop may occur if subtasks are continuously started (START command) and ended (EXIT TASK command) by a main task in an alternating manner.
This occurs for the following reason: if the main task and subtask priority levels are the same, a task transition to the main task occurs during subtask END processing, and an illegal task status then occurs when the main task attempts to start a subtask.
Therefore, in order to properly execute the program, the subtask priority level must be set higher than that of the main task. This prevents a task transition condition from occurring during execution of the EXIT TASK command.
In the sample program shown below, the priority level of task 1 (main task) is set as 32, and the priority level of task 2 is set as 31 (the lower the value, the higher the priority).

SAMPLE

```
FLAG1 = 0
*MAIN_TASK:
IF FLAG1=0 THEN
FLAG1 = 1
START *TASK2,T2,31 ······Task 2 (*TASK2) is started at the
priority level of 31.
ENDIF
GOTO *MAIN_TASK
'==============
' TASK2
'==============
*TASK2:
DRIVE(1,P1)
WAIT ARM(1)
DRIVE(1,P2)
WAIT ARM(1)
FLAG1 = 0
EXIT TASK
HALT
```

Chapter 6

Sequence function

1 Sequence function 6-1
2 Creating a sequence program 6-1
3 Executing a sequence program 6-4
4 Creating a sequence program 6-5

1 Sequence function

Besides normal robot programs, this YRC controller can execute high-speed processing programs (sequence programs) in response to the robot input/output (DI, DO, MO, LO TO, SI, SO) signals. This means that when a sequence program is running, it is running simultaneously with the robot program (2 programs are running).

NOTE

- The "DO12: Sequence program running" dedicated signal output occurs while a sequence program is being executed.

When the dedicated "DI10: sequence control input" is ON, the sequence program runs according to its own cycle in the AUTO or MANUAL mode, regardless of robot program starts and stops.
The sequence program starts running as soon as the controller is turned on (normally, the MANUAL mode), so it can be used to monitor the status of sensors, push button switches, solenoid valves, etc.
The sequence program can be written in the same robot language used for robot programs. This eliminates the need to learn a new language and making it easier to program.
General-purpose outputs are not reset while the sequence function is running, even if a program reset is executed. However, a setting can be specified which allows these outputs to be reset at the sequence program compiling operation. For details regarding settings required to execute a sequence program, see section "3 Executing Sequence Programs".

2 Creating a sequence program

2.1 Programming method

The following explains how to create a sequence program in order to make use of the sequence function.
First, enter "PROGRAM" mode and create a file with the file name "SEQUENCE". The controller automatically recognizes that a file with this name is a sequence program.

Naming a sequence program file

```
PROGRAM>DIR                              <TEST10  >
 No.       Name    Line    Byte   RW/RO
   1    TEST10       12     145      RW
   2    LOCATE20     25     320      RW

 Enter program name >SEQUENCE
```

Next, input a program. This is no different from the standard robot program creation method. Commands which can be input are explained later in this manual.

Creating a sequence program

```
PROGRAM>EDIT                             <SEQUENCE>
   1   DO(20)=DI(21) AND DI(22)
   2   MO(30)=DO(23) OR DI(22)
   3   MO(31)=~MO(30)
   4   DO(21)=(DI(36) OR DI(25)) AND DI(2
   5   DO(30)=MO(30) OR DI(27)

 SELECT   COPY   CUT   PASTE   BS
```

2.2 Compiling

After editing the program, it must be compiled as a sequence program. Compiling is performed in the same way as for robot programs. Press the F5 key on the highest-level screen in "PROGRAM" mode.

Sequence program

```
PROGRAM>EDIT                                   <SEQUENCE>
-----+----+----1----+----2----+----3----+
    1   DO(20)=DI(21)  AND DI(22)
    2   MO(30)=DO(23)  OR DI(22)
    3   MO(31)=~MO(30)
    4   DO(21)=(DI(36)  OR DI(25))AND DI(2
    5   DO(30)=MO(30)  OR DI(27)
 SELECT |        | DIR    |        | COMPILE
```

A check message appears asking if you want to compile the sequence program. Press the F4 key to compile the program. To cancel this compiling, press the F5 key. The display changes to the compiling screen for normal robot programs.

Press the F4 key to compile the sequence program.

Compile the sequence program before compiling the main program.

Compiling the sequence program

```
PROGRAM>COMPILE                                <SEQUENCE>
-----+----+----1----+----2----+----3----+
    1   DO(20)=DI(21)  AND DI(22)
    2   MO(30)=DO(23)  OR DI(22)
    3   MO(31)=~MO(30)
    4   DO(21)=(DI(36)  OR DI(25))AND DI(2
    5   DO(30)=MO(30)  OR DI(27)
 Compile for SEQUENCE OK?YES     | NO
```

If there is a syntax error in the program, an error message appears and the program will be listed from the line with the error When the compiling ends without any error, the program will be listed from its first line.

Compiling error

```
PROGRAM                                        <SEQUENCE>
-----5.1:Syntax error-----2----+----3----+
    3   MO(31)=~MO(30)'AB
    4   DO(21)=MO(36)  OR DI(27)
    5   DO(30)=MO(30)  OR DI(27)
    6   DO(25)=DI(26)  AND DO(32)
    7   DO(30)=MO(30)  OR DI(27)
 SELECT |        | DIR    |        | COMPILE
```

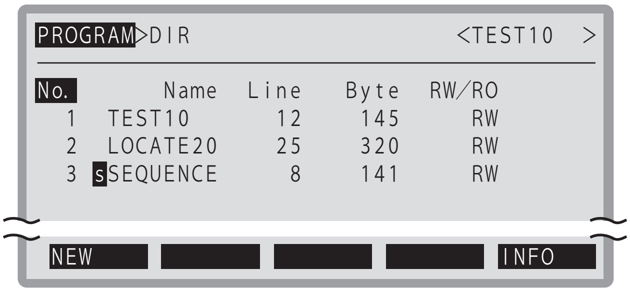
MEMO

- The sequence execution program is erased and the letter "s" disappears in the following cases. In these cases the sequence function cannot be used in "UTILITY" mode.

1. When the sequence program was erased
2. When the sequence program was edited
3. When normal robot program compiling was performed for the sequence program
   (The same processing occurs even if the mode is changed to AUTO while in the SEQUENCE program is selected.)
4. Program data was initialized.
5. A "9.39: Sequence object destroyed.

When you display the directory after the compiling the sequence program, a letter "s" appears to the left of the program name "SEQUENCE". This means that the sequence program has been compiled successfully and is ready for use.

Sequence execution program after compiling



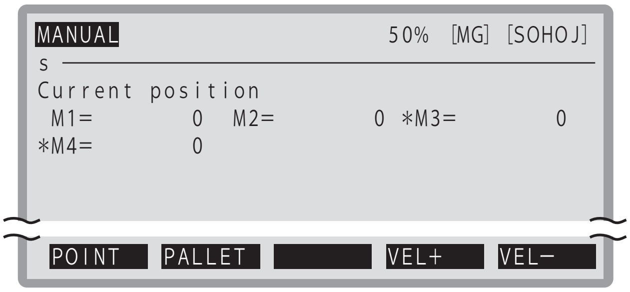
3 Executing a sequence program

The following conditions must be satisfied to execute a sequence program. If any of these conditions is not met, the sequence program cannot be executed.

1. The sequence execution program has been created by compiling.
2. The sequence function is enabled in "UTILITY" mode.
   (For details regarding the UTILITY mode, refer to the controller manual.)
3. The external sequence control input (DI10) contact is closed.
4. The current operation mode is "MANUAL" or "AUTO".

When all of the above conditions are met, the sequence program can now be executed. While the program is running, the letter "s" will appear at the left end of the second line of the screen.

Sequence program execution in progress

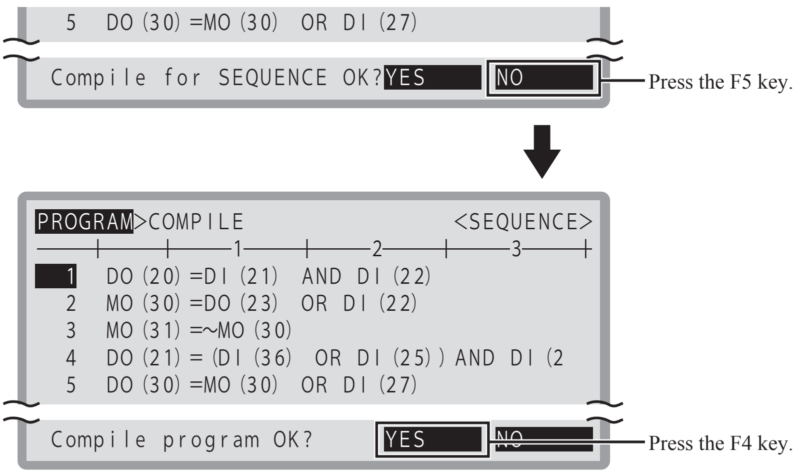


For details regarding the UTILITY mode setting procedure, refer to the controller manual.

3.1 Sequence program STEP execution

The sequence program may be executed line by line while checking one command line at a time.
To do this, press the F5 key on the compile screen. Sequence program compiling is canceled and the normal robot compile screen then appears.
Press the F4 key to compile and create a normal execution program. Then, execute this program with the STEP statement in "AUTO" mode to check the operation.

Sequence program STEP execution



4 Creating a sequence program

When creating a sequence program, you may use only assignment statements comprised of input/output variables and logical operators. An error will occur during compiling if any statement other than assignment statements is used in the program, and the compiling cannot be completed.

4.1 Assignment statements

Format

<output variable <internal auxiliary output variable> <arm lock output variable> <timer output variable>	=<expression>

Values <expression> .... Any one of the following can be used.

- Parallel input/output variables
- Internal auxiliary output variables
- Arm lock output variables
- Timer output variables
- Serial input/output variable
- The logic operation expression shown above

4.2 Input/output variables

Each variable must be specified in a 1-bit format

• Correct examples	DO(35)
	MO(24)
	DI(16)
• Incorrect examples	DO(37, 24)
	DI3(4)
	MO3()

- Parallel input variables

Format

DI(mb)	m: Port number ······················	0 to 7, 10 to 17, 20 to 27
	b: bit definition ······················	0 to 7

These variables show the status of the parallel input signal.

- Parallel output variables

Format

DO(mb)	m: Port number ······················	0 to 7, 10 to 17, 20 to 27
	b: bit definition ······················	0 to 7

A parallel output is specified, or the output status is referenced. Ports 0 and 1 are for referencing only, and no outputs can occur there.

● Internal output variables

Format		
MO(mb)	m: Port number ······················	0 to 7, 10 to 17, 20 to 27
	b: bit definition ······················	0 to 7

These variables are used within the controller and are not output externally. Ports 0 and 1 are for referencing only, and no outputs can occur there.

● Arm lock output variables

Format		
LO(mb)	m: port number ······················	0
	b: bit definition ······················	0 to 7

These variables are used to prohibit the arm movement. Movement is prohibited when ON. LO(00) to LO(07) corresponds to arm 1 to arm 8.

● Timer output variables

Format		
TO(mb)	m: port number ······················	0
	b: bit definition ······················	0 to 7

There are a total of 8 timer output variables: TO(00) to TO(07). The timer of each variable is defined by the timer definition statement TIM00 to 07.

● Serial input variables

Format		
SI(mb)	m: Port number ······················	0 to 7, 10 to 17, 20 to 27
	b: bit definition ······················	0 to 7

Indicates a serial input signal status. Only referencing can occur. No settings are possible.

● Serial output variables

Format		
SO(mb)	m: Port number ······················	0 to 7, 10 to 17, 20 to 27
	b: bit definition ······················	0 to 7

Sets or references a serial output signal status. Ports 0 and 1 are for referencing only, and no outputs can occur there.

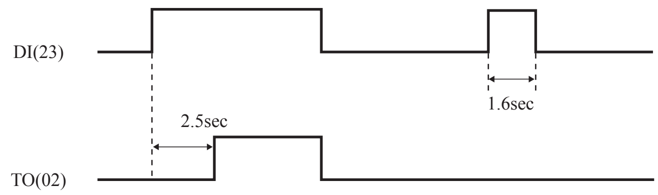
Timer example

SAMPLE

TIM02 = 2500 ··························· Timer 02 is set to 2.5 seconds.
TO(02) = DI(23) ························· Timer starts when DI(23) switches ON.

- When DI(23) is ON, after 2.5 seconds, TO(02) is set ON.
- When DI(23) is OFF, TO(02) is also OFF.
- When DI(23) isn't ON after 2.5 second or more, TO(02) does not change to ON.

Timer usage example: Timing chart



4.3 Timer definition statement

Format

TIMmb=<time> m: Port number ·················· 0
b: bit definition ·················· 0 to 7

Values <time> .... 100 to 999,900msec (0.1 to 999.9 second)

Meaning The timer definition statement sets the timer value of the timer output variable. This definition statement may be anywhere in the program.
When the timer definition statement is omitted, the timer setting value of the variable is 0.
TIM00 to 07 correspond to the timer output variables TO(00) to (07).
However, since the units are set every 100msec, values less than 99msec are truncated.

4.4 Logical operators

Operators	Functions	Meaning
NOT, ~	Logical NOT	Reverses the bits.
AND, &	Logical AND	Becomes "1" when both bits are "1".
OR, |	Logical OR	Becomes "1" when either of the bits is "1".

4.5 Priority of logic operations

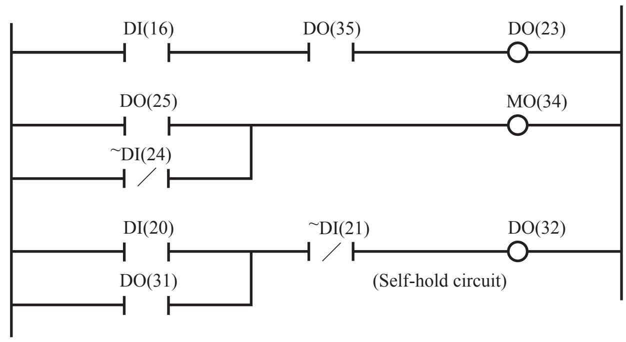
Priority Ranking	Operation Content
1	Expressions in parentheses
2	NOT, ~ (Logical NOT)
3	AND, & (Logical AND)
4	OR, | (Logical OR)

● Example with a ladder statement substitution

SAMPLE

```
DO(23)=DI(16)&DO(35)
MO(34)=DO(25) | ~DI(24)
DO(31)=(DI(20) | DO(31))&~DI(21)
```

Ladder diagram



MEMO

- NOT cannot be used prior to the first parenthesis " (" or on the left of an expression. For example, the following commands cannot be used.
  •DO(21)=~(DI(30) | DI(32))
  •~DO(30)=DI(22)&DI(27)
- Numeric values cannot be assigned on the right of an expression.
  •MO(35)=1
  •DO(26)=0
- There is no need to define a "HALT" or "HOLD" statement at the end of the program.
- The I/O and internal auxiliary output variables used in sequence programs are shared with robot programs, so be careful not to make improper changes when using the same variables between them.

4.6 Sequence program specifications

Item	Specification
Commands	Logical NOT, AND, OR
I/O	Same as robot language
Program capacity	4096 bytes (A maximum of 512 variables can be specified.)
Scan time	10 to 30ms depending on the number of steps (This changes automatically.)

Chapter 7

Robot Language Lists

How to read the robot language table.........................7-1

Command list in alphabetic order 7-3

Function Specific...7-7

Functions: in alphabetic order...................................7-13

Functions: operation-specific 7-15

1 ABS..7-17

2 ABSINIT...7-18

3 ABSRPOS...7-20

4 ABSRST...7-21

5 ACCEL...7-22

6 ARCH 7-23

7 ARMCND...7-25

8 ARMTYPE 7-26

9 ATN..7-27

10 ASPEED 7-28

11 AXWGHT...7-29

12 CALL 7-30

13 CHANGE...7-31

14 CHGPRI..7-32

15 CHR$ 7-33
16 COS 7-34
17 CURTRQ 7-34
18 CUT 7-35
19 DATE$ 7-36
20 DECEL 7-37
21 DECLARE 7-38
22 DEF FN 7-40
23 DEGRAD 7-41
24 DELAY 7-42
25 DI 7-43
26 DIST 7-44
27 DIM 7-45
28 DO 7-46
29 DRIVE 7-47
30 DRIVEI 7-55
31 END SELECT 7-60
32 END SUB 7-61
33 ERR / ERL 7-62
34 EXIT FOR 7-63
35 EXIT SUB 7-64
36 EXIT TASK 7-65
37 FOR to NEXT 7-66
38 GOSUB to RETURN 7-67
39 GOTO 7-68
40 HALT 7-69
41 HAND 7-70
42 HOLD 7-73
43 IF 7-74
44 INPUT 7-76
45 INT 7-77
46 JTOXY 7-78
47 LABEL Statement 7-79
48 LEFT$ 7-80
49 LEFTY 7-81
50 LEN 7-82

51	LET	7-83
52	LO	7-86
53	LOCx	7-87
54	LSHIFT	7-89
55	MCHREF	7-90
56	MID$	7-91
57	MO	7-92
58	MOVE	7-93
59	MOVEI	7-109
60	OFFLINE	7-114
61	ORD	7-115
62	ON ERROR GOTO	7-116
63	ON to GOSUB	7-117
64	ON to GOTO	7-118
65	ONLINE	7-119
66	ORGORD	7-120
67	ORIGIN	7-121
68	OUT	7-122
69	OUTPOS	7-123
70	PATH	7-125
71	PATH END	7-131
72	PATH SET	7-132
73	PATH START	7-134
74	PDEF	7-135
75	PMOVE	7-136
76	Pn	7-140
77	PPNT	7-142
78	PRINT	7-143
79	RADDEG	7-144
80	REM	7-145
81	RESET	7-146
82	RESTART	7-147
83	RESUME	7-148
84	RETURN	7-149
85	RIGHT$	7-150
86	RIGHTY	7-151

87 RSHIFT....7-152
88 Sn....7-153
89 SELECT CASE....7-154
90 SEND....7-155
91 SERVO....7-157
92 SET....7-158
93 SHARED....7-159
94 SHIFT....7-160
95 SIN....7-161
96 SO....7-162
97 SPEED....7-163
98 START....7-164
99 STR$....7-165
100 SQR....7-166
101 SUB to END SUB....7-167
102 SUSPEND....7-169
103 SWI....7-170
104 TAN....7-171
105 TCOUNTER....7-172
106 TIME$....7-173
107 TIMER....7-174
108 TO....7-175
109 TOLE....7-176
110 TORQUE....7-177
111 TRQSTS....7-179
112 TRQTIME....7-180
113 VAL....7-182
114 WAIT....7-183
115 WAIT ARM....7-184
116 WEIGHT....7-185
117 WEND....7-186
118 WHERE....7-187
119 WHILE to WEND....7-188
120 WHRXY....7-189
121 XYTOJ....7-190
122 _SYSFLG....7-190

How to read the robot language table

The key to reading the following robot language table is explained below.

	No. (1)	Function (2)	Conditions (3)	Direct (4)	Type (5)
DIM	27	Declares the array variable name and the number of elements.	6	×	Command

(1) No.

Indicates the Item No. where this robot language is explained in detail.

Example of "No." column

No.

27 DIM
Declares array variable

Format
```
DIM <array definition> [, <array definition>,…]
```

Format
```
<name> [| % |] (<constant> [, <constant> [, <constant>]])
        | ! |
        | $ |
```

Values <constant> ....Array subscript: 0 to 32,767 (positive integer)

Explanation Directly declares the name and length (number of elements) of an array variable. A maximum of 3 dimensions may be used for the array subscripts. Multiple arrays can be declared in a single line by using comma (,) breakpoints to separate the arrays.

MEMO
- Array subscripts can be "0 to a specified value", with their total number being the <constant> + 1.
- A "9.31: Memory full" error may occur depending on the size of each dimension defined in an array.

SAMPLE
```
DIM A%(10) ························Defines a integer array
                                   variable A% (0) to A% (10).
                                   (Number of elements: 11).
DIM B(2,3,4) ······················Defines a real array variable
                                   B (0, 0, 0) to B (2, 3, 4).
                                   (Number of elements: 60).
DIM C%(2,2),D!(10) ················Defines an integer array C%
                                   (0,0) to C% (2,2) and a real
                                   array D! (0) to D! (10).
```

(2) Function

Explains the function of the robot language.

(3) Condition

Lists the conditions under which command execution is enabled.

Condition 1:	Commands that can be executed by both direct commands and online commands.
Condition 2:	In addition to Condition 1, commands that execute task 1 (main task) only.
Condition 3:	In addition to condition 1, commands containing operands that cannot be executed by direct commands or online commands.
Condition 4:	In addition to condition 1, commands which are executed after positioning is completed.
Condition 5:	MOVE L and MOVE C can be executed by both direct commands and online commands, although they are executed after positioning is completed. The STOPON option cannot be executed by direct commands and online commands.
Condition 6:	Commands that cannot be executed by direct commands and online commands.

Regarding robot languages which can be used as both commands and functions, the "execution enabled" conditions for a "command execution" may differ from those for a "function execution". In such cases, the respective conditions for the command and function are divided by a slash mark (/). For example, if condition 4 is applies for a "Command", but there are no conditions for the "Function", this would be expressed as follows: 4/-

(4) Direct

If "○" is indicated at this item, both direct commands and online commands can be used.

MEMO

- Direct commands are input directly from the programming box while in the AUTO mode, and are used to perform temporary operations. For details, refer to the controller manual.

(5) Type

Indicates the robot language type as "Command" or "Function".

When a command is used as both a "Command" and "Function", this is expressed as follows: Command/Function

Command list in alphabetic order

No.	Command	Function	Condition	Direct	Type
A					
1	ABS	Acquires the absolute value of a specified value.	-	-	Functions
2	ABSINIT	Resets the current position of a specified main group axis.	4	○	Command Statements
3	ABSRPOS	Acquires the machine reference of the specified main group axis. (Valid only for axes where the return-to-origin method is set as "mark method".)	-	-	Functions
4	ABSRST	Executes a return-to-origin at the robot absolute motor axes.	4	○	Command Statements
5	ACCEL	Specifies/acquires the acceleration coefficient parameter of the main group.	4/-	○	Command Statements/ Functions
6	ARCH	Specifies/acquires the arch position parameter of the main group.	4/-	○	Command Statements/ Functions
7	ARMCND	Acquires the current arm status of the main robot.	-	-	Functions
8	ARMTYPE	Acquires the current "hand system" setting of the main robot.	-	-	Functions
10	ASPEED	Changes the AUTO movement speed of the main group.	4	○	Command Statements
9	ATN	Acquires the arctangent of the specified value.	-	-	Functions
11	AXWGHT	Specifies/acquires the axis tip weight parameter of the main group.	4/-	○	Command Statements/ Functions
C					
12	CALL	Executes (calls) another program.	6	×	Command Statements
13	CHANGE	Switches the main robot hand.	4	○	Command Statements
14	CHGPRI	Changes the priority ranking of a specified task.	6	×	Command Statements
15	CHR$	Acquires a character with the specified character code.	-	-	Functions
16	COS	Acquires the cosine value of a specified value.	-	-	Functions
17	CURTRQ	Acquires the current torque value of the specified main group axis.	-	×	Functions
18	CUT	Terminates a task currently being executed or temporarily stopped.	6	×	Command Statements
D					
19	DATE$	Acquires the date as a "yy/mm/dd" format character string.	-	-	Functions
20	DECEL	Specifies/acquires the deceleration rate parameter of the main group.	4/-	○	Command Statements/ Functions
23	DEGRAD	Converts a specified value to radians (↔RADDEG).	-	-	Functions
24	DELAY	Waits for the specified period (units: ms).	6	×	Command Statements
27	DIM	Declares the array variable name and the number of elements.	6	×	Command Statements
26	DIST	Acquires the distance between 2 specified points.	-	-	Functions
28	DO	Outputs a specified value to the DO port.	1	○	Command Statements
29	DRIVE	Moves a specified main group axis to an absolute position.	4	○	Command Statements
29	DRIVE	(With T-option) Executes an absolute movement command for a specified axis.	4	○	Command Statements
30	DRIVEI	Moves a specified main group axis to a relative position.	4	○	Command Statements
E					
33	ERL	Gives the line No. where an error occurred.	-	-	Functions
33	ERR	Gives the error code number of an error which has occurred.	-	-	Functions
34	EXIT FOR	Terminates the FOR to NEXT statement loop.	6	×	Command Statements
36	EXIT TASK	Terminates its own task which is in progress.	6	×	Command Statements

No.	Command	Function	Condition	Direct	Type
F					
37	FOR to NEXT	Controls repetitive operations. Executes the FOR to NEXT statement repeatedly until a specified value is reached.	6	×	Command Statements
G					
38	GOSUB to RETURN	Jumps to a subroutine with the label specified by a GOSUB statement, and executes that subroutine.	6	×	Command Statements
39	GOTO	Unconditionally jumps to the line specified by a label.	6	×	Command Statements
H					
40	HALT	Stops the program and performs a reset.	6	×	Command Statements
41	HAND	Defines the main robot hand.	4	○	Command Statements
42	HOLD	Temporarily stops the program.	6	×	Command Statements
I					
43	IF	Allows control flow to branch according to conditions.	6	×	Command Statements
44	INPUT	Assigns a value to a variable specified from the programming box.	1	○	Command Statements
45	INT	Acquires an integer for a specified value by truncating all decimal fractions.	-	-	Functions
J					
46	JTOXY	Converts joint coordinate data to main group Cartesian coordinate data. (↔XYTOJ)	-	-	Functions
L					
48	LEFT$	Extracts a character string comprising a specified number of digits from the left end of a specified character string.	-	-	Functions
49	LEFTY	Sets the main robot hand system to "Left".	4	○	Command Statements
50	LEN	Acquires the length (number of bytes) of a specified character string.	-	-	Functions
51	LET	Executes a specified assignment statement.	1	○	Command Statements
52	LO	Outputs a specified value to the LO port to enable/disable axis movement.	1	○	Command Statements
53	LOCx	Specifies/acquires point data or shift data for a specified axis.	-	-	Command Statements/ Functions
54	LSHIFT	Shifts a value to the left by the specified number of bits. (↔RSHIFT)	-	-	Functions
M					
55	MCHREF	Acquires the return-to-origin or absolute-search machine reference for a specified main group axis.	-	-	Functions
56	MID$	Extracts a character string of a desired length from a specified character string.	-	-	Functions
57	MO	Outputs a specified value to the MO port.	1	○	Command Statements
58	MOVE	Performs absolute movement of all main robot axes.	5	○	Command Statements
59	MOVEI	Performs relative movement of all main robot axes.	4	○	Command Statements
O					
60	OFFLINE	Sets a specified communication port to the "offline" mode.	1	○	Command Statements
62	ON ERROR GOTO	If an error occurs during program execution, this command allows the program to jump to the error processing routine specified by the label without stopping the program, or it stops the program and displays the error message.	6	×	Command Statements
63	ON to GOSUB	Jumps to a subroutine with labels specified by a GOSUB statement in accordance with the conditions, and executes that subroutine.	6	×	Command Statements

No.	Command	Function	Condition	Direct	Type
64	ON to GOTO	Jumps to label-specified lines in accordance with the conditions.	6	×	Command Statements
65	ONLINE	Sets the specified communication port to the "online" mode.	1	○	Command Statements
61	ORD	Acquires the character code of the first character in a specified character string.	-	-	Functions
66	ORGORD	Specifies/acquires the axis sequence parameter for performing return-to-origin and absolute search operations in the main group.	4/-	○	Command Statements/ Functions
67	ORIGIN	Executes a return-to-origin for incremental specs. axes.	4	○	Command Statements
68	OUT	Turns ON the bits of the specified output ports and the command statement ends.	6	×	Command Statements
69	OUTPOS	Specifies/acquires the OUT enable position parameter of the main group.	4/-	○	Command Statements/ Functions
P					
70	PATH	Sets the movement path.	6	×	Command Statements
71	PATH END	Ends the movement path setting.	6	×	Command Statements
72	PATH SET	Starts the movement path setting.	6	×	Command Statements
73	PATH START	Starts the PATH motion.	6	×	Command Statements
74	PDEF	Defines the pallet used to execute pallet movement commands.	1	○	Command Statements
75	PMOVE	Executes the main robot pallet movement command.	4	○	Command Statements
76	Pn	Defines points within a program.	1	○	Command Statements
77	PPNT	Creates point data specified by a pallet definition number and pallet position number.	-	-	Functions
78	PRINT	Displays a character string at the programming box screen.	1	○	Command Statements
R					
79	RADDEG	Converts a specified value to degrees. (↔DEGRAD)	-	-	Functions
80	REM	Expresses a comment statement.	6	×	Command Statements
81	RESET	Turns the bit of a specified output port OFF.	1	○	Command Statements
82	RESTART	Restarts another task during a temporary stop.	6	×	Command Statements
83	RESUME	Resumes program execution after error recovery processing.	6	×	Command Statements
85	RIGHT$	Extracts a character string comprising a specified number of digits from the right end of a specified character string.	-	-	Functions
86	RIGHTY	Sets the main robot hand system to "Right".	4	○	Command Statements
87	RSHIFT	Shifts a value to the right by the specified number of bits. (↔LSHIFT)	-	-	Functions
S					
88	Sn	Defines the shift coordinates within the program.	4	○	Command Statements
89	SELECT CASE to END SELECT	Allows control flow to branch according to conditions.	6	×	Command Statements
90	SEND	Sends a file.	1	○	Command Statements
91	SERVO	Controls the servo ON/OFF of specified main group axes or all main group axes.	4	○	Command Statements
92	SET	Turns the bit at the specified output port ON.	3	In part ×	Command Statements
94	SHIFT	Sets the shift coordinates for the main robot by using the shift data specified by a shift variable.	4	○	Command Statements
95	SIN	Acquires the sine value for a specified value.	-	-	Functions
96	SO	Outputs a specified value to the SO port.	1	○	Command Statements
97	SPEED	Changes the main group's program movement speed.	4	○	Command Statements

No.	Command	Function	Condition	Direct	Type
98	START	Specifies the task number and priority ranking of a specified task, and starts that task.	6	×	Command Statements
99	STR$	Converts a specified value to a character string (↔VAL)	-	-	Functions
100	SQR	Acquires the square root of a specified value.	-	-	Functions
102	SUSPEND	Temporarily stops another task which is being executed.	6	×	Command Statements
103	SWI	Switches the program being executed, performs compiling, then begins execution from the first line.	2	○	Command Statements
T					
104	TAN	Acquires the tangent value for a specified value.	-	-	Functions
105	TCOUNTER	Outputs count-up values at 10ms intervals starting from the point when the TCOUNTER variable is reset.	-	-	Functions
106	TIME$	Acquires the current time as an "hh:mm:ss" format character string.	-	-	Functions
107	TIMER	Acquires the current time in seconds, counting from 12:00 midnight.	-	-	Functions
108	TO	Outputs a specified value to the TO port.	1	○	Command Statements
109	TOLE	Specifies/acquires the main group tolerance parameter.	4/-	○	Command Statements/ Functions
110	TORQUE	Specifies/acquires the maximum torque command value which can be set for a specified main group axis.	4/-	○	Command Statements/ Functions
111	TRQSTS	Acquires the command end status for the DRIVE command with torque limit option executed at the main group.	-	-	Functions
112	TRQTIME	Specifies/acquires the current limit time-out period at the specified main group axis when using a torque limit option in the DRIVE statement.	1/-	○	Command Statements/ Functions
V					
113	VAL	Converts the numeric value of a specified character string to an actual numeric value. (↔STR$)	-	-	Functions
W					
114	WAIT	Waits until the conditions of the DI/DO conditional expression are met (with time-out).	6	×	Command Statements
115	WAIT ARM	Waits until the main group robot axis operation is completed.	6	×	Command Statements
116	WEIGHT	Specifies/acquires the main robot tip weight parameter.	4/-	○	Command Statements/ Functions
118	WHERE	Reads out the current position of the main group robot arm in joint coordinates (pulses).	-	-	Functions
119	WHILE to WEND	Controls repeated operations.	6	×	Command Statements
120	WHRXY	Reads out the current position of the main group arm as Cartesian coordinates (mm, degrees).	-	-	Functions
X					
121	XYTOJ	Converts the point variable Cartesian coordinate data to the main group's joint coordinate data (↔JTOXY).	-	-	Functions
122	_SYSFLG	Axis status monitoring flag.	-	-	Functions

Function Specific

Program commands

General commands

No.	Command	Function	Condition	Direct	Type
27	DIM	Declares the array variable name and the number of elements.	6	×	Command Statements
51	LET	Executes a specified assignment statement.	1	○	Command Statements
80	REM	Expresses a comment statement.	6	×	Command Statements

Arithmetic commands

No.	Command	Function	Condition	Direct	Type
1	ABS	Acquires the absolute value of a specified value.	-	-	Functions
2	ABSINIT	Resets the current position of a specified main group axis.	4	○	Command Statements
9	ATN	Acquires the arctangent of the specified value.	-	-	Functions
16	COS	Acquires the cosine value of a specified value.	-	-	Functions
23	DEGRAD	Converts a specified value to radians (↔RADDEG).	-	-	Functions
26	DIST	Acquires the distance between 2 specified points.	-	-	Functions
45	INT	Acquires an integer for a specified value by truncating all decimal fractions.	-	-	Functions
54	LSHIFT	Shifts a value to the left by the specified number of bits. (↔RSHIFT)	-	-	Functions
79	RADDEG	Converts a specified value to degrees. (↔DEGRAD)	-	-	Functions
87	RSHIFT	Shifts a value to the right by the specified number of bits. (↔LSHIFT)	-	-	Functions
95	SIN	Acquires the sine value for a specified value.	-	-	Functions
100	SQR	Acquires the square root of a specified value.	-	-	Functions
104	TAN	Acquires the tangent value for a specified value.	-	-	Functions

Date / time

No.	Command	Function	Condition	Direct	Type
19	DATE $	Acquires the date as a "yy/mm/dd" format character string.	-	-	Functions
105	TCOUNTER	Outputs count-up values at 10ms intervals starting from the point when the TCOUNTER variable is reset.	-	-	Functions
106	TIME $	Acquires the current time as an "hh:mm:ss" format character string.	-	-	Functions
107	TIMER	Acquires the current time in seconds, counting from 12:00 midnight.	-	-	Functions

Character string operation

No.	Command	Function	Condition	Direct	Type
15	CHR $	Acquires a character with the specified character code.	-	-	Functions
48	LEFT $	Extracts a character string comprising a specified number of digits from the left end of a specified character string.	-	-	Functions
50	LEN	Acquires the length (number of bytes) of a specified character string.	-	-	Functions
56	MID $	Extracts a character string of a desired length from a specified character string.	-	-	Functions
61	ORD	Acquires the character code of the first character in a specified character string.	-	-	Functions
85	RIGHT $	Extracts a character string comprising a specified number of digits from the right end of a specified character string.	-	-	Functions
99	STR $	Converts a specified value to a character string (↔VAL)	-	-	Functions
113	VAL	Converts the numeric value of a specified character string to an actual numeric value. (↔STR$)	-	-	Functions

Point, coordinates, shift coordinates

No.	Command	Function	Condition	Direct	Type
13	CHANGE	Switches the main robot hand.	4	○	Command Statements
41	HAND	Defines the main robot hand.	4	○	Command Statements
46	JTOXY	Converts joint coordinate data to main group Cartesian coordinate data. (↔XYTOJ)	-	-	Functions
49	LEFTY	Sets the main robot hand system to "Left".	4	○	Command Statements
76	Pn	Defines points within a program.	1	○	Command Statements
77	PPNT	Creates point data specified by a pallet definition number and pallet position number.	-	-	Functions
86	RIGHTY	Sets the main robot hand system to "Right".	4	○	Command Statements
88	Sn	Defines the shift coordinates in the program.	4	○	Command Statements
94	SHIFT	Sets the shift coordinates for the main robot by using the shift data specified by a shift variable.	4	○	Command Statements
121	XYTOJ	Converts the point variable Cartesian coordinate data to the main group's joint coordinate data (↔JTOXY).	-	-	Functions
53	LOCx	Specifies/acquires point data or shift data for a specified axis.	-	-	Command Statements/ Functions

Branching commands

No.	Command	Function	Condition	Direct	Type
34	EXIT FOR	Terminates the FOR to NEXT statement loop.	6	×	Command Statements
37	FOR to NEXT	Controls repetitive operations. Executes the FOR to NEXT statement repeatedly until a specified value is reached.	6	×	Command Statements
38	GOSUB to RETURN	Jumps to a subroutine with the label specified by a GOSUB statement, and executes that subroutine.	6	×	Command Statements
39	GOTO	Unconditionally jumps to the line specified by a label.	6	×	Command Statements
43	IF	Allows control flow to branch according to conditions.	6	×	Command Statements
63	ON to GOSUB	Jumps to a subroutine with labels specified by a GOSUB statement in accordance with the conditions, and executes that subroutine.	6	×	Command Statements
64	ON to GOTO	Jumps to label-specified lines in accordance with the conditions.	6	×	Command Statements
89	SELECT CASE to END SELECT	Allows control flow to branch according to conditions.	6	×	Command Statements
119	WHILE to WEND	Controls repeated operations.	6	×	Command Statements

Error control

No.	Command	Function	Condition	Direct	Type
62	ON ERROR GOTO	If an error occurs during program execution, this command allows the program to jump to the error processing routine specified by the label without stopping the program, or it stops the program and displays the error message.	6	×	Command Statements
83	RESUME	Resumes program execution after error recovery processing.	6	×	Command Statements
33	ERL	Gives the line No. where an error occurred.	-	-	Functions
33	ERR	Gives the error code number of an error which has occurred.	-	-	Functions

Program & task control

Program control

No.	Command	Function	Condition	Direct	Type
12	CALL	Executes (calls) another program.	6	×	Command Statements
40	HALT	Stops the program and performs a reset.	6	×	Command Statements
42	HOLD	Temporarily stops the program.	6	×	Command Statements
103	SWI	Switches the program being executed, performs compiling, then begins execution from the first line.	2	○	Command Statements

Task control

No.	Command	Function	Condition	Direct	Type
14	CHGPRI	Changes the priority ranking of a specified task.	6	×	Command Statements
18	CUT	Terminates a task currently being executed or temporarily stopped.	6	×	Command Statements
36	EXIT TASK	Terminates its own task which is in progress.	6	×	Command Statements
82	RESTART	Restarts another task during a temporary stop.	6	×	Command Statements

No.	Command	Function	Condition	Direct	Type
98	START	Specifies the task number and priority ranking of a specified task, and starts that task.	6	×	Command Statements
102	SUSPEND	Temporarily stops another task which is being executed.	6	×	Command Statements

Robot control

Robot operations

No.	Command	Function	Condition	Direct	Type
4	ABSRST	Executes a return-to-origin at the robot absolute motor axes.	4	○	Command Statements
13	CHANGE	Switches the main robot hand.	4	○	Command Statements
29	DRIVE	Moves a specified main group axis to an absolute position.	4	○	Command Statements
30	DRIVEI	Moves a specified main group axis to a relative position.	4	○	Command Statements
41	HAND	Defines the main robot hand.	4	○	Command Statements
49	LEFTY	Sets the main robot hand system to "Left".	4	○	Command Statements
58	MOVE	Performs absolute movement of all main robot axes.	5	○	Command Statements
59	MOVEI	Performs relative movement of all main robot axes.	4	○	Command Statements
67	ORIGIN	Executes a return-to-origin for incremental specs. axes.	4	○	Command Statements
75	PMOVE	Executes the main robot pallet movement command.	4	○	Command Statements
86	RIGHTY	Sets the main robot hand system to "Right".	4	○	Command Statements
91	SERVO	Controls the servo ON/OFF of specified main group axes or all main group axes.	4	○	Command Statements

Status acquisition

No.	Command	Function	Condition	Direct	Type
3	ABSRPOS	Acquires the machine reference of the specified main group axis. (Valid only for axes where the return-to-origin method is set as "mark method".)	-	-	Functions
7	ARMCND	Acquires the current arm status of the main robot.	-	-	Functions
8	ARMTYPE	Acquires the current "hand system" setting of the main robot.	-	-	Functions
55	MCHREF	Acquires the return-to-origin or absolute-search machine reference for a specified main group axis.	-	-	Functions
111	TRQSTS	Acquires the command end status for the DRIVE command with torque limit option executed at the main group.	-	-	Functions
118	WHERE	Reads out the current position of the main group robot arm in joint coordinates (pulses).	-	-	Functions
120	WHRXY	Reads out the current position of the main group arm as Cartesian coordinates (mm, degrees).	-	-	Functions
115	WAIT ARM	Waits until the main group robot axis operation is completed.			

Status change

No.	Command	Function	Condition	Direct	Type
5	ACCEL	Specifies/acquires the acceleration coefficient parameter of the main group.	4/-	○	Command Statements/ Functions
6	ARCH	Specifies/acquires the arch position parameter of the main group.	4/-	○	Command Statements/ Functions
10	ASPEED	Changes the AUTO movement speed of the main group.	4	○	Command Statements
11	AXWGHT	Specifies/acquires the axis tip weight parameter of the main group.	4/-	○	Command Statements/ Functions
20	DECEL	Specifies/acquires the deceleration rate parameter of the main group.	4/-	○	Command Statements/ Functions
66	ORGORD	Specifies/acquires the axis sequence parameter for performing return-to-origin and absolute search operations in the main group.	4/-	○	Command Statements/ Functions
69	OUTPOS	Specifies/acquires the OUT enable position parameter of the main group.	4/-	○	Command Statements/ Functions
74	PDEF	Defines the pallet used to execute pallet movement commands.	1	○	Command Statements
97	SPEED	Changes the main group's program movement speed.	4	○	Command Statements
109	TOLE	Specifies/acquires the main group tolerance parameter.	4/-	○	Command Statements/ Functions
116	WEIGHT	Specifies/acquires the main robot tip weight parameter.	4/-	○	Command Statements/ Functions

Path control

No.	Command	Function	Condition	Direct	Type
70	PATH	Sets the movement path.	6	×	Command Statements
71	PATH END	Ends the movement path setting.	6	×	Command Statements
72	PATH SET	Starts the movement path setting.	6	×	Command Statements
73	PATH START	Starts the PATH motion.	6	×	Command Statements

Torque control

No.	Command	Function	Condition	Direct	Type
17	CURTRQ	Acquires the current torque value of the specified main group axis.	-	×	Functions
29	DRIVE	(With T-option) Executes an absolute movement command for a specified axis.	4	○	Command Statements
110	TORQUE	Specifies/acquires the maximum torque command value which can be set for a specified main group axis.	4/-	○	Command Statements/ Functions
112	TRQTIME	Specifies/acquires the current limit time-out period at the specified main group axis when using a torque limit option in the DRIVE statement.	1/-	○	Command Statements/ Functions

7

8

9

10

11

12

13

14

Input/output & communication control

Input/output control

No.	Command	Function	Condition	Direct	Type
24	DELAY	Waits for the specified period (units: ms).	6	×	Command Statements
28	DO	Outputs a specified value to the DO port.	1	○	Command Statements
52	LO	Outputs a specified value to the LO port to enable/disable axis movement.	1	○	Command Statements
57	MO	Outputs a specified value to the MO port.	1	○	Command Statements
68	OUT	Turns ON the bits of the specified output ports and the command statement ends.	6	×	Command Statements
81	RESET	Turns the bit of a specified output port OFF.	1	○	Command Statements
92	SET	Turns the bit at the specified output port ON.	3	In part ×	Command Statements
96	SO	Outputs a specified value to the SO port.	1	○	Command Statements
108	TO	Outputs a specified value to the TO port.	1	○	Command Statements
114	WAIT	Waits until the conditions of the DI/DO conditional expression are met (with time-out).	6	×	Command Statements

Programming box

No.	Command	Function	Condition	Direct	Type
44	INPUT	Assigns a value to a variable specified from the programming box.	1	○	Command Statements
78	PRINT	Displays a character string at the programming box screen.	1	○	Command Statements

Communication control

No.	Command	Function	Condition	Direct	Type
65	ONLINE	Sets the specified communication port to the "online" mode.	1	○	Command Statements
60	OFFLINE	Sets a specified communication port to the "offline" mode.	1	○	Command Statements
90	SEND	Sends a file.	1	○	Command Statements

Other

Other

No.	Command	Function	Condition	Direct	Type
122	_SYSFLG	Axis status monitoring flag.	-	-	Functions

Functions: in alphabetic order

No.	Function	Type	Function
A			
1	ABS	Arithmetic function	Acquires the absolute value of a specified value.
3	ABSRPOS	Arithmetic function	Acquires the machine reference of the specified main group axis. (Valid only for axes where the return-to-origin method is set as "mark method".)
5	ACCEL	Arithmetic function	Acquires the acceleration coefficient parameter of the main group.
6	ARCH	Arithmetic function	Acquires the arch position parameter of the main group.
7	ARMCND	Arithmetic function	Acquires the current arm status of the main robot.
8	ARMTYPE	Arithmetic function	Acquires the current "hand system" setting of the main robot.
9	ATN	Arithmetic function	Acquires the arctangent of the specified value.
11	AXWGHT	Arithmetic function	Acquires the axis tip weight parameter of the main group.
C			
15	CHR$	Character string function	Acquires a character with the specified character code.
16	COS	Arithmetic function	Acquires the cosine value of a specified value.
17	CURTRQ	Arithmetic function	Acquires the current torque value of the specified main group axis.
D			
19	DATE$	Character string function	Acquires the date as a "yy/mm/dd" format character string.
20	DECEL	Arithmetic function	Acquires the deceleration rate parameter of the main group.
23	DEGRAD	Arithmetic function	Converts a specified value to radians (↔RADDEG).
26	DIST	Arithmetic function	Acquires the distance between 2 specified points.
E			
33	ERL	Arithmetic function	Gives the line No. where an error occurred.
33	ERR	Arithmetic function	Gives the error code number of an error which has occurred.
I			
45	INT	Arithmetic function	Acquires an integer for a specified value by truncating all decimal fractions.
J			
46	JTOXY	Point function	Converts joint coordinate data to main group Cartesian coordinate data. (↔XYTOJ)
L			
48	LEFT$	Character string function	Extracts a character string comprising a specified number of digits from the left end of a specified character string.
50	LEN	Arithmetic function	Acquires the length (number of bytes) of a specified character string.
53	LOCx	Point function	Acquires point data or shift data for a specified axis.
54	LSHIFT	Arithmetic function	Shifts a value to the left by the specified number of bits. (↔RSHIFT)
M			
55	MCHREF	Arithmetic function	Acquires the return-to-origin or absolute-search machine reference for a specified main group axis.
56	MID$	Character string function	Extracts a character string of a desired length from a specified character string.

No.	Function	Type	Function
O			
61	ORD	Arithmetic function	Acquires the character code of the first character in a specified character string.
66	ORGORD	Arithmetic function	Acquires the axis sequence parameter for performing return-to-origin and absolute search operations in the main group.
69	OUTPOS	Arithmetic function	Acquires the OUT enable position parameter of the main group.
P			
77	PPNT	Point function	Creates point data specified by a pallet definition number and pallet position number.
R			
79	RADDEG	Arithmetic function	Converts a specified value to degrees. (↔DEGRAD)
85	RIGHT$	Character string function	Extracts a character string comprising a specified number of digits from the right end of a specified character string.
87	RSHIFT	Arithmetic function	Shifts a value to the right by the specified number of bits. (↔LSHIFT)
S			
95	SIN	Arithmetic function	Acquires the sine value for a specified value.
100	SQR	Arithmetic function	Acquires the square root of a specified value.
99	STR$	Character string function	Converts a specified value to a character string (↔VAL)
T			
104	TAN	Arithmetic function	Acquires the tangent value for a specified value.
105	TCOUNTER	Arithmetic function	Outputs count-up values at 10ms intervals starting from the point when the TCOUNTER variable is reset.
106	TIME$	Character string function	Acquires the current time as an "hh:mm:ss" format character string.
107	TIMER	Arithmetic function	Acquires the current time in seconds, counting from 12:00 midnight.
109	TOLE	Arithmetic function	Acquires the main group tolerance parameter.
110	TORQUE	Arithmetic function	Acquires the maximum torque command value which can be set for a specified main group axis.
111	TRQSTS	Arithmetic function	Acquires the command end status for the DRIVE command with torque limit option executed at the main group.
112	TRQTIME	Arithmetic function	Acquires the current limit time-out period at the specified main group axis when using a torque limit option in the DRIVE statement.
V			
113	VAL	Arithmetic function	Converts the numeric value of a specified character string to an actual numeric value. (↔STR$)
W			
116	WEIGHT	Arithmetic function	Acquires the main robot tip weight parameter.
118	WHERE	Point function	Reads out the current position of the main group robot arm in joint coordinates (pulses).
120	WHRXY	Point function	Reads out the current position of the main group arm as Cartesian coordinates (mm, degrees).
X			
121	XYTOJ	Point function	Converts the point variable Cartesian coordinate data to the main group's joint coordinate data (↔JTOXY).
122	_SYSFLG	Arithmetic function	Axis status monitoring flag.

Functions: operation-specific

Point related functions

No.	Function name	Function
46	JTOXY	Converts joint coordinate data to main group Cartesian coordinate data. (↔XYTOJ)
53	LOCx	Acquires point data or shift data for a specified axis.
77	PPNT	Creates point data specified by a pallet definition number and pallet position number.
118	WHERE	Reads out the current position of the main group robot arm in joint coordinates (pulses).
120	WHRXY	Reads out the current position of the main group arm as Cartesian coordinates (mm, degrees).
121	XYTOJ	Converts the point variable Cartesian coordinate data to the main group's joint coordinate data (↔JTOXY).

Parameter related functions

No.	Function name	Function
3	ABSRPOS	Acquires the machine reference of the specified main group axis. (Valid only for axes where the return-to-origin method is set as "mark method".)
5	ACCEL	Acquires the acceleration coefficient parameter of the main group.
6	ARCH	Acquires the arch position parameter of the main group.
7	ARMCND	Acquires the current arm status of the main robot.
8	ARMTYPE	Acquires the current "hand system" setting of the main robot.
11	AXWGHT	Acquires the axis tip weight parameter of the main group.
17	CURTRQ	Acquires the current torque value of the specified main group axis.
20	DECEL	Acquires the deceleration rate parameter of the main group.
50	LEN	Acquires the length (number of bytes) of a specified character string.
55	MCHREF	Acquires the return-to-origin or absolute-search machine reference for a specified main group axis.
61	ORD	Acquires the character code of the first character in a specified character string.
66	ORGORD	Acquires the axis sequence parameter for performing return-to-origin and absolute search operations in the main group.
69	OUTPOS	Acquires the OUT enable position parameter of the main group.
109	TOLE	Acquires the main group tolerance parameter.
110	TORQUE	Acquires the maximum torque command value which can be set for a specified main group axis.
111	TRQSTS	Acquires the command end status for the DRIVE command with torque limit option executed at the main group.
112	TRQTIME	Acquires the current limit time-out period at the specified main group axis when using a torque limit option in the DRIVE statement.
116	WEIGHT	Acquires the main robot tip weight parameter.

Numeric calculation related functions

No.	Function name	Function
1	ABS	Acquires the absolute value of a specified value.
9	ATN	Acquires the arctangent of the specified value.
16	COS	Acquires the cosine value of a specified value.
23	DEGRAD	Converts a specified value to radians (↔RADDEG).
26	DIST	Acquires the distance between 2 specified points.
45	INT	Acquires an integer for a specified value by truncating all decimal fractions.
54	LSHIFT	Shifts a value to the left by the specified number of bits. (↔RSHIFT)
79	RADDEG	Converts a specified value to degrees. (↔DEGRAD)
87	RSHIFT	Shifts a value to the right by the specified number of bits. (↔LSHIFT)
95	SIN	Acquires the sine value for a specified value.
100	SQR	Acquires the square root of a specified value.
104	TAN	Acquires the tangent value for a specified value.
113	VAL	Converts the numeric value of a specified character string to an actual numeric value. (↔STR$)

Character string calculation related functions

No.	Function name	Function
15	CHR $	Acquires a character with the specified character code.
19	DATE $	Acquires the date as a "yy/mm/dd" format character string.
48	LEFT $	Extracts a character string comprising a specified number of digits from the left end of a specified character string.
56	MID $	Extracts a character string of a desired length from a specified character string.
85	RIGHT $	Extracts a character string comprising a specified number of digits from the right end of a specified character string.
99	STR $	Converts a specified value to a character string (↔VAL)

Parameter related functions

No.	Function name	Function
122	_SYSFLG	Axis status monitoring flag.
33	ERL	Gives the line No. where an error occurred.
33	ERR	Gives the error code number of an error which has occurred.
105	TCOUNTER	Outputs count-up values at 10ms intervals starting from the point when the TCOUNTER variable is reset.
106	TIME $	Acquires the current time as an "hh:mm:ss" format character string.
107	TIMER	Acquires the current time in seconds, counting from 12:00 midnight.

1 ABS

Acquires absolute values

Format
ABS (<expression>)

Explanation Returns a value specified by an <expression> as an absolute value.

SAMPLE	
A=ABS(-326.55)...	The absolute value of -362.54 (=362.54) is assigned to variable A.

2 ABSINIT

Resets the current position of a specified axis

NOTE

- ABSINIT is available in software version 1.66M or higher.
- The ABSINIT statements can be used only when the "Limitless motion" parameter is set to "VALID" in the robot axis parameters. (For details, refer to the User's Manual.)

CAUTION

- When the <expression> is 0, the "17.42: Cannot reset position" error will occur if the robot's current position is at a position where a reset is impossible.

Format

1.ABSINIT (<axis number>)
2.ABSINIT (<axis number>)=<expression>

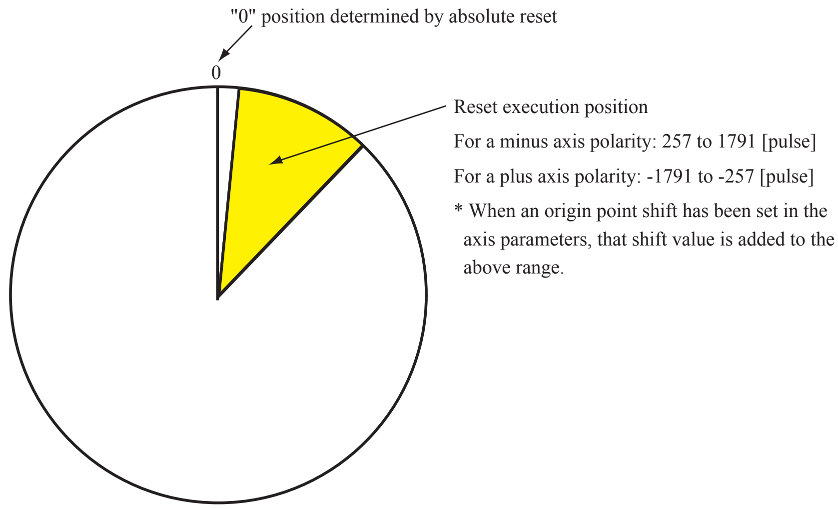
Values

<axis number>............................main group: 1 to 6
<expression> ....0 to 1

Explanation

Resets the current position of the axis specified by <axis number>.
If the <expression> is "0", a reset is possible only if the robot is positioned as shown in the figure below. To perform multi-turn movement in the same direction, the command statement must be executed after each 360° of movement, and the current position must be reset.
If the <expression> is "1", a reset occurs regardless of the robot's current coordinates. In this case, the robot's absolute function is disabled.
The format 1 operation is identical to a format 2 operation where the <expression> = 0.

"Reset possible" range within a mechanical angle of 360° (16384 [pulse] × speed reduction ratio)



SAMPLE

ABSINIT1..Resets the main group's 1st axis at the position

MEMO

- Following the reset, the current position and target position values become values from which a distance equivalent to the motor's number-of-turns has been subtracted.
- The reset time per axis is approximately 100ms.
- If a "Limitless motion INVALID" axis is specified, the "5.37: Specification mismatch" error message displays, and execution is stopped.

● Restrictions

1. The ABSINIT statement cannot be used at YC-Link specification axes.
2. The ABSINIT statement cannot be used at electric gripper specification axes.

3 ABSRPOS

Acquires a machine reference

Format
ABSRPOS (<axis 1>)

Values <axis 1>......................................main group: 1 to 6

Explanation The machine reference value for a specified <axis number> is acquired (units: %). This function is valid only for axes where the return-to-origin method is set as "mark method". It is not valid at axes where the return-to-origin method is set as "sensor" or "stroke end".

MEMO

- At axes where return-to-origin method is set to "mark" method, absolute reset is possible when the machine reference value is in a 44 to 56% range.

SAMPLE
A=ABSRPOS(4) .... The machine reference value for the main group's axis 4 is assigned to variable A.

4 ABSRST

Absolute motor axis return-to-origin operation

Format
ABSRST

Explanation This statement executes a direct return-to-origin operation for the robot's absolute motor axes (absolute reset).

The return-to-origin will fail if the robot stops en route.

MEMO

- This command is valid at axes where the return-to-origin method is set to other than "mark".
- This command cannot be executed if a return-to-origin is incomplete at an axis where the return-to-origin method is set as "mark".
- In systems with both absolute motor axes and incremental motor axes, a return-to-origin will occur only at the absolute motor axes when the ABSRST command is executed.
- The ORIGIN command must be used to perform a return-to-origin at incremental motor axes. Moreover, the return-to-origin operations occur in the parameter-specified sequence, and the incremental motor axes will not operate.

SAMPLE

```
*ABS_RST:
IF DI(20)=1 THEN................................................ABSRST executed when DI (20) is "1".
 ABSRST
ENDIF
*START:ABSRST..................................................Absolute motor return-to-origin occurs.
```

Related commands	ORIGIN, ORGORD, MCHREF

A B C D E F G H I J K L M

5 ACCEL

Specifies/acquires the acceleration coefficient parameter

Format
1. ACCEL <expression>
2. ACCEL (<axis number>)=<expression>

Values <axis number>............................main group: 1 to 6
<expression> ....1 to 100 (units: %)

Explanation Directly changes the acceleration coefficient parameters to the value specified by the <expression>.
In format 1, the change occurs at all the group axes.
In format 2, the change occurs at the axis specified in <axis number>.

MEMO

- If an axis that is set to "no axis" in the system generation is specified, a "5.37: Specification mismatch" error message displays and command execution is stopped.
- Changes the value which has been set at SYSTEM > PARAMETER > AXIS > ACCEL. Program-declared values have priority.

Functions

Format
ACCEL (<axis 1>)

Values <axis 1>......................................main group: 1 to 6

Explanation The acceleration parameter value is acquired for the axis specified at <axis number>.

SAMPLE

```
A=50
ACCEL A ...............................................................The acceleration coefficient for all axes becomes 50%.
ACCEL(3)=100.........................................................Only axis 3 becomes 100%.
'CYCLE WITH INCREASING ACCELERATION
FOR A=10  TO 100 STEP 10..............................The acceleration coefficient parameter is increased from
                                                      10% to 100% in 10% increments.
    ACCEL A
    MOVE P,P0
    MOVE P,P1
NEXT A
A=ACCEL(3)  .....................................................The acceleration coefficient parameter for the main group's
                                                      axis 3 is assigned to variable A.
HALT "END TEST"
```

6 ARCH

Specifies/acquires the acceleration coefficient parameter

Format

1. ARCH <expression>
2. ARCH (<axis number>)=<expression>

Values <axis number>............................main group: 1 to 6
<expression> ....1 to 6144000 (Unit: pulses)

Explanation Changes the parameter's arch position to the value indicated in the <expression>.
In format 1, the change occurs at all the group axes.
In format 2, the change occurs at the arch position parameter for the axis specified in <axis number> to the value specified in <expression>.

MEMO

- If an axis that is set to "no axis" in the system generation is specified, a "5.37: Specification mismatch" error message displays and command execution is stopped.

Functions

Format

ARCH (<axis 1>)

Values <axis 1>......................................main group: 1 to 6

Explanation Acquires the arch position parameter value of the axis specified at <axis number>.

6 ARCH

SAMPLE

```
DIM SAV(3)
GOSUB *SAVE_ARCH
FOR A=1000  TO 10000  STEP 1000
       GOSUB *CHANGE_ARCH
       MOVE P,P0,Z=0
       DO3(0)=1 ......................................................Chuck CLOSE
       MOVE P,P1,Z=0
       DO3(0)=0 ......................................................Chuck OPEN
NEXT A
GOSUB *RESTORE_ARCH
HALT
*CHANGE_ARCH:
FOR B=1 TO 4 ......................................................The arch position parameters ARCH (1) to (4) are assigned
                                                                    to array variables SAV (0) to (3).
       ARCH(B)=A
NEXT B
RETURN
*SAVE_ARCH:
FOR B=1 TO 4
       SAV(B-1)=ARCH(B)
NEXT B
RETURN
*RESTORE_ARCH:
FOR B=1 TO 4
       ARCH(B)=SAV(B-1)
NEXT B
RETURN
```

A
B
C
D
E
F
G
H
I
J
K
L
M

ARMCND
Arm status acquisition

Format

ARMCND

Explanation This function acquires the current arm status of the SCARA robot. The arm status is "1" for a left-handed system and "0" for a right-handed system.
This function is enabled only when a SCARA robot is used.

SAMPLE

```
A=ARMCND   ....................................The main robot's current arm status is assigned to variable A.
IF A=0 THEN  ....................................Right-handed system status.
     MOVE P, P100, Z=0
ELSE         ....................................Left-handed system status.
     MOVE P, P200, Z=0
ENDIF
```

8 ARMTYPE

SCARA robot hand system

Format

```
ARMTYPE
```

Explanation This function acquires the hand system currently selected for the SCARA robot.
The arm type is "0" for a right-handed system, and "1" for a right-handed system. This function is enabled only when a SCARA robot is used.

SAMPLE

```
A=ARMTYPE   ......................................The main robot's arm type value is assigned.
IF A=0 THEN ......................................The arm type is a right-handed system.
    MOVE P,P100,Z=0
ELSE        ......................................The arm type is a left-handed system.
    MOVE P,P200,Z=0
ENDIF
```

ATN

Acquires the arctangent of the specified value

Format
ATN (<expression>)

Explanation ATN: Acquires the arctangent values of the specified <expression> values. The acquired values are radians within the following range: -π / 2 to +π / 2

SAMPLE

A(0)=A*ATN(Y/X)..The product of the expression (Y/X) arctangent value and variable A is assigned to array A (0).
A(0)=ATN(0.5) ....The 0.5 arctangent value is assigned to array A (0).

Related commands	COS, DEGRAD, RADDEG, SIN, TAN

10 ASPEED

Sets the automatic movement speed

Format

ASPEED <expression>

Values <expression> ....1 to 100 (units: %)

NOTE

- Automatic movement speed specified by programming box operation or by the ASPEED command.
- Program movement speed specified by SPEED command or MOVE, DRIVE speed settings.

Explanation Directly changes the automatic movement speed to the value indicated in the <expression>. This speed change applies to all the robot axes and auxiliary axes. The operation speed is determined by the product of the automatic movement speed (specified by programming box operation and by the ASPEED command), and the program movement speed (specified by SPEED command, etc.).

Operation speed = automatic movement speed x program movement speed.

Example:
Automatic movement speed 80%
Program movement speed 50%
Movement speed = 40% (80% × 50%)

SAMPLE

```
SPEED 70
ASPEED 100
MOVE P,P0       .........................Movement from the current position to P0 occurs at 70% speed (=100 * 70).
ASPEED 50
MOVE P,P1       .........................Movement from the current position to P1 occurs at 35% speed (=50 * 70).
MOVE P,P2,S=10......................Movement from the current position to P2 occurs at 5% speed (=50 * 10).
HALT
```

Related commands	SPEED

11 AXWGHT

Sets/acquires the axis tip weight

Format
AXWGHT (<axis number>)=<expression>

Values <axis number>............................main group: 1 to 6
<expression> ....Varies according to the specified robot.

Explanation Directly changes the axis tip weight parameter for the group's axis specified by the <axis number> to the <expression> value.
This statement is valid in systems with "MULTI" axes and auxiliary axes (the robot type and auxiliary axes are factory set prior to shipment).

Functions

Format
AXWGHT (<axis 1>)

Values <axis 1>......................................main group: 1 to 6

Explanation Acquires the value axis tip weight parameter value for the axis specified by the <expression>.
This statement is valid in systems with "MULTI" axes and auxiliary axes.

SAMPLE

```
A=5
B=0
C=AXWGHT(1)..................................................Axis tip weight value is acquired (the current value is saved
                                                              to variable C).
AXWGHT(1)=A
DRIVE(1,P0)
AXWGHT(1)=B
DRIVE(1,P1)
AXWGHT(1)=C.................................................The axis tip weight value is set again.
HALT
```

Related commands	WEIGHT

7

12 CALL

Calls a sub-procedure

NOTE

- When a value is passed on to a sub-procedure, the original value of the actual argument will not be changed even if it is changed in the sub-procedure.
- When a reference is passed on to a sub-procedure, the original value of the actual argument will also be changed if it is changed in the sub-procedure.
- For details, see Chapter 3 "8 Value Pass-Along & Reference Pass-Along".

Format

CALL <label> [(<actual argument> [, <actual argument>…])]

Explanation This statement calls up sub-procedures defined by the SUB to END SUB statements.

The <label> specifies the same name as that defined by the SUB statement.

1. When a constant or expression is specified as an actual argument, its value is passed on to the sub-procedure.
2. When a variable or array element is specified as an actual argument, its value is passed on to the sub-procedure. It will be passed on as a reference if "REF" is added at the head of the actual argument.
3. When an entire array (array name followed by parentheses) is specified as an actual argument, it is passed along as a reference.

MEMO

- CALL statements containing one actual argument can be used up to 15 times in succession. Note that this number is reduced if commands which use stacks such as an IF statement or GOSUB statement are used, or depending on the number of arguments in the CALL statement.
- Always use the END SUB statement to end a sub-procedure which has been called with the CALL statement. If another statement such as GOTO is used to jump out of the sub- routine, a "5.12: Stack overflow" error, etc., may occur.

SAMPLE 1

```
X%=4
Y%=5
CALL *COMPARE ( REF X%, REF Y% )
HALT
'SUB ROUTINE:  COMPARE
SUB *COMPARE ( A%,  B% )
    IF A% < B% THEN
        TEMP%=A%
        A%=B%
        B%=TEMP%
    ENDIF
END SUB
```

SAMPLE 2

```
I = 1
CALL  *TEST( I )
HALT
'SUB ROUTINE:  TEST
SUB *TEST
    X = X + 1
    IF X < 15 THEN
        CALL *TEST( X )
    ENDIF
END SUB
```

Related commands	SUB, END SUB, CALL, DECLARE, EXIT SUB, SHARED

13 CHANGE

Switches the hand

Format
CHANGE Hn

Values n: The range of hand Nos. which can specified for the main group.

main group 0 to 3

Explanation CHANGE is used to switch the robot hand.

Before hand switching can occur, the hands must be defined at the HAND statement. For details, see section "39 HAND".

SAMPLE

```
HAND H1=           0       150.0       0.0
HAND H2=       -5000       20.00       0.0
P1=150.00 300.00 0.00 0.00 0.00 0.00
CHANGE H2   ..........................Changes to hand 2.
MOVE P,P1   ..........................Moves the hand 2 tip to P1 (1).
CHANGE H1   ..........................Changes to hand 1.
MOVE P,P1   ..........................Moves the hand 1 tip to P1 (2).
HALT
```

14 CHGPRI

Changes the priority ranking of a specified task

Format

```
CHGPRI Tn, p
```

Values

n: Task No ....2 to 8
p: Task priority ranking ....17 to 47

Explanation Directly changes the priority ranking of the specified task ("n") to "p".
The priority ranking of the main task (Task 1) is fixed as 32. Even if a priority ranking is not specified, "32" is adopted as the priority ranking for this task.
The smaller the priority number, the higher the priority (high priority: 17 - low priority: 47).
When a READY status occurs at a task with higher priority, all tasks with lower priority also remain in a READY status.

SAMPLE

```
START *SUBTASK,T2,33
*ST:
    MOVE P,P0,P1
    IF DI(20) = 1 THEN
        CHGPRI T2,32
    ELSE
        CHGPRI T2,33
    ENDIF
GOTO *ST
HALT
'SUBTASK ROUTINE
*SUBTASK:
    IF LOCZ(WHERE) > 10000 THEN
        DO(20) = 1
        GOTO *SUBTASK
    ENDIF
    DO(20) = 0
GOTO *SUBTASK
EXIT TASK
```

Related commands	CUT, EXIT TASK, RESTART, SUSPEND, START

15 CHR$

Acquires a character with the specified character code

Format
CHR$ (<expression>)

Values <expression> ....0 to 255

Explanation Acquires a character with the specified character code. An error occurs if the <expression> value is outside the 0 to 255 range.

SAMPLE
A$=CHR$(65) "A" is assigned to A$.

Related commands	ORD

16 COS

Acquires the cosine value of a specified value

Format

COS (<expression>)

Values <expression> ....Angle (units: radians)

Explanation Acquires a cosine value for the <expression> value.

SAMPLE

A(0)=B*COS(C)...The product of the C42 variable's cosine value and variable B is assigned to array A (0).
A(1)=COS(DEGRAD(20)).....................................The 20.0° cosine value is assigned to array A (1).

Related commands	ATN, DEGRAD, RADDEG, SIN, TAN

17 CURTRQ

Acquires the current torque of the specified axis

Format

CURTRQ (<expression>)

Values <expression> ....1 to 6

Explanation Acquires the current torque value (-100 to 100) of the axis specified by the <expression>. The current torque value is expressed as a percentage of the maximum torque command value. Plus/minus signs indicate the direction.

SAMPLE

A = CURTRQ(3)..The current torque value of the main group's axis 3 is assigned to variable "A".

18 CUT

Terminates another sub task which is currently being executed

Format

```
CUT Tn
```

Values n: Task No ....2 to 8

Explanation Directly terminates another task which is currently being executed or which is temporarily stopped.
This statement cannot terminate its own task, nor can it terminate Task 1.

SAMPLE

```
'TASK1 ROUTINE
*ST:
    MO(20) = 0
    START *SUBTASK2,T2
    MOVE P,P0
    MOVE P,P1
    WAIT MO(20) = 1
    CUT T2
GOTO *ST
HALT
'TASK2 ROUTINE
*SUBTASK2:
    P100=JTOXY(WHERE)
    IF LOCZ(P100) >= 100.0 THEN
        MO(20) = 1
    ELSE
        DELAY 100
    ENDIF
GOTO *SUBTASK2
EXIT TASK
```

Related commands	EXIT TASK, CUT, RESTART, START, SUSPEND

A
B
C
D
E
F
G
H
I
J
K
L
M

19 DATE$

Acquires the date

Format

```
DATE$
```

Explanation Acquires the date as a "yy/mm/dd" format character string.
"yy" indicates the year (last two digits), "mm" indicates the month, and "dd" indicates the day.
Date setting is performed at SYSTEM mode initial processing.

SAMPLE

```
A$=DATE$
PRINT DATE$
HALT
```

Related commands	TIME$

20 DECEL

Specifies/acquires the deceleration rate parameter

Format

1. DECEL <expression>
2. DECEL (<axis number>)=<expression>

Values <axis number>............................main group: 1 to 6
<expression> ....1 to 100 (units: %)

Explanation Changes the deceleration rate parameter to the <expression> value.
In format 1, the change occurs at all the group axes.
In format 2, the change occurs at the axis specified in <axis number>.

MEMO

- If an axis that is set to "no axis" in the system generation is specified, a "5.37: Specification mismatch" error message displays and command execution is stopped.
- Command statements DECEL can be used to change the acceleration parameter.

Functions

Format

DECEL (<axis 1>)

Values <axis 1>......................................main group: 1 to 6

Explanation Acquires the deceleration rate parameter value for the axis specified by the <expression>.

SAMPLE

```
A =50
DECEL A
DECEL(3)=100
'CYCLE WITH INCREASING DECELERATION
FOR A =10 TO 100 STEP 10
      DECEL A
      MOVE P ,P0
      MOVE P ,P1
NEXT A
A=DECEL(3)  ......................................................The deceleration rate parameter for the main group's axis 3
                                                                   is assigned to variable A.
HALT "END TEST "
```

21 DECLARE

Declares that a sub-routine or sub-procedure is to be used within the COMMON program

Format

1. DECLARE <label> [, <label>…]
2. DECLARE SUB <name> [(<dummy argument> [, <dummy argument>]…)]

CAUTION

- Only the following external labels can be used: GOSUB, CALL, ON to GOSUB.

Values

<label>..Label of the sub-routine defined in the COMMON program.

<name>..Name of the sub-procedure defined in the COMMON program.

<dummy argument>...................Sub-procedure argument. Only the "number of arguments" and the "data type" are significant.

Explanation Directly declares that a label or sub-procedure exists in the COMMON program. If a sub-procedure is declared, the argument's data type is also checked.

This statement cannot be defined within a sub-procedure.

Because the DECLARE statement declares the existence of a label or sub-procedure within the COMMON program, it cannot be used within the COMMON program itself. The DECLARE statement is valid throughout the entire program.

SAMPLE

```
COMMON program shared label
Program name: DIST1
'================================================
'   MAIN  PROGRAM
'================================================
DECLARE *DISTANCE,*AREA
X!=2. 5
Y!=1. 2
GOSUB *DISTANCE
GOSUB *AREA
HALT

Program name: COMMON
'================================================
'  'COMMON'PROGRAM
'================================================
*DISTANCE:
    PRINT SQR(X!^2+Y!^2)
RETURN
*AREA:
    PRINT X!*Y!
RETURN
```

SAMPLE

External program shared sub-procedure
Program name: DIST2

```
'==========================================================
'      MAIN  PROGRAM
'==========================================================
       DECLARE SUB *DISTANCE(X!,Y!,D!)
       DECLARE SUB *AREA(X!,Y!,A!)
       CALL  *DISTANCE(2. 5,1. 2,REF D!)
       PRINT D!
       CALL  *AREA(2. 5,1. 2,REF A!)
       PRINT A!
       HALT
       Program name: COMMON
'==========================================================
'      'COMMON' PROGRAM
'==========================================================
       SUB *DISTANCE(X!,Y!,D!)
              D!=SQR(X!^2+Y!^2)
       END SUB
       SUB *AREA(X!,Y!,A!)
              A!=X!*Y!
       END SUB
```

Related commands	CALL, EXIT SUB, GOSUB, ON to GOSUB, SUB, END SUB

22 DEF FN

Defines functions which can be used by the user

Format

```
DEF FN <name> [ % ] [(<dummy argument>, [<dummy argument>…])] = <function definition expression>
                !
                $
```

Values

<name> ....Function name. Max. of 16 chars., including "FN".
<dummy argument> ....Numeric or character string variable.

Explanation Defines the functions which can be used by the user. Defined functions are called in the FN <name> (<variable>) format.

MEMO

- The <dummy argument> names are the same as the variable names used in the <function definition expression>. The names of these variables are valid only when the <function definition expression> is evaluated. There may be other variables with the same name in the program.
- When calling a function that uses a <dummy argument>, specify the constant, variable, or expression type which is the same as the <dummy argument> type.
- If a variable used in the <function definition expression> is not included in the <dummy argument> list, the current value of that particular variable is used for the calculation.
- A space must be entered between "DEF" and "FN". If no space is entered, DEFFN will be handled as a variable.
- The DEF FN statement cannot be used in sub-procedures.
- Definition by the DEF FN statement must be declared before statements which use functions.

SAMPLE

```
DEF FNPAI=3.141592
DEF FNASIN(X)=ATN(X/SQR(-X^2+1))
                    ...................................................Both the <dummy argument> and <function definition
                                                                       expression> use "X".
        .
        .
A=FNASIN(B)*10................................................"X" is not required for calling.
```

23 DEGRAD

Angle conversion (angle → radian)

Format

DEGRAD (<expression>)

Values <expression> ....Angle (units: degrees)

Explanation The <expression> value is converted to radians.
To convert radians to degrees, use RADDEG.

SAMPLE

A=COS(DEGRAD(30))..A 30° cosine value is assigned to variable A.

Related commands	ATN, COS, RADDEG, SIN, TAN

24 DELAY

Program execution waits for a specified period of time

Format

DELAY <expression>

Values <expression> ....1 to 3600000 (units: ms)

Explanation A "program wait" status is established for the period of time specified by the <expression>. The minimum wait period is 10ms.

SAMPLE

```
DELAY 3500    3,500ms (3.5 secs) wait
DELAY A*10
```

25 DI

Acquires the input status from the parallel port

Format

1. [LET] <expression> = DIm([b,........,b])
2. [LET] <expression> = DI(mb,.......,mb)

Values
m...Port No.: 0 to 7, 10 to 17, 20 to 27
b..Bit definition: 0 to 7

Explanation Indicates the parallel input signal status.
If multiple bits are specified, they are expressed from the left in descending order (large to small).
Enter "0" if no input port exists.
If the [b,…,b] data is omitted, all 8 bits are processed.

SAMPLE

A%=DI2() ....The input status from DI (27) to DI (20) is assigned to variable A%.
A%=DI5(7,4,0) ....The DI (57), DI (54), DI (50) input status is assigned to variable A% (when all the above signals are "1" (ON), A% = 7).
A%=DI(37,25,20)..The DI (37), DI (25), DI (20) input status is assigned to variable A% (when all the above signals except DI (20) are "1" (ON), A% = 6).

Reference For details, refer to Chapter 3 "9.5 Parallel input variable".

26 DIST

Acquires the distance between 2 specified points

Format

DIST (<point expression 1>,<point expression 2>)

Values <point expression 1>..................Cartesian coordinate system point
<point expression 2>..................Cartesian coordinate system point

Explanation Acquires the distance (X,Y,Z)between the 2 points specified by <point expression 1> and <point expression 2>. An error occurs if the 2 points specified by each <point expression> do not have a Cartesian coordinates.

SAMPLE

A=DIST(P0,P1)..The distance between P0 and P1 is assigned to variable A.

27 DIM

Declares array variable

Format

DIM <array definition> [, <array definition>,…]

Format

<name> [%] (<constant> [, <constant> [, <constant>]])
          !
          $

Values <constant> ....Array subscript: 0 to 32,767 (positive integer)

Explanation Directly declares the name and length (number of elements) of an array variable. A maximum of 3 dimensions may be used for the array subscripts. Multiple arrays can be declared in a single line by using comma (,) breakpoints to separate the arrays.

MEMO

- Array subscripts can be "0 to a specified value", with their total number being the <constant> + 1.
- A "9.31: Memory full" error may occur depending on the size of each dimension defined in an array.

SAMPLE

DIM A%(10) ....Defines a integer array variable A% (0) to A% (10). (Number of elements: 11).

DIM B(2,3,4) ....Defines a real array variable B (0, 0, 0) to B (2, 3, 4). (Number of elements: 60).

DIM C%(2,2),D!(10)...Defines an integer array C% (0,0) to C% (2,2) and a real array D! (0) to D! (10).

28 DO

Outputs to parallel port

Format

1. [LET] DOm ([b,.........,b]) = <expression>
2. [LET] DO (mb,........,mb) = <expression>

Values

m: Port No. ....2 to 7, 10 to 17, 20 to 27
b: Bit definition...........................0 to 7
The output value is the lower left-side bit of the integer-converted <expression> value.

Explanation

Directly outputs the specified value to the DO port.
If multiple bits are specified, they are expressed from the left in descending order (large to small).
No output will occur if a nonexistent DO port is specified.
If the [b,…,b] data is omitted, all 8 bits are processed.
Outputs are not possible to DO0() and DO1(). These ports are for referencing only.

SAMPLE

```
DO2() = &B10111000...........................................DO (27, 25, 24, 23) are turned ON, and DO (26, 22, 21,
                                                              20) are turned OFF.
DO2(6,5,1) = &B010...........................................DO (25) are turned ON, and DO (26, 21) are turned OFF.
DO3() = 15      .............................................DO (33, 32, 31, 30) are turned ON, and DO (37, 36, 35,
                                                              34) are turned OFF.
DO(37,35,27,20) = A..........................................The contents of the 4 lower bits acquired when variable A is converted
                                                              to an integer are output to DO (37, 35, 27, 20) respectively.
```

Related commands	RESET, SET

29 DRIVE

Executes absolute movement of specified axes

Format

DRIVE(<axis number>, <expression>)[,(<axis number>, <expression>)...] [, option]

Values <axis number>............................main group: 1 to 6
<expression> ....Motor position (mm, degrees, pulses) or point expression

Explanation Executes absolute movement commands for specified axes within a group.
This command is also used in the same way for the group's auxiliary axes.
• Movement type: PTP movement of specified axis.
• Point setting method: By direct numeric value input and point definition.
• Options: Speed setting, STOPON conditions setting, torque limit setting, XY setting. movement direction setting.

Movement type

● PTP (Point to Point) movement of specified axis:

PTP movement begins after positioning of all axes specified at <axis number> is complete (within the tolerance range), and the command terminates when the specified axes enter the OUT position range. When two or more axes are specified, they will reach their target positions simultaneously.

If the next command following the DRIVE command is an executable command such as a signal output command, that next command will start when the movement axis enters the OUT position range. In other words, that next command starts before the axis arrives within the target position tolerance range.

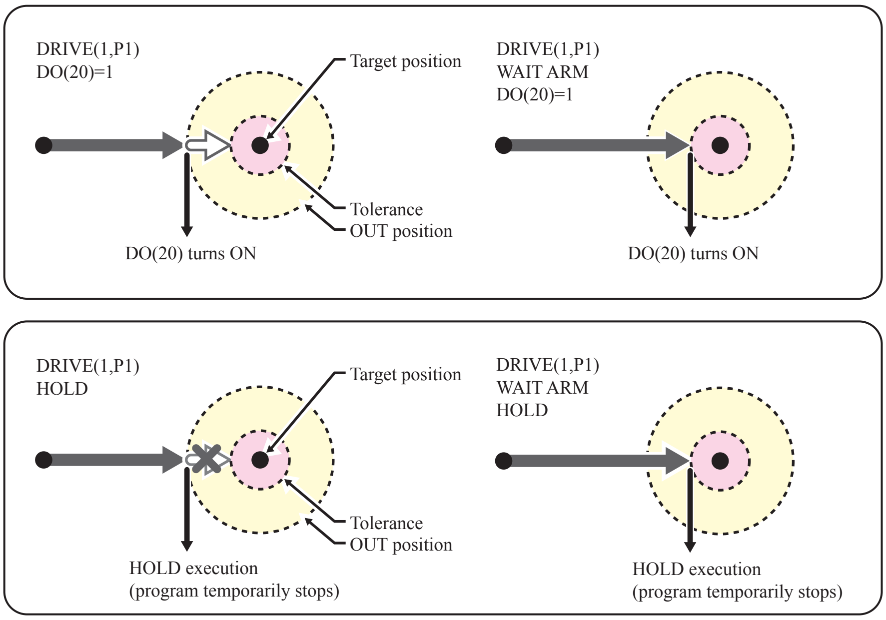
Example:

Signal output (DO, etc.)	Signal is output when axis enters within OUT position range.
DELAY	DELAY command is executed and standby starts, when axis enters the OUT position range.
HALT	Program stops and is reset when axis enters the OUT position range. Therefore, axis movement also stops.
HOLD	Program temporarily stops when axis enters the OUT position range. Therefore, axis movement also stops.
WAIT	WAIT command is executed when axis enters the OUT position range.

29 DRIVE

The WAIT ARM statement is used to execute the next command after the axis enters the tolerance range.

DRIVE command



SAMPLE

DRIVE(1,P0) Axis 1 moves from its current position to the position specified by P0.

Point data setting types

- Direct numeric value input

The motor position is specified directly in <expression>.

If the motor position's numeric value is an integer, this is interpreted as a "pulse" units. If the motor position's numeric value is a real number, this is interpreted as a "mm/degrees" units, and each axis will move from the 0-pulse position to a pulse-converted position.

However, when using the optional XY setting, movement occurs from the coordinate origin position.

SAMPLE

DRIVE(1,10000) ....Main group's axis 1 moves from its current position to the 10000 pulses position.

● Point definition

Point data is specified in <expressions>. The axis data specified by the <axis number> is used. If the point expression is in "mm/degrees" units, movement for each axis occurs from the 0-pulse position to the pulse-converted position.

However, when using the optional XY setting, movement occurs from the coordinate origin position.

NOTE

- If point data is specified with both integers and real numbers in the same statement, all values are handled in "mm/degrees" units.

SAMPLE

DRIVE(1,P1) ....Main group's axis 1 moves from its current position to the position specified by P1.

DRIVE(4,P90)..Axis 4 moves from its current position to the position specified by P90 (deg) relative to the 0 pulse position. (When axis 4 is a rotating axis.)

Option types

● Speed setting

Format

1. SPEED =<expression>
2. S =<expression>

NOTE

- This defines the maximum speed, and does not guarantee that all movement will occur at specified speed.

Values <expression> ....1 to 100 (units: %)

Explanation The program's movement speed is specified as an <expression>.

The actual speed is determined as shown below.

- Robot's max. speed (mm/sec, or deg/sec) × automatic movement speed (%) × program movement speed (%).

This option is enabled only for the specified DRIVE statement.

Format

1. DSPEED =<expression>
2. DS =<expression>

NOTE

- SPEED option and DSPEED option cannot be used together

Values <expression> ....0.01 to 100.00 (units: %)

Explanation The axis movement speed is specified in <expression>.

The actual speed is determined as shown below.

- Robot's max. speed (mm/sec, or deg/sec) × axis movement speed (%).

This option is enabled only for the specified DRIVE statement.

- Movement always occurs at the DSPEED <expression> value (%) without being affected by the automatic movement speed value (%).

29 DRIVE

● STOPON conditions setting

Format
STOPON <conditional expression>

Explanation Stops movement when the conditions specified by the conditional expression are met. **Because this is a deceleration type stop, there will be some movement (during deceleration) after the conditions are met.**
If the conditions are already met before movement begins, no movement occurs, and the command is terminated.
This option is enabled only by program execution.

SAMPLE

```
DRIVE(1,10000),STOPON DI(20)=1
```

Axis 1 moves from its current position toward the "10000 pulses" position and stops at an intermediate point if the "DI (20) = 1" condition is met. The next step is then executed.

MEMO

- When the conditional expression used to designate the STOPON condition is a numeric expression, the conditions for determining a TRUE or FALSE status can be changed at the controller's "TRUE conditions" in the "Other parameters" mode.
  These conditions apply to all the IF, WHILE, WAIT, STOPON, etc., conditional expressions. For details, refer to the controller manual.
  1) -1 (default setting) An expression value of "-1" indicates a TRUE status, and "0" indicates a FALSE status. A "6.35: Incorrect condition expression" error occurs if the expression value is other than "-1" or "0".
  2) not 0 Any expression value other than "0" indicates a TRUE status, and "0" indicates a FALSE status.

CAUTION

- The torque limit setting cannot be used at axes where YC-Link is connected, or at axes where a power gripper is being used. Attempts to specify this setting at these axes results in a "5.37 Specification mismatch" message, and command execution is stopped.
- The torque limit setting range differs depending on the robot model. Setting a torque limit higher than the maximum level may cause robot malfunctions or failure.
- If the specified torque limit value is too small, the axis movement may not occur. Moreover, vertical axes may fall.

● Torque limit setting

Format
1. T =<torque limit value> 2. T =(<torque limit value> [,<torque offset value>])

Values <torque limit value>...................1 to 100 (units: %)
<torque offset value> ....-100 to 100

Explanation Moves the axis while under torque control.
The maximum torque at this time is limited to a value calculated as follows: Rated torque × <torque limit value> / 100.
A <torque offset value> is specified to control the torque at vertical axes, etc., where a fixed load is constantly applied. When the torque limit value setting is omitted, a setting of "0" is adopted (the setting is also "0" in Format 1).
Specify the torque offset value with reference to the value which displays at the current command monitor while axis movement is stopped by servo HOLD.
Note, however, that the value displayed at the current command monitor is the maximum torque ratio. As a general guideline, the torque offset value should be set as approximately 1/3 of the displayed value in order to acquire the rated torque ratio.

MEMO

- The current command monitor can be displayed by pressing the [DISPLAY] key at the programming box. For details, refer to the controller manual.

When the DRIVE statement is executed with this option specified, the axis moves to the target position while controlling the torque by changing the maximum torque for the axis to the <torque limit value>.
The maximum movement speed at this time is 10% of the normal operating speed. No errors will occur even if the axis strikes an obstacle during movement, and the axis torque (thrust) will not exceed the limit value.

• **Command END conditions**

1. The command ends when the axis has reached the target position.
2. The command ends when the time (timeout period) specified by the TRQTIME statement has elapsed while the axis torque (thrust) has reached the limit value.

• **TRQSTS command value**

1. 1 is set at the TRQSTS function when this command has ended due to a time-out during which the axis torque has reached its limit value.
2. "0" is set if the command was ended for any other reason.

• **Cautions**

1. Maximum torque command value which have been changed by the TORQUE statement do not immediately become effective. They become effective at the next movement command (MOVE or DRIVE statement, etc.).
2. Even after this command ends, the maximum torque limit and torque control status remain in effect. The same applies if a stop occurs due to an interlock, etc., while this command is being executed.
3. Torque control is canceled when an axis related operation is executed. Such operations include servo ON/OFF switching, and a MOVE command execution, etc.
4. To cancel the maximum torque limit, use the TORQUE statement to specify a new maximum torque command value.
5. Maximum torque limit is cancelled at the following times regardless of whether or not a TORQUE statement is used:
   • When the controller power is turned ON.
   • When the servo is turned OFF.
   • When a return-to-origin or an absolute reset (except by the mark method) is executed.
   • When parameter data has been changed or initialized.

29 DRIVE

• Restrictions

1. Two or more axes cannot be specified with this option.
2. Maximum movement speed is set as 10% of the normal operating speed.
3. Manual movement is not possible at axes which are under torque control (axes where this command has been executed).

SAMPLE

TRQTIME(3)=2500......................................Sets the axis 3 torque control time-out period as 2.5 seconds.
DRIVE(3,P1),T=(20,15)................................Sets the maximum torque value to 20% of the rated torque, and the torque offset to 15, and moves the axis 3 from its current position to the point specified by P1 (pushing action).
IF TRQSTS(3)=1 THEN................................Checks to see if a time-out has occurred.
DO(21)=1..Time-out occurred (pushing is complete). (Result is output to DO(21) in this example.)
ELSE
DO(21)=0..Time-out has not occurred. (Reached target position but failed to complete pushing.) (Result is output to DO(21) in this example.)
ENDIF
TORQUE(3)=100..Maximum torque command value is returned to original value (100%).
DRIVE(3,P0) ....Torque limit and torque control end, and movement to P0 occurs.

● XY setting

Format
XY

Explanation Moves multiple specified axes to a position specified by Cartesian coordinates.
All the specified axes arrive at the target position at the same time.
If all axes which can be moved by MOVE statement has been specified, operation is identical to that which occurs when using MOVE statement.
The following restrictions apply to this command:
1. Axes specified by <axis number> must include the axis 1 and 2.
2. This command can be specified at SCARA robots.
3. Point settings must be in "mm" or "deg" units (real number setting).

SAMPLE

DRIVE(1,P100),(2,P100),(4,P100),XY
..............................The axis 1, 2 and 4 move from their current positions to the Cartesian coordinates position specified by P100.

● Movement direction setting

Format
PLS MNS

Explanation <With a "movement direction setting">

- Movement occurs in the specified direction. A PLS setting always results in plus-direction movement, and a MNS setting always results in minus-direction movement.
- If the target position and the current position are the same, a 1-cycle movement mount occurs in the specified direction, then operation stops.

<Without a "movement direction setting">

- Movement occurs in the direction in which the movement distance is shortest.
- If the target position and the current position are the same, no movement occurs.

• Cautions

1. When using this option, the maximum movement distance per operation is the distance equivalent to 1 cycle (360°). If movement which exceeds the 1-cycle distance is desired, the movement must be divided into 2 or more operations.
2. When using this option, the DRIVE statement's soft limit values are as shown below.
   Plus-direction soft limit: 67,000,000 [pulse]
   Minus-direction soft limit: -67,000,000 [pulse]

• Restrictions

1. Only the axis of a single-axis rotary type robot can be specified.
2. Simultaneous movement of multiple axes is not possible when a movement direction has been specified. If such movement is attempted, the "5.37: Specification mismatch" error will occur (see below).
   Example: DRIVE (3,P1), (4,P1), PLS
3. The PLS and MNS options cannot both be specified simultaneously.
4. Attempting to use this option for a "limitless motion INVALID" axis will result in the "2.29: Cannot move without the limit" error.
5. If a stop is executed by pressing the [STOP] key, etc., during movement which uses this option (including during a deceleration), the movement distance when restarted will be equivalent to a 1-cycle distance (360°).

29 DRIVE

SAMPLE

DRIVE (4,270.00), PLS
......................When the robot current position is 260°:
Moves 10° in the plus direction from the current position.
When the robot current position is 280°:
Moves 350° in the plus direction from the current position.
DRIVE (4,270.00), MNS
......................When the robot current position is 260°:
Moves 350° in the minus direction from the current position.
When the robot current position is 280°:
Moves 10° in the minus direction from the current position.
DRIVE (4,270.00)
......................When the robot current position is 260°:
Moves 10° in the plus direction from the current position.
When the robot current position is 280°:
Moves 10° in the minus direction from the current position.

Related commands	TORQUE, TRQTIME, TRQSYS, CURTRQ

30 DRIVEI

Moves the specified robot axes in a relative manner

Format

```
DRIVEI(<axis number>, <expression>)[,(<axis number>, <expression>)...][,option]
```

Values <axis number>............................1 to 4

<expression> ....Motor position (mm, deg, pulses) or point expression.

Explanation Directly executes relative movement of each axis of a group, including the group's auxiliary axes.

- Movement type : PTP movement of a specified axis
- Point data setting : Direct coordinate data input, point definition
- Options : Speed setting, STOPON conditions setting

MEMO

- When DRIVEI motion to the original target position is interrupted and then restarted, the target position for the resumed movement can be selected as the "MOVEI/DRIVEI start position" in the controller's "other parameters". For details, refer to the controller manual.

  1) KEEP (default setting) Continues the previous (before interruption) movement. The original target position remains unchanged.

  2) RESET Relative movement begins anew from the current position. The new target position is different from the original one (before interruption). (Backward compatibility)

Movement type

● PTP (point-to-point) of specified axis

PTP movement begins after positioning of all axes specified at <axis number> is complete (within the tolerance range), and the command terminates when the specified axes enter the OUT position range. When two or more axes are specified, they will reach their target positions simultaneously.

If the next command following the DRIVEI command is an executable command such as a signal output command, that next command will start when the movement axis enters the OUT position range. In other words, that next command starts before the axis arrives within the target position tolerance range.

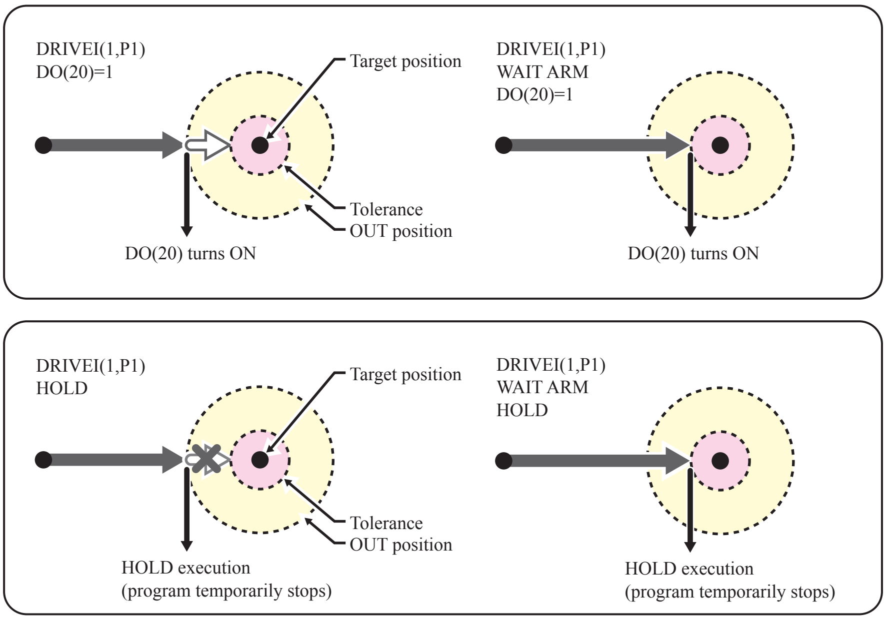
Example:

Signal output (DO, etc.)	Signal is output when axis enters within OUT position range.
DELAY	DELAY command is executed and standby starts, when axis enters the OUT position range.
HALT	Program stops and is reset when axis enters the OUT position range. Therefore, axis movement also stops.
HOLD	Program temporarily stops when axis enters the OUT position range. Therefore, axis movement also stops.
WAIT	WAIT command is executed when axis enters the OUT position range.

30 DRIVEI

The WAIT ARM statement is used to execute the next command after the axis enters the tolerance range.

DRIVEI command
WAIT ARM statement



Limitless motion related cautions

• When the "limitless motion" parameter is enabled, the DRIVEI statement soft limit check values are as follows:

Plus-direction soft limit: 67,000,000 [pulse]
Minus-direction soft limit: -67,000,000 [pulse]

•When using the DRIVEI statement, the above values represent the maximum movement distance per operation.

SAMPLE

```
DRIVEI(1,P0)   The axis 1 moves from its current position to the position specified by P0.
```

Point data setting types

● Direct numeric value input

The motor position is specified directly in <expression>.

If the motor position's numeric value is a real number, this is interpreted as a "mm / deg" units, and each axis will move from the 0-pulse position to a pulse-converted position.

SAMPLE

DRIVEI(1,10000) ....The axis 1 moves from its current position to the "+10000 pulses" position.

DRIVEI(4,90.00) ....The axis 4 moves from its current position to the +90° position (when axis 4 is a rotating axis).

NOTE

- If point data is specified with both integers and real numbers in the same statement, all values are handled in "mm/degrees" units.

● Point definition

Point data is specified in <expression>. The axis data specified by the <axis number> is used. The motor position is determined in accordance with the point data defined by the point expression. If the point expression is in "mm/degrees" units, movement for each axis occurs from the 0-pulse position to the pulse-converted position.

SAMPLE

DRIVEI(1,P1) ....The axis 1 moves from its current position the distance specified by P1.

DRIVEI(4,P90)...The axis 4 moves from its current position the number of degrees specified by P90 (when axis 4 is a rotating axis).

30 DRIVEI

Option types

● Speed setting

Format

1. SPEED=<expression>
2. S=<expression>

Values <expression> ....1 to 100 (units: %)

NOTE
• This defines the maximum speed, and does not guarantee that all movement will occur at specified speed.

Explanation The program's movement speed is specified by the <expression>.
The actual speed is as follows:
• Robot's max. speed (mm/sec, or deg/sec) × automatic movement speed (%) × program movement speed (%).
This option is enabled only for the specified DRIVEI statement.

SAMPLE

DRIVEI(1,10000),S=10..................................The axis 1 moves from its current position to the +10000 pulses position at 10% of the automatic movement speed.

Format

1. DSPEED=<expression>
2. DS=<expression>

Values <expression> ....0.01 to 100.00 (units: %)

NOTE
• SPEED option and DSPEED option cannot be used together.

Explanation The axis movement speed is specified as an <expression>.
The actual speed is determined as shown below.
• Robot's max. speed (mm/sec, or deg/sec) × axis movement speed (%).
This option is enabled only for the specified DRIVEI statement.
• Movement always occurs at the DSPEED <expression> value (%) without being affected by the automatic movement speed value (%).

SAMPLE

DRIVEI(1,10000),DS=0.1..............................The axis 1 moves from its current position to the +10000 pulses position at 0.1% of the automatic movement speed.

● STOPON conditions setting

Format
STOPON <conditional expression>

Explanation Stops movement when the conditions specified by the conditional expression are met. **Because this is a deceleration type stop, there will be some movement (during deceleration) after the conditions are satisfied.**
If the conditions are already satisfied before movement begins, no movement occurs, and the command is terminated.
This option is enabled only by program execution.

MEMO

- When the conditional expression used to designate the STOPON condition is a numeric expression, the conditions for determining a TRUE or FALSE status can be changed at the controller's "TRUE conditions" in the "Other parameters" mode. These conditions apply to all the IF, WHILE, WAIT, STOPON, etc., conditional expressions. For details, refer to the controller manual.

1) -1 (default setting)	An expression value of "-1" indicates a TRUE status, and "0" indicates a FALSE status. A "6.35: Incorrect condition expression" error occurs if the expression value is other than "-1" or "0".
2) not 0	Any expression value other than "0" indicates a TRUE status, and "0" indicates a FALSE status.

SAMPLE

DRIVEI(1,10000),STOPON DI(20)=1
..............................Axis 1 moves from its current position toward the "+10000 pulses" position and stops at an intermediate point if the "DI (20) = 1" condition become satisfied. The next step is then executed.

31 END SELECT

Ends the SELECT CASE statement

Format

```
SELECT [CASE] <expression>
    CASE <expression's list 1>
        [command block 1]
    CASE <expression's list 2>
        [command block 2]
        :
    [CASE ELSE
        [command block n]
END SELECT
```

Explanation Directly ends the SELECT CASE command block.
For details, see section "87 SELECT CASE".

SAMPLE

```
WHILE -1
SELECT CASE DI3()
    CASE 1,2,3
        CALL *EXEC(1,10)
    CASE 4,5,6,7,8,9,10
        CALL *EXEC(11,20)
    CASE ELSE
        CALL *EXEC(21,30)
END SELECT
WEND
HALT
```

Related commands	SELECT CASE

32 END SUB

Ends the sub-procedure definition

Format

```
SUB <label> [(<dummy argument> [, <dummy argument>…])]
    <command block>
END SUB
```

Explanation Ends the sub-procedure definition which begins at the SUB statement.
For details, see section "99 SUB to ENDSUB".

SAMPLE 1

```
I=1
CALL  *TEST
PRINT I
HALT
'SUB ROUTINE: TEST
SUB *TEST
 I=50
END SUB
```

Related commands	CALL, EXIT SUB, SUB to END SUB

33 ERR / ERL

Acquires the error code / error line No

Format

```
ERR
ERL
```

Explanation Variables ERR and ERL are used in error processing routines specified by the ON ERROR GOTO statement.
ERR gives the error code of the error that has occurred, and ERL gives the line number in which the error occurred.

SAMPLE 1

```
IF ERR <> &H604 THEN HALT
IF ERL=20 THEN RESUME NEXT
```

Related commands	ON ERROR GOTO, RESUME

34 EXIT FOR

Terminates the FOR to NEXT statement loop

Format

EXIT FOR

Explanation Directly terminates the FOR to NEXT statement loop, then jumps to the command which follows the NEXT statement.
This statement is valid only between the FOR to NEXT statements.

MEMO

- The FOR to NEXT statement loop will end when the FOR statement condition is satisfied or when the EXIT FOR statement is executed. A "5.12: Stack overflow" error, etc., will occur if another statement such as GOTO is used to jump out of the loop.

SAMPLE

```
*ST:
WAIT DI(20)=1
FOR  A%=101 TO 109
        MOVE  P,P100,Z=0
        DO(20)=1
        MOVE P,P[A%],Z=0
        DO(20)=0
        IF DI(20)=0 THEN EXIT FOR
NEXT A%
GOTO  *ST
HALT
```

Related commands	FOR, NEXT

35 EXIT SUB

Terminates the sub-procedure defined by SUB to END

Format
EXIT SUB

Explanation The EXIT SUB statement terminates the sub-procedure defined by the SUB to END SUB statements, then jumps to the next command in the CALL statement that called up the sub-procedure.

This statement is valid only within the sub-procedure defined by the SUB to END SUB statements.

MEMO

- To end the sub-procedure defined by the SUB to END SUB statements, use the END SUB statement or EXIT SUB statement. A "5.12: Stack overflow" error, etc., will occur if another statement such as GOTO is used to jump out of the loop.

SAMPLE

```
MAIN ROUTINE
CALL  *SORT2(REF X%,REF Y%)
HALT
'SUB ROUTINE: SORT
SUB *SORT2(X%, Y%)
        IF X%>=Y% THEN EXIT SUB
        TMP%=Y%
        Y%=X%
        X%=TMP%
END SUB
```

Related commands	CALL, SUB to END SUB, END SUB

A B C D E F G H I J K L M

36

EXIT TASK

Terminates its own task which is in progress

7

Format

EXIT TASK

Explanation Terminates its own task which is currently being executed.
This statement is valid for all tasks other than task 1 (Task 1 cannot be terminated).

MEMO

- Even if a task that has started as a subtask jumps to another task processing routine with a statement such as GO TO, that processing routine is then executed as this subtask processing.

SAMPLE

```
'TASK1 ROUTINE
*ST:
    MO(20)=0
    START *SUBTASK2,T2
    MOVE P,P0,P1
    WAIT MO(20)=1
    GOTO *ST
HALT
'TASK2 ROUTINE
*SUBTASK2:
    P100=JTOXY(WHERE)
    IF LOCZ(P100)>=100.0 THEN
        MO(20)=1
        EXIT TASK
    ENDIF
    DELAY 100
GOTO *SUBTASK2
EXIT TASK
```

Related commands	CUT, RESTART, START, SUSPEND, CHGPRI

A
B
C
D
E
F
G
H
I
J
K
L
M

37 FOR to NEXT

Performs loop processing until the variable-specified value is exceeded

Format

```
FOR <control variable> = <start value> TO <end value> [STEP] <step>]
    <command block>
NEXT [<control variable>]
```

Explanation These direct statements repeatedly execute commands between the FOR to NEXT statements for the <start value> to <end value> number of times, while changing the <control variable> value in steps specified by <STEP>. If <STEP> is omitted, its value becomes "1".

The <STEP> value may be either positive or negative.

The <control variable> must be a numeric <simple variable> or <array variable>.

The FOR and NEXT statements are always used as a set.

SAMPLE

```
'CYCLE WITH CYCLE NUMBER OUTPUT TO DISPLAY
FOR A=1 TO 10
    MOVE P,P0
    MOVE P,P1
    MOVE P,P2
    PRINT"CYCLE NUMBER=";A
NEXT A
HALT
```

Related commands	EXIT FOR

A B C D E F G H I J K L M

38 GOSUB to RETURN

Jumps to a sub-routine

Format

```
GOSUB <label>
  :
<label>:
  :
RETURN
```

* GOSUB can also be expressed as "GO SUB".

Explanation Jumps to the <label> sub-routine specified by the GOSUB statement.
A RETURN statement within the sub-routine causes a jump to the next line of the GOSUB statement.

MEMO

- The GOSUB statement can be used up to 29 times in succession. Note that this number of times is reduced if commands containing a stack such as an IF statement or CALL statement are used.
- When a jump to a subroutine was made with the GOSUB statement, always use the RETURN statement to end the subroutine. If another statement such as GOTO is used to jump out of the subroutine, an error such as "5.12: Stack overflow" may occur.

SAMPLE

```
*ST:
MOVE P,P0
GOSUB *CLOSEHAND
MOVE P,P1
GOSUB *OPENHAND
GOTO *ST
HALT
'SUB ROUTINE
*CLOSEHAND:
    DO(20) = 1
RETURN
*OPENHAND:
    DO(20) = 0
RETURN
```

Related commands	RETURN

39 GOTO

Executes an unconditional jump to the specified line

Format

GOTO <label>* GOTO can also be expressed as "GO TO".

Explanation Executes an unconditional jump to the line specified by <label>.
To select a conditional jump destination, use the ON to GOTO, or IF statements.

SAMPLE

```
'MAIN ROUTINE
*ST:
 MOVE P,P0,P1
 IF DI(20) = 1 THEN
  GOTO *FIN
 ENDIF
GOTO *ST
*FIN:
HALT
```

40 HALT

Stops the program and performs a reset

Format

HALT[| <expression> / <character string> |]

Explanation Directly stops the program and resets it. If restarted after a HALT, the program runs from its beginning.

MEMO

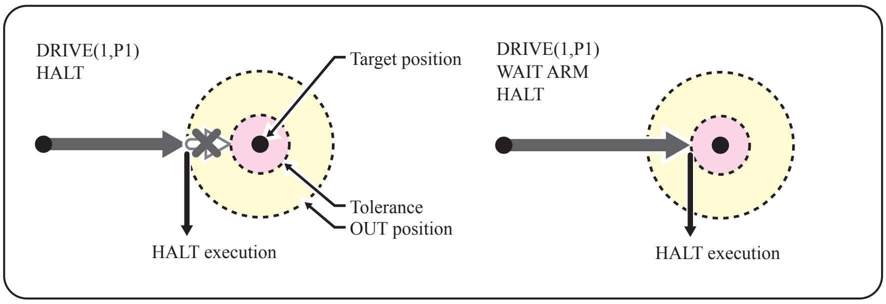
- If an <expression> or <character string> is written in the statement, the contents of the <expression> or <character string> are displayed on the programming box screen.
- If a "_SELECT" program name exists, processing will switch to that "_SELECT" program after the HALT command is executed.

SAMPLE

```
'MAIN ROUTINE
*ST:
 MOVE P,P0,P1
 IF DI(20) = 1 THEN
  GOTO *FIN
 ENDIF
GOTO *ST
*FIN:
HALT "PROGRAM FIN"
```

In PTP movement specified by movement commands such as MOVE and DRIVE, the next line's command is executed when the axis enters the OUT position range.
Therefore, if a HALT command exists immediately after a PTP movement command, that HALT command is executed before the axis arrives in the target position tolerance range.
Likewise, in interpolation movement during MOVE command, the next command is executed immediately after movement starts. Therefore, if a HALT command exists immediately after the interpolation movement command during MOVE, that HALT command is executed immediately after movement starts.
In either of the above cases, use the WAIT ARM command if desiring to execute the HALT command after the axis arrives within the target position tolerance range.

HALT command



Reference For details regarding "_SELECT", see Chapter 1 "4 Program Names".

41 HAND

Defines the hand

Format

Definition statement:
HAND Hn = <1st parameter> <2nd parameter> <3rd parameter> [R]
Selection statement:
CHANGE Hn

Values n: hand No ....main group: 0 to 3 → HAND is used.

Explanation The HAND statement only defines the hand. To actually change hands, the CHANGE statement must be used.
For CHANGE statement details, see section "12 CHANGE".

MEMO
- If a power OFF occurs during execution of the hand definition statement, the "9.7 Hand check-sum error" may occur.

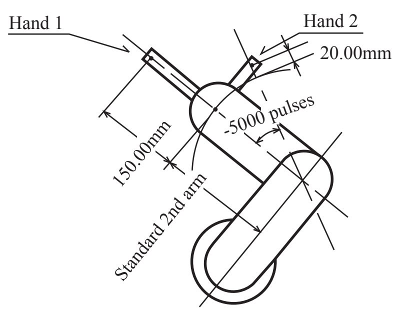
41.1 For SCARA Robots

1. When the <4th parameter> "R" is not specified

Hands installed on the second arm tip are selected (see below).

<1st parameter> .... Number of offset pulses between the standard second arm position and the virtual second arm position of hand "n". "+" indicates the counterclockwise direction [pulse].
<2nd parameter> .... Difference between the hand "n" virtual second arm length and the standard second arm length. [mm]
<3rd parameter> .... Z-axis offset value for hand "n". [mm]

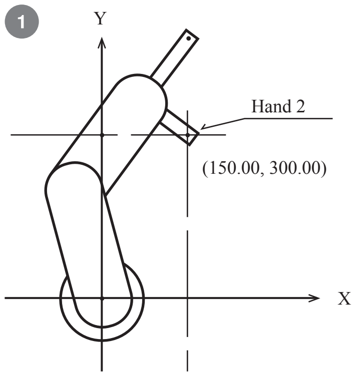
When the <4th parameter> "R" is not specified:

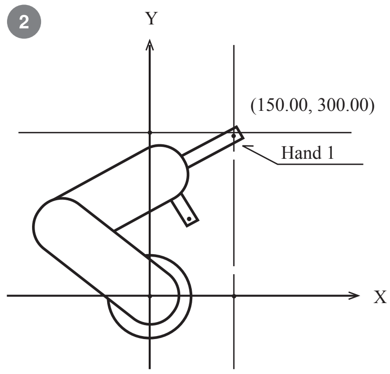


SAMPLE

```
HAND H1=        0          150.0       0.0
HAND H2=        -5000      20.00       0.0
P1=             150.00     300.00      0.00      0.00      0.00      0.00
CHANGE H2
MOVE P,P1    ...........................................................Tip of hand 2 moves to P1. ❶
CHANGE H1
MOVE P,P1    ...........................................................Tip of hand 1 moves to P1. ❷
HALT
```

SAMPLE:HAND



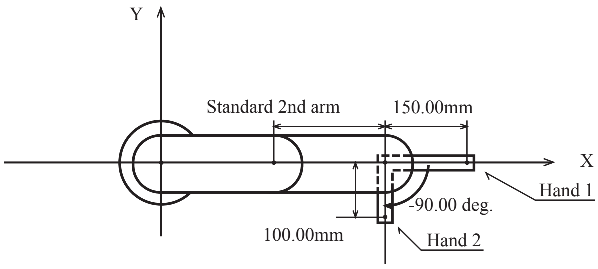


2. When the <4th parameter> "R" is specified

If the R-axis uses a servo motor, the hands that are offset from the R-axis rotating center are selected (see below).

<1st parameter> .... When the current position of R-axis is 0.00, this parameter shows the angle of hand "n" from the X-axis plus direction in a Cartesian coordinate system. ("+"indicates the counterclockwise direction.) [degree]
<2nd parameter> .... Length of hand "n". [mm] (>0)
<3rd parameter> .... Z-axis offset amount for hand "n". [mm]

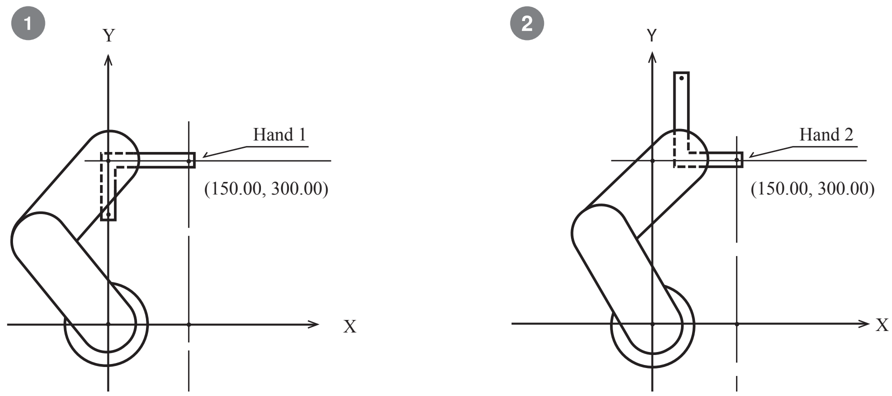
When the <4th parameter> "R" is specified



SAMPLE

```
HAND H1=      0.00     150.0     0.0      R
HAND H2=    -90.00    100.00     0.0      R
P1=         150.00    300.00    0.00    0.00    0.00    0.00
CHANGE H1
MOVE P,P1   ......................................................Tip of hand 1 moves to P1. ❶
CHANGE H2
MOVE P,P1   ......................................................Tip of hand 2 moves to P1. ❷
HALT
```

SAMPLE:HAND



42 HOLD

Temporarily stops the program

Format

```
HOLD[ | <expression>       | ]
      | <character string> |
```

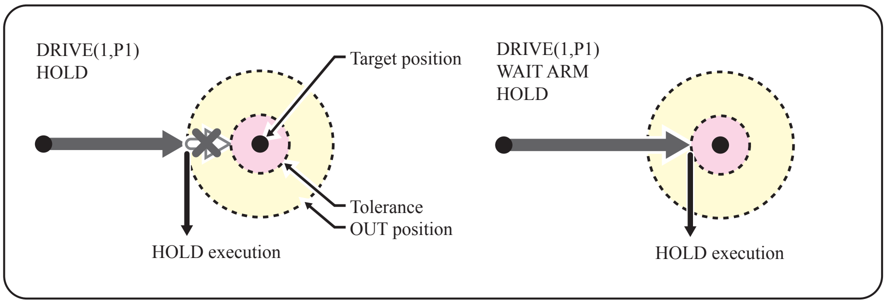
Explanation Temporarily stops the program. When restarted, processing resumes from the next line after the HOLD statement. If an <expression> or <character string> is written in the statement, the contents of the <expression> or <character string> display on the programming box screen.

SAMPLE

```
'MAIN ROUTINE
*ST:
    MOVE P,P0,P1
    IF DI(20)=1 THEN
        HOLD "PROGRAM STOP"
    ENDIF
GOTO *ST
HALT
```

In PTP movement specified by movement commands such as MOVE and DRIVE, the next line's command is executed when the axis enters the effective OUT position range.
Therefore, if a HOLD command exists immediately after a PTP movement command, that HOLD command is executed before the axis arrives in the target position tolerance range.
Likewise, in interpolation movement during MOVE command, the next command is executed immediately after movement starts. Therefore, if a HOLD command exists immediately after the interpolation movement command during MOVE, that HOLD command is executed immediately after movement starts.
In either of the above cases, use the WAIT ARM command if desiring to execute the HOLD command after the axis arrives within the target position tolerance range.

HOLD command



43 IF

Evaluates a conditional expression value, and executes the command in accordance with the conditions

MEMO

- When the conditional expression used to designate the IF statement conditions is a numeric expression, the conditions for determining a TRUE or FALSE status can be changed at the controller's "TRUE conditions" in the "Other parameters" mode. These conditions apply to all the IF, WHILE, WAIT, STOPON, etc., conditional expressions. For details, refer to the controller manual.

1) -1 (default setting)	An expression value of "-1" indicates a TRUE status, and "0" indicates a FALSE status. A "6.35 Incorrect condition expression" error occurs if the expression value is other than "-1" or "0".
2) not 0	Any expression value other than "0" indicates a TRUE status, and "0" indicates a FALSE status.

43.1 Simple IF statement

Format

IF <conditional expression> THEN | <label 1> / <command statement 1> | [ELSE | <label 2> / <command statement 2> |]

Explanation If the condition specified by the <conditional expression> is met, processing jumps either to the <label 1> which follows THEN, or to the next line after <command statement 1> is executed.

If the condition specified by the <conditional expression> is not met, the following processing occurs:

1. Processing either jumps to the <label 2> specified after the ELSE statement, or to the next line after <command statement 2> is executed.
2. If nothing is specified after the ELSE statement, no action is taken, and processing simply jumps to the next line.

SAMPLE

```
'MAIN ROUTINE
*ST:
    MOVE P,P0,P1
    IF DI(20)=1 THEN *L1................................If DI (20) is "1", a jump to *L1 occurs.
    DO(20)=1
    DELAY 100
*L1:
    IF DI(21)=1 THEN *ST ELSE *FIN
                ......................................................If DI (21) is "1", a jump to *ST occurs. If other than "1",
                                                                      a jump to *FIN occurs.
*FIN:
HALT
```

43.2 Block IF statement

Format

```
IF <conditional expression 1> THEN
  <command block 1>
[ELSEIF <conditional expression 2> THEN
  <command block 2>]
[ELSE
  <command block n>]
ENDIF
```

Explanation If the condition specified by <conditional expression 1> is met, this statement executes the instructions specified in <command block 1>, then jumps to the next line after ENDIF.
When an ELSEIF statement is present and the condition specified by <conditional expression 2> is met, the instructions specified in <command block 2> are executed.
If all the conditions specified by the conditional expression are not met, <command block n> is executed.

MEMO

- The IF statement can be used up to 48 times in succession. Note that the maximum number of times is reduced if commands containing a stack such as a GOSUB or CALL statement are used.

SAMPLE

```
'MAIN ROUTINE
*ST:
      MOVE P,P0,P1
      IF DI(21,20)=1 THEN
            DO(20)= 1
            DELAY 100
            WAIT DI(20)=0
      ELSEIF DI(21,20)=2 THEN
            DELAY 100
      ELSE
            GOTO *FIN
      ENDIF
GOTO *ST
*FIN:
HALT
```

44 INPUT

Assigns a value to a variable specified from the programming box

Format

INPUT [<prompt statement>	;	]	<variable>	[,	<variable>	,...]
	,		<point variable>		<point variable>	
			<shift variable>		<shift variable>	

Explanation Assigns a value to the variable specified from the programming box.
The input definitions are as follows:

1. When two or more variables are specified by separating them with a comma (,), the specified input data items must also be separated with a comma (,).
2. At the <prompt statement>, enter a character string enclosed in quotation marks (" ") that will appear as a message requiring data input. When a semicolon (;) is entered following the prompt statement, a question mark (?) and a space will appear at the end of the message. When a comma (,) is entered, nothing will be displayed following the message.
3. When the <prompt statement> is omitted, only a question mark (?) and a space will be displayed.
4. The input data type must match the type of the corresponding variables. When data is input to a point variable or shift variable, insufficient elements are set to "0".
5. If only the ENTER key is pressed without making any entry, the program interprets this as a "0" or "null string" input. However, if specifying two or more variables, a comma (,) must be used to separate them.
6. If the specified variable is a character type and a significant space is to be entered before and after a comma (,), double quotation mark (") or character string, the character string must be enclosed in double quotation marks ("). Note that in this case, you must enter two double quotation marks in succession so that they will be identified as a double quotation mark input.
7. Pressing the ESC key skips this command.

MEMO

- If the variable and the value to be assigned are different types, an "Input again" message displays and a "waiting for input" status is established.
- When assigning alphanumeric characters to a character variable, it is not necessary to enclose the character string in double quotation marks (").

SAMPLE

```
INPUT  A
INPUT  "INPUT POINT NUMBER";A1
INPUT  "INPUT STRING",B$(0),B$(1)
INPUT  P100
HALT
```

45 INT

Truncates decimal fractions

Format

INT (<expression>)

Explanation This function acquires an integer with decimal fractions truncated. The maximum integer value which does not exceed the <expression> value is acquired.

SAMPLE

```
A=INT(A(0))
B=INT(-1. 233)"-2" is assigned to B.
```

46 JTOXY

Performs axis unit system conversions (pulse → mm)

NOTE

- X-arm and Y-arm rotation information is only available in software Ver.1.66M or higher.

Format

JTOXY (<point expression>)

Explanation Converts the joint coordinate data (unit: pulses) specified by the <point expression> into Cartesian coordinate data (unit: mm, deg.).
On R6YXTW500 model robots, the X-arm and Y-arm rotation information is also set.

SAMPLE

P10=JTOXY(WHERE)..Current position data is converted to Cartesian coordinate data.

Related commands	XYTOJ

47

LABEL Statement

Defines labels at program lines

Format

```
*<label> :
```

Explanation Defines a <label> on a program line. A <label> must always begin with an asterisk mark (*), and it must be located at the beginning of the line.

Although a colon mark (:) is required at the end of the <label> when defining it, this mark is not required when specifying program jump destinations.

1. A <label> must begin with an alphabetic or numeric character.
2. Alphanumeric and underbars (_) can be used as the remaining <label> characters. Special symbols, etc., cannot be used.
3. The <label> must not exceed 16 characters (all characters beyond the 16th character are ignored).

SAMPLE

```
*ST:       ......................................................*ST label is defined.
 MOVE P,P0
 DO(20) = 1
 MOVE P,P1
 DO(20) = 0
GOTO       ......................................................*ST   Jumps to *ST.
HALT
```

48 LEFT$

Extracts character strings from the left end

Format

LEFT$ (<character string expression> , <expression>)

Values <expression> ....0 to 75

Explanation This function extracts a character string with the digits specified by the <expression> from the left end of the character string specified by <character string expression>.
The <expression> value must be between 0 and 75, otherwise an error will occur.
If the <expression> value is 0, then LEFT$ will be a null string (empty character string).
If the <expression> value has more characters than the <character string expression>, LEFT$ will become the same as the <character string expression>.

SAMPLE

```
B$=LEFT$(A$,4).......................................4 characters from the left end of A$ are assigned to B$.
```

Related commands	MID$, RIGHT$

49 LEFTY

Sets the SCARA robot hand system as a left-hand system

Format
LEFTY

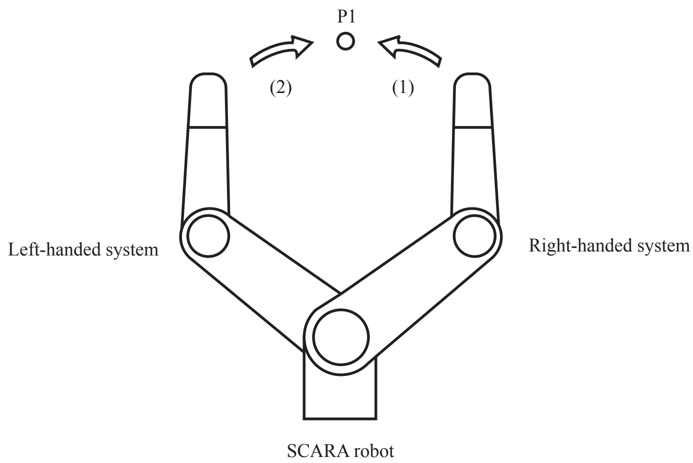
Explanation This statement specifies left-handed movement to a point specified in Cartesian coordinates. **This statement only selects the hand system, and does not move the robot.** If executed while the robot arm is moving, execution waits until movement is complete (positioned within tolerance range).

This command is only valid for SCARA robots.

SAMPLE

```
RIGHTY
MOVE P,P1
LEFTY
MOVE P,P1
RIGHTY
HALT
```

SAMPLE: LEFTY/RIGHTY



Related commands	RIGHTY

50 LEN

Acquires a character string length

Format

```
LEN(<character string expression>)
```

Explanation Returns the length (number of bytes) of the <character string expression>.

SAMPLE

```
B=LEN(A$)
```

51 LET

Assigns values to variables

Format		
[LET]	<arithmetic assignment statement> <character string assignment statement> <point assignment statement> <shift assignment statement>	

Explanation Directly executes the specified assignment statement. The right-side value is assigned to the left side. An assignment statement can also be directly written to the program without using a LET statement.

MEMO

- If the controller power is turned off during execution of a <point assignment statement> or <shift assignment statement>, a memory-related error such as the "9.2: Point check-sum error" or the "9.6: Shift check-sum error" may occur.

1. Arithmetic assignment statement

Format		
[LET]	<arithmetic variable> <point element variable> <shift element variable> <parallel output variable> <internal output variable> <arm lock output variable> <timer output variable> <serial output variable>	= <expression>

Values <expression> ....Variable, function, numeric value

Explanation The <expression> value is assigned to the left-side variable.

SAMPLE

```
A!=B!+1
B%(1,2,3)=INT(10.88)
LOCZ(P0)=A!
LOCX(S1)=100.00
DO2()=&B00101101
MO(21,20)=2
LO(00)=1
TO(01)=0
SO12()=255
```

2. Character string assignment statement

Format

[LET] <character string variable> = <character string expression>

Explanation The <character string expression> value is assigned to the character string variable.
Only the plus (+) arithmetic operator can be used in the <character string expression>.
Other arithmetic operators and parentheses cannot be used.

SAMPLE

```
A$ ="OMRON"
B$ ="ROBOT"
D$ = A$ + "-" + B$

Execution result: OMRON-ROBOT
```

MEMO

- The "+" arithmetic operator is used to link character strings.

3. Point assignment statement

Format

[LET] <point variable> = <point expression>

Explanation Assigns <point expression> values to point variables.
Only 4 arithmetic operators (+, - *, /) can be used in the <point expression>.
Multiplication and division are performed only for constant or variable arithmetic operations.

- Addition/subtraction .... Addition/subtraction is performed for each element of each axis.
- Multiplication .... Multiplication by a constant or variable is performed for each element of each axis.
- Division .... Division by a constant or variable is performed for each element of each axis.

Multiplication results vary according to the point data type.

- For pulse units .... Assigned after being converted to an integer.
- For "mm" units .... Assigned after being converted to a real number down to the 2nd decimal position.

SAMPLE

```
P1 =P10 ..................Point 10 is assigned to P1.
P20=P20+P5................Each element of point 20 and point 5 is summed and
                          assigned to P20.
P30=P30-P3................Each element of point 3 is subtracted from point 30 and
                          assigned to P30.
P80=P70*4.................Each element of point 70 is multiplied by 4 and assigned
                          to P80.
P60=P5/3 .................Each element of point 5 is divided by 3 and assigned to
                          P60.
```

MEMO

- Multiplication & division examples are shown below.
  - Permissible examples .... P15 * 5, P[E]/A, etc.
  - Prohibited examples .... P10 * P11, 3/P10, etc.

4. Shift assignment statement

Format

[LET] <shift variable> = <shift expression>

Explanation Assigns <shift expression> values to shift variables.
Only shift elements can be used in <shift expressions>, and only addition and subtraction arithmetic operators are permitted. Parentheses cannot be used.

- Addition/subtraction .... Addition/subtraction is performed for each element of each axis.

SAMPLE

```
S1=S0 ....................."shift 0" is assigned to "shift 1".
S2=S1+S0...................Each element of "shift 1" and "shift 0" is summed and
                           assigned to "shift 2".
```

MEMO

- Examples of <shift expression> addition/subtraction:
  - Permissible examples .... S1 + S2
  - Prohibited examples .... S1 + 3

52 LO

Arm lock output

Format

1. LO0 ([b,.....,b]) = <expression>
2. LO (0b,.....,0b) = <expression>

Values

b: Bit definition ....0 to 7
<expression> ....Converted to an integer. The lower bits corresponding to the bits specified on the left side are valid.

Explanation

This statement outputs the specified value to the LO port to either prohibit or allow axis movement.

LO(00) to LO(07) respectively correspond to axes 1 to 8. An arm lock ON status occurs at axes where bits are set, and axis movement is prohibited. Multiple bits must be specified in descending order from left to right (large → small).

MEMO

- This statement is valid at axes where movement is started.

SAMPLE

```
LO0()=&B00001010.............................................Prohibits movement at axes 2 and 4.
LO0(2,1)=&B10....................................................Prohibits movement at axis 3.
```

Related commands	RESET, SET

A B C D E F G H I J K L M

53 LOCx

Specifies/acquires point data or shift data for a specified axis

NOTE

- X-arm and Y-arm rotation information is only available in software Ver.1.66M or higher.

Format

1. LOCx (<point expression>) = <expression>
2. LOCx (<shift expression>) = <expression>

Values

Format 1: x ....X,Y,Z,R,A,B (axis setting), F (hand system flag setting), F1 (X-arm rotation information), F2 (Y-arm rotation information).

Format 2: x ....X,Y,Z,R (axis setting)

<expression> ....For axis setting: coordinate value.
For hand system flag setting:
1 (right-handed system) or 2 (left-handed system)
0 (no setting)
For specifies the X-arm rotation information and specifies the Y-arm rotation information:
0, 1, -1

*1: For details regarding the X-arm and Y-arm rotation information, refer to Chapter 4 "3. Point data format".

Explanation Direct format 1: Changes the value of the point data specified axis, the hand system flag, and the X-arm and Y-arm rotation information.

Format 2: Changes the value of a specified axis from the shift data value.

MEMO

- Points where data is to be changed must be registered in advance. An error will occur if a value change is attempted at an unregistered point (where there are no coordinate values).

53 LOCx

Functions

Format

1. LOCx (<point expression>)
2. LOCx (<shift expression>)

Values Format 1: x ....X,Y,Z,R,A,B (axis setting), F (hand system flag setting), F1 (X-arm rotation information), F2 (Y-arm rotation information).

Format 2: x ....X,Y,Z,R (axis setting)

Explanation Format 1: Acquires the value of the point data specified axis, the hand system flag, and the X-arm and Y-arm rotation information.

Format 2: Acquires a specified axis value from the shift data.

SAMPLE

```
LOCX(P10)=A(1).................................................The X-axis data of P10 is changed to the array A (1) value.
LOCY(S1)=B   ....................................................The Y-axis data of S1 is changed to the B value.

A(1)=LOCX(P10)..................................................The X-axis data of P10 is assigned to array A (1).
B(2)=LOCX(S1)....................................................The X-axis data of S1 is assigned to array B (2).
```

Related commands	Point element variable, shift element variable

54

LSHIFT

Left-shifts a bit

Format

LSHIFT (<expression 1> ,<expression 2>)

Explanation Shifts the <expression 1> bit value to the left by the amount of <expression 2>. Spaces left blank by the shift are filled with zeros (0).

SAMPLE

A=LSHIFT(&B10111011,2)..................................The 2-bit-left-shifted &B10111011 value (&B11101100) is assigned to A.

Related commands	RSHIFT

7

55 MCHREF

Acquires a machine reference

Format

MCHREF (<axis 1>)

Values <axis 1>......................................main group: 1 to 6

Explanation This function provides the return-to-origin or absolute-search machine reference (unit: %) for the axis specified by <axis number>.

This function can only be used for axes whose return-to-origin method is set to the sensor or stroke end method.

SAMPLE

A=MCHREF(1)...The main group's axis 1 return-to-origin machine reference is assigned to variable A.

A
B
C
D
E
F
G
H
I
J
K
L
M

56 MID$

Acquires a character string from a specified position

Format

MID$ (<character string expression>,<expression 1>[,<expression 2>])

Values

<expression 1> ....1 to 75
<expression 2> ....0 to 75

Explanation

This function extracts a character string of a desired length (number of characters) from the character string specified by <character string expression>. <expression 1> specifies the character where the extraction is to begin, and <expression 2> specifies the number of characters to be extracted.

An error will occur if the <expression 1> and <expression 2> values violate the permissible value ranges.

If <expression 2> is omitted, or if the number of characters to the right of the character of <expression 1> is less than the value of <expression 2>, then all characters to the right of the character specified by <expression 1> will be extracted.

If <expression 1> is longer than the character string, MID$ will be a null string (empty character string).

SAMPLE

B$=MID$(A$,2,4)...The 2nd to 4th characters (up to the 5th char.) of A$ are assigned to B$.

Related commands	LEFT$, RIGHT$

57 MO

Outputs a specified value to the MO port (internal output)

Format

1. MO0([b,.....,b]) = <expression>
2. MO(0b,.....,0b) = <expression>

REFERENCE

- For details regarding bit definitions, see Chapter 3 "10 Bit Settings".

Values

m: port number ....2 to 7, 10 to 17, 20 to 27
b: bit definition ....0 to 7
<expression> ....The integer-converted lower bits corresponding to the bit definition specified at the left side are valid.

Explanation

Outputs a specified value to the MO port.
In order to maintain the sensor status and axis HOLD status at each axis, ports "0" and "1" cannot be used as output ports (these ports are for referencing only).
If multiple bits are specified, they are expressed from the left in descending order (large to small).
If the [b,…,b] data is omitted, all 8 bits are processed.

• Ports "0" and "1" outputs

Bit	7	6	5	4	3	2	1	0
Port 0	Axis 8	Axis 7	Axis 6	Axis 5	Axis 4	Axis 3	Axis 2	Axis 1
	Origin sensor status 0: OFF; 1: ON							
Port 1	Axis 8	Axis 7	Axis 6	Axis 5	Axis 4	Axis 3	Axis 2	Axis 1
	HOLD status 0: No HOLD / 1: HOLD (1 axis is not used)							

MEMO

- For details regarding MO ports "0" and "1", see Chapter 3 "9.7 Internal output variable".

SAMPLE

MO2()=&B10111000..MO (27, 25, 24, 23) are turned ON, and MO (26, 22, 21, 20) are turned OFF.
MO2(6,5,1)=&B010...MO (25) are turned ON, and MO (26, 21) are turned OFF.
MO3() = 15 ....MO (33, 32, 31, 30) are turned ON, and MO (37, 36, 35, 34) are turned OFF.
MO(37,35,27,20)=A...The contents of the 4 lower bits acquired when variable A is converted to an integer are output to MO (37, 35, 27, 20), respectively.

Related commands	RESET, SET

58 MOVE

Performs absolute movement of all robot axes

Format

MOVE | PTP / P / L / C | , <point definition> [, option [, option] ...]

Explanation Executes direct absolute movement of a group's axes.

The MOVE command is used for axes which have been specified as main robot axes. It is not enabled for other groups, or for auxiliary axes.

- Movement type : PTP, linear interpolation, circular interpolation.
- Point data setting : Direct coordinate data input, point definition.
- Options : Speed setting, arch motion setting, STOPON condition setting, CONT setting, acceleration setting, deceleration setting, plane coordinate setting, port output setting.

Options	PTP	Linear interpolation	Arch interpolation	Remarks
Speed setting (SPEED)	○	○	○	Enabled only for specified MOVE statement
Speed setting (VEL)	×	○	○	Enabled only for specified MOVE statement
Arch motion	○	×	×	Enabled only for specified MOVE statement
STOPON condition setting	○	○	×	Enabled only by program execution
CONT setting	○	×	×	Enabled only for specified MOVE statement
Acceleration setting	×	○	×	Enabled only for specified MOVE statement
Deceleration setting	×	○	×	Enabled only for specified MOVE statement
Plane coordinate setting	×	×	○	Enabled only for specified MOVE statement
Port output setting	×	○	○	Enabled only for specified MOVE statement

58 MOVE

Movement type

- PTP (point-to-point) movement

Execution START condition: Movement of all specified axes is complete (within the tolerance range).
Execution END condition: All specified axes have entered the OUT position range.
When two or more axes are specified, they will reach their target positions simultaneously. The movement path of the axes is not guaranteed.

- Caution regarding commands which follow the MOVE P command:

If the next command following the MOVE P command is an executable command such as a signal output command, that next command will start when the movement axis enters the OUT position range. In other words, that next command starts before the axis arrives within the target position tolerance range.
Example:

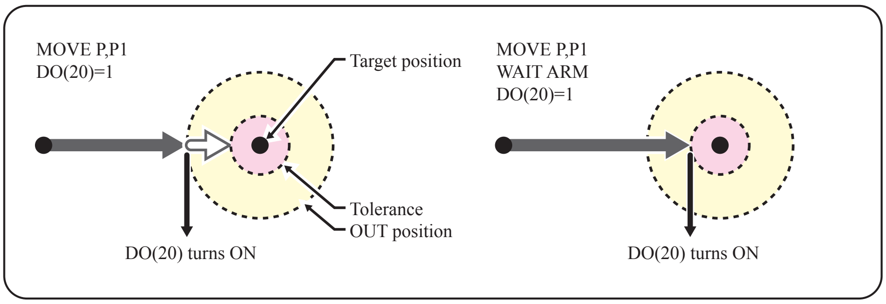
Signal output (DO, etc.)	Signal is output when axis enters within OUT position range.
DELAY	DELAY command is executed and standby starts, when axis enters the OUT position range.
HALT	Program stops and is reset when axis enters the OUT position range. Therefore, axis movement also stops.
HOLD	Program temporarily stops when axis enters the OUT position range. Therefore, axis movement also stops.
WAIT	WAIT command is executed when axis enters the OUT position range.

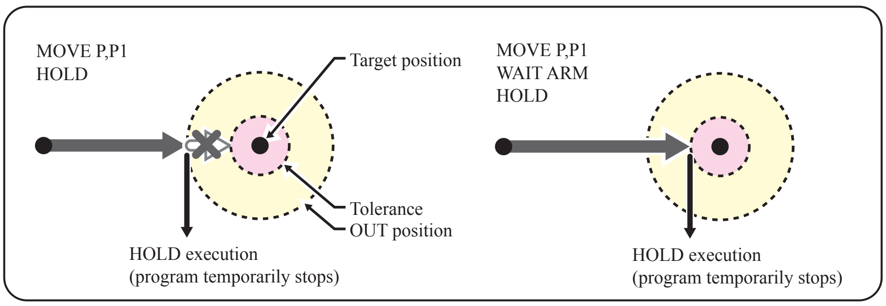
The WAIT ARM statement is used to execute the next command after the axis enters the tolerance range.

MEMO

- The OUT position value is specified by parameter setting.
  This value can be changed from within the program by using the OUTPOS command.

MOVE command





SAMPLE

MOVE P,P0 ....The main robot axis moves from its current position to the position specified by P0 (the same occurs for MOVE PTP, P0).

MEMO

- PTP movement is faster than interpolation movement, but when executing continuous movement to multiple points, a positioning stop occurs at each point.

● Linear interpolation movement

Execution START condition: Movement of all specified axes is complete (within the tolerance range).
Execution END condition: Movement of all specified axes has begun.

Execution of the immediately following command occurs immediately after axis movement begins.

When executing linear or circular interpolation in a continuous manner, the 2 movement paths are linked by connecting the deceleration and acceleration sections, enabling continuous movement without intermediate stops.
All movement axes arrive at the same time.
However, the following execution END condition applies if a STOPON condition has been specified:

Execution END condition: Movement of all specified axes is complete (within the tolerance range).
In this case, continuous movement for the 2 linked paths is not possible.

● Caution regarding commands which follow a MOVE L command:

If the next command following the MOVE L command is an executable command such as a signal output command, that next command will start immediately after axis movement begins.
Example:

Signal output (DO, etc.)	Signal is output immediately after movement begins.
DELAY	DELAY command is executed and standby starts immediately after movement begins.
HALT	Program stops and is reset immediately after movement begins. Therefore, axis movement also stops.
HOLD	Program temporarily stops immediately after movement begins. Therefore, axis movement also stops.
WAIT	WAIT command is executed immediately after movement begins.

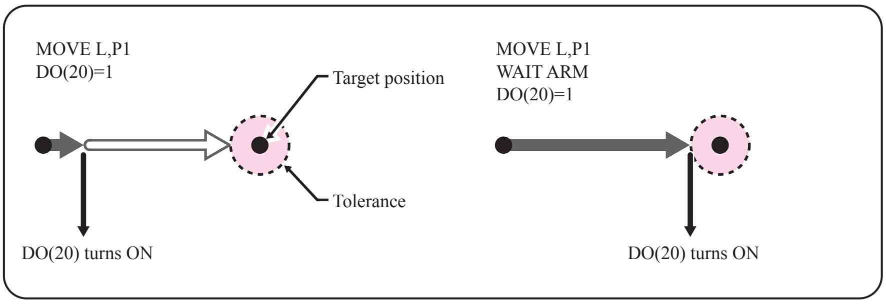
The WAIT ARM statement is used to execute the next command after the axis enters the tolerance range.

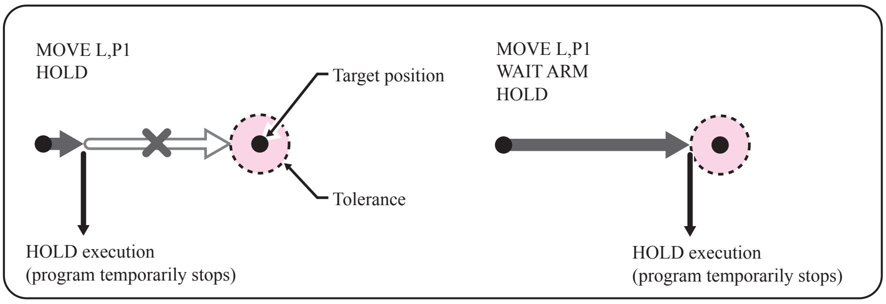
58 MOVE

MEMO

- Smooth travel paths and constant speeds between paths may not always be possible, depending on the axis movement speed, acceleration, and distance between the target points.
- On robots with an R-axis, the R-axis speed may become too fast and cause an error depending on the R-axis movement distance.
- If a DELAY statement is executed after a MOVE L command, a DELAY timer is activated after the MOVE L command is executed. Therefore, **if a DELAY is desired after reaching the target point, use the WAIT ARM statement after the MOVE command.**
  The same applies for other commands such as HALT, etc.
- If the direction changes at an acute angle during interpolation movement, the acceleration/deceleration speed of the connection section may become too fast, causing an error. In this case, specify a slower acceleration/deceleration speed at the connection section, or use the WAIT ARM command to revise the operation pattern.

MOVE command

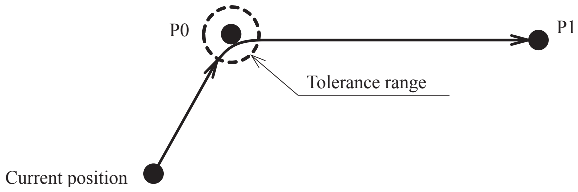




SAMPLE

MOVE L,P0,P1..The main robot axis moves from its current position to the position specified by P0, P1.

SAMPLE: MOVE L



● Circular interpolation movement

Execution START condition: Movement of all specified axes is complete (within the tolerance range).

Execution END condition: Movement of all specified axes has begun.

Execution of the immediately following command occurs immediately after axis movement begins.

When executing linear or circular interpolation in a continuous manner, the 2 movement paths are linked by connecting the deceleration and acceleration sections, enabling continuous movement without intermediate stops.

All movement axes arrive at the same time.

In circular interpolation, an arc is generated based on 3 points: the current position, an intermediate position, and the target position. **Therefore, circular interpolation must be specified by an even number of points.**

● Caution regarding commands which follow a MOVE C command:

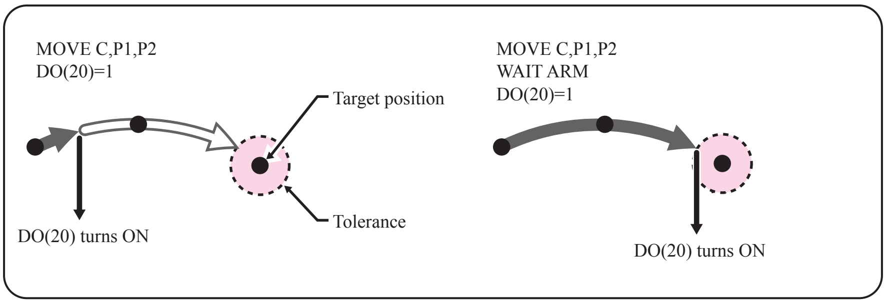
If the next command following the MOVE C command is an executable command such as a signal output command, that next command will start immediately after axis movement begins.

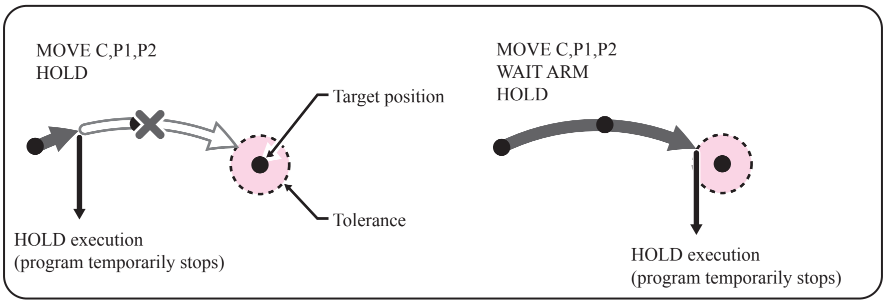
Example:

Signal output (DO, etc.)	Signal is output immediately after movement begins.
DELAY	DELAY command is executed and standby starts immediately after movement begins.
HALT	Program stops and is reset immediately after movement begins. Therefore, axis movement also stops.
HOLD	Program temporarily stops immediately after movement begins. Therefore, axis movement also stops.
WAIT	WAIT command is executed immediately after movement begins.

The WAIT ARM statement is used to execute the next command after the axis enters the tolerance range.

MOVE command



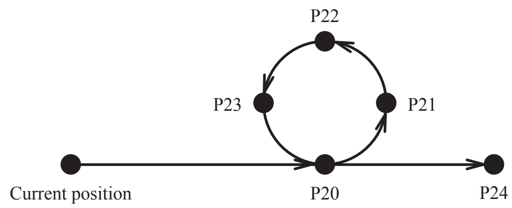


58 MOVE

SAMPLE

MOVE L,P20..Linear movement occurs from the current position to P20.
MOVE C,P21,P22,P23,P20...................Arc movement occurs through points P21, P22, P23, P20.
MOVE L,P24..Linear movement occurs to P24.

SAMPLE: MOVE C

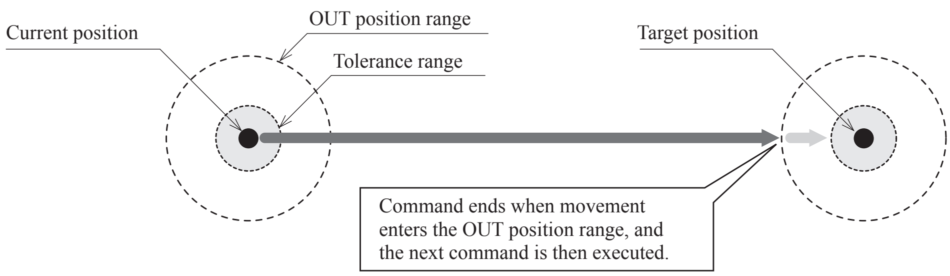


MEMO

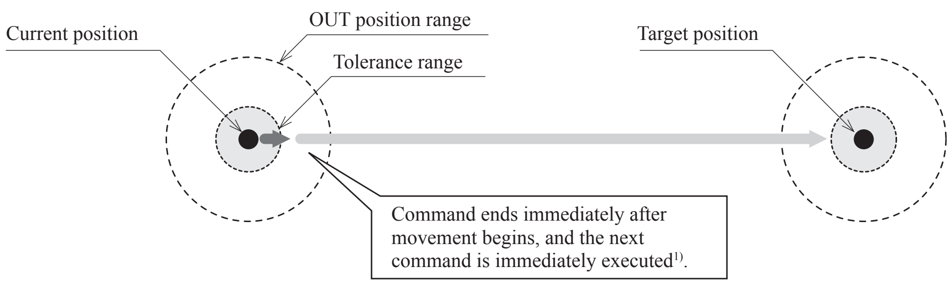
- In continuous interpolation operations, too, there are no stops at intermediate points. However, the maximum speed is slower than the PTP speed.
- Circular interpolation is possible within the following range: radius 1.00mm to 5,000.00mm.
- Circle distortion may occur, depending on the speed, acceleration, and the distance between points.
- On robots with an R-axis, the R-axis speed may become too fast and cause an error, depending on the R-axis movement distance.
- If a DELAY statement is executed after a MOVE L command, a DELAY timer is activated after the MOVE L command is executed. Therefore, **if a DELAY is desired after reaching the target point, use the WAIT ARM statement after the MOVE statement.**
  The same applies for other commands such as HALT, etc.
- If the direction changes at an acute angle during interpolation movement, the acceleration/deceleration speed of the connection section may become too fast, causing an error. In this case, specify a slower acceleration/deceleration speed at the connection section, or use the WAIT ARM command to revise the operation pattern.

Movement command types and the corresponding movement

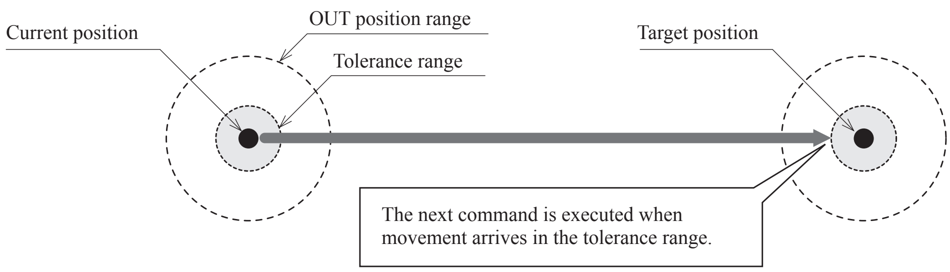
1. PTP movement



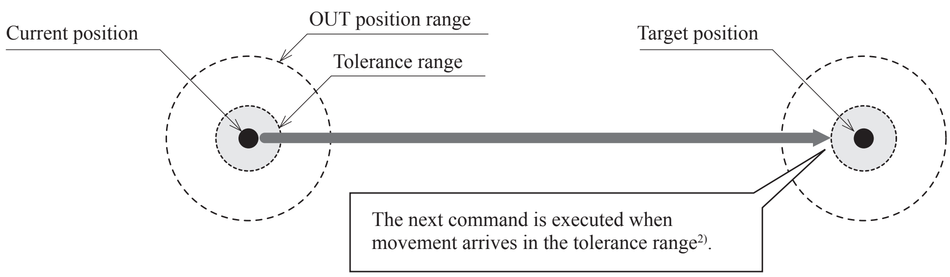
2. Interpolation movement



3. WAIT ARM



4. STOPON conditional expression



1) If a DELAY statement is executed after an interpolation operation, a DELAY timer is activated immediately after the movement starts. Therefore, if a DELAY is desired after reaching the target point, use the WAIT ARM statement after the MOVE command.
2) A deceleration and stop occurs at an intermediate point if the condition specified by the STOPON conditional expression is met (for details, see the "STOPON Condition Setting" item).

A B C D E F G H I J K L M

58 MOVE

Point data setting types

● Direct numeric value input PTP Linear interpolation Circular interpolation

Format
X Y Z R A B [F] [F1] [F2]

Values
X, Y, Z, R, A, B ....Space-separated coordinate values for each axis.
F..Hand system flag
F1..X-arm rotation information (R6YXTW500 model only).
F2..Y-arm rotation information (R6YXTW500 model only).

Explanation
Directly specifies coordinate values by a numeric value. If an integer is used, this is interpreted as "pulse" units, and if a real number (with decimal point) is used, this is interpreted as "mm/deg" units, with movement occurring accordingly. If both integers and real numbers are used together (mixed), all coordinate values will be handled in "mm/deg" units.
This setting method can be used only for PTP and linear interpolation movement types.
Hand system flags can be specified for SCARA robots when directly specifying the coordinate values in "mm" units.
To specify an extended hand system flag for SCARA robots, set either 1 or 2 at "F". If a number other than 1 or 2 is set, or if no number is designated, 0 will be set to indicate that there is no hand system flag.

1: Right-handed system is used to move to a specified position.
2: Left-handed system is used to move to a specified position.

Direct numeric value inputs can be used to set the X-arm and Y-arm rotation information (*1) only on R6YXTW500 model robots where the coordinate system-of-units has been set as "mm".
To set extended X-arm and Y-arm rotation information at the R6YXTW500 model robot, a "-1", "0", or "1" value must be specified at F1 and F2. Any other value, or no X-arm and Y-arm rotation information at all, will be processed as "0".
0: Indicates arm rotation information where movement to the "0" position has been specified.
1: Indicates arm rotation information where movement to the "1" position has been specified.
-1: Indicates arm rotation information where movement to the "-1" position has been specified.

*1: For details regarding the X-arm and Y-arm rotation information, refer to Chapter 4 "3. Point data format".

NOTE
- If both integers and real numbers are used together (mixed), all coordinate values will be handled in "mm/deg" units.
- X-arm and Y-arm rotation information is only available in software Ver.1.66M or higher.

CAUTION
- When performing linear interpolation with a hand system flag specified, be sure that the same hand system is used at the current position and target position. If the same hand system is not used, an error will occur and robot movement will be disabled.
- When performing a linear interpolation, the current position's X-arm and Y-arm rotation information must be the same as the movement destination's X-arm and Y-arm rotation information. If the two are different, an error will occur and movement will be disabled.

MEMO
- At SCARA robots with a hand system flag set in the movement destination's coordinate data, the specified hand system will have priority over the current arm type or LEFTY/RIGHTY setting.

SAMPLE
MOVE P,10000 10000 1000 1000 0 0 ....PTP movement occurs from current position to the specified position.

CAUTION

- When moving the robot by linear or circular interpolation to a point where a hand system flag is specified, be sure that the same hand system is used at both the current and target positions. If the same hand system is not used, an error will occur and robot movement will be disabled.

● Point definition PTP linear interpolation Circular interpolation

Format
<point expression>[,<point expression>...]

Explanation Specifies a <point expression>. Two or more data items can be designated by separating them with a comma (,).
Circular interpolation must be specified by an even number of points.

MEMO

- At SCARA robots with a hand system flag set in the movement destination's coordinate data, the specified hand system will have priority over the current arm type or LEFTY/RIGHTY setting.

CAUTION

- When performing a linear and circular interpolation, the current position's X-arm and Y-arm rotation information must be the same as the movement destination's X-arm and Y-arm rotation information. If the two are different, an error will occur and movement will be disabled.

SAMPLE

```
MOVE  P,P1 .................................................Moves from the current position to the position specified
                                                            by P1.
MOVE  P,P20,P0,P100..................................Moves in sequence from the current position to positions
                                                            specified by P20, P0, P100.
```

58 MOVE

Option types

● Speed setting 1 PTP linear interpolation Circular interpolation

Format
1. SPEED = <expression> 2. S = <expression>

NOTE
- This option specifies only the maximum speed and does not guarantee movement at the specified speed.

Values <expression> ....1 to 100 (units: %)

Explanation Specifies the program speed in an <expression>.
The actual speed will be as follows:
- [Robot max. speed (mm/sec)] × [automatic movement speed (%)] × [program movement speed (%)].

This option is enabled only for the specified MOVE statement.

SAMPLE

```
MOVE  P,P10,S=10.......................................Moves from the current position to the position specified
                                                        by P10, at 10% of the program movement speed.
```

● Speed setting 2 PTP linear interpolation Circular interpolation

Format
VEL = <expression>

NOTE
- This option specifies only the maximum composite speed and does not guarantee movement at the specified speed.

Values <expression> ....For SCARA robot: 1 to 750

Explanation Specifies the maximum composite speed (in "mm/sec" units) of the XYZ axes in an <expression>. This option can be used for linear interpolation and circular interpolation movements of SCARA robots.
This option is enabled only for the specified MOVE statement.

SAMPLE

```
MOVE L,P10,VEL=100.................................Moves from the current position to the position specified
                                                    by P10, at the XYZ maximum composite speed of 100mm/
                                                    sec.
```

● Arch motion setting **PTP** linear interpolation Circular interpolation

Format
x = <expression>

Values x..Specifies the X,Y,Z,R,A,B axis.
<expression> ....An integer value is processed in "pulse" units.
A real number (with decimal point) is process in "mm/deg" units.

Explanation
1. The "x" specified axis begins moving toward the position specified by the <expression> (see "1" in the Fig. below).
2. When the "x" specified axis enters the arch position range, all other axes move toward the target position (see "2" in the Fig. below).
3. When all axes other than the "x" specified axis enter the arch position range, the "x" specified axis moves to the target position ("3" in the Fig. below).
4. The command ends when all axis enter the OUT position range.

This option can be used only for PTP movement.
If the "x" specified axis is the X or Y axis, the target position and <expression> must be specified as an integer (pulse units) for SCARA robots

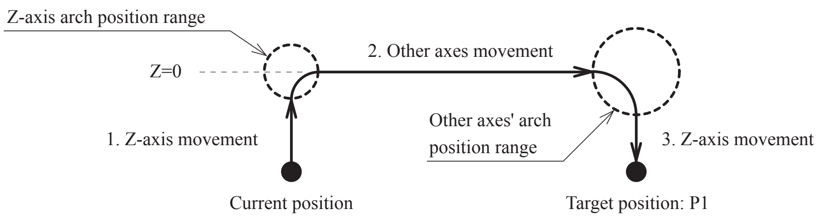
<expression> value: SCARA type robots.
Target position value: SCARA type robot.

The values are indicated as the motor position rather than the coordinate values.

SAMPLE

MOVE P,P1,Z=0...Z-axis moves 0 pulses from its current position, then other axes move to P1. Finally, the Z-axis moves to P1.

SAMPLE: MOVE Z



58 MOVE

● STOPON condition setting PTP linear interpolation Circular interpolation

Format
STOPON <conditional expression>

Explanation Stops movement when the conditions specified by the conditional expression are met. Because this is a deceleration type stop, **there will be some movement (during deceleration) after the conditions are met.**

If the conditions are already met before movement begins, no movement occurs, and the command is terminated.

This option can only be used for PTP movement and linear interpolation movement.

This option is only possible by program execution.

SAMPLE

```
MOVE P,P100,STOPON DI(20)=1
```

..............................Moves from the current position to the position specified by P100. If the "DI (20) = 1" condition is met during movement, a deceleration and stop occurs, and the next step is then executed.

MEMO

- When the conditional expression used to designate the STOPON condition is a numeric expression, the conditions for determining a TRUE or FALSE status can be changed at the controller's "TRUE conditions" in the "Other parameters" mode. These conditions apply to all the IF, WHILE, WAIT, STOPON, etc., conditional expressions. For details, refer to the controller user's manual.

1) -1 (default setting)	An expression value of "-1" indicates a TRUE status, and "0" indicates a FALSE status. A "6.35 Incorrect condition expression" error occurs if the expression value is other than "-1" or "0".
2) not 0	Any expression value other than "0" indicates a TRUE status, and "0" indicates a FALSE status.

● CONT setting .... PTP linear interpolation Circular interpolation

Format
CONT

NOTE

- The CONT setting can be used to reduce the PTP movement START positioning time.
- The path to the target point is not guaranteed.

Explanation When PTP movement is executed with the CONT setting option, the PTP movement which begins immediately after all movable axes enter the OUT position range (with the command being terminated at that point), will begin without waiting for the movable axes to complete their movement into the tolerance range.

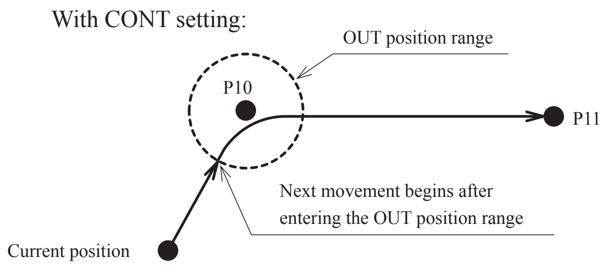
This option can be used only for PTP movement and is enabled only for the specified MOVE statement.

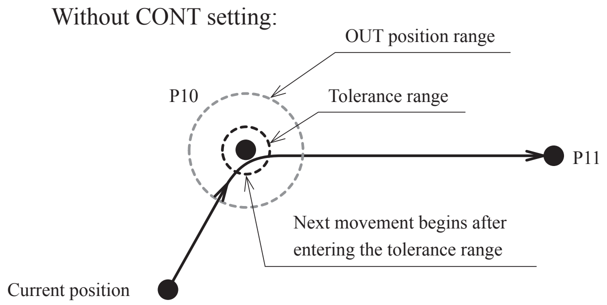
SAMPLE

MOVE P,P10,P11,CONT

..............................Moves from the current position to the position specified by P10, and then moves to P11 without waiting for the moving axes to arrive in the tolerance range.

SAMPLE: MOVE CONT





58 MOVE

● Acceleration setting   PTP **linear interpolation** Circular interpolation

Format

```
ACC = <expression>
```

Values <expression> ....1 to 100 (units: %)

Explanation Specifies the robot acceleration rate in an <expression>. The actual robot acceleration is determined by the acceleration coefficient parameter setting.
This option can be used only for linear interpolation movement and is enabled only for the specified MOVE statement.

SAMPLE

MOVE L,P100,ACC=10..............................Moves at an acceleration rate of 10% from the current position to the position specified by P100.

● Deceleration setting   PTP **linear interpolation** Circular interpolation

Format

```
DEC = <expression>
```

Values <expression> ....1 to 100 (units: %)

Explanation Specifies the robot deceleration rate in an <expression>. The actual robot deceleration is determined by the acceleration coefficient parameter setting (the setting is specified as a percentage of the acceleration setting value (100%)).
This option can be used only for linear interpolation movement and is enabled only for the specified MOVE statement.

SAMPLE

MOVE L,P100,DEC=20.........................Moves at a deceleration rate of 20% from the current position to the position specified by P100.

- Coordinate plane setting PTP linear interpolation **Circular interpolation**

Format
| XY | | YZ | | ZX |

Values XY ....XY coordinate plane
YZ...YZ coordinate plane
ZX...ZX coordinate plane

NOTE

- If no coordinate plane is specified, the robot moves along a 3-dimensional circle.
- When a 2-axis robot is used, the robot moves along a circle on the XY plane.

Explanation When circular interpolation is executed by setting coordinates, this option executes circular interpolation so that the projection on the specified coordinate plane becomes a circle.
This option can be used for circular interpolation movement and is enabled only for the specified MOVE statement.

SAMPLE

```
P10 =  100.00 100.00 20.00 0.00 0.00 0.00
P11 =  150.00 100.00  0.00 0.00 0.00 0.00
P12 =  150.00 150.00 20.00 0.00 0.00 0.00
P13 =  100.00 150.00 40.00 0.00 0.00 0.00
MOVE P,P10
MOVE C,P11,P12
MOVE C,P13,P10........................................Moves continuously along a 3-dimensional circle generated
                                                         at P10, P11, P12, and P12, P13, P10.
MOVE C,P11,P12,XY
MOVE C,P13,P10,XY.....................................Moves continuously along a circle on an XY plane generated at P10, P11, P12, and P12,
                                                         P13, P10. Z-axis moves to the position specified by P12 and P10 (the circle's target
                                                         position).
```

● Port output setting **PTP** **linear interpolation** **Circular interpolation**

Format 1

```
|DO|m([b,.....,b])=<expression 1>@<expression 2>
|MO|
|SO|
```

Format 2

```
|DO|(mb,.....,mb)=<expression 1>@<expression 2>
|MO|
|SO|
```

Values

m: port number ....2 to 7, 10 to 17, 20 to 27

b: bit definition ....0 to 7

<expression 1> ....Value which is output to the specified port (only integers are valid).

<expression 2> ....Position where the port output occurs. This position can be specified in "mm" units down to the 2nd decimal position.

Explanation

During linear interpolation or circular interpolation movement, this command option outputs the value of <expression 1> to the specified port when the robot reaches the <expression 2> distance (units: "mm") from the start position.

The <expression 2> numeric value represents a circle radius centered on the movement START point.

This command option can only be used with linear or circular interpolation movement, and it can be specified no more than 2 times per each MOVE statement.

If multiple bits are specified, they are expressed from the left in descending order (large to small).

If the [b,…,b] data is omitted in format 1, all 8 bits are processed.

If no hardware port exists, nothing is output.

CAUTION

- Output to ports "0" and "1" is not allowed at DO, MO, and SO.

REFERENCE

- For bit setting details, see Chapter 3 "10 Bit Settings".

SAMPLE 1

```
MOVE P,P0
MOVE L,P1,DO2()=105@25.85
          ..............................During linear interpolation movement to P1, 105 (&B01101001) is output to DO2()
                                        when the robot reaches a distance of 25.85mm from P0.
```

SAMPLE 2

```
A!=10
B!=20
MOVE L,P2,MO(22)=1@A!,MO(22)=0@B!
          .............................After movement START toward P2, MO(22) switches ON when the robot
                                       has moved a distance of 10mm, and switches OFF when the robot has
                                       moved a distance of 20mm.
```

Related commands	MOVEI, DRIVE, DRIVEI, WAIT ARM

59 MOVEI

Performs absolute movement of all robot axes

Format

MOVEI | PTP / P | ,<point definition> [, option [, option]...]

Explanation Executes relative position movement commands for the robot.
It is not enabled for other groups, or for auxiliary axes.

- Movement type : PTP
- Point data setting : Direct coordinate data input, point definition.
- Options : Speed setting

Options	PTP	Linear interpolation	Arch interpolation	Remarks
Speed setting (SPEED)	○	○	○	Enabled only for specified MOVE statement

MEMO

- If the MOVEI statement is interrupted and then re-executed, the movement target position can be selected at the "MOVEI/DRIVEI start position" setting at "Other parameters" in the controller. For details, refer to the controller user's manual.
  1) KEEP (default setting) Continues the previous (before interruption) movement. The original target position remains unchanged.
  2) RESET Relative movement begins anew from the current position. The new target position is different from the original one (before interruption). (Backward compatibility)

Movement type

● PTP (point-to-point) movement

Execution START condition: Movement of all specified axes is complete (within the tolerance range).
Execution END condition: All specified axes have entered the OUT position range.
When two or more axes are specified, they will reach their target positions simultaneously. The movement path of the axes is not guaranteed.

● Caution regarding commands which follow the MOVEI command:

If the next command following the MOVEI command is an executable command such as a signal output command, that next command will start when the movement axis enters the OUT position range. In other words, that next command starts before the axis arrives within the target position tolerance range.

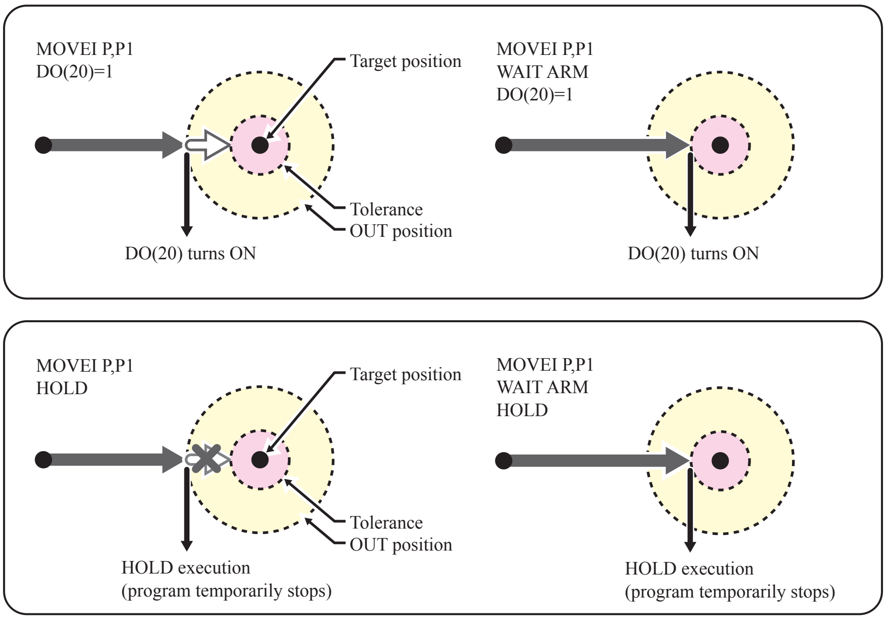
59 MOVEI

Example:

Signal output (DO, etc.)	Signal is output when axis enters within OUT position range.
DELAY	DELAY command is executed and standby starts, when axis enters the OUT position range.
HALT	Program stops and is reset when axis enters the OUT position range. Therefore, axis movement also stops.
HOLD	Program temporarily stops when axis enters the OUT position range. Therefore, axis movement also stops.
WAIT	WAIT command is executed when axis enters the OUT position range.

The WAIT ARM statement is used to execute the next command after the axis enters the tolerance range.

MOVEI command



SAMPLE

MOVEI P,P0 ....From its current position, the main robot axis moves (PTP movement) the amount specified by P0.

Point data setting types

● Direct numeric value input **PTP**

Format
X Y Z R A B [F] [F1] [F2]

Values X, Y, Z, R, A, B ....Space-separated coordinate values for each axis.
F..Hand system flag
F1..X-arm rotation information (R6YXTW500 model only).
F2..Y-arm rotation information (R6YXTW500 model only).

NOTE

- If both integers and real numbers are used together (mixed), all coordinate values will be handled in "mm/deg" units.
- X-arm and Y-arm rotation information is only available in software Ver.1.66M or higher.

Explanation Directly specifies coordinate values by a numeric value. If an integer is used, this is interpreted as "pulse" units, and if a real number is used, this is interpreted as "mm/deg" units, with movement occurring accordingly.
Hand system flags can be specified for SCARA robots when directly specifying the coordinate values in "mm" units.
To specify an extended hand system flag for SCARA robots, set either 1 or 2 at "F". If a number other than 1 or 2 is set, or if no number is designated, 0 will be set to indicate that there is no hand system flag.

1: Right-handed system is used to move to a specified position.
2: Left-handed system is used to move to a specified position.

Direct numeric value inputs can be used to set the X-arm and Y-arm rotation information (*1) only on R6YXTW500 model robots where the coordinate system-of-units has been set as "mm".
To set extended X-arm and Y-arm rotation information at the R6YXTW500 model robot, a "-1", "0", or "1" value must be specified at F1 and F2. Any other value, or no X-arm and Y-arm rotation information at all, will be processed as "0".

0: Indicates arm rotation information where movement to the "0" position has been specified.
1: Indicates arm rotation information where movement to the "1" position has been specified.
-1: Indicates arm rotation information where movement to the "-1" position has been specified.

*1: For details regarding the X-arm and Y-arm rotation information, refer to Chapter 4 "3. Point data format".

MEMO

- At SCARA robots with a hand system flag set in the movement destination's coordinate data, the specified hand system will have priority over the current arm type or LEFTY/RIGHTY setting.

SAMPLE

MOVEI P, 10000 10000 1000 1000 0 0
.............................From its current position, the axis moves (PTP movement) the specified amount (pulse units).

59 MOVEI

● Point definition **PTP**

Format
<point expression>[,<point expression>...]

Explanation Specifies a <point expression>. Two or more data items can be designated by separating them with a comma (,).

MEMO

- At SCARA robots with a hand system flag set in the movement destination's coordinate data, the specified hand system will have priority over the current arm type or LEFTY/RIGHTY setting.

SAMPLE
MOVEI P,P1 From its current position, the axis moves (PTP movement) the amount specified by P1.

Option types

● Speed setting PTP

Format

1. SPEED = <expression>
2. S = <expression>

Values <expression> ....1 to 100 (units: %)

Explanation Specifies the program speed in an <expression>.
The actual speed will be as follows:
• [Robot max. speed (mm/sec)] × [automatic movement speed (%)] × [program movement speed (%)].
This option is enabled only for the specified MOVEI statement.

NOTE
• This option specifies only the maximum speed and does not guarantee movement at the specified speed.

SAMPLE

MOVEI P,P10,S=10..................................From its current position, the axis moves (PTP movement) the amount specified by P1, at 10% of the program movement speed.

Related commands	MOVE, DRIVE, DRIVEI, WAIT ARM

60 OFFLINE

Sets a specified communication port to the "offline" mode

Format

```
OFFLINE     [ |ETH | ]
            [ |CMU | ]
```

Values <expression> ....ETH, CMU, or no setting

Explanation Changes the communication mode parameter in order to switch the communication mode to OFFLINE.

ETH .... Changes the Ethernet communication mode parameter to OFFLINE and clears the transmission and reception buffers.

CMU or no setting .... Changes the RS-232C communication mode parameter to OFFLINE, resets the communication error, and clears the reception buffer.

SAMPLE

```
OFFLINE
SEND CMU TO A$
SEND CMU TO P10
ONLINE
HALT
```

7

61 ORD

Acquires a character code

Format
ORD (<character string expression>)

Explanation Acquires the character code of the first character in a <character string expression>.

SAMPLE
A=ORD("B")   66 (=&H42) is assigned to A.

Related commands	CHR$

N O P Q R S T U V W X Y Z

62 ON ERROR GOTO

Jumps to a specified label when an error occurs

Format

1. ON ERROR GOTO <label>
2. ON ERROR GOTO 0

Values Error output information ....ERR: Error code number
ERL: Line number where error occurred

Explanation Even if an error occurs during execution of the robot language, this statement allows the program to jump to the error processing routine specified by the <label>, allowing the program to continue without being stopped (this is not possible for some serious errors.)
If "0" is specified instead of the <label>, the program stops when an error occurs, and an error message displays.
If ON ERROR GOTO "0" is executed at any place other than an error processing routine, the ON ERROR GOTO command is canceled (interruption canceled).
The error processing routine can process an error using the RESUME statement and the error output information (ERR, ERL).

MEMO

- If a serious error such as "17.4: Overload" occurs, the program execution stops.
- The most recently executed "ON ERROR GOTO <label>" statement is valid.
- If an error occurs during an error processing routine, the program will stop.
- "ON ERROR GOTO <label>" statements cannot be used within error processing routines.

SAMPLE

```
ON ERROR GOTO *ER1
FOR A = 0 TO 9
    P[A+10] = P[A]
NEXT A
*L99: HALT
'ERROR ROUTINE
*ER1:
IF ERR = &H0604 THEN *NEXT1.....................Checks to see if a "Point doesn't exist" error has occurred.
IF ERR = &H0606 THEN *NEXT2......................Checks to see if a "Subscript out of range" error has
                                                 occurred.
ON ERROR GOTO 0............................................Displays the error message and stops the program.
*NEXT1:
    RESUME NEXT.........................................Jumps to the next line after the error line and resumes
                                                 program execution.
*NEXT2:
    RESUME *L99............................................Jumps to label *L99 and resumes program execution.
```

Related commands	RESUME

63 ON to GOSUB

Executes the subroutine specified by the <expression> value

Format

```
ON<expression>GOSUB<label 1>[,<label 2>...]
```

* GOSUB can also be expressed as "GO SUB".

Values <expression> ....0 or positive integer

Explanation The <expression> value determines the program's jump destination.
An <expression> value of "1" specifies a jump to <label 1>, "2" specifies a jump to <label 2>, etc.
Likewise, (<expression> value "n" specifies a jump to <label n>.)
If the <expression> value is "0" or if the <expression> value exceeds the number of existing labels, no jump occurs, and the next command is executed.
After executing a jump destination subroutine, the next command after the ON to GOSUB statement is executed.

SAMPLE

```
'MAIN ROUTINE
*ST:
ON DI3() GOSUB *SUB1,*SUB2,*SUB3............*SUB1 to *SUB3 are executed.
GOTO *ST      .....................................................Returns to *ST.
HALT
'SUB ROUTINE
*SUB1:
      MOVE P,P10,Z=0
      RETURN
*SUB2:
      DO(30) = 1
      RETURN
*SUB3:
      DO(30) = 0
      RETURN
```

Related commands	GOSUB, RETURN, DECLARE

64 ON to GOTO

Jumps to the label specified by the <expression> value

Format

ON<expression>GOTO<label 1>[,<label 2>...]
* GOTO can also be expressed as "GO TO".

Values <expression> ....0 or positive integer

Explanation The <expression> value determines the program's jump destination.
An <expression> value of "1" specifies a jump to <label 1>, "2" specifies a jump to <label 2>, etc.
Likewise, (<expression> value "n" specifies a jump to <label n>.)
If the <expression> value is "0" or if the <expression> value exceeds the number of existing labels, no jump occurs, and the next command is executed.

SAMPLE

```
'MAIN ROUTINE
*ST:
ON DI3() GOTO *L1,*L2,*L3........................Jumps to *L1 to *L3 in accordance
                                                 with the DI3() value.
GOTO *ST      ...................................Returns to *ST.
HALT
'SUB ROUTINE
*L1:
    MOVE P,P10,Z=0
    GOTO *ST
*L2:
    DO(30) = 1
    GOTO *ST
*L3:
    DO(30) = 0
    GOTO *ST
```

Related commands	GOTO, DECLARE

65

ONLINE

Sets the specified communication port to the "online" mode

Format

```
ONLINE    |[ ETH ] |
          |[ CMU ] |
```

Values <expression> ....ETH, CMU, or no setting

Explanation Changes the communication mode parameter in order to switch the communication mode to ONLINE.

ETH ....Changes the Ethernet communication mode parameter to ONLINE and clears the transmission and reception buffers.

CMU or no setting........Changes the RS-232C communication mode parameter to ONLINE, resets the communication error, and clears the reception buffer.

SAMPLE

```
OFFLINE
SEND CMU TO A$
SEND CMU TO P10
ONLINE
HALT
```

66 ORGORD

Specifies/acquires the robot's return-to-origin sequence

Format
ORGORD <expression>

Values <expression> ....main group: n to nnnnnn (n : 0 to 6)

Explanation Sets the axis sequence parameter for the robot's return-to-origin and absolute search operations.

The 1 to 6 axes are expressed as "1 to 6" values, respectively, and the <expression> value must be 1-digit to 6-digit integer.

The same axis cannot be specified twice.

After the specified axes are returned to their origin points in sequence, from left to right, the remaining axes return to their origin points simultaneously.

If the <expression> value is "0", all axes will be returned to their origin points simultaneously.

Functions

Format
ORGORD

Explanation Acquires the axis sequence parameter for return-to-origin and absolute search operations.

SAMPLE

```
A=3
ORGORD A    .......................................................A return-to-origin is executed first for axis 3.
ABSRST      .......................................................After the main group's axis 3 return-to-origin is completed, a return-to-
                                                                   origin is executed for the remaining axes, followed by an absolute reset.
MOVE P,P0
A=ORGORD    ...................................................The main group's return-to-origin sequence parameter is
                                                                   assigned to variable A.
HALT
```

Related commands	ABSRST, ORIGIN

67 ORIGIN

Performs an incremental mode axis return-to-origin

Format
ORIGIN

Explanation This statement performs a return-to-origin for an incremental mode axis, or an absolute search for a semi-absolute axis.

If the movement is stopped at an intermediate point, an "incomplete return-to-origin" status will occur.

SAMPLE	
ORIGIN	....Performs an incremental mode return-to-origin.

Related commands	ABSRST, ORGORD, MCHREF

A value other than "0" must be set for the execution <level> in order to execute the ORIGIN command. For details regarding how to check and change the execution <level> value, see section 13 "3.10 Setting the UTILITY mode", and section 15 "6 Execution Level".

68 OUT

Turns ON the specified port output

Format

OUT	DOm([b,.....,b]) DO(mb,.....,mb) MOm([b,.....,b]) MO(mb,.....,mb) SOm([b,.....,b]) SO(mb,.....,mb) LO0([b,.....,b]) LO(0b,.....,0b) TO0([b,.....,b]) TO(0b,.....,0b)	[, <time>]

Values
m: port number ....2 to 7, 10 to 17, 20 to 27
b: bit definition ....0 to 7
<expression> ....1 to 3600000 (units: ms)

CAUTION

- Output to ports "0" and "1" are not allowed at DO, MO, and SO.

REFERENCE

- For bit setting details, see Chapter 3 "10 Bit Settings".

Explanation This statement turns ON the specified port output and terminates the command. (The program proceeds to the next line.) Output to that port is then turned OFF after the time specified by the <expression> has elapsed. If the operation is stopped temporarily at an intermediate point and then restarted, that port's output is turned OFF when the remaining <expression> specified time has elapsed.

<expression> values are rounded downward to the nearest even 10 (e.g., 113 → 110). Or, if a value is less than 10, it becomes 10.
If this <expression> is omitted, the specified port's output remains ON.
Up to 16 OUT statements using <expressions> can be executed at the same time. Attempting to execute 17 or more OUT statements will activate error "6.26: Insufficient memory for OUT".

If multiple bits are specified, they are expressed from the left in descending order (large to small).
If no hardware port exists, nothing is output.

SAMPLE

```
OUT DO2(),200....................................................Turns DO(27 to 20) ON, then turns them OFF 200ms
                                                                  later.
OUT DO(37,35,27,20)..........................................Turns DO(37, 35, 27, 20) ON.
```

Related commands	DO, MO, SO, TO, LO

69 OUTPOS

Specifies/acquires the OUT enable position parameter of the robot

Format

1. OUTPOS <expression>
2. OUTPOS (<axis number>)=<expression>

Values <axis number>............................main group: 1 to 6
<expression> ....1 to 6144000 (Unit: pulses)

Explanation Changes the parameter's OUT position to the value indicated by the <expression>.
Format 1: The change is applied to the group axes.
Format 2: The change is applied only to the axis specified by <axis number>.

MEMO

- If an axis that is set to "no axis" in the system generation is specified, a "5.37 : Specification mismatch" error message displays and command execution is stopped.

Functions

Format

OUTPOS(<axis 1>)

Values <axis 1>.....................................main group: 1 to 6

Explanation Acquires the OUT position parameter value for the axis specified by the <expression>.

MEMO

- If an axis that is set to "no axis" in the system generation is specified, a "5.37 : Specification mismatch" error message displays and command execution is stopped.

69 OUTPOS

SAMPLE

```
'CYCLE WITH DECREASING OUTPOS
DIM SAV(3)
GOSUB *SAVE_OUTPOS
FOR A=1000  TO 10000  STEP 1000
    GOSUB *CHANGE_OUTPOS
    MOVE P,P0
    DO3(0)=1
    MOVE P,P1
    DO3(0)=0
NEXT A
GOSUB *RESTORE_OUTPOS
HALT
*CHANGE_OUTPOS:
    FOR B=1 TO 4
        OUTPOS(B)=A
    NEXT B
    RETURN
*SAVE_OUTPOS:
    FOR B=1 TO 4
        SAV(B-1)=OUTPOS(B)
    NEXT B
    RETURN
*RESTORE_OUTPOS:
    FOR B=1 TO 4
        OUTPOS(B)=SAV(B-1)
    NEXT B
    RETURN
```

70 PATH

Specifies the main robot axis PATH motion path

Format		
PATH	L C	,<point definition> [, option [, option...]]

Explanation Sets the PATH motion path for the main robot axis. This command can only be executed between the PATH SET and PATH END commands. If execution is attempted elsewhere, an error will occur.

- Movement type: Linear interpolation and circular interpolation.
- Point setting: By direct numeric value input and by point definition.
- Options: Speed setting, coordinate plane setting (for circular interpolation only), and port output setting.

PATH motion types

- Linear interpolation movement

"PATH L…" is set for linear interpolation movement.

- Circular interpolation movement

"PATH C…" is set for circular interpolation movement.

Only the X, Y and Z coordinate values of the specified points are valid for PATH motion. Any other coordinates use the coordinate values of the PATH motion START point.
The motion path can be connected by repeated PATH commands ("PATH L", "PATH C") to allow movement without stopping.

70 PATH

Point data setting types

● Direct numeric value input **linear interpolation** Circular interpolation

NOTE

- X-arm and Y-arm rotation information is only available in software Ver.1.66M or higher.

Format
X Y Z R A B [F] [F1] [F2]

Values X, Y, Z, R, A, B ....Space-separated coordinate values for each axis.
F..Hand system flag.
F1..X-arm rotation information (R6YXTW500 model only).
F2..Y-arm rotation information (R6YXTW500 model only).

Explanation Directly specifies coordinate data by a numeric value. **If an integer is used, this is interpreted as "pulse" units, and if a real number (with decimal point) is used, this is interpreted as "mm" units. If both integers and real numbers are used together (mixed), all coordinate values will be handled in "mm" units.**

With this format, only 1 point can be specified as the movement destination coordinates. The only type of movement specified by this point data setting is linear interpolation.

Hand system flags can be specified for SCARA robots when directly specifying the coordinate data in "mm" units.
To specify an extended hand system flag for SCARA robots, set either 1 or 2 at "F". If a number other than 1 or 2 is set, or if no number is set, 0 will be set to indicate that there is no hand system flag.

1 : Right-handed system is used to move to a specified position.
2 : Left-handed system is used to move to a specified position.

Direct numeric value inputs can be used to set the X-arm and Y-arm rotation information (*1) only on R6YXTW500 model robots where the coordinate system-of-units has been set as "mm".
To set extended X-arm and Y-arm rotation information at the R6YXTW500 model robot, a "-1", "0", or "1" value must be specified at "F1" and "F2". Any other value, or no X-arm and Y-arm rotation information at all, will be processed as "0".
0: Indicates arm rotation information where movement to the "0" position has been specified.
1: Indicates arm rotation information where movement to the "1" position has been specified.
-1: Indicates arm rotation information where movement to the "-1" position has been specified.

*1: For details regarding the X-arm and Y-arm rotation information, refer to Chapter 4 "3. Point data format".

The same hand system must always be used between a motion path's START and END points. The hand system cannot be changed between these points.

CAUTION

- The hand system used during PATH motion must be the same as the hand system used at the path motion route's start point. The same applies if the path is to pass through points where hand system flags are set. Differing hand systems will cause an error and disable motion.
- The X-arm and Y-arm rotation information during PATH movement must be the same as the X-arm and Y-arm rotation information at the PATH movement's START point. If the two are different, an error will occur and movement will be disabled.

Moreover, the X-arm and Y-arm rotation information must be the same throughout the movement path, from the path's START to END points. The X-arm and Y-arm rotation information cannot be changed at any point along the path.

MEMO

• At SCARA robots with a hand system flag set in the movement destination's coordinate data, the specified hand system will have priority over the current arm type or LEFTY/RIGHTY setting.

SAMPLE

PATH L,10000 10000 1000 1000 0 0
............................The target position is set in "pulse" units, and linear interpolation movement occurs.
PATH L,150.00 250.00 10.00 30.00 0.00 0.00 1
............................The target position is set in the coordinate values specified by the right-handed system, and linear interpolation movement occurs.

CAUTION

• The hand system used during PATH motion must be the same as the hand system used at the path motion route's start point. The same applies if the path is to pass through points where hand system flags are set. Differing hand systems will cause an error and disable motion.

● Point definition **linear interpolation** **Circular interpolation**

Format

<point definition> [,<point definition>...]

Explanation Specifies the movement destination as <point expression> value. Two or more data items can be designated by separating them with a comma (,).
For circular interpolation movement, 2 points must be specified for each arc.

MEMO

• At SCARA robots with a hand system flag set in the movement destination's coordinate data, the specified hand system will have priority over the current arm type or LEFTY/RIGHTY setting.

CAUTION

• The X-arm and Y-arm rotation information during PATH movement must be the same as the X-arm and Y-arm rotation information at the PATH movement's START point. If the two are different, an error will occur and movement will be disabled.

SAMPLE

PATH L,P1,P2,P3......................................Specifies sequential linear interpolation movement to positions specified by P1, P2 and P3.
PATH C P5,P6,P7,P8................................Specifies circular interpolation movement through the following points: current position, P5, P6, and P6, P7, P8.

70 PATH

Option types

● Speed setting **linear interpolation** **Circular interpolation**

Format

1. SPEED = <expression>
2. S = <expression>

> **NOTE**
> • This defines the maximum speed, and does not guarantee that all movement will occur at specified speed.

Values <expression> ....1 to 100 (units: %)

Explanation The program's movement speed is specified as the <expression> value (units: %).
The actual speed is determined as shown below.

• Robot's max. speed (mm/sec) × automatic movement speed (%)× program movement speed (%).

This option is enabled only for the specified PATH statement.

SAMPLE

PATH L,P5,S=40...Movement to the position specified by P5 occurs at 40% of the program movement speed.

Format

VEL = <expression>

> **NOTE**
> • This option specifies only the maximum composite speed and does not guarantee movement at the specified speed.

Values <expression> ....The permissible setting range varies according to the robot type (units: mm/sec).

Explanation The movement speed is specified by the <expression> value (units: mm/sec). An error will occur if the speed is too fast.
This command is enabled only for the specified PATH statement.

SAMPLE

PATH L,P10,VEL=150.................................Movement to the position specified by P10 occurs at a speed of 150mm/sec.

● Coordinate plane setting

linear interpolation Circular interpolation

Format

| XY |
| YZ |
| ZX |

Values
XY ....XY coordinate plane
YZ...YZ coordinate plane
ZX...ZX coordinate plane

Explanation Specifies the coordinate plane on which to draw a circular arc for circular interpolation movement. If no coordinate plane is specified, 3-dimensional circular interpolation movement is used.
Only circular interpolation movement can be specified by this coordinate plane setting.
This command is enabled only for the specified PATH statement.

SAMPLE

PATH C,P1,P2,XY..Circular interpolation movement occurs within the XY plane, with the Z-axis moving to the P2 Z-axis coordinates position.

● Port output setting **linear interpolation** **Circular interpolation**

Format 1

```
|DO |  m ([b,.....,b])=<expression 1> @<expression 2>
|MO |
|SO |
```

Format 2

```
| DO | (mb,.....,mb)=<expression 1> @<expression 2>
| MO |
| SO |
```

Values

m: port number ....2 to 7, 10 to 17, 20 to 27
b: bit definition ....0 to 7
<expression 1> ....Value which is output to the specified port (only integers are valid).
<expression 2> ....Position where the port output occurs. This position can be specified in "mm" units down to the 2nd decimal position.

CAUTION

- Output to ports "0" and "1" is not allowed at DO, MO, and SO.

REFERENCE

- For details regarding bit definitions, see Chapter 3 "10 Bit Settings".

Explanation

During PATH motion, this command option outputs the value of <expression 1> to the specified port when the robot reaches the <expression 2> distance from the start position.
The <expression 2> numeric value represents a circle radius centered on the movement START point.
If multiple bits are specified, they are expressed from the left in descending order (large to small). If the [b,…,b] data is omitted in format 1, all 8 bits are processed.
If no hardware port exists, nothing is output.

SAMPLE

```
PATH SET
PATH L,P1,DO(20)=1@10.................................During linear interpolation movement to P1, "1" is output to
                                                      DO(20) at a 10mm radius position from the START position.
PATH L,P2,DO(21)=1@12.5...........................During linear interpolation movement to P2, "1" is
                                                      output to DO(21) at a 12.5mm radius position from
                                                      P1.
PATH END
PATH START
```

Related commands	PATH SET, PATH END, PATH START

Reference For PATH function details, see Chapter 9 "PATH Statements".

71 PATH END

Ends the movement path setting

Format
PATH END

Explanation Ends the path setting for PATH motion.

The PATH END command must always be paired with a PATH SET command. The PATH motion path end-point is the final point specified by the final PATH command (PATH L, PATH C) which exists between the PATH SET and PATH END commands.

Attempting to execute a PATH END command when no PATH SET command has been executed will result in an error.

Related commands	PATH, PATH SET, PATH START

Reference For PATH function details, see Chapter 9 "PATH Statements".

72 PATH SET

Starts the movement path setting

Format
PATH SET [<point definition>]

Explanation Starts the path setting for PATH motion.

Specifies the <point definition> position as the PATH motion start-point. (This only sets the PATH motion start point and does not actually begin robot motion.) If the <point definition> value is omitted, the current robot position is set as the start point.

However, if robot movement is in progress, the target position of that movement becomes the start point. (Example: The OUT position range is wider for the MOVE command which precedes the PATH SET command, so the robot is still moving when the PATH SET command is executed.)

The PATH SET command must always be paired with a PATH END.

When a PATH SET command is executed, the previously set PATH motion path data is deleted.

• Point data setting : By direct numeric value input and by point definition

Point data setting types

● Direct numeric value input

Format
X Y Z R A B [F] [F1] [F2]

Values

X, Y, Z ,R, A, B ....Space-separated coordinate values for each axis.

F..Hand system flag.

F1..X-arm rotation information (R6YXTW500 model only).

F2..Y-arm rotation information (R6YXTW500 model only).

Explanation Directly specifies the path's start-point coordinates for PATH motion. If an integer is used, this is interpreted as "pulse" units, and if a real number is used, this is interpreted as "mm" units (valid down to the 2nd decimal position).

Hand system flags can be specified for SCARA robots when directly specifying the coordinate data in "mm" units.

To specify an extended hand system flag for SCARA robots, set either 1 or 2 at "F". If a number other than 1 or 2 is set, or if no number is set, 0 will be set to indicate that there is no hand system flag.

1: Indicates that a right-handed system is specified for the PATH motion's start-point.

2: Indicates that a left-handed system is specified for the PATH motion's start-point.

Direct numeric value inputs can be used to set the X-arm and Y-arm rotation information (*1) only on R6YXTW500 model robots where the coordinate system-of-units has been set as "mm".

To set extended X-arm and Y-arm rotation information at the R6YXTW500 model robot, a "-1", "0", or "1" value must be specified at F1 and F2. Any other value, or no X-arm and Y-arm rotation information at all, will be processed as "0".

NOTE

- If both integers and real numbers are used together (mixed), all coordinate values will be handled in "mm/deg" units.
- X-arm and Y-arm rotation information is only available in software Ver.1.66M or higher.

CAUTION

- The hand system used during PATH motion must be the same hand system as that at the PATH motion's start-point. An error will occur if the hand systems are different.
- The X-arm and Y-arm rotation information during PATH movement must be the same as the X-arm and Y-arm rotation information at the PATH movement's START point. If the two are different, an error will occur and movement will be disabled.

0: Indicates that the PATH movement START point's arm rotation information has been set at the "0" position.

1: Indicates that the PATH movement START point's arm rotation information has been set at the "1" position.

-1: Indicates that the PATH movement START point's arm rotation information has been set at the "-1" position.

*1: For details regarding the X-arm and Y-arm rotation information, refer to Chapter 4 "3. Point data format".

MEMO

- At SCARA robots with a hand system flag set in the movement destination's coordinate data, the specified hand system will have priority over the current arm type or LEFTY/RIGHTY setting.

SAMPLE

PATH SET 120 250.00 55.2 20.33 0 0
...............................PATH motion's start-point is specified in "mm" units as follows: (120.00 250.00 55.20 20.33 0.00 0.00).
PATH SET -51200 80521 7045 204410 0 0
..............................PATH motion's start-point is specified in "pulse" units.

CAUTION

- The hand system used during PATH motion must be the same as the hand system used at the path motion route's start point.Differing hand systems will cause an error and disable motion.

● Point definition

Format

<point definition>

Explanation The PATH motion's start-point is specified by the <point expression>.

MEMO

- At SCARA robots with a hand system flag set in the movement destination's coordinate data, the specified hand system will have priority over the current arm type or LEFTY/RIGHTY setting.

CAUTION

- The X-arm and Y-arm rotation information during PATH movement must be the same as the X-arm and Y-arm rotation information at the PATH movement's START point. If the two are different, an error will occur and movement will be disabled.

SAMPLE

PATH SET P10..The PATH motion's start-point is set as P10.
PATH SET WHERE..The PATH motion's start-point is set as the robot's current position.

Related commands	PATH, PATH END, PATH START

Reference For PATH function details, see Chapter 9 "PATH Statements".

73 PATH START

Starts the PATH motion

Format
PATH START

Explanation Starts PATH motion.

Before PATH START can be executed, the PATH motion path must be specified by the PATH SET command, PATH commands (PATH L, PATH C) and the PATH END command. The robot must also be positioned at the motion path's start-point which was specified by the PATH SET command.

The robot's PATH motion speed is the automatic movement speed (%) which was in effect when the PATH START was executed, multiplied by the program movement speed (%) specified by the SPEED command or the (SPEED or S) option of the PATH command. A speed specified by the "VEL" option of the PATH command does not rely on the automatic movement speed.

After PATH motion begins, the PATH START command is terminated when the robot reaches the PATH motion end-point, or when movement is stopped by an interlock, etc.

This command can only be executed in Task 1 (main task).

Related commands	PATH, PATH SET, PATH END

Reference For PATH function details, see Chapter 9 "PATH Statements".

74 PDEF

Defines the pallet used to execute pallet movement commands

Format

PDEF(<Pallet definition number>)=<expression 1>, <expression 2> [, <expression 3>]

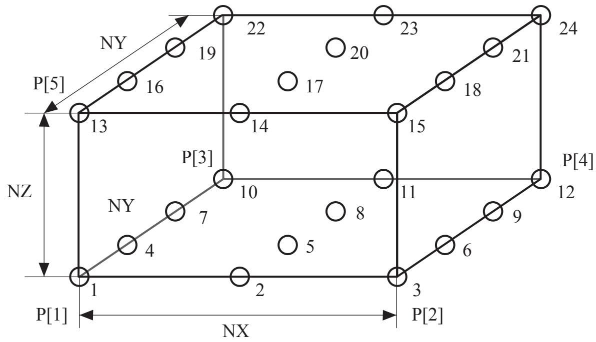
Values

<Pallet definition number>.........0 to 19
<expression 1> ....Number of points (NX) between P[1] and P[2].
<expression 2> ....Number of points (NY) between P[1] and P[3].
<expression 3> ....Number of points (NZ) between P[1] and P[5].
Total number of points: <expression 1> × <expression 2> × <expression 3> must be 32767 or less.
Regarding the P[1] to P[5] definition, see the figure below.

Explanation

Defines the pallets to permit execution of the pallet movement command.
Also changes the dividing conditions of previously defined pallet data.
After specifying the number of points per axis, the equally-spaced points for each axis are automatically calculated and defined in the sequence shown in the figure below.
If <expression 3> (Z-axis direction) is omitted, the height direction value becomes "1".
The total number of points defined for a single pallet must not exceed 32,767.

Automatic point calculation



The point data for pallet definition uses the following data areas.

Pallet definition	P[1]	P[2}	P[3}	P[4]	P[5]
Pallet 0	P3996	P3997	P3998	P3999	P4000
Pallet 1	P3991	P3992	P3993	P3994	P3995
:	:	:	:	:	:
Pallet 19	P3901	P3902	P3903	P3904	P3905

SAMPLE

PDEF(1)=3,4,2 ....Pallet definition 1 is defined as 3 × 4 × 2.

75 PMOVE

Executes a pallet movement command for the robot

Format

```
PMOVE (<pallet definition number> , <pallet position number>)[,option[,option]...]
```

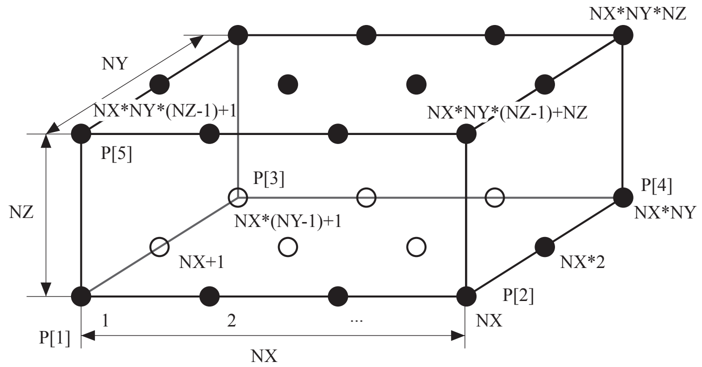
Values <pallet definition number>.........0 to 19
<pallet position number> ....1 to 32767

Explanation Executes a robot axis "pallet move" command. (The specified pallet numbers must be registered in advance.)
The PMOVE command applies to all main robot axes. This command do not apply to any other group axes, or to auxiliary axes.

- Movement type: PTP
- Pallet definition number: Numeric expression
- Pallet position number: Numeric expression
- Options: Speed setting, arch motion setting, STOPON condition setting

The position numbers for each pallet definition are shown below.

Position numbers for each pallet definition



MEMO

- Although the XYZ axes move to the positions determined by calculated values, the R-axis moves to the position specified by pallet point data P[1].

Options	PTP	Remarks
Speed setting (SPEED)	○	Enabled only for specified PMOVE statement
Arch motion	○	Enabled only for specified PMOVE statement
STOPON condition setting	○	Enabled only by program execution

SAMPLE

```
PMOVE(1,16)  ......................................................The main robot axis moves from its current position to the
                                                                   position specified by pallet position number 16 of pallet definition
                                                                   number 1.
```

Movement type

- PTP (point-to-point) movement

PTP movement begins after positioning of all movement axes is complete (within the tolerance range), and **the command terminates when the movement axes enter the OUT position range.** Although the movement axes reach their target positions simultaneously, their paths are not guaranteed.

- Caution regarding commands which follow the PMOVE command:

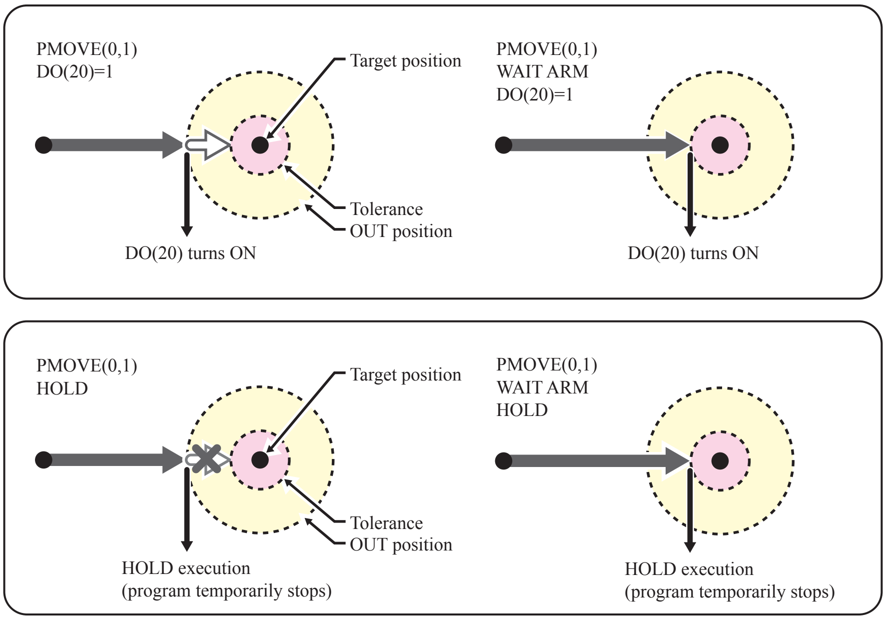
If the next command following the PMOVE command is an executable command such as a signal output command, that next command will start when the movement axis enters the OUT position range. In other words, that next command starts before the axis arrives within the target position OUT position range.

Example:

Signal output (DO, etc.)	Signal is output when axis enters within OUT position range.
DELAY	DELAY command is executed and standby starts, when axis enters the OUT position range.
HALT	Program stops and is reset when axis enters the OUT position range. Therefore, axis movement also stops.
HOLD	Program temporarily stops when axis enters the OUT position range. Therefore, axis movement also stops.
WAIT	WAIT command is executed when axis enters the OUT position range.

The WAIT ARM statement is used to execute the next command after the axis enters the tolerance range.

PMOVE command



75 PMOVE

Option types

● Speed setting **PTP**

Format

1. SPEED = <expression>
2. S = <expression>

Values <expression> ....1 to 100 (units: %)

NOTE
- This option specifies only the maximum speed and does not guarantee movement at the specified speed.

Explanation Specifies the program speed in an <expression>. The movement speed is the automatic movement speed multiplied by the program movement speed.
This option is enabled only for the specified PMOVE statement.

SAMPLE

PMOVE(1,3),S=10... Movement occurs at 10% of the program speed, from the current position to the position specified by pallet position number 3 of pallet definition number 1.

● Arch motion setting **PTP**

Format

x = <expression>[, x = <expression>...]

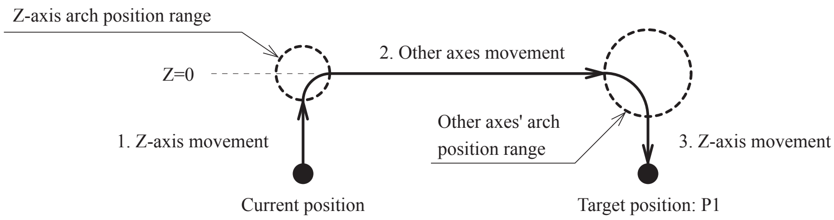
Values
x ....Specifies the Z,R,A,B axis.
<expression> ....An integer value is processed in "pulse" units. A real number (with decimal point) is process in "mm/deg" units.

Explanation
1. The "x" specified axis begins moving toward the position specified by the <expression>.
2. When the "x" specified axis enters the arch position range, all other axes move toward the target position.
3. When all axes other than the "x" specified axis enter the arch position range, and the "x" specified axis enters the tolerance range of the position specified by the <expression>, the "x" specified axis then moves to the target position.
4. The command ends when all axis enter the OUT position range.

SAMPLE

PMOVE(1,A),Z=0........................... The Z-axis first moves from the current position to the "0 pulse" position. Then the other axes move to the position specified by pallet position number A of pallet definition number 1. Finally the Z-axis moves to the position specified by pallet position number A.

SAMPLE: PMOVE Z



- STOPON condition setting **PTP**

Format
STOPON <conditional expression>

Explanation Stops movement when the conditions specified by the conditional expression are met. Because this is a deceleration type stop, there will be some movement (during deceleration) after the conditions are met.
If the conditions are already met before movement begins, no movement occurs, and the command is terminated.
This option is only possible by program execution.

SAMPLE

```
PMOVE(A,16),STOPON DI(20)=1
```
...............................Moves from the current position to the position specified by pallet position number 16 of pallet definition number A, then decelerates and stops when the condition "DI(20) = 1" is met.

MEMO

- When the conditional expression used to designate the STOPON condition is a numeric expression, the conditions for determining a TRUE or FALSE status can be changed at the controller's "TRUE conditions" in the "Other parameters" mode. These conditions apply to all the IF, WHILE, WAIT, STOPON, etc., conditional expressions. For details, refer to the controller user's manual.

1) -1 (default setting)	An expression value of "-1" indicates a TRUE status, and "0" indicates a FALSE status. A "6.35 Incorrect condition expression" error occurs if the expression value is other than "-1" or "0".
2) not 0	Any expression value other than "0" indicates a TRUE status, and "0" indicates a FALSE status.

76 Pn

Defines points within a program

Format
Pn = x y z r a b [f] [f1] [f2]

Values

n...Point number: 0 to 9999.
x, y, z, r, a, b ....Point data: the range varies according to the format.
f...Hand system flag: 1 or 2.
f1..X-arm rotation information : -1, 0, 1 (R6YXTW500 model only).
f2..Y-arm rotation information : -1, 0, 1 (R6YXTW500 model only).

Explanation Defines the point data.

1. "n" indicates the point number.
2. Input data for "x" to "b" must be separated with a space (blank).
3. If all input data for "x" to "b" are integers (no decimal points), the movement units are viewed as "pulses". "x" through "b" then correspond to axis 1 through axis 6.
4. **If there is even 1 real number (with decimal point)** in the input data for "x" through "b", **the movement units are recognized as "mm".** In this case, "x" to "z" correspond to the x, y and z coordinates of a Cartesian coordinate system, while "r" to "b" correspond to axes 4 to 6.
5. The input data ranges are as follows:
   For "pulse" units: -6,144,000 to 6,144,000 range
   For "mm" units: -99,999.99 to 99,999.99 range

NOTE
- If both integers and real numbers are used together (mixed), all coordinate values will be handled in "mm/deg" units.
- X-arm and Y-arm rotation information is only available in software Ver.1.66M or higher.

Hand system flags can be specified for SCARA robots when specifying point definition data in "mm" units.
To specify an extended hand system flag for SCARA robots, set either 1 or 2 at "f". If a number other than 1 or 2 is set, or if no number is designated, 0 will be set, indicating that there is no hand system flag.

1: Indicates a right-handed system point setting.
2: Indicates a left-handed system point setting.

X-arm and Y-arm rotation information (*1) can be specified on R6YXTW500 where point data is defined in "mm" units.
To set extended X-arm and Y-arm rotation information at the R6YXTW500 model robot, a "-1", "0", or "1" value must be specified at f1 and f2. Any other value, or no X-arm and Y-arm rotation information at all, will be processed as "0".

0: Indicates arm rotation information where "0" has been specified.
1: Indicates arm rotation information where "1" has been specified.
-1: Indicates arm rotation information where "-1" has been specified.

*1: For details regarding the X-arm and Y-arm rotation information, refer to Chapter 4 "3. Point data format".

76 Pn

NOTE

- All input values are handled as constants.
- If controller power is turned off during execution of a point definition statement, a memory-related error such as "9.2: Point check-sum error" may occur.

SAMPLE

```
P1   =        0         0         0        0        0        0
P2   =   100.00    200.00     50.00     0.00     0.00     0.00
P3   =    10.00      0.00      0.00     0.00     0.00     0.00
P10=P2
FOR A=10 TO 15
      P[A+1]=P[A]+P3
NEXT A
FOR A=10 TO 16
      MOVE P,P1,P[A]
NEXT A
HALT
```

Related commands	Point assignment statement (LET)

77 PPNT

Creates pallet point data

Format
PPNT(pallet definition number,pallet position number)

Explanation Creates the point data specified by the pallet definition number and the pallet position number.

SAMPLE
P10=PPNT(1,24)..Creates, at P10, the point data specified by pallet position number 24 of pallet definition number 1.

Related commands	PDEF, PMOVE

78 PRINT

Displays the specified expression value at the programming box

Format

PRINT [<expression>][, or ; <expression>...][, or ;]

Values <expression> ....character string, numeric value, variable.

Explanation Displays a specified variable on the programming box screen.
Output definitions are as follows:

1. If numbers or character strings are specified in an <expression>, they display as they are. If variables or arrays are specified, the values assigned to the specified variables or arrays display.
2 If no <expression> is specified, only a line-feed occurs.
3. If the data length exceeds the screen width, a line-feed occurs, and the data wraps to the next line.
4. If a comma (,) is used as a display delimiter, a space (blank) is inserted between the displayed items.
5. If a semicolon (;) is used as a display delimiter, the displayed items appear in succession without being separated.
6. If the data ends with a delimiter, the next PRINT statement is executed without a line-feed. When not ended with a display delimiter, a line-feed occurs.

- Data communication to the programming box screen occurs in order for the PRINT statement to be displayed there. Therefore, program execution may be delayed when several PRINT statements are executed consecutively.

SAMPLE

```
PRINT A  Displays the value of variable A.
PRINT "A1 =";A1 ................................................Displays the value of variable A1 after "A1 =".
PRINT "B(0),B(1) = ";B(0);",";B(1)
PRINT P100     .....................................................Displays the P100 value.
```

Related commands	INPUT

79 RADDEG

Performs a unit conversion (radians → degrees)

Format
RADDEG(<expression>)

Values <expression> ....Angle (units: radians)

Explanation Converts the <expression> value to degrees.

SAMPLE

```
LOCR(P0)=RADDEG(ATN(B))...........................Converts the variable B arctangent value to degrees, and
                                                  assigns it to R-data of P0.
```

Related commands	ATN, COS, DEGRAD, SIN, TAN

80 REM

Inserts a comment

Format

1. REM <character string>
2. ' <character string>

Explanation All characters which follow REM or an apostrophe (') are handled as a comment. This comment statement is used only to insert comments in the program, and it does not execute any command. The apostrophe (') can be entered at any point in the line.

SAMPLE

```
REM *** MAIN PROGRAM ***
      ’*** SUBROUTINE ***
HALT   ’HALT COMMAND
```

81 RESET

Turns OFF the bits of specified ports, or clears variables

Format 1	
RESET	DOm([b,.....,b]) DO(mb,.....,mb) MOm([b,.....,b]) MO(mb,.....,mb) TO0([b,.....,b]) TO(0b,.....,0b) LO0([b,.....,b]) LO(0b,.....,0b) SOm([b,.....,b]) SO(mb,.....,mb)

Format 2

RESET TCOUNTER

Values m: port number ....2 to 7, 10 to 17, 20 to 27
b: bit definition ....0 to 7

Explanation Format 1: Turns the bits of specified ports OFF.
Format 2: Clears the 10ms counter variables (10ms counter variables are used to measure the time in 10ms units).

If multiple bits are specified, they are expressed from the left in descending order (large to small).
If the [b,…,b] data is omitted, all 8 bits are processed.

CAUTION
- Output to ports "0" and "1" is not allowed at DO, MO, and SO.

REFERENCE
- For details regarding bit definitions, see Chapter 3 "10 Bit Settings".

SAMPLE

RESET DO2() ....Turns OFF DO(27 to 20).
RESET DO2(6,5,1) ....Turns OFF DO(26, 25, 21).
RESET (37,35,27,20)..Turns OFF DO(37, 35, 27, 20).
RESET TCOUNTER...Clears the 10ms counter variables.

Related commands	SET, DO, MO, SO, TO, LO

82 RESTART

Restarts another task during a temporary stop

7

Format

RESTART Tn

Values n: Task number...........................2 to 8

Explanation Restarts another task that has been temporarily stopped (SUSPEND status).
RESTART cannot be executed for Task 1.

SAMPLE

```
START *SUBTASK,T2
    FLAG=1
*L0:
    IF FLAG=1 AND DI2(0)=1 THEN
        SUSPEND T2
        FLAG=2
    WAIT DI2(0)=0
    ENDIF
    IF FLAG=2 AND DI2(0)=1 THEN
        RESTART T2
        FLAG=1
        WAIT DI2(1)=0
    ENDIF
    MOVE P,P0
    MOVE P,P1
    GOTO *L0
    HALT
'SUBTASK ROUTINE
*SUBTASK:
    DO2(0)=1
    DELAY 1000
    DO2(0)=0
    DELAY 1000
    GOTO *SUBTASK
    EXIT TASK
```

Related commands	CUT, EXIT TASK, START, SUSPEND

Reference For details, refer to the "Multi-Task" item.

N O P Q R S T U V W X Y Z

83 RESUME

Resumes program execution after error recovery processing

Format
1. RESUME NEXT
2. RESUME <label>

REFERENCE

- For details, see Chapter 8 "60 ON ERROR GOTO".

Explanation Resumes program execution after recovery from an error.
Depending on its location, a program can be resumed in the following 3 ways:

1. RESUME	The program resumes from the command which caused the error.
2. RESUME NEXT	The program resumes from the next command after the command which caused the error.
3. RESUME <label>	The program resumes from the command specified by the <label>.

MEMO

- The RESUME statement can also be executed in an error processing routine.
- "Error recovery processing is not possible for serious errors such as "17.4 : Overload", etc.

Related commands	ON ERROR GOTO

84 RETURN

Processing which was branched by GOSUB, is returned to the next line after GOSUB

Format

```
GOSUB <label>
    :
<label>:
    :
RETURN
```

* GOSUB can also be expressed as "GO SUB".

Explanation Ends the subroutine and returns to the next line after the jump source GOSUB statement. All subroutines (jump destinations) specified by a GOSUB statement must end with a RETURN statement. Using the GOTO statement, etc., to jump from a subroutine will cause an error such as the "5.12: Stack overflow", etc.

SAMPLE

```
*ST:
    MOVE P,P0
    GOSUB *CLOSEHAND
    MOVE P,P1
    GOSUB *OPENHAND
GOTO *ST
HALT
'SUB ROUTINE
*CLOSEHAND:
    DO(20) = 1
RETURN
*OPENHAND:
    DO(20) = 0
RETURN
```

Related commands	GOSUB

85 RIGHT$

Extracts a character string from the right end of another character string

Format
RIGHT$(<character string expression>,<expression>)

Values <expression> ....0 to 75

Explanation This function extracts a character string with the digits specified by the <expression> from the right end of the character string specified by <character string expression>.
The <expression> value must be between 0 and 75, otherwise an error will occur.
If the <expression> value is 0, then RIGHT$ will be a null string (empty character string).
If the <expression> value has more characters than the <character string expression>, RIGHT$ will become the same as the <character string expression>.

SAMPLE

```
B$=RIGHT$(A$,4)........................................4 characters from the right end of A$ are assigned to B$.
```

Related commands	LEFT$, MID$

86

RIGHTY

Sets the SCARA robot hand system to "Right"

Format

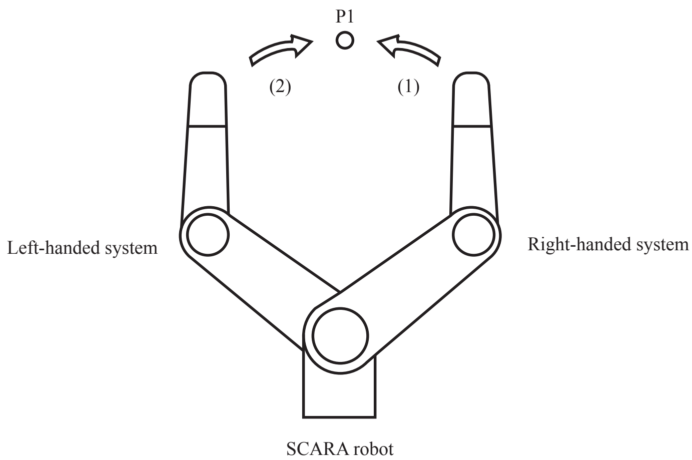
RIGHTY

Explanation This statement specifies right-handed movement to a point specified in Cartesian coordinates. This statement only selects the hand system, and does not move the robot. If executed while the robot arm is moving, execution waits until movement is complete (positioned within tolerance range).

SAMPLE

```
RIGHTY      ......................................Specifies a robot "right-handed system" setting (see Fig.1
                                                  below).
MOVE P,P1
LEFTY       ......................................Specifies a robot "left-handed system" setting (see Fig.2
                                                  below).
MOVE P,P1
RIGHTY
HALT
```

SAMPLE: LEFTY/RIGHTY



Related commands	LEFTY

87 RSHIFT

Shifts a bit value to the right

Format
RSHIFT(<expression 1>,<expression 2>)

Explanation Shifts the <expression 1> bit value to the right by the amount of <expression 2>. Spaces left blank by the shift are filled with zeros (0).

SAMPLE	
A=RSHIFT(&B10111011,2).................................	The 2-bit-right-shifted &B10111011 value (&B00101110) is assigned to A.

Related commands	LSHIFT

88 Sn

Defines the shift coordinates in the program

Format

```
Sn = x y z r
```

Values

n..0 to 9
x, y, z, r.......................................-99,999.99 to 99,999.99

NOTE

- All input values are handled as constants.
- If the controller power is turned off during execution of a shift coordinate definition statement, a memory-related error such as "9.6: Shift check-sum error" may occur.

Explanation Defines shift coordinate values in order to shift the coordinates for robot movement. Only "mm" units can be used for these coordinate values ("pulse" units cannot be used).

1. "n" indicates the shift number.
2. The "x" to "r" input data must be separated with spaces (blanks).
3. The "x" to "r" input data is recognized as "mm" unit data.
4. "x" to "z" correspond to the Cartesian coordinate system's x, y, z coordinate shift values, and "r" corresponds to the xy coordinates' rotational shift values.

SAMPLE

```
S0   =   0.00     0.00     0.00    0.00
S1   =   100.00   200.00   50.00   90.00
P3   =   100.00   0.00     0.00    0.00    0.00    0.00
SHIFT S0
MOVE P,P3
SHIFT S1
MOVE P,P3
HALT
```

Related commands	Shift assignment statement, SHIFT

7

N
O
P
Q
R
S
T
U
V
W
X
Y
Z

89 SELECT CASE

Executes the specified command block in accordance with the <expression> value

Format

```
SELECT [CASE] <expression>
    CASE <expression list 1>
        [command block 1]
    [CASE <expression list 2>
        [command block 2]]
        :
    [CASE ELSE
        [command block n]]
END SELECT
```

Explanation These statements execute multiple command blocks in accordance with the <expression> value. The setting method is as follows.

1. The <expression list> following CASE statement comprises multiple numerical expressions and character expressions separated from each other by a comma (,).
2. If the <expression> value matches one of expressions contained in the <expression list>, the specified command block is executed. After executing the command block, the program jumps to the next command which follows the END SELECT statement.
3. If the <expression> value does not match any of the expressions contained in the <expression list>, the command block indicated after the CASE ELSE statement is executed. After executing the command block, the program jumps to the next command which follows the END SELECT statement.
4. If the <expression> value does not match any of the expressions contained in <expression list> and no CASE ELSE statement exists, the program jumps to the next command following the END SELECT statement.

SAMPLE

```
WHILE -1
SELECT CASE DI3()
    CASE 1,2,3
        CALL *EXEC(1,10)
    CASE 4,5,6,7,8,9,10
        CALL *EXEC(11,20)
    CASE ELSE
        CALL *EXEC(21,30)
END SELECT
WEND
HALT
```

90 SEND

Sends <read file> data to the <write file>

Format

SEND <read file> TO <write file>

NOTE

- Examples of erroneous writing to a read-only file:
  SEND CMU TO DIR
  SEND PNT TO SI()
- Examples of data format mismatches:
  SEND PGM TO PNT
  SEND SI() TO SFT

Explanation Sends <read file> data to the <write file>.

An entire DO, MO, TO, LO, SO, or SOW port (DO(), MO(), etc.), cannot be specified as a write file.

Moreover, some individual files (DOn(), MOn(), etc.) cannot be specified as a write file. For details, refer to Chapter 11 "Data file description".

Writing to read-only files (indicated by a "×" in the "WRITE" column of the table shown below) is not permitted.

Even if the READ and WRITE files are specified correctly, it may not be possible to execute them if there is a data format mismatch between the files.

Type	File Name	Definition Format		READ	WRITE
		All	Individual File		
User	All files	ALL	———	○	○
	Program	PGM	<bbbbbbbb>	○	○
	Point	PNT	Pn	○	○
	Point comment	PCM	PCn	○	○
	Parameter	PRM	/cccccc/	○	○
	Shift definition	SFT	Sn	○	○
	Hand definition	HND	Hn	○	○
	Pallet definition	PLT	PLn	○	○
Variable, Constant	Variable	VAR	ab...by	○	○
	Array variable	ART	ab...by(x)	○	○
	Constant		“cc...c”	○	×
Status	Program directory	DIR	<<bbbbbbbb>>	○	×
	Parameter directory	DPM	———	○	×
	Machine reference	MRF	———	○	×
	Error log	LOG	———	○	×
	Remaining memory size	MEM	———	○	×
Device	DI port	DI()	DIn()	○	×
	DO port	DO()	DOn()	○	○
	MO port	MO()	MOn()	○	○
	TO port	TO()	TOn()	○	○
	LO port	LO()	LOn()	○	○
	SI port	SI()	SIn()	○	×
	SO port	SO()	SOn()	○	○
	SIW port	SIW()	SIWn()	○	×
	SOW port	SOW()	SOWn()	○	○
	RS-232C	CMU	———	○	○
	Ethernet	ETH	———	○	○
Other	File END code	EOF	———	○	×

N: Number a: Alphabetic character b: Alphanumeric character or underscore (_) ○: Permitted
c: Alphanumeric character or special symbol x: Expression (array argument) y: Variable type × : Not permitted

90 SEND

MEMO

- The following cautions apply when a restart is performed after a stop occurred during execution of the SEND statement:
  1. When reading from RS-232C / Ethernet (SEND CMU TO XXX, SEND ETH TO XXX): When the SEND statement is stopped during data reading from the reception buffer, the data acquired up to that point is discarded.
  2. When writing to RS-232C / Ethernet (SEND XXX TO CMU, SEND XXX TO ETH): When the SEND statement is stopped during data writing to the transmission buffer, the data is written from the beginning.

SAMPLE

```
SEND PGM TO CMU.......................Outputs all user programs from the RS-232C port.
SEND <PRG1> TO CMU....................Outputs the PRG1 program from the RS-232C port.
SEND CMU TO PNT.......................Inputs a point data file from the RS-232C port.
SEND "T1" TO CMU......................Outputs the "T1" character string from the RS-232C port.
SEND CMU TO A$........................Inputs character string data to variable A$ from the RS-
                                      232C port.
```

Reference For details, refer to Chapter 11 "Data file description".

91 SERVO

Controls the servo status

Format

SERVO	ON OFF FREE PWR	[(<axis number>)]

Values <axis number>............................main group: 1 to 6

Explanation This statement controls the servo ON/OFF at the specified axes or all axes. When the axes have been specified by an <axis number> setting, this statement applies only to the specified axes within the group. If no axes have been specified by an <axis number> setting, this statement applies to all the main axes. In this case, motor power supply ON/OFF switching occurs simultaneously with the servo ON/OFF operations.

- ON .... Turns the servo ON. If no axis is specified, the motor power supply also turns ON.
- OFF....... Turns the servo OFF and applies the dynamic brake. Axes equipped with brakes are all locked by the brake. If no axis is specified, the motor power supply also turns OFF.
- FREE .... Turns the servo OFF and releases the dynamic brake. The brakes are released at all axes with brakes. If no axis is specified, the motor power supply also turns OFF.
- PWR ..... Turns only the motor power supply ON.

CAUTION

- Keep out of the robot movement range while the motor power is turned OFF by the SERVO OFF statement. Always check that the Emergency Stop is ON when working within the robot movement area.

MEMO

- This statement is executed after positioning of all axes is complete (within the tolerance range).
- Individual axis servos cannot be turned ON as long as the motor power is OFF.

SAMPLE

```
SERVO ON      .....................................................Turns servos ON at all axes after turning the motor power ON.
SERVO OFF     .....................................................Turns the motor power OFF, then turns the servos at all axes OFF.
                                                                    Brakes are applied, and a lock status is established at axes equipped
                                                                    with brakes.
SERVO FREE(3)...................................................The axis 3 (Z-axis) servo is turned OFF, and the brake is released.
```

92 SET

Turns the bit at the specified output port ON

Format

SET	DOm([b,.....,b])	[, <time>]
	DO (mb,.....,mb)	
	MOm([b,.....,b])	
	MO (mb,.....,mb)	
	TO0([b,.....,b])	
	TO (0b,.....,0b)	
	LO0([b,.....,b])	
	LO (0b,.....,0b)	
	SOm([b,.....,b])	
	SO (mb,.....,mb)	

Values

m: port number ....2 to 7, 10 to 17, 20 to 27
b: bit definition ....0 to 7
<time> ....10 to 3600000 (units: ms)

Explanation

Turns ON the bits of specified ports.
The pulse output time (unit: ms) is specified by the <time> value. When the specified time elapses, the output is turned OFF, and command execution ends.

If multiple bits are specified, they are expressed from the left in descending order (large to small).
If the [b,…,b] data is omitted, all 8 bits are processed.
If no hardware port exists, nothing is output.

CAUTION

- Output to ports "0" and "1" are not allowed at DO, MO, and SO.

REFERENCE

- For bit setting details, see Chapter 3 "10 Bit Settings".

SAMPLE

SET DO2()..Turns ON DO(27 to 20).
SET DO2(6,5,1),200..DO(26,25,21) switches ON for 200ms.
SET DO(37,35,27,20)..Turns DO(37, 35, 27, 20) ON.

Related commands	RESET, DO, MO, SO, TO, LO

93 SHARED

Enables sub-procedure referencing without passing on the variable

Format

SHARED <variable>[()][,<variable>[()]...]

Explanation This statement allows variables declared with a program level code to be referenced with a sub-procedure without passing on the variables as dummy arguments.
The program level variable used by the sub-procedure is specified by the <variable> value.
A simple variable or an array variable followed by parentheses is specified. If an array is specified, that entire array is selected.

MEMO

- Normally, a <dummy argument> passes along the variable to a sub-procedure, but the SHARED statement allows referencing to occur without passing along the variable.
- The SHARED statement allows variables to be shared only between a program level code and sub-procedure which are within the same program level.

SAMPLE

```
DIM Y!(10)
X!=2. 5
Y!(10)=1. 2
CALL *DISTANCE
CALL *AREA
HALT
SUB *DISTANCE
    SHARED X!,Y!( )..........................................Variable referencing is declared by SHARED.
    PRINT X!^2+Y!(10)^2..................................The variable is shared.
END SUB
SUB *AREA
    DIM Y!(10)
    PRINT X!*Y!(10)..........................................The variable is not shared.
END SUB
```

Related commands	SUB, END SUB

94 SHIFT

Sets the shift coordinates

Format
SHIFT <shift variable>

Explanation Sets the shift coordinates in accordance with the shift data specified by the <shift variable>.

MEMO

- This statement is executed after axis positioning is complete (within the tolerance range).
- The default shift setting is S0 (no shift values).

SAMPLE

```
SHIFT S1
MOVE P,P10
SHIFT S[A]
MOVE P,P20
HALT
```

Related commands	Shift definition statement, shift assignment statement

95

SIN

Acquires the sine value for a specified value

Format
SIN(<expression>)

Values <expression> ....Angle (units: radians)

Explanation This function gives the sine value for the <expression> value.

SAMPLE

A(0)=SIN(B*2+C)..Assigns the expression B*2+C sine value to array A (0).
A(1)=SIN(DEGRAD(30))......................................Assigns a 30.0° sine value to array A (1).

Related commands	ATN, COS, DEGRAD, RADDEG, TAN

96 SO

Outputs a specified value to the serial port

Format

1. [LET]SOm([b,.....,b]) = <expression>
2. [LET]SO (mb,.....,mb) = <expression>

Values m: port number...........................2 to 7, 10 to 17, 20 to 27
b: bit definition...........................0 to 7

Explanation Outputs a specified value to the SO port.
Only the <value> data's integer-converted lower bits corresponding to the bits defined at the left side can be output.
If multiple bits are specified, they are expressed from the left in descending order (large to small).
If the [b,…,b] data is omitted, all 8 bits are processed.
If no hardware port exists, nothing is output.

CAUTION

- Outputs to SO0() and SO1() are not possible.

REFERENCE

- For bit setting details, see Chapter 3 "10 Bit Settings".

SAMPLE

SO2()=&B10111000 ....SO (27, 25, 24, 23) are turned ON, and SO (26, 22, 21, 20) are turned OFF.
SO2(6,5,1)=&B010 ....SO (25) are turned ON, and SO (26, 21) are turned OFF.
SO3()=15 ....SO (33, 32, 31, 30) are turned ON, and SO (37, 36, 35, 34) are turned OFF.
SO(37,35,27,20)=A...The lower 4 bits of integer-converted variable A are output to SO (37, 35, 27, 20).

Related commands	RESET, SET

97 SPEED

Changes the program movement speed

Format
SPEED <expression>

Values <expression> ....1 to 100 (units: %)

Explanation Changes the program movement speed to the speed indicated by the <expression>.
This speed change applies to all the robot axes.
The operation speed is determined by multiplying the automatic movement speed (specified from the programming box and by the ASPEED command), by the program movement speed (specified by SPEED command, etc.).
Operation speed = automatic movement speed x program movement speed.

Example:
Automatic movement speed ... 80%
Program movement speed ... 50%
Movement speed = 40% (80% × 50%)

SAMPLE

```
ASPEED 100
SPEED 70
MOVE P,P0............................................................Moves from current position to P0 at a speed of 70% (=100 *
                                                                     70).
SPEED 50
MOVE P, P1..........................................................Moves from current position to P1 at a speed of 50% (=100 *
                                                                     50).
MOVE P,P2, S=10.................................................Moves from current position to P2 at a speed of 10% (=100 *
                                                                     10).
HALT
```

Related commands	ASPEED

98 START

Starts a new task

Format

START <label>, Tn[, p]

Values

n: Task number...........................2 to 8
p: Task priority ranking.............17 to 47

Explanation

Starts task "n" specified by the <label> with the "p" priority ranking.
If a priority ranking is not specified, "32" is adopted as the priority ranking for this task.
The smaller the priority number, the higher the priority (high priority: 17 ↔ low priority: 47).
When a READY status occurs at a task with higher priority, all tasks with lower priority also remain in a READY status.

SAMPLE

```
START *SUBTASK,T2,33
*ST:
    MOVE P,P0,P1
GOTO *ST
HALT
'SUBTASK ROUTINE
*SUBTASK:
    P100 = WHERE
    IF LOCZ(P100) > 10000 THEN
        DO(20) = 1
    ELSE
        DO(20) = 0
    ENDIF
GOTO *SUBTASK
EXIT TASK
```

Related commands	CUT, EXIT TASK, RESTART, SUSPEND, CHGPRI

99 STR$

Converts a numeric value to a character string

Format

STR$(<expression>)

Explanation Converts the value specified by the <expression> to a character string. The <expression> specifies an integer or real number value.

SAMPLE

```
B$=STR$(10.01)
```

Related commands	VAL

100 SQR

Acquires the square root of a specified value

Format

```
SQR(<expression>)
```

Values <expression> ....0 or positive number.

Explanation Gives the square root of the <expression> value. An error occurs if the <expression> value is a negative number.

SAMPLE

```
A=SQR(X^2+Y^2)........................................The square root of X^2+Y^2 is assigned to variable A.
```

101 SUB to END SUB

Defines a sub-procedure

Format

```
SUB <label> [(<dummy argument> [, <dummy argument> ...])]
    <command block>
END SUB
```

Explanation Defines a sub-procedure.

The sub-procedure can be executed by a CALL statement. When the END SUB statement is executed, the program jumps to the next command after the CALL statement that was called. Definitions are as follows.

1. All variables declared within the sub-procedure are local variables, and these are valid only within the sub-procedure. Local variables are initialized each time the sub-procedure is called up.
2. Use a SHARED statement in order to use global variables (program level).
3. Use a <dummy argument> when variables are to be passed on. If two or more dummy arguments are used, separate them by a comma (,).
4. A valid <dummy argument> consists of a name of variable and an entire array (array name followed by parentheses). An error will occur if array elements (a <subscript> following the array name) are specified.

MEMO

- Sub-procedures cannot be defined within a sub-procedure.
- The DECLARE statement cannot be used within a sub-procedure.
- A label can be defined within a sub-procedure, but it cannot jump (by a GOTO or GOSUB statement) to a label outside the sub-procedure.
- Local variables cannot be used with PRINT and SEND statements.

SAMPLE 1

```
A=1
CALL *TEST
PRINT A
HALT
'SUB ROUTINE: TEST
SUB *TEST
    A=50      ...................Handled as a different variable than the "A" shown above.
END SUB
```

MEMO

- In the above example, the program level variable "A" is unrelated to the variable "A" within the sub-procedure. Therefore, the value indicated in the 3rd line PRINT statement becomes "1".

101 SUB to END SUB

SAMPLE 2

```
X% = 4
Y% = 5
CALL *COMPARE( REF X%, REF Y% )
PRINT X%,Y%
Z% = 7
W% = 2
CALL *COMPARE( REF Z%, REF W% )
PRINT Z%,W%
HALT
'SUB ROUTINE: COMPARE
SUB *COMPARE( A%, B% )
    IF A% < B% THEN
        TEMP% = A%
        A% = B%
        B% = TEMP%
    ENDIF
END SUB
```

MEMO

- In the above example, different variables are passed along as arguments to call the sub-procedure 2 times.

Related commands	CALL, DECLARE, EXIT SUB, SHARED

102 SUSPEND

Temporarily stops another task which is being executed

Format
SUSPEND Tn

Values n: Task number...........................2 to 8

Explanation Temporarily stops (suspends) another task which is being executed.
This statement can also be used for tasks with a higher priority ranking than this task itself.
This statement cannot be specified for the main task (Task number 1).

SAMPLE

```
START *SUBTASK,T2
SUSFLG=0
*L0:
    MOVE P,P0
    MOVE P,P1
    WAIT SUSFLG=1
    SUSPEND T2
    SUSFLG=0
GOTO *L0
HALT
'SUBTASK ROUTINE
*SUBTASK:
    WAIT SUSFLG=0
    DO2(0)=1
    DELAY 1000
    DO2(0)=0
    DELAY 1000
    SUSFLG=1
    GOTO *SUBTASK
    EXIT TASK
```

Related commands	CUT, EXIT TASK, RESTART, SUSPEND

103 SWI

Switches the program being executed

Format

SWI "<"<program name>">"

Explanation This statement switches from the current program to the specified program, starting from the first line after compiling is completed.

Although the output variable status is not changed when the program is switched, the dynamic variables and array variables are cleared. Operation stops if an error occurs during compiling. The program name to be switched to must be enclosed in angular brackets (< >).

This command can be executed only in Task 1 (main task).

MEMO

- If the program specified as the switching target does not exist, message "3.3: Program doesn't exist" (code: &H0303) displays and operation stops.
- Execution of a SWI statement is always accompanied by compiling, and the time required for this compiling depends on the size of the switching target program.
- If an error occurs during compiling, an error message line displays and the program stops.
- The SWI statement can only be executed within task 1 (main task). If used within tasks 2 through 8, the message "6.1: Illegal command" displays and operation stops.
- The STOP key is disabled during compiling.

SAMPLE

SWI <ABC> ....Switches the execution program to "ABC".

104 TAN

Acquires the tangent value for a specified value

Format

TAN(<expression>)

Values <expression> ....Angle (units: radians)

Explanation Gives a tangent value for the <expression> value. An error will occur if the <expression> value is a negative number.

SAMPLE

A(0)=B-TAN(C)...The difference between the tangent values of variable B and variable C is assigned to array A (0).

A(1)=TAN(DEGRAD(20))....................................The 20.0° tangent value is assigned to array A (1).

Related commands	ATN, COS, DEGRAD, RADDEG, SIN

105 TCOUNTER

Timer & counter

Format
TCOUNTER

Explanation Outputs count-up values at 10ms intervals starting from the point when the TCOUNTER variable is reset (counter variable value 1 = 10ms).
After counting up to 65,535, the count is reset to 0.

SAMPLE

```
MOVE P,P0
WAIT ARM
RESET TCOUNTER
MOVE P,P1
WAIT ARM
A = TCOUNTER
PRINT TCOUNTER..............................................Displays the P0 to P1 movement time at the programming
                                                             box until movement enters the tolerance range.
```

Related commands	RESET

106 TIME$

Acquires the current time

Format

```
TIME$
```

Explanation Acquires the current time in an hh:mm:ss format character string. "hh" is the hour, "mm" is the minutes, and "ss" is the seconds. The clock can be set in the SYSTEM mode's initial processing.

SAMPLE

```
A$=TIME$
PRINT TIME$
```

Related commands	DATE$, TIMER

107 TIMER

Acquires the current time

CAUTION

- The time indicated by the internal clock may differ somewhat from the actual time.

Format
TIMER

Functions Acquires the current time in seconds, counting from 12:00 midnight. This function is used to measure a program's run time, etc.
The clock can be set in the SYSTEM mode's initial processing.

SAMPLE

```
A%=TIMER
FOR B=1 TO 10
MOVE P,P0
MOVE P,P1
NEXT
A%=TIMER-A%
PRINT A%/60;":";A% MOD 60
HALT
```

Related commands	TIME$

108 TO

Outputs a specified value to the TO port

Format

1. [LET]TO0([b,.....,b]) = <expression>
2. [LET]TO (0b,.....,0b) = <expression>

Values b: bit definition ....0 to 7

Explanation Outputs the specified value to the TO port. The output value is the expression's integer-converted lower bits corresponding to the bit definition specified at the left side.

If multiple bits are specified, they are expressed from the left in descending order (large to small).

If the [b,…,b] data is omitted, all 8 bits are processed.

The OFF/ON settings for bits which are being used in a SEQUENCE program have priority while the SEQUENCE program is running.

SAMPLE

```
TO0() = &B00000110
```

Related commands	RESET, SET

109 TOLE

Specifies/acquires the tolerance parameter

Format
1. TOLE <expression>
2. TOLE (<axis number>) = <expression>

Values <axis number>............................main group: 1 to 6
<expression> ....Varies according to the motor which has been specified (units: pulse)

Explanation Changes the tolerance parameter to the <expression> value.
Format 1: The change is applied to all axes of each group.
Format 2: The change is applies to only the group axes specified by <axis number>.

MEMO

- If an axis that is set to "no axis" in the system generation is specified, a "5.37 : Specification mismatch" error message displays and command execution is stopped.
- This statement is executed after positioning of the specified axes is complete (within the tolerance range).

Functions

Format
TOLE(<axis 1>)

Values <axis 1>......................................main group: 1 to 6

Explanation Acquires the tolerance parameter value for the axis specified by <axis number>.

SAMPLE

```
'CYCLE WITH DECREASING TOLERANCE
DIM TOLE(5)
FOR A=200 TO 80 STEP -20
    GOSUB *CHANGE_TOLE
    MOVE P,P0
    MOVE P,P1
NEXT A
C=TOLE(2)...............................................The tolerance parameter of the main group's axis 2 is assigned to variable C.
HALT
*CHANGE_TOLE:
FOR B=1 TO 4
    TOLE(B)=A
NEXT B
RETURN
```

N O P Q R S T U V W X Y Z

110 TORQUE

Specifies/acquires the maximum torque command value which can be set for a specified axis

CAUTION

- If the specified torque limit is too small, the axis may not move. In this case, press the emergency stop button before proceeding with the operation.
- If the specified value is less than the rated torque, an error may not occur even if the robot strikes an obstacle.

Format

```
TORQUE(<axis number>) = <expression>
```

Values

<axis number>............................main group: 1 to 6
<expression> ....1 to 100 (units: %)

Explanation Changes the maximum torque command value for each group's axes which have been specified by <axis number>. The new value is enabled after the next movement command (MOVE or DRIVE statement, etc.) is executed. The torque parameter value does not change.

The maximum torque specified by this statement remains valid until any of the following operations occur.

- Until another TORQUE command for the same axis is executed.
- Until a torque limit option is executed in a DRIVE statement for the same axis.
- Until controller power is turned off and then on again.
- Until parameters are changed or initialized.
- Until a return-to-origin or an absolute reset & return-to-origin is performed.
- Until the servo is turned off.

MEMO

- If an axis that is set to "no axis" in the system generation is specified, a "5.37 : Specification mismatch" error message displays and command execution is stopped.

Functions

Format

```
TORQUE(<axis 1>)
```

Values

<axis 1>......................................main group: 1 to 6

Explanation Acquires the torque setting value for the axis specified by <axis number>.

110 TORQUE

SAMPLE

TRQTIME(3) = 2500	Sets the torque control time-out period as 2.5 seconds for axis 3.
DRIVE(3,P1),T = (20,15)	Sets the maximum torque value to 20% of the rated torque, and the torque offset to 15, then moves axis 3 from its current position to the point specified by P1 (pushing action).
IF TRQSTS(3) = 1 THEN	Checks if a time-out has occurred.
DO(21) = 1	Time-out has occurred (pushing is complete). (Result is output to DO(21) in this example.)
ELSE	
DO(21) = 0	Time-out has not occurred. (Reached target position but failed to complete pushing.) (Result is output to DO(21) in this example.)
ENDIF	
TORQUE(3) = 100	Returns the max. torque command value to the original value (100%).
DRIVE(3,P0)	Ends the torque limit and torque control, and moves to P0.
A=TORQUE(2)	The torque setting value for the main group's axis 2 is assigned to variable A.

Related commands	DRIVE, TRQTIME, TRQSTS, CURTRQ

111 TRQSTS

Acquires the status when DRIVE statement ends

Format

TRQSTS (<axis 1>)

Values <axis 1>......................................main group: 1 to 6

Explanation Acquires the status at the completion of a "DRIVE statement with torque limit option" that was executed for the main group axis specified by <axis number>.

0 ...... The DRIVE statement was ended for a reason other than a torque limit time-out.
1 ...... The DRIVE statement was ended by torque limit time-out.

MEMO

- If an axis that is set to "no axis" in the system generation is specified, a "5.37 : Specification mismatch" error message displays and command execution is stopped.

SAMPLE

```
DRIVE(3,P1), T=20...............................................Moves the main group's axis 3 under torque limit control.
IF TRQSTS(3)=1 THEN.........................................Ended by a time-out with the torque limit value reached.
GOTO *OK
ELSE......................................................................Movement ended without a time-out occurring.
GOTO *NG
ENDIF
```

Related commands	DRIVE, TRQTIME, CURTRQ

112 TRQTIME

Sets/acquires the time-out period for the torque limit setting option

Format

TRQTIME(<axis number>) = <time-out period>

Values <axis number>............................main group: 1 to 6
<time-out period>.......................1 to 10000 (units: ms)

Explanation Specifies the torque control time-out period when using the DRIVE statement's torque limit setting option. This command specifies the time-out period (<time-out period>) for the axis specified by <axis number>.

A DRIVE statement executed with a torque limit option ends when the axis has reached the target position, or when the "specified toque limit reached" time has exceeded the time-out period specified by the TRQTIME statement.
A value is then set in the TRQSTS function, depending on whether or not the "specified toque limit reached" period has exceeded the time-out period specified by the TRQTIME statement and this command has ended.
When the controller power is turned on, the time-out period is set to 1 second (1,000ms).

MEMO

- Although the time-out period is specified in "ms" units, it actually operates in "10ms" units. Therefore, settings are rounded upward to 10ms. For example, if the setting is a value from 1 to 9, this becomes 10ms. However, if "0" is specified, this becomes 1 second (1000ms).
- If an axis that is set to "no axis" in the system generation is specified, a "5.37 : Specification mismatch" error message displays and command execution is stopped.

Functions

Format

TRQTIME(<axis 1>)

Values <axis 1>......................................main group: 1 to 6

Explanation Acquires the torque limit time-out period for the axis specified by <axis number>. The time-out period is specified in "ms" units.

112 TRQTIME

SAMPLE

TRQTIME(3)=2500	Sets the torque control time-out period as 2.5 seconds for axis 3.
DRIVE(3,P1),T=(20,15)	Sets the maximum torque value to 20% of the rated torque, and the torque offset to 15, then moves axis 3 from its current position to the point specified by P1 (pushing action).
IF TRQSTS(3)=1 THEN	Checks if a time-out has occurred.
DO(21)=1	Time-out has occurred (pushing is complete). (Result is output to DO(21) in this example.)
ELSE	
DO(21)=0	Time-out has not occurred. (Reached target position but failed to complete pushing.) (Result is output to DO(21) in this example.)
ENDIF	
TORQUE(3)=100	Returns the max. torque command value to the original value (100%).
DRIVE(3,P0)	Ends the torque limit and torque control, and moves to P0.
A%=TRQTIME(3)	The torque limit time-out period for the main group's axis 3 is assigned to variable A.

Related commands	DRIVE, TRQSTS, CURTRQ

113 VAL

Converts character strings to numeric values

Format

VAL (<character string expression>)

Explanation Converts the numeric value of the character string specified in the <character string expression> into an actual numeric value.
The value may be expressed in integer format (binary, decimal, hexadecimal), or real number format (decimal point format, exponential format).
The VAL value becomes "0" if the first character of the character string is "+", "-", "&" or anything other than a numeric character.
If there are non-numeric characters or spaces elsewhere in the character string, all subsequent characters are ignored by this function.
However, for hexadecimal expressions, A to F are considered numeric characters.

SAMPLE

```
A=VAL("&B100001")
```

114 WAIT

Waits until the conditions of the DI/DO conditional expression are met

Format
WAIT <conditional expression> [,<expression>]

Values <expression> ....10 to 3600000 (units: ms)

Meaning Establishes a "wait" status until the condition specified by the <conditional expression> is met. Specify the time-out period (unit: ms) in the <expression>.
If a time-out period has been specified, this command terminates if the time-out period elapses before the WAIT condition is met.
The minimum wait time is 10 ms.

MEMO

- When the conditional expression is a numeric expression, the conditions for determining a TRUE or FALSE status can be changed at the controller's "TRUE conditions" in the "Other parameters" mode. These conditions apply to all the IF, WHILE, WAIT, STOPON, etc., conditional expressions. For details, refer to the controller's user manual.

1) -1 (default setting)	An expression value of "-1" indicates a TRUE status, and "0" indicates a FALSE status. A "6.35: Incorrect condition expression" error occurs if the expression value is other than "-1" or "0".
2) not 0	Any expression value other than "0" indicates a TRUE status, and "0" indicates a FALSE status.

SAMPLE

```
WAIT A=10.............................................................A wait status continues until variable A becomes 10.
WAIT DI2( )=&B01010110......................Waits until DI(21),(22),(24),(26) are turned on, and
                                                                      DI(20),(23),(25),(27) is turned off.
WAIT DI2(4,3,2)=&B101......................................Waits until DI(22) and DI(24) are turned on, and DI(23)
                                                                      is turned off.
WAIT DI(31)=1 OR DO(21)=1..............................A wait status continues until either DI (31) or DO(21) turns ON.
WAIT DI(20)=1,1000.............................................A wait status continues until DI(20) turns ON. If DI(20) fails
                                                                      to turn ON within 1 second, the command is terminated.
```

Related commands	DRIVE, DRIVEI, MOVE, MOVEI

115 WAIT ARM

Waits until the robot axis operation is completed

Format
WAIT ARM [(<axis number>)]

Values <axis number>............................main group: 1 to 6

Explanation Establishes a "wait" status until robot axis movement is completed (within the positioning tolerance range).

If a specific axis in a group has been specified by <axis number>, this command will apply only to that axis. If there is no <axis number> setting, this command applies to all the group axes.

SAMPLE

```
WAIT ARM...........................................................Waits for main robot movement completion.
```

Related commands	DRIVE, DRIVEI, MOVE, MOVEI

116 WEIGHT

Specifies/acquires the tip weight parameter

Format

WEIGHT <expression>

Values <expression> ....The range varies according to the robot which has been specified.

Explanation Changes the tip weight parameter of the main robot to the <expression> value. This change does not apply to auxiliary axes.

Functions

Format

WEIGHT

Explanation Acquires the tip weight parameter of the robot.

SAMPLE

```
A=5
B=2
C=WEIGHT
WEIGHT A
MOVE P,P0
WEIGHT B
MOVE P,P1
WEIGHT C
D=WEIGHT.........................................................The main robot's tip weight parameter is assigned to
                                                                 variable D.
HALT
```

117 WEND

Ends the WHILE statement's command block

Format

```
WHILE  <conditional expression>
        <command block>
WEND
```

Explanation Ends the command block which begins with the WHILE statement. A WEND statement must always be paired with a WHILE statement.
Jumping out of the WHILE to WEND loop is possible by using the GOTO statement, etc.

SAMPLE

```
A=0
WHILE DI3(0)=0
    A=A+1
    MOVE P,P0
    MOVE P,P1
    PRINT "COUNTER=";A
WEND
HALT
```

Related commands	WHILE

118 WHERE

Acquires the arm's current position (pulse coordinates)

Format

WHERE

Explanation Acquires the arm's current position in joint coordinates.

SAMPLE

P10=WHERE ....The current position's pulse coordinate value is assigned to P10.

Related commands	WHRXY

119 WHILE to WEND

Repeats an operation for as long as a condition is met

Format

```
WHILE   <conditional expression>
          <command block>
WEND
```

Explanation Executes the command block between the WHILE and WEND statements when the condition specified by the <conditional expression> is met, and then returns to the WHILE statement to repeat the same operation.

When the <conditional expression> condition is no longer met (becomes false), the program jumps to the next command after the WEND statement.

If the <conditional expression> condition is not met from the beginning (false), the command block between the WHILE and WEND statements is not executed, and a jump occurs to the next statement after the WEND statement.

Jumping out of the WHILE to WEND loop is possible by using the GOTO statement, etc.

MEMO

- When the conditional expression is a numeric expression, the conditions for determining a TRUE or FALSE status can be changed at the controller's "TRUE conditions" in the "Other parameters" mode. These conditions apply to all the IF, WHILE, WAIT, STOPON, etc., conditional expressions. For details, refer to the controller's user manual.

1) -1 (default setting)	An expression value of "-1" indicates a TRUE status, and "0" indicates a FALSE status. A "6.35: Incorrect condition expression" error occurs if the expression value is other than "-1" or "0".
2) not 0	Any expression value other than "0" indicates a TRUE status, and "0" indicates a FALSE status.

SAMPLE 1

```
A=0
WHILE DI3(0)=0
    A=A+1
    MOVE P,P0
    MOVE P,P1
    PRINT "COUNTER=";A
WEND
HALT
```

SAMPLE 2

```
A=0
WHILE -1 ..........................................Becomes an endless loop because the conditional expression is always TRUE (-1).
    A=A+1
    MOVE P,P0
    IF DI3(0)=1 THEN *END
    MOVE P,P1
    PRINT "COUNTER=";A
    IF DI3(0)=1 THEN *END
WEND
*END
HALT
```

120 WHRXY

Acquires the arm's current position in Cartesian coordinates

NOTE

- X-arm and Y-arm rotation information is only available in software Ver.1.66M or higher.

Format

```
WHRXY
```

Explanation Acquires the arm's current position in Cartesian coordinates.
On R6YXTW500 model robots, the X-arm and Y-arm rotation information is also set.

SAMPLE

```
P10=WHRXY.......................................................The current position Cartesian coordinate value is assigned
                                                                 to P10.
```

Related commands	WHERE

121 XYTOJ

Converts the main group axes Cartesian coordinate data ("mm") to joint coordinate data ("pulse")

NOTE

- X-arm and Y-arm rotation information is only available in software Ver.1.66M or higher.

Format

XYTOJ (<point expression>)

Explanation This function converts the Cartesian coordinate data (unit: mm, deg.) specified by the <point expression> to joint coordinate data (unit: pulses).

- When the command is executed, the data is converted based on the standard coordinates, shift coordinates and hand definition that were set.
- The converted result differs depending on whether right-handed or left-handed is specified.
- On the R6YXTW500 model robot, the result varies, depending on the X-arm and Y-arm rotation information settings.
- To convert joint coordinate data to Cartesian coordinate data, use the JTOXY statement.

SAMPLE

```
P10=XYTOJ(P10)....................................P10 is converted to joint coordinate data.
```

122 _SYSFLG

Axis status monitoring flag

Format

_SYSFLG

Explanation Used as an axis status monitoring flag in accordance to the value specified by the _SYSFLG variable.

SAMPLE

```
_SYSFLG = 1
```

Related commands	RESET

Chapter 8

PATH Statements

1 Overview 8-1
2 Features 8-1
3 How to use 8-1
4 Cautions when using this function......................8-2

1 Overview

This function moves the robot at a specified speed along a path composed of linear and circular segments. Because speed fluctuations during movement are minimal, the PATH function is ideal for applications such as sealing, etc.

2 Features

- Moves the robot at a constant speed along the entire movement path (except during acceleration from a stop, and during deceleration just prior to the operation end).
- Permits easy point teaching because the robot speed is not affected by the point teaching positions' level of precision.
- Permits movement speed changes for the entire movement path, or speed changes for only one portion of the path (using the speed option).
- Using the DO option permits signal outputs to a specified port at any desired position during movement.

3 How to use

The following robot language commands must be used as a set in order to use the PATH function.

- PATH SET .... Start of path setting.
- PATH (PATH L, PATCH C) .... Specifies the path to be used.
- PATH END .... End of path setting.
- PATH START.. Starts actual movement along the path.

As shown below, the motion path is specified between the PATH SET and PATH END statements. Simply specifying a path, however, does not begin robot motion.
Robot motion only occurs when the PATH START statement is executed after the path setting procedure has been completed.

SAMPLE

```
MOVE P,P0,Z=0
PATH SET        ·························  Start of path setting
PATH L,P1,DO(20)=1@10.0
PATH L,P2
    •
    •
    •
PATH C,P12,P13
PATH L,P14,DO(20)=0@20.0
PATH END        ·························  End of path setting
MOVE P,P1,Z=0
    •
    •
    •
MOVE P,P0,Z=0
PATH START   Path motion is executed
HALT
```

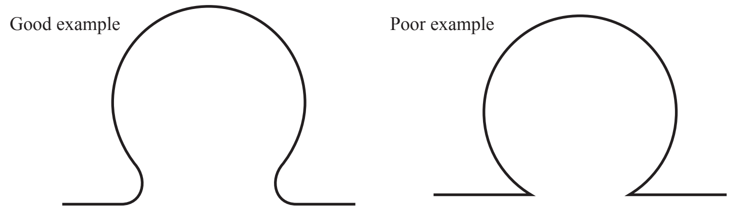
4 Cautions when using this function

- Paths may comprise no more than 300 (total) linear and circular segments.

- The robot must be positioned at the path start point when PATH motion is executed (by PATH START statement).

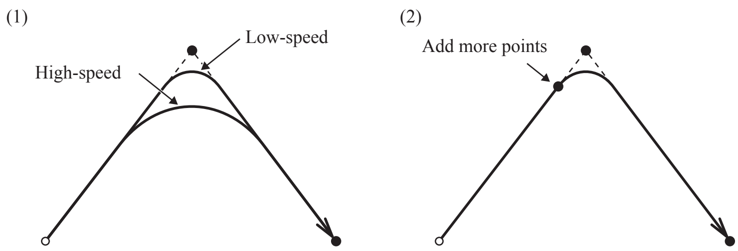
- At points where circular and linear segments connect, the motion direction of the two connecting segments should be a close match (as close as possible). An excessive difference in their motion directions could cause vibration and robot errors.

Circular and linear segment connection point:
if there is a large difference between the motion directions of the connecting segments



- Where a linear segment connects to another linear segment, the motion path passes to the inner side of the connection point. Moreover, as shown in fig. (1) below, the faster the speed, the further to the inner side the path becomes. To prevent significant speed-related path shifts, add more points as shown in fig. (2). Note also, that in some cases, the speed may have to be reduced in order to prevent errors from occurring.

Connection point of 2 linear segments:
suppressing the path shift



- If an error occurs due the robot's inability to move at the specified speed:
  Robot acceleration/deceleration occurs if the speed setting is changed when PATH motion begins, stops, or at some point along the path. At such times, an error may occur before motion begins if the distance between points is too short for the specified speed to be reached. In such cases, a slower speed must be specified. If the error still occurs after the speed is lowered, adjust the PATH points to increase the length of the linear or circular segments which contain acceleration or deceleration zones.

- The hand system used during PATH motion must be the same as the hand system used at the path's start point. The same applies if the path is to pass through points where hand flags are set. Differing hand systems will cause an error and disable motion.

- The X-arm and Y-arm rotation information during PATH movement must be the same as the X-arm and Y-arm rotation information at the PATH movement's START point. If the two are different, an error will occur and movement will be disabled.

- If the robot is stopped by an interlock function, etc., during PATH motion, this is interpreted as an execution termination, and the remaining path motion will not be completed even if a restart is executed.

Be sure to read the cautions relating to each command.

Chapter 9

Limitless motion

1 Overview 9-1

2 Operation Procedure 9-1

3 Restrictions 9-3

1 Overview

NOTE

- The limitless motion function is available in software version 1.66M or higher.

Generally speaking, controllers have a soft limit function which allows the soft limits to be specified by parameter settings, and operation beyond the soft limits is normally prohibited. However, the "limitless motion" function permits multi-turn same-direction movement without that soft limit restriction.

2 Operation Procedure

2.1 Parameters

The "limitless motion" parameter can be enabled in the robot axis parameters.
(For details, see the controller user's manual.)

2.2 Robot language

CAUTION

- Limitless motion applies to the axis which has been specified as the additional axis in the "system generation" settings (robot factory settings).
- Do not attempt to enable limitless motion at an axis which has not been specified as an additional axis. Doing so will result in the "2.29: Cannot move without the limit" error when movement is attempted using the MOVE statement or a point trace movement. This error also disables movement.

The robot language shown below is required in order to use the limitless motion function. For command details, see Chapter 8 "Robot Language Lists".

Robot movement	DRIVE statement (PLS or MNS option specified)
	DRIVEI statement
Current position reset	ABSINIT statement

8

9

10

11

12

13

14

2.3 Sample program

2.3.1 For Axis 4 limitless motion

The following program executes limitless movement in the plus direction, in 180.0° increments.
(Settings: Resolver pulse: 16384, speed reduction ratio: 25, limitless motion specified for the main robot's axis 4, minus axis polarity.)

P0 = 0.0 0.0 0.0 0.9 0.0 0.0.........................0.9°=1024[pulse]

P1 = 0.0 0.0 0.0 180.9 0.0 0.0

* It is recommended that point data be created so that the position to reset the current position is at the middle of the resettable range.

Axis polarity	Recommended reset range
For a "minus" setting	Approximately 1024 ± 256 [pulse]
For a "plus" setting	Approximately -1024 ± 256 [pulse]

SAMPLE

```
WHILE -
DRIVE(4,P1),PLS
WAIT ARM(4)
P100 = WHERE
A%=LOCR(P100) MOD 409600 ・・・・・409600 [pulse] = Machine angle 360°.
IF(A%<257) OR (A%>1791) THEN ・・・Checks current position to see if
                                   it is the reset execution position.
DRIVE(4,P0),PLS ・・・・・・・・・・・・・・・・・Moves to the current position reset execution position.
WAIT ARM(4)
ENDIF
ABSINIT 4 ・・・・・・・・・・・・・・・・・・・・・・Resets the main robot's axis 4 current position.
WEND
HALT
```

2.3.2 For Axis 2 limitless motion

The following program executes limitless movement in the plus direction, in 180.0° increments.
(Settings: Resolver pulse: 16384, speed reduction ratio: 25, limitless motion specified for the main robot's axis 2, minus axis polarity.)

P0 = 0.0 0.9 0.0 0.0 0.0 0.0.........................0.9°=1024[pulse]

P1 = 0.0 180.0 0.0 0.0 0.0 0.0

* It is recommended that point data be created so that the position to reset the current position is at the middle of the resettable range.

Axis polarity	Recommended reset range
For a "minus" setting	Approximately 1024 ± 256 [pulse]
For a "plus" setting	Approximately -1024 ± 256 [pulse]

SAMPLE

```
WHILE -1
DRIVE(2,P1),PLS
WAIT ARM(2)
P100 = WHERE
A%=LOCY(P100) MOD 409600 ・・・・・409600 [pulse] = Machine angle 360°.
IF(A%<257) OR (A%>1791) THEN ・・・Checks current position to see if
                                   it is the reset execution position.
DRIVE(2,P0),PLS ・・・・・・・・・・・・・・・・・Moves to the current position reset execution position.
WAIT ARM(2)
ENDIF
ABSINIT 2 ・・・・・・・・・・・・・・・・・・・・・・Resets the main robot's axis 2 current position.
WEND
HALT
```

3 Restrictions

- OMRON recommends that the return-to-origin method be specified as "sensor" in the axis parameters.
- The limitless motion function cannot be used at YC-Link specification axes.
- The limitless motion function cannot be used at electric gripper specification axes.
- The limitless motion function cannot be used at SCARA robot X and Y axes.

Chapter 10

Data file description

1	Overview	10-1
2	Program file	10-2
3	Point file	10-4
4	Point comment file	10-8
5	Parameter file	10-10
6	Shift coordinate definition file	10-13
7	Hand definition file	10-15
8	Pallet definition file	10-17
9	All file	10-23
10	Program directory file	10-24
11	Parameter directory file	10-26
12	Variable file	10-27
13	Constant file	10-30
14	Array variable file	10-31
15	DI file	10-33
16	DO file	10-35
17	MO file	10-37
18	LO file	10-39

19 TO file 10-41
20 SI file 10-43
21 SO file 10-45
22 Error message history file 10-47
23 Error Message History Details File.................10-48
24 Machine reference file....................................10-49
25 EOF file 10-50
26 Serial port communication file........................10-51
27 SIW file 10-52
28 SOW file..10-54
29 Ethernet port communication file....................10-56

1 Overview

1.1 Data file types

This section explains data files used with a SEND statement and READ/WRITE online commands. There are 27 different types of data files.

1. Program file
2. Point file
3. Point comment file
4. Parameter file
5. Shift coordinate definition file
6. Hand definition file
7. Pallet definition file
8. All file
9. Program directory file
10. Parameter directory file
11. Variable file
12. Constant file
13. Array variable file
14. DI file
15. DO file
16. MO file
17. LO file
18. TO file
19. SI file
20. SO file
21. Error message history file
22. Machine reference file
23. EOF file
24. Serial port communication file
25. SIW file
26. SOW file
27. Ethernet port communication file

1.2 Cautions

Observe the following cautions when handling data files.

- Only 1-byte characters can be used.
- All data is handled as character strings conforming to ASCII character codes.
- Only upper case alphabetic characters may be used in command statements (lower case characters are prohibited).
- Line lengths must not exceed 75 characters.
- A [cr/lf] data format designation indicates CR code (0Dh) + LF code (0Ah).
- The terms "reading" and "writing" used in this manual indicate the following data flow directions:
  Reading: controller → external communication device
  Writing: External communication device → controller

2 Program file

2.1 All programs

Format
PGM

Meaning

- Expresses all programs.
- When used as a readout file, all programs currently stored are read out.
- Write files are registered at the controller under the program name indicated at the NAME = <program name> line.

DATA FORMAT

```
NAME = <program name> [cr/lf]
aaaaa ...aaaaaaaaaaaaaa[cr/lf]
             :
aaaaa ...aaaaaaaaaaaaaa[cr/lf]
             :
NAME = <program name> [cr/lf]
aaaaa ...aaaaaaaaaaaaaa[cr/lf]
             :
aaaaa ...aaaaaaaaaaaaaa[cr/lf]
[cr/lf]
```

Values a ....Character code

- <Program names> are shown with 8 characters or less consisting of alphanumeric characters and underscore (_).
- A line containing only [cr/lf] is added at the end of the file, indicating the end of the file.
- A TAB code (09H) is converted to a space.

SAMPLE

```
SEND PGM TO CMU ···············Outputs all programs from communication port.
SENDCMU TO PGM  ···············Inputs all programs from communication port.

Response:
NAME=TEST[cr/lf]
A=1[cr/lf]
RESET DO2()[cr/lf]
  :
HALT[cr/lf]
[cr/lf]
```

2.2 One program

Format

<program name>

Meaning

- Expresses a specified program.
- Program name may be up to 8 characters consisting of alphanumeric characters and underscore "_" and must be enclosed by "<" and ">".
- If no program name is specified, the currently selected program is specified.
- An error occurs if the specified program name differs from the program name on the data.

DATA FORMAT

```
NAME=program name[cr/lf]
aaaaa ...aaaaaaaaaaaaaa[cr/lf]
            :
aaaaa ...aaaaaaaaaaaaaa[cr/lf]
[cr/lf]
```

Values a ....Character code

- <Program names> are shown with 8 characters or less consisting of alphanumeric characters and underscore (_).
- A line containing only [cr/lf] is added at the end of the file, indicating the end of the file.
- A TAB code is converted to a space.

MEMO

- At program writing operations, be sure to use the NAME statement to specify the program name. Program writing cannot occur if the program name is not specified.
- When the current mode is "AUTO" or "PROGRAM" mode, and writing into the currently selected program is not possible.
- When a sequence program is being executed, writing into the program name "SEQUENCE" is not possible.

SAMPLE

```
SEND <TEST1> TO CMU・・・・・・・・・・・・・・・・Outputs the program "TEST1" from communication port.
SEND CMU TO <TEST1>・・・・・・・・・・・・・・・・Inputs the program "TEST1" from communication port

Response:
NAME=TEST1[cr/lf]
A=1[cr/lf]
RESET DO2()[cr/lf]
       :
HALT[cr/lf]
[cr/lf]
```

3 Point file

3.1 All points

Format
PNT

Meaning
- Expresses all point data.
- When used as a readout file, all points currently stored are read out.
- When used as a write file, writing is performed with a point number.

DATA FORMAT (On robots other than R6YXTW500)

```
Pmmmm=fxxxxxx fyyyyyy fzzzzzz frrrrrr faaaaaa fbbbbbb t[cr/lf]
Pmmmm=fxxxxxx fyyyyyy fzzzzzz frrrrrr faaaaaa fbbbbbb t[cr/lf]
        :
Pmmmm=fxxxxxx fyyyyyy fzzzzzz frrrrrr faaaaaa fbbbbbb t[cr/lf]
Pmmmm=fxxxxxx fyyyyyy fzzzzzz frrrrrr faaaaaa fbbbbbb t[cr/lf]
[cr/lf]
```

DATA FORMAT (On robot R6YXTW500, with software Ver.1.66M or higher)

```
Pmmmm= fxxxxxx fyyyyyy fzzzzzz frrrrrr faaaaaa fbbbbbb t xr yr [cr/lf]
Pmmmm= fxxxxxx fyyyyyy fzzzzzz frrrrrr faaaaaa fbbbbbb t xr yr [cr/lf]
        :
Pmmmm= fxxxxxx fyyyyyy fzzzzzz frrrrrr faaaaaa fbbbbbb t xr yr [cr/lf]
Pmmmm= fxxxxxx fyyyyyy fzzzzzz frrrrrr faaaaaa fbbbbbb t xr yr [cr/lf]
[cr/lf]
```

NOTE
- Integer point data is recognized in pulse units, and real number point data is recognized in "mm" units.
- The X-arm and Y-arm rotation information is only available in software version 1.66M or higher.

Values

mmmm .... Point No.: 0 to 9999
f................................. Coordinate sign: + / - / space
xxxxxx/../bbbbbb....... Represent a numeric value of 8 digits or less. When a dot is included, this is treated as point data in "mm" units. Each piece of data is separated by one or more spaces.
t................................. Extended hand system flag setting for SCARA robots. 1: right hand system; 2: left hand system.
xr................................ Extended setting's X-arm rotation information.
0: The "mm → pulse" converted angle data x (*1) range is –180.00° < x < = 180.00°.
1: The "mm → pulse" converted angle data x (*1) range is 180.00° < x < = 540.00°.
-1: The "mm → pulse" converted angle data x (*1) range is -540.00° < x < = -180.00°.
yr................................ Extended setting's Y-arm rotation information.
0: The "mm → pulse" converted angle data x (*1) range is –180.00° < y < = 180.00°.
1: The "mm → pulse" converted angle data x (*1) range is 180.00° < y < = 540.00°.
-1: The "mm → pulse" converted angle data x (*1) range is -540.00° < y < = -180.00°.

*1: The joint-coordinates-converted pulse data represents each arm's distance (converted to angular data) from its mechanical origin point.

- Hand system flags are valid only for SCARA robots, with the coordinate data specified in "mm" units.
- If a number other than "1" or "2" is specified for a hand system flag, or if no number is specified, this is interpreted as "0" setting (no hand system flag).

- X-arm and Y-arm rotation information settings are available only on the R6YXTW500 robot model where a "mm" units coordinate system has been set.
- X-arm and Y-arm rotation information is processed as "0" if a numeral other than 0, 1, -1 has been specified, or if no numeral has been specified.
- A line containing only [cr/lf] is added at the end of the file to indicate the end of the file.

SAMPLE (On robots other than R6YXTW500)

SEND PNT TO CMU・・・・・・・・・・・・・・・・Outputs all points from communication port.
SEND CMU TO PNT・・・・・・・・・・・・・・・・Inputs all points from communication port.

Response:

P0 =	1	2	3	4	5	6	[cr/lf]
P1 =	1.00	2.00	3.00	4.00	5.00	6.00	[cr/lf]
P2 =	1.00	0.00	0.00	0.00	0.00	0.00	[cr/lf]
:							
P9999=	-1.00	0.00	0.00	0.00	0.00	0.00	[cr/lf]
[cr/lf]							

SAMPLE (On robot R6YXTW500, with software Ver.1.66m or higher)

SEND PNT TO CMU・・・・・・・・・・・・・・・・Outputs all points from communication port.
SEND CMU TO PNT・・・・・・・・・・・・・・・・Inputs all points from communication port.

Response:

P0 =	1	2	3	4	5	6			[cr/lf]
P1 =	426.20	-160.77	0.01	337.21	0.00	0.00	0	1	0 [cr/lf]
P2 =	-27.57	-377.84	0.36	193.22	0.00	0.00	0	-1	0 [cr/lf]
:									
P9999=	-251.66	-419.51	0.00	-127.79	0.00	0.00	2	-1	-1 [cr/lf]
[cr/lf]									

3.2 One point

Format

Pmmmm

Meaning
- Expresses a specified point.
- "mmmm" must be from 0 to 9999

DATA FORMAT (On robots other than R6YXTW500)

Pmmmm=fxxxxxx fyyyyyy fzzzzzz frrrrrr faaaaaa fbbbbbb t[cr/lf]

DATA FORMAT (On robot R6YXTW500, with software Ver.1.66M or higher)

Pmmmm= fxxxxxx fyyyyyy fzzzzzz frrrrrr faaaaaa fbbbbbb t xr yr [cr/lf]

Values

mmmm .... Point No.: 0 to 9999

f................................. Coordinate sign: + / - / space

xxxxxx/../bbbbbb....... Represent a numeric value of 8 digits or less. When a dot is included, this is treated as point data in "mm" units. Each piece of data is separated by one or more spaces.

t................................. Extended hand system flag setting for SCARA robots. 1: right hand system; 2: left hand system.

xr................................ X-arm rotation information for the R6YXTW500 robot.
0: The "mm → pulse" converted pulse data x (*1) range is –180.00° < x < = 180.00°.
1: The "mm → pulse" converted pulse data x (*1) range is 180.00° < x < = 540.00°.
-1: The "mm → pulse" converted pulse data x (*1) range is –540.00° < x < = -180.00°.

yr................................ Y-arm rotation information for the R6YXTW500 robot.
0: The "mm → pulse" converted pulse data x (*1) range is –180.00° < y < = 180.00°.
1: The "mm → pulse" converted pulse data x (*1) range is 180.00° < y < = 540.00°.
-1: The "mm → pulse" converted pulse data x (*1) range is –540.00° < y < = -180.00°.

*1: The joint-coordinates-converted pulse data represents each arm's distance (converted to angular data) from its mechanical origin point.

NOTE
- Integers indicate point data in "pulse" units, and real numbers in "mm" units.
- The X-arm and Y-arm rotation information is only available in software version 1.66M or higher.

- Hand system flags are valid only for SCARA robots, with the coordinate data specified in "mm" units.
- If a number other than "1" or "2" is specified for a hand system flag, or if no number is specified, this is interpreted as "0" setting (no hand system flag).
- X-arm and Y-arm rotation information settings are available only on the R6YXTW500 robot model where a "mm" units coordinate system has been set.
- X-arm and Y-arm rotation information is processed as "0" if a numeral other than 0, 1, -1 has been specified, or if no numeral has been specified.

SAMPLE (On robots other than R6YXTW500)

SEND P100 TO CMU················Outputs the specified point from communication port.
SEND CMU TO P100················Inputs the specified point from communication port.

Response:
P100= 1 2 3 4 5 6[cr/lf]

SAMPLE (On robot R6YXTW500, with software Ver.1.66M or higher)

SEND P100 TO CMU················Outputs the specified point from communication port.
SEND CMU TO P100················Inputs the specified point from communication port.

Response:
P100= 1 2 3 4 5 6 0 1 0 [cr/lf]

8
9
10
11
12
13
14

4 Point comment file

4.1 All point comments

Format

PCM

Meaning
- Expresses all point comments.
- When used as a readout file, all point comments currently stored are read out.
- When used as a write file, writing is performed with a point comment number.

DATA FORMAT

```
PCmmmm= sssssssssssssss[cr/lf]
PCmmmm= sssssssssssssss[cr/lf]
        :
PCmmmm= sssssssssssssss[cr/lf]
PCmmmm= sssssssssssssss[cr/lf]
[cr/lf]
```

Values

mmmm ....Point comment number: a number from 0 to 9999

ss...ss..Comment data: which can be up to 15 one-byte characters. If comment data exceeds 15 characters, then the 16th character onwards will be deleted.

- A line containing only [cr/lf] is added at the end of the file, indicating the end of the file.

SAMPLE

SEND PCM TO CMU················Outputs all point comments from communication port.
SEND CMU TO PCM················Inputs all point comments from communication port.

Response:
PC1=ORIGIN POS[cr/lf]
PC3=WAIT POS[cr/lf]
:
PC3999= WORK100[cr/lf]
[cr/lf]

4.2 One point comment

Format

PCmmmm

Meaning
- Expresses a specified point comment.
- "mmmm" represents a number from 0 to 9999.

DATA FORMAT

PCmmmm= sssssssssssssss[cr/lf]

Values mmmm ....Point comment number: a number from 0 to 9999

ss...ss...Comment data: which can be up to 15 one-byte characters. If comment data exceeds 15 characters, then the 16th character onwards will be deleted.

SAMPLE

SEND PC1 TO CMU・・・・・・・・・・・・・・・・・Outputs the specified point comment from communication port.
SEND CMU TO PC1・・・・・・・・・・・・・・・・・Inputs the specified point comment from communication port.

Response:
PC1=ORIGIN POS[cr/lf]

5 Parameter file

5.1 All parameters

Format
PRM

Meaning
- Expresses all parameters (including settings in "UTILITY" mode).
- When used as a readout file, all parameters currently stored are read out.
- When used as a write file, only the parameters specified by parameter labels are written.

DATA FORMAT

```
/parameter label/ ' <comment> [cr/lf]
RC= xxxxxx [cr/lf]
/parameter label/   '<comment> [cr/lf]
R1= xxxxxx  R2= yyyyyy [cr/lf]
/parameter label/   '<comment> [cr/lf]
A1= xxxxxx  A2= yyyyyy  A3= zzzzzz  A4= rrrrrr[cr/lf]
/parameter label/   '<comment> [cr/lf]
        :
[cr/lf]
```

Values
RC...Indicates the entire controller.
R? ....Robot setting: ? / 1: main robot
A? ....Axis setting: ?: axis No.
<Comment> ....Parameter name

- Parameter labels are shown with 6 alphabetic characters.
- A line containing only [cr/lf] is added at the end of the file, indicating the end of the file.

MEMO
- When writing parameter data, be sure that the servo is off.
- Parameters are already compatible with upper versions. However, parameters might not always be compatible with lower versions (upward compatibility).
- When you attempt to load a parameter file of new version into a controller of an earlier version, an error "10.14 : Undefined parameter found" may appear. If this happens, you may load the parameter by setting the "Skip undefined parameters" parameter to "VALID". For more details, refer to the "SYSTEM mode" – "Other parameters" section in the robot controller user's manual.

SAMPLE

```
SEND PRM TO CMU················Outputs all parameters from communication port.
SEND CMU TO PRM ················Inputs all parameters from communication port.

Response:
/RBTNUM/ 'Robot number (V8.01/R1001)[cr/lf]
 R1=            3000    R2=         3010    [cr/lf]
/AXES  / 'Number of axes[cr/lf]
 R1=               2    R2=            2    [cr/lf]
/AXSNUM/ 'Axis number (V1.01/V1.01/V1.01/V1.01/-----/-----/-----/-----/)[cr/lf]
 A1=            5000    A2=         5001    A3=         5010    A4=         5011 [cr/lf]
 A5=               0    A6=            0    A7=            0    A8=            0 [cr/lf]
/ATTRIB/ 'Axis attribute[cr/lf]
 A1=           33792    A2=        33792    A3=        33792    A4=        33792 [cr/lf]
 A5=             256    A6=          256    A7=          256    A8=          256 [cr/lf]
/WEIGHT/ 'Tip weight[kg][cr/lf]
 R1=               2    R2=           12    [cr/lf]
       :
/CURPNO/ 'Port number of output[cr/lf]
  RC=      20    [cr/lf]
/CURPT1/ 'Compare point number1[cr/lf]
  RC=       0    [cr/lf]
/CURPT2/ 'Compare point number2[cr/lf]
  RC=       0    [cr/lf]
[cr/lf]
```

8
9
10
11
12
13
14

5.2 One parameter

Format

/parameter label/

Meaning

- Parameter labels are shown with 6 alphabetic characters.
- When used as a readout file, only the parameter specified by a parameter label is read out.
- When used as a write file, only the parameter specified by a parameter label is written.

DATA FORMAT 1

/parameter label/ '<comment>[cr/lf]
RC= xxxxxx [cr/lf]
[cr/lf]

DATA FORMAT 2

/parameter label/ '<comment>[cr/lf]
R?= xxxxxx [cr/lf]
[cr/lf]

DATA FORMAT 3

/parameter label/ '<comment>[cr/lf]
A?= xxxxxx[cr/lf]
[cr/lf]

Values

RC..Indicates the entire controller.
R? ....Robot setting: ? / 1: main robot
A? ....Axis setting: ?: axis No.
<Comment> ....Parameter name

- Parameter labels are shown with 6 alphabetic characters.
- A line containing only [cr/lf] is added at the end of the file, indicating the end of the file.

MEMO

- When writing parameter data, be sure that the servo is off.
- Parameters are already compatible with upper versions. However, parameters might not always be compatible with lower versions (upward compatibility).
- When you attempt to load a parameter file of new version into a controller of an earlier version, an error "10.14 : Undefined parameter found" may appear.

SAMPLE

SEND /ACCEL / TO CMU･････････････････Outputs the acceleration parameter from communication port.
SEND CMU TO /ACCEL /･････････････････Inputs the acceleration parameter from communication port.

Response:
/ACCEL / 'Accel coefficient[%]
A1= 100 A2= 100 A3= 100 A4= 100[cr/lf]
[cr/lf]

6 Shift coordinate definition file

6.1 All shift data

Format
SFT

Meaning
- Expresses all shift data.
- When used as a readout file, all shift data currently stored are read out.
- When used as a write file, writing is performed with a shift number.

DATA FORMAT

Sm = fxxxxxx fyyyyyy fzzzzzz frrrrr [cr/lf]
SPm = fxxxxxx fyyyyyy fzzzzzz frrrrr [cr/lf]
SMm = fxxxxxx fyyyyyy fzzzzzz frrrrr [cr/lf]
:
Sm = fxxxxxx fyyyyyy fzzzzzz frrrrr [cr/lf]
SPm = fxxxxxx fyyyyyy fzzzzzz frrrrr [cr/lf]
SMm = fxxxxxx fyyyyyy fzzzzzz frrrrr [cr/lf]
[cr/lf]

Values

m..Shift No.: 0 to 9
f...Coordinate sign: + / - / space
xxxxxx/yyyyyy/../rrrrrr...............Represent a numeric value of 8 digits or less, having 2 or less places below the decimal point.

- The SPm and SMm inputs are optional in writing files.
  SPm: shift coordinate range plus-side; SMm: shift coordinate range minus-side
- A line containing only [cr/lf] is added at the end of the file, indicating the end of the file.

SAMPLE

SEND SFT TO CMU・・・・・・・・・・・・・・・・Outputs all shift data from communication port.
SEND CMU TO SFT・・・・・・・・・・・・・・・・Inputs all shift data from communication port.

Response:

S0 =	0.00	0.00	0.00	0.00	[cr/lf]
SP0=	0.00	0.00	0.00	0.00	[cr/lf]
SM0=	0.00	0.00	0.00	0.00	[cr/lf]
S1 =	1.00	1.00	1.00	1.00	[cr/lf]
:					
SM9=	9.00	9.00	9.00	9.00	[cr/lf]
[cr/lf]					

6.2 One shift definition

Format
Sm

Meaning • Expresses a specified shift definition.

DATA FORMAT
Sm = fxxxxxx fyyyyyy fzzzzzz frrrrrr[cr/lf]

Values

m..Shift No.: 0 to 9

f..Coordinate sign: + / - / space

xxxxxx/yyyyyy/../rrrrrr...............Represent a numeric value of 8 digits or less, having 2 or less places below the decimal point.

SAMPLE

SEND S0 TO CMU・・・・・・・・・・・・・・・・Outputs the specified shift coordinate from communication port.
SEND CMU TO S0・・・・・・・・・・・・・・・・Inputs the specified shift coordinate from communication port.

Response:
S0 =0.00 0.00 0.00 0.00[cr/lf]
SP0=0.00 0.00 0.00 0.00[cr/lf]
SM0=0.00 0.00 0.00 0.00[cr/lf]
[cr/lf]

7 Hand definition file

7.1 All hand data

Format

HND

Meaning

- Expresses all hand data.
- When used as a readout file, all hand data currently stored are read out.
- When used as a write file, writing is performed with a hand number.

DATA FORMAT

```
Hm = fxxxxxx fyyyyyy fzzzzzz {R}[cr/lf]
   :
Hm = fxxxxxx fyyyyyy fzzzzzz {R}[cr/lf]
[cr/lf]
```

Values

m..Hand number / 0 to 3: used for main robot

f..Coordinate sign: + / - / space

xxxxxx/yyyyyy/zzzzzz..............Represent a numeric value of 8 digits or less, having 2 or less places below the decimal point, or an integer of 7 digits or less. (This numeric format depends on the robot type setting and hand definition type.)

{R} ....Indicates whether a hand is attached to the R-axis.

■ A line containing only [cr/lf] is added at the end of the file, indicating the end of the file.

SAMPLE

SEND HND TO CMU················Outputs all hand data from communication port.
SEND CMU TO HND················Inputs all hand data from communication port.

Response:

```
H0 = 0.00 0.00 0.00 [cr/lf]
H1 = 1.00 1.00 1.00 [cr/lf]
H2 = 2.00 2.00 2.00 [cr/lf]
H3 = 3.00 3.00 3.00 [cr/lf]
H4 = 4.00 4.00 4.00 [cr/lf]
H5 = 5.00 5.00 5.00 [cr/lf]
H6 = 6.00 6.00 6.00 [cr/lf]
H7 = 7.00 7.00 7.00 [cr/lf]
[cr/lf]
```

8 9 10 11 12 13 14

7.2 One hand definition

Format

Hm

Meaning • Expresses a specified hand definition.

DATA FORMAT

Hm = fxxxxxx fyyyyyy fzzzzzz {R}[cr/lf]

Values

m...Hand number / 0 to 3: used for main robot
f...Coordinate sign: + / - / space
xxxxxx/yyyyyy/zzzzzz ....Represent a numeric value of 8 digits or less, having 2 or less places below the decimal point, or an integer of 7 digits or less. (This numeric format depends on the robot type setting and hand definition type.)
{R} ....Indicates whether a hand is attached to the R-axis.

SAMPLE

SEND H3 TO CMU・・・・・・・・・・・・・・・・Outputs the specified hand definition data from communication port.
SEND CMU TO H3・・・・・・・・・・・・・・・・Inputs the specified hand definition data from communication port.

Response:
H3 = 3.00 3.00 3.00[cr/lf]

8 Pallet definition file

8.1 All pallet definitions

Format
PLT

Meaning

- Expresses all pallet definitions.
- When used as a readout file, all pallet definitions currently stored are read out.
- When used as a write file, writing is performed with a pallet number.

DATA FORMAT (On robots other than R6YXTW500)

```
PLm [cr/lf]
PLN =  XY [cr/lf]
NX  =  nnn [cr/lf]
NY  =  nnn [cr/lf]
NZ  =  nnn [cr/lf]
P[1] = fxxxxxx fyyyyyy fzzzzzz frrrrrr faaaaaa fbbbbbb t[cr/lf]
             :
P[5] = fxxxxxx fyyyyyy fzzzzzz frrrrrr faaaaaa fbbbbbb t[cr/lf]
PLm [cr/lf]
             :
[cr/lf]
```

DATA FORMAT (On robot R6YXTW500, with software Ver.1.66M or higher)

```
PLm [cr/lf]
PLN =  XY [cr/lf]
NX  =  nnn [cr/lf]
NY  =  nnn [cr/lf]
NZ  =  nnn [cr/lf]
P[1] = fxxxxxx fyyyyyy fzzzzzz frrrrrr faaaaaa fbbbbbb t xr yr[cr/lf]
             :
P[5] = fxxxxxx fyyyyyy fzzzzzz frrrrrr faaaaaa fbbbbbb t xr yr[cr/lf]
PLm [cr/lf]
             :
[cr/lf]
```

Values

m........................Pallet number: 0 to 19
nnn.....................Number of axis points: positive integer
f..........................Coordinate sign: + / - / space
xxxxxx/yyyyyy/../bbbbbb...........
Represent a numeric value of 8 digits or less. When a dot is included, this is treated as coordinate data in "mm" units. Each piece of data is separated by one or more spaces.
t..........................An extended hand system flag setting for SCARA robots. 1: Right-handed system, 2: Left-handed system
xr........................Extended setting for X-arm rotation information.
0: "mm" → pulse converted angle data x (*1) range: -180.00°< x <= 180.00°
1: "mm" → pulse converted angle data x (*1) range: 180.00° < x <= 540.00°
-1: "mm" → pulse converted angle data x (*1) range: -540.00° < x <= -180.00°

yr.........................Extended setting for Y-arm rotation information.

0: "mm" → pulse converted angle data x (*1) range: -180.00° < y <= 180.00°

1: "mm" → pulse converted angle data x (*1) range: 180.00° < y <= 540.00°

-1: "mm" → pulse converted angle data x (*1) range: -540.00° < y <= -180.00°

*1: The joint-coordinates-converted pulse data represents each arm's distance (converted to angular data) from its mechanical origin point.

NOTE

- X-arm and Y-arm rotation information is only available in software Ver.1.66M or higher.

- Hand system flags are enabled only when specifying the coordinate data in "mm" units for SCARA robots.
- Hand system flags and the X-arm and Y-arm rotation information are ignored during movement where pallet definitions are used.
- If a number other than 1 or 2 is set, or if no number is designated, then 0 will be set to indicate that there is no hand system flag.
- X-arm and Y-arm rotation information settings are available only on the R6YXTW500 robot model where a "mm" units coordinate system has been set.
- If a value other than "0", "1", "-1" is specified at the X-arm and Y-arm rotation information, or if no value is specified, this will be processed as "0".
- A line containing only [cr/lf] is added at the end of the file, indicating the end of the file.

SAMPLE

SEND PLT TO CMU・・・・・・・・・・・・・・・・Outputs all pallet definitions from communication port.
SEND CMU TO PLT・・・・・・・・・・・・・・・・Inputs all pallet definitions from communication port.
Response:
PL0[cr/lf]
PLN=XY[cr/lf]
NX = 3 [cr/lf]
NY = 4 [cr/lf]
NZ = 2 [cr/lf]

P[1]=	0.00	0.00	0.00	0.00	0.00	0.00	[cr/lf]
P[2]=	100.00	0.00	0.00	0.00	0.00	0.00	[cr/lf]
P[3]=	0.00	100.00	0.00	0.00	0.00	0.00	[cr/lf]
P[4]=	100.00	100.00	0.00	0.00	0.00	0.00	[cr/lf]
P[5]=	0.00	0.00	50.00	0.00	0.00	0.00	[cr/lf]

PL1[cr/lf]
PLN= XY[cr/lf]
NX = 3[cr/lf]
NY = 4[cr/lf]
NZ = 2[cr/lf]

P[1]=	0.00	0.00	0.00	0.00	0.00	0.00	[cr/lf]
P[2]=	100.00	100.00	0.00	0.00	0.00	0.00	[cr/lf]
P[3]=	0.00	200.00	0.00	0.00	0.00	0.00	[cr/lf]
P[4]=	100.00	200.00	0.00	0.00	0.00	0.00	[cr/lf]
P[5]=	0.00	0.00	100.00	0.00	0.00	0.00	[cr/lf]

[cr/lf]

8.2 One pallet definition

Format

PLm

Meaning

- Expresses a specified pallet definition.
- "m" must be from 0 to 19.

DATA FORMAT (On robots other than R6YXTW500)

```
PLm [cr/lf]
PLN = XY [cr/lf]
NX  = nnn [cr/lf]
NY  = nnn [cr/lf]
NZ  = nnn [cr/lf]
P[1] = fxxxxxx fyyyyyy fzzzzzz frrrrrr faaaaaa fbbbbbb t[cr/lf]
    :
P[5] = fxxxxxx fyyyyyy fzzzzzz frrrrrr faaaaaa fbbbbbb t[cr/lf]
[cr/lf]
```

DATA FORMAT (On robot R6YXTW500, with software Ver.1.66M or higher)

```
PLm [cr/lf]
PLN = XY [cr/lf]
NX  = nnn [cr/lf]
NY  = nnn [cr/lf]
NZ  = nnn [cr/lf]
P[1] = fxxxxxx fyyyyyy fzzzzzz frrrrrr faaaaaa fbbbbbb t xr yr[cr/lf]
    :
P[5] = fxxxxxx fyyyyyy fzzzzzz frrrrrr faaaaaa fbbbbbb t xr yr[cr/lf]
[cr/lf]
```

NOTE

- Integers indicate point data in "pulse" units, and real numbers in "mm" units.

Values

m..........................Pallet number: 0 to 19
nnn.......................Number of points for each axis: positive integer
f...........................Coordinate sign: + / - / space
xxxxxx/yyyyyy/../bbbbbb............
Represent a numeric value of 8 digits or less. When a dot is included, this is treated as point data in "mm" units. Each piece of data is separated by one or more spaces.
t...........................An extended hand system flag setting for SCARA robots. 1: Right-handed system, 2: Left-handed system
xr.........................Extended setting for X-arm rotation information.
0: "mm" → pulse converted angle data x (*1) range: -180.00° < x <= 180.00°
1: "mm" → pulse converted angle data x (*1) range: 180.00° < x <= 540.00°
-1: "mm" → pulse converted angle data x (*1) range: -540.00° < x <= -180.00°
yr.........................Extended setting for Y-arm rotation information.
0: "mm" → pulse converted angle data x (*1) range: -180.00° < y <= 180.00°
1: "mm" → pulse converted angle data x (*1) range: 180.00° < y <= 540.00°
-1: "mm" → pulse converted angle data x (*1) range: -540.00° < y <= -180.00°

*1: The joint-coordinates-converted pulse data represents each arm's distance (converted to angular data) from its mechanical origin point.

NOTE

- X-arm and Y-arm rotation information is only available in software Ver.1.66M or higher.

- Hand system flags are enabled only when specifying the coordinate data in "mm" units for SCARA robots.
- Hand system flags and the X-arm and Y-arm rotation information are ignored during movement where pallet definitions are used.
- If a number other than 1 or 2 is set, or if no number is designated, then 0 will be set to indicate that there is no hand system flag.
- X-arm and Y-arm rotation information settings are available only on the R6YXTW500 robot model where a "mm" units coordinate system has been set.
- If a value other than "0", "1", "-1" is specified at the X-arm and Y-arm rotation information, or if no value is specified, this will be processed as "0".
- A line containing only [cr/lf] is added at the end of the file, indicating the end of the file.

8 Pallet definition file

SAMPLE

SEND PL2 TO CMU ················ Outputs the specified pallet definition from communication port as shown below.
SEND CMU TO PL2 ················ Inputs the specified pallet definition from communication port as shown below.

Response:

```
PL2[cr/lf]
PLN=    XY[cr/lf]
NX=     3[cr/lf]
NY=     3[cr/lf]
NZ=     2[cr/lf]
P[1]=   100.00   100.00    50.00   90.00   0.00   0.00   [cr/lf]
P[2]=   200.00   100.00    50.00   90.00   0.00   0.00   [cr/lf]
P[3]=   100.00   200.00    50.00   90.00   0.00   0.00   [cr/lf]
P[4]=   200.00   200.00    50.00   90.00   0.00   0.00   [cr/lf]
P[5]=   100.00    10.00   100.00   90.00   0.00   0.00   [cr/lf]
[cr/lf]
```

9 All file

9.1 All files

Format
ALL

Meaning Expresses the minimum number of data files required to operate the robot system.

NOTE

- For details of each file, refer to that file's explanation.

DATA FORMAT

```
[PGM]     All program format
NAME=< program name >
aaaaa....aaaaaaaaaaaaaaa[cr/lf]
:
aaaaa....aaaaaaaaaaaaaaa[cr/lf]
[cr/lf]
[PNT]     All point format
Pmmmm=fxxxxxx fyyyyyy fzzzzzz frrrrrr faaaaaa fbbbbbb t[cr/lf]
:
Pmmmm=fxxxxxx fyyyyyy fzzzzzz frrrrrr faaaaaa fbbbbbb t[cr/lf]
[cr/lf]
[PCM]     All point comment format
PCmmmm= sssssssssssssss[cr/lf]
:
PCmmmm= sssssssssssssss[cr/lf]
[cr/lf]
[PRM]     All parameter format
/parameter label/'<comment> [cr/lf]
RC= xxxxxx [cr/lf]
/parameter label/'<comment> [cr/lf]
R1= xxxxxx R2= yyyyyy [cr/lf]
:
[cr/lf]
[END]     ALL files end
```

MEMO

- In writing files, [xxx] determines the data file's format, and this format is saved at the controller. Example: [HND]…All text data up the next [xxx] is saved at the controller as "all hand" format data.

SAMPLE

SEND ALL TO CMU ················ Outputs all files of the entire system from communication port.
SEND CMU TO ALL ················ Inputs all files of the entire system from communication port.

8

9

10

11

12

13

14

10 Program directory file

10.1 Entire program directory

Format

DIR

Meaning

- Expresses entire program directory.
- When used as a readout file, information on entire program directory is read out.
- Cannot be used as a write file.

DATA FORMAT

```
No.      Name           Line     Byte      RW/RO   Date          Time[cr/lf]
nnnfgssssssss           llll     bbbbbb    xx      yy/mm/dd      hh:mm[cr/lf]
   :
nnnfgssssssss           llll     bbbbbb    xx      yy/mm/dd      hh:mm[cr/lf]
END[cr/lf]
```

Values

nnn..Program directory number: 3 digits

f..."o" at program compiling when a program object is created. "s" at sequence program compiling when a sequence object is created.

g...Shows an asterisk "*" for the currently selected program.

ssssssss ....Program name: 8 digits

llll ....Number of program lines: 4 digits

bbbbbb..Byte size of program: 6 digits

xx ....File attribute: 2-digit
RW: Readable/writable
RO: Not writable (read only)

yy/mm/dd ....Date when the program was updated: 8 digits (including the "/" marks)

hh:mm ....Time when the program was updated: 5 digits

SAMPLE

SEND DIR TO CMU・・・・・・・・・・・・・・・・Outputs information on all program directory from communication port.

Response:

No.	Name	Line	Byte	RW/RO	Date	Time[cr/lf]
1o*	12345678	5	21	RW	01/06/20	10:35[cr/lf]
2	PGM1	5	66	RW	01/06/20	10:35[cr/lf]
3	PGM2	5	66	RW	01/06/20	10:35[cr/lf]
4	PGM3	5	66	RW	01/06/20	10:35[cr/lf]
5	PGM4	5	66	RW	01/06/20	10:35[cr/lf]
6	PGM5	5	66	RW	01/06/20	10:35[cr/lf]
7	PGM6	5	66	RW	01/06/20	10:35[cr/lf]
8s	SEQUENCE	1	15	RW	01/06/20	10:35[cr/lf]

END[cr/lf]

10.2 One program

Format

“<<”<program name>”>>”

Meaning

- Expresses information on one program.
- The program name is enclosed in <<>> double brackets.

DATA FORMAT

No.	Name	Line	Byte	RW/RO	Date	Time[cr/lf]
nnnfgssssssss		llll	bbbbbb	xx	yy/mm/dd	hh:mm[cr/lf]

Values

nnn..Program directory number: 3 digits
f..."o" at program compiling when a program object is created. "s" at sequence program compiling when a sequence object is created.
g...Shows an asterisk "*" for the currently selected program.
ssssssss ....Program name: 8 digits
llll ....Number of program lines: 4 digits
bbbbbb..Byte size of program: 6 digits
xx...File attribute: 2-digit
RW: Readable/writable
RO: Not writable (read only)
yy/mm/dd ....Date when the program was updated: 8 digits (including the "/" marks)
hh:mm ....Time when the program was updated: 5 digits

MEMO

- Indicates the compiled execution program (program objects compiled for a robot program, or sequence objects compiled for a sequence program).

SAMPLE

SEND <<TEST>> TO CMU･･････････Outputs information on the specified program from communication port.

Response:

No.	Name	Line	Byte	RW/RO	Date	Time[cr/lf]
3o*	PGM2	5	66	RW	01/06/20	10:35[cr/lf]

11 Parameter directory file

11.1 Entire parameter directory

Format
DPM

Meaning
- Expresses entire parameter directory.
- When used as a readout file, information on entire parameter directory is read out.
- Cannot be used as a write file.

DATA FORMAT

```
/parameter label/ '<comment>[cr/lf]
/parameter label/ '<comment>[cr/lf]
            :
/parameter label/ '<comment>[cr/lf]
 [cr/lf]
```

Values <comment> .... Parameter name

- Parameter labels are shown with 6 alphabetic characters.
- A line containing only [cr/lf] is added at the end of the file, indicating the end of the file.

SAMPLE

```
SEND DPM TO CMU ••••••••••••••••Outputs information on all parameter directory from communication port.
Response:
/RBTNUM/ 'Robot number (V8.01/R1001)[cr/lf]
/AXES  / 'Number of axes[cr/lf]
/AXSNUM/ 'Axis number(V1.01/V1.01/V1.01/V1.01/-----/-----/-----/-----/)[cr/lf]
/ATTRIB/ 'Axis attribute[cr/lf]
/WEIGHT/ 'Tip weight[kg][cr/lf]
/ORIGIN/ 'Origin sequence[cr/lf]
/RORIEN/ 'R axis orientation[cr/lf]
      :
/CURPNO/ 'Output port number[cr/lf]
/CURPT1/ 'Number of compare point 1[cr/lf]
/CURPT2/ 'Number of compare point 2[cr/lf]
[cr/lf]
```

12 Variable file

12.1 All variables

Format
VAR

Meaning
- Expresses all global variables.
- When used as a readout file, all global variables currently stored are read out.
- When used as a write file, a specified global variable is written.

DATA FORMAT

```
<variable name>t = xxxxxx [cr/lf]
<variable name>t = xxxxxx [cr/lf]
            :
<variable name>t = xxxxxx [cr/lf]
 [cr/lf]
```

Values

<Variable name> Global variable defined in the program. Variable name is shown with 16 characters or less consisting of alphanumeric characters and underscore ("_").

t..Type of variable / !: real type, %: integer type, $: character string type

xxxxxx..Differs depending on the type of variable:

Integer type:	integer of 8 digits or less
Real type:	real number of 7 digits or less including decimal fractions
Character type:	character string of 70 characters or less

- A line containing only [cr/lf] is added at the end of the file, indicating the end of the file.

12 Variable file

SAMPLE 1

SEND VAR TO CMU････････････････Outputs all global variables from communication port.

Response:
SGI0=0[cr/lf]
SGI1=1111[cr/lf]
SGI2=2222[cr/lf]
SGI3=3333[cr/lf]
SGI4=4444[cr/lf]
SGI5=5555[cr/lf]
SGI6=6666[cr/lf]
SGI7=7777[cr/lf]
SGR0=0[cr/lf]
SGR1=1.1111E3[cr/lf]
SGR2=2.2222E3[cr/lf]
SGR3=3.3333E3[cr/lf]
SGR4=4.4444E3[cr/lf]
SGR5=5.5555E3[cr/lf]
SGR6=6.6666E3[cr/lf]
SGR7=7.7777E3[cr/lf]
B1%=111[cr/lf]
B2%=222[cr/lf]
C1$="CNS_1"[cr/lf]
C2$="CNS_2"[cr/lf]
[cr/lf]

SAMPLE 2

SEND CMU TO VAR････････････････Inputs all global variables from communication port.

12.2 One variable

Format

<variable name>t

Meaning
- Expressed one variable.

DATA FORMAT

xxxxxx [cr/lf]

NOTE
- SGIx indicates an integer type static variable.
- SGRx indicates a real type static variable.

Values

<Variable name> .... Global variable defined in the program. Variable name is shown with 16 characters or less consisting of alphanumeric characters and underscore ("_").

t .... Type of variable / !: real type, %: integer type, $: character string type

xxxxxx .... Differs depending on the type of variable:

Integer type:	integer of 8 digits or less
Real type:	real number of 7 digits or less including decimal fractions
Character type:	character string of 70 characters or less

MEMO
- Dynamic global variables are registered during compiling. Variables cannot be referred to unless they are registered.

SAMPLE 1

SEND SGI6 TO CMU[cr/lf] ················ Outputs the specified variable SGI6 from communication port.

Response:
6666[cr/lf]

SAMPLE 2

SEND CMU TO SGI6[cr/lf] ················ Inputs the specified variable SGI6 from communication port.

Response:
6666 [cr/lf] Data input to the controller.
OK [cr/lf] Result output from the controller.

13 Constant file

13.1 One character string

Format
"<character string>"

Meaning
- Expresses a specified character string.
- When used as a readout file, the specified character string is read out.
- Cannot be used as a write file.

DATA FORMAT
sssss...ssssss[cr/lf]

Values sssss...ssssss................................Character string: 61 characters or less

- Output of a double quotation (") is shown with two successive double quotations.

SAMPLE
SEND ""OMRON ROBOT"" TO CMU
·················Outputs the specified character string from communication port.
Response:
"OMRON ROBOT"[cr/lf]

14 Array variable file

14.1 All array variables

Format

ARY

Meaning
- Expresses all array variables.
- When used as a readout file, all array variables are read out.
- When used as a write file, writing is performed with a specified array variable.

DATA FORMAT

```
<variable name>t(l{,m{,n}}) = xxxxxx [cr/lf]
<variable name>t(l{,m{,n}}) = xxxxxx [cr/lf]
            :
<variable name>t(l{,m{,n}}) = xxxxxx [cr/lf]
[cr/lf]
```

Values

<Variable name> ....Global variable defined in the program. Variable name is shown with 16 characters or less consisting of alphanumeric characters and underscore ("_").
t..Type of variable / !: real type, %: integer type, $: character string type
l, m, n..Indicate array arguments.
xxxxxx..Differs depending on the type of array variable.
Integer type: integer of 8 digits or less
Real type: real number of 7 digits or less including decimal fractions
Character type: character string of 70 characters or less

■ A line containing only [cr/lf] is added at the end of the file, indicating the end of the file.

SAMPLE 1

SEND ARY TO CMU················Outputs all global array variables from communication port.

```
Response:
A!(0)=0[cr/lf]
A!(1)=1.E2[cr/lf]
A!(2)=2.E2[cr/lf]
B%(0,0)=0[cr/lf]
B%(0,1)=1111[cr/lf]
B%(1,0)=2222[cr/lf]
B%(1,1)=3333[cr/lf]
C$(0,0,0)="ARY1"[cr/lf]
C$(0,0,1)="ARY2"[cr/lf]
C$(0,1,0)="ARY3"[cr/lf]
C$(0,1,1)="ARY4"[cr/lf]
C$(1,0,0)="ARY5"[cr/lf]
C$(1,0,1)="ARY6"[cr/lf]
C$(1,1,0)="ARY7"[cr/lf]
C$(1,1,1)="ARY8"[cr/lf]
[cr/lf]
```

SAMPLE 2

SEND CMU TO ARY················Inputs all global array variables from communication port.

14.2 One array variable

Format

<variable name> t(l {,m {,n }})

Meaning • Expresses one array variable.

DATA FORMAT

xxxxxx [cr/lf]

Values

<Variable name> ....Global variable defined in the program. Variable name is shown with 16 characters or less consisting of alphanumeric characters and underscore ("_").

t...Type of variable / !: real type, %: integer type, $: character string type

l, m, n..Indicate array arguments.

xxxxxx..Differs depending on the type of array variable.

Integer type:	integer of 8 digits or less
Real type:	real number of 7 digits or less including decimal fractions
Character type:	character string of 70 characters or less

MEMO

- Array variables defined by the DIM statement are registered during compiling. Array variables cannot be referred to unless they are registered.

SAMPLE 1

SEND C1$(2) TO CMU[cr/lf] • • • • • • • • • • • • • • • • Outputs the specified array variable C1$(2) from communication port.

Response:
OMRON ROBOT[cr/lf]

SAMPLE 2

SEND CMU TO C1$(2)[cr/lf] • • • • • • • • • • • • • • • • Inputs the specified array variable C1$(2) from communication port.

Response:
OK[cr/lf]

15 DI file

15.1 All DI information

Format
DI()

Meaning

- Expresses all DI (parallel input variable) information.
- When used as a readout file, all DI information is read out.
- Cannot be used as a write file.

DATA FORMAT

```
DI0()=&Bnnnnnnnn [cr/lf]
DI1()=&Bnnnnnnnn [cr/lf]
    :
DI27()=&Bnnnnnnnn [cr/lf]
[cr/lf]
```

Values

n ...."0" or "1" (total of 8 digits). Corresponds to m7, m6, …, m0, reading from the left ("m" is the port No.).

- A line containing only [cr/lf] is added at the end of the file, indicating the end of the file.

SAMPLE

```
SEND DI() TO CM················Outputs all DI information from communication port.

Response:
DI0()=&B10001001[cr/lf]
DI1()=&B00000010[cr/lf]
DI2()=&B00000000[cr/lf]
          :
DI7()=&B00000000[cr/lf]
DI10()=&B00000000[cr/lf]
DI11()=&B00000000[cr/lf]
DI12()=&B00000000[cr/lf]
          :
DI17()=&B00000000[cr/lf]
DI20()=&B00000000[cr/lf]
          :
DI26()=&B00000000[cr/lf]
DI27()=&B00000000[cr/lf]
[cr/lf]
```

15.2 One DI port

Format

DIm()

Meaning
- Expresses the status of one DI port.
- When used as a readout file, the specified DI port status is read out.
- Cannot be used as a write file.

DATA FORMAT

DIm()=&Bnnnnnnnn[cr/lf]

Values

m...0 to 7, 10 to 17, 20 to 27
n..."0" or "1" (total of 8 digits). Corresponds to m7, m6, …, m0, reading from the left ("m" is the port No.).

SAMPLE

SEND DI5() TO CMU·················Outputs the DI5 port status from communication port.

Response:
DI5()=&B00000000[cr/lf]

16 DO file

16.1 All DO information

Format

DO()

Meaning

- Expresses all DO (parallel output variable) information.
- When used as a readout file, all DO information is read out.
- Cannot be used as a write file.

DATA FORMAT

```
DO0()=&Bnnnnnnnn [cr/lf]
DO1()=&Bnnnnnnnn [cr/lf]
    :
DO27()=&Bnnnnnnnn [cr/lf]
[cr/lf]
```

Values

n.."0" or "1" (total of 8 digits). Corresponds to m7, m6, …, m0, reading from the left ("m" is the port No.).

- A line containing only [cr/lf] is added at the end of the file, indicating the end of the file.

SAMPLE

```
SEND DO() TO CMU················Outputs all DO information from communication port.

Response:
DO0()=&B10001001[cr/lf]
DO1()=&B00000010[cr/lf]
DO2()=&B00000000[cr/lf]
          :
DO7()=&B00000000[cr/lf]
DO10()=&B00000000[cr/lf]
DO11()=&B00000000[cr/lf]
DO12()=&B00000000[cr/lf]
          :
DO17()=&B00000000[cr/lf]
DO20()=&B00000000[cr/lf]
          :
DO26()=&B00000000[cr/lf]
DO27()=&B00000000[cr/lf]
[cr/lf]
```

8

9

10

11

12

13

14

16.2 One DO port

Format
DOm()

Meaning

- Expresses the status of one DO port.
- When used as a readout file, the specified DO port status is read out.
- When used as a write file, the value is written to the specified DO port. However, writing to DO0() and DO1() is prohibited.

- Readout file

DATA FORMAT
DOm()=&Bnnnnnnnn[cr/lf]

- Write file

DATA FORMAT
&Bnnnnnnnn[cr/lf] or k[cr/lf]

Values

m..Port number: 0 to 7, 10 to 17, 20 to 27
n.."0" or "1" (total of 8 digits). Corresponds to m7, m6, …, m0, reading from the left ("m" is the port No.).
k..Integer from 0 to 255

MEMO

- Writing to DO0() and DO1() is prohibited. Only referencing is permitted.

SAMPLE 1

SEND DO5() TO CMU････････････････Outputs the DO5 port status from communication port.

Response:
DO5()=&B00000000[cr/lf]

SAMPLE 2

SEND CMU TO DO5()････････････････Inputs the DO5 port status from communication port.
&B00000111

Response:
OK[cr/lf]

17 MO file

17.1 All MO information

Format
MO()

Meaning
- Expresses all MO (internal output variable) information.
- When used as a readout file, all MO information is read out.
- Cannot be used as a write file.

DATA FORMAT

```
MO0()=&Bnnnnnnnn [cr/lf]
MO1()=&Bnnnnnnnn [cr/lf]
            :
MO27()=&Bnnnnnnnn [cr/lf]
[cr/lf]
```

Values

n.."0" or "1" (total of 8 digits). Corresponds to m7, m6, …, m0, reading from the left ("m" is the port No.).

- A line containing only [cr/lf] is added at the end of the file, indicating the end of the file.

SAMPLE

```
SEND MO() TO CMU················Outputs all MO information from communication port.

Response:
MO0()=&B10001001[cr/lf]
MO1()=&B00000010[cr/lf]
MO2()=&B00000000[cr/lf]
:
MO7()=&B00000000[cr/lf]
MO10()=&B00000000[cr/lf]
MO11()=&B00000000[cr/lf]
MO12()=&B00000000[cr/lf]
:
MO17()=&B00000000[cr/lf]
MO20()=&B00000000[cr/lf]
:
MO26()=&B00000000[cr/lf]
MO27()=&B00000000[cr/lf]
[cr/lf]
```

8
9
10
11
12
13
14

17.2 One MO port

Format
MOm()

Meaning

- Expresses the status of one MO port.
- When used as a readout file, the specified MO port status is read out.
- When used as a write file, the value is written to the specified MO port. However, writing to MO0() and MO1() is prohibited.

- Readout file

DATA FORMAT
MOm()=&Bnnnnnnnn[cr/lf]

- Write file

DATA FORMAT
&Bnnnnnnnn[cr/lf] or k[cr/lf]

Values

m..Port number: 0 to 7, 10 to 17, 20 to 27
n..."0" or "1" (total of 8 digits). Corresponds to m7, m6, …, m0, reading from the left ("m" is the port No.).
k...Integer from 0 to 255

MEMO

- Writing to MO0() and MO1() is prohibited. Only reference is permitted.

SAMPLE 1

SEND MO5() TO CMU················Outputs the MO5 port status from communication port.

Response:
MO5()=&B00000000[cr/lf]

SAMPLE 2

SEND CMU TO MO5()···············Inputs the MO5 port status from communication port.
&B00000111

Response:
OK[cr/lf]

18 LO file

18.1 All LO information

Format
LO()

Meaning
- Expresses all LO (internal output variable) information.
- When used as a readout file, all LO information is read out.
- Cannot be used as a write file.

DATA FOMAT
LO0()=&Bnnnnnnnn [cr/lf] [cr/lf]

Values n..."0" or "1" (total of 8 digits). Corresponds to 07, 06, …, 00, reading from the left.

- A line containing only [cr/lf] is added at the end of the file, indicating the end of the file.

SAMPLE
SEND LO() TO CMU・・・・・・・・・・・・・・・・Outputs all LO status from communication port. Response: LO0()=&B10001001[cr/lf] [cr/lf]

8
9
10
11
12
13
14

18.2 One LO port

Format
LO0()

Meaning
- Expresses the status of one LO port.
- When used as a readout file, the specified LO port status is read out.
- When used as a write file, the value is written to the specified LO port.

- Readout file

DATA FORMAT
LO0()=&Bnnnnnnnn[cr/lf]

- Write file

DATA FORMAT
&Bnnnnnnnn[cr/lf] or k[cr/lf]

Values

n.."0" or "1" (total of 8 digits). Corresponds to 07, 06, …, 00, reading from the left.

k..Integer from 0 to 255

SAMPLE 1

```
SEND LO0() TO CMU・・・・・・・・・・・・・・・・Outputs the LO0 port status from communication port.

Response:
LO0()=&B00000000[cr/lf]
```

SAMPLE 2

```
SEND CMU TO LO0()・・・・・・・・・・・・・・・・Inputs the LO0 port status from communication port.
&B00000111

Response:
OK[cr/lf]
```

19 TO file

19.1 All TO information

Format
TO()

Meaning
- Expresses all TO (timer output variable) information.
- When used as a readout file, all TO information is read out.
- Cannot be used as a write file.

DATA FORMAT

TO0()=&Bnnnnnnnn [cr/lf]
[cr/lf]

Values n.."0" or "1" (total of 8 digits). Corresponds to 07, 06, …, 00, reading from the left ("m" is the port No.).

■ A line containing only [cr/lf] is added at the end of the file, indicating the end of the file.

SAMPLE

SEND TO() TO CMU················Outputs all TO status from communication port.

Response:
TO0()=&B10001001[cr/lf]
[cr/lf]

19.2 One TO port

Format

TO()

Meaning
- Expresses the status of one TO port.
- When used as a readout file, the specified TO port status is read out.
- When used as a write file, the value is written to the specified TO port.

- Readout file

DATA FORMAT

TO0()=&Bnnnnnnnn[cr/lf]

- Write file

DATA FORMAT

&Bnnnnnnnn[cr/lf] or k[cr/lf]

Values

n.."0" or "1" (total of 8 digits). Corresponds to 07, 06, …, 00, reading from the left ("m" is the port No.).

k..Integer from 0 to 255

SAMPLE 1

SEND TO0() TO CMU・・・・・・・・・・・・・・・・Outputs the TO0 port status from communication port.

Response:
TO0()=&B00000000[cr/lf]

SAMPLE 2

SEND CMU TO TO0()・・・・・・・・・・・・・・・・Inputs the TO0 port status from communication port.
&B00000111

Response:
OK[cr/lf]

20 SI file

20.1 All SI information

Format

SI()

Meaning

- Expresses all SI (serial input variable) information.
- When used as a readout file, all SI information is read out.
- Cannot be used as a write file.

DATA FORMAT

```
SI0()=&Bnnnnnnnn [cr/lf]
SI1()=&Bnnnnnnnn [cr/lf]
    :
SI27()=&Bnnnnnnnn [cr/lf]
[cr/lf]
```

Values

n......................................"0" or "1" (total of 8 digits). Corresponds to m7, m6, …, m0, reading from the left ("m" is the port No.).

- A line containing only [cr/lf] is added at the end of the file, indicating the end of the file.

SAMPLE

```
SEND SI() TO CMU················Outputs all SI status from communication port.

Response:
SI0()=&B10001001[cr/lf]
SI1()=&B00000010[cr/lf]
SI2()=&B00000000[cr/lf]
:
SI7()=&B00000000[cr/lf]
SI10()=&B00000000[cr/lf]
SI11()=&B00000000[cr/lf]
SI12()=&B00000000[cr/lf]
:
SI17()=&B00000000[cr/lf]
SI20()=&B00000000[cr/lf]
:
SI26()=&B00000000[cr/lf]
SI27()=&B00000000[cr/lf]
[cr/lf]
```

8
9
10
11
12
13
14

20.2 One SI port

Format
SIm()

Meaning
- Expresses the status of one SI port.
- When used as a readout file, the specified SI port status is read out.
- Cannot be used as a write file.

DATA FORMAT

SIm()=&Bnnnnnnnn[cr/lf]

Values

m..Port number: 0 to 7, 10 to 17, 20 to 27

n.."0" or "1" (total of 8 digits). Corresponds to m7, m6, …, m0, reading from the left ("m" is the port No.).

SAMPLE

SEND SI5() TO CMU••••••••••••••••Outputs the SI5 port status from communication port.

Response:
SI5()=&B00000000[cr/lf]

21 SO file

21.1 All SO information

Format

SO()

Meaning

- Expresses all SO (serial output variable) information.
- When used as a readout file, all SO information is read out.
- Cannot be used as a write file.

DATA FORMAT

```
SO0()=&Bnnnnnnnn [cr/lf]
SO1()=&Bnnnnnnnn [cr/lf]
        :
SO27()=&Bnnnnnnnn [cr/lf]
[cr/lf]
```

Values

n.."0" or "1" (total of 8 digits). Corresponds to m7, m6, …, m0, reading from the left ("m" is the port No.).

- A line containing only [cr/lf] is added at the end of the file, indicating the end of the file.

SAMPLE

```
SEND SO() TO CMU················Outputs all SO status from communication port.

Response:
SO0()=&B10001001[cr/lf]
SO1()=&B00000010[cr/lf]
SO2()=&B00000000[cr/lf]
:
SO7()=&B00000000[cr/lf]
SO10()=&B00000000[cr/lf]
SO11()=&B00000000[cr/lf]
SO12()=&B00000000[cr/lf]
:
SO17()=&B00000000[cr/lf]
SO20()=&B00000000[cr/lf]
:
SO26()=&B00000000[cr/lf]
SO27()=&B00000000[cr/lf]
[cr/lf]
```

21.2 One SI port

Format
SOm()

Meaning

- Expresses the status of one SO port.
- When used as a readout file, the specified SO port status is read out.
- When used as a write file, the value is written to the specified SO port. However, writing to SO0() and SO1() is prohibited.

- Readout file

DATA FORMAT
SOm()=&Bnnnnnnnn[cr/lf]

- Write file

DATA FORMAT
&Bnnnnnnnn[cr/lf] or k[cr/lf]

Values

m..Port number: 0 to 7, 10 to 17, 20 to 27
n..."0" or "1" (total of 8 digits). Corresponds to m7, m6, …, m0, reading from the left ("m" is the port No.).
k...Integer from 0 to 255

MEMO

- Writing to SO0() and SO1() is prohibited. Only reference is permitted.

SAMPLE 1

SEND SO5() TO CMU················Outputs the SO5 port status from communication port.

Response:
SO5()=&B00000000[cr/lf]

SAMPLE 2

SEND CMU TO SO5()················Inputs the SO5 port status from communication port.
&B00000111

Response:
OK[cr/lf]

22 Error message history file

22.1 All error message history

Format
LOG

Meaning
- Expresses all error message history.
- When used as a readout file, all error message history is read out.
- Cannot be used as a write file.

DATA FORMAT

```
nnn:yy/mm/dd,hh:mm:ss  gg.bb:msg[cr/lf]
nnn:yy/mm/dd,hh:mm:ss  gg.bb:msg[cr/lf]
   :
nnn:yy/mm/dd,hh:mm:ss  gg.bb:msg[cr/lf]
[cr/lf]
```

Values

nnn..Represents an error history serial number and may be up to 500.
yy, mm ,dd.................................Year/Month/Day
hh, mm, ss..................................Hour, Minute, Second
gg..Error message group
bb..Error message category
msg ....Error message

- A line containing only [cr/lf] is added at the end of the file, indicating the end of the file.

SAMPLE

SEND LOG TO CMU････････････････Outputs all error message history from communication port.

Response:
1 :01/06/14,13:19:20 14.22:No start code (@)[cr/lf]
2 :01/06/14,13:18:34 22.3: DC24V power low[cr/lf]
:
498:01/06/12,21:49:54 5.39:Illegal identifier[cr/lf]
499:01/06/12,21:49:14 14.22:No start code (@)[cr/lf]
500:01/06/12,21:49:00 22.3: DC24V power low[cr/lf]
[cr/lf]

8
9
10
11
12
13
14

23 Error Message History Details File

23.1 General error message history details

Format

LEX

Meaning
- Displays the general error message history details.
- This file cannot be specified as a WRITE file.

DATA FORMAT

nnn.l:yy/mm/dd hh:mm:ss gg.bb:msg,A,N,L,P[cr/lf]
nnn.l:M=x,y,z,r,a,b[cr/lf]
nnn.l:S=x,y,z,r,a,b[cr/lf]

Values

nnn...Represents an error history serial number and may be up to 500.
l..Output line number (max. of 3)
yy, mm ,dd.................................Year/Month/Day
hh, mm, ss..................................Hour, Minute, Second
gg..Error message group
bb..Error message category
msg ....Error message
A ....AUTO operation execution status. 1: AUTO operation in progress; 0: Other status.
N ....Execution program name.
L ....Program's execution line No.
P..No. of most recently referenced point.
M= ....Indicates that all subsequent coordinates are main robot coordinates.
x,y,z,r,a,b ....Current position of each axis.
[cr/lf]...CR code (0Dh) + LF code (0Ah).

- A line containing only [cr/lf] is added at the end of the file, indicating the end of the file.

SAMPLE

```
SEND LEX to CMU ••••••••••••••••• Outputs the following general error message history details from the communication port.
Response:
1    .1:01/06/14,13:19:20 14.22: No START code (@),0,TEST    ,2,2500[cr/lf]
1    .2:M=   23.80   23.59   0.00   0.00   0.00   0.00[cr/lf]
1    .3:S=   48.20   47.99   0.00   0.00   0.00   0.00[cr/lf]
2    .1:01/06/12,13:18:34 22.3: 24VDC low voltage,0,TEST    ,6,2500[cr/lf]
2    .2:M=   11568   23985   0.00   0.00   0.00   0.00[cr/lf]
2    .3:S=   35693   12582   0.00   0.00   0.00   0.00]cr/lf]
           :
500.1:01/06/12,21:49:00 22.3: 24VDC low voltage,0,TEST    ,6,2500[cr/lf]
500.2:M=   38.71   38.50   0.00   0.00   0.00   0.00[cr/lf]
500.3:S=   51.48   51.27   0.00   0.00   0.00   0.00[cr/lf]
[cr/lf]
```

NOTE
- Error information is output only for the connected robot.

NOTE
- Robot coordinates display in pulse units for axes where an "incomplete return-to-origin" status exists. Robot coordinates also display in pulse units on SCARA robots where reference coordinates have not been specified.
- Robot coordinates display in mm/deg units for axes where a "completed return-to-origin" status exists.

24 Machine reference file

24.1 All machine reference file

Format
MRF

Meaning

- Expresses the machine reference data obtained after robots have performed return- to-origin and absolute search.
- Reads out all machine reference data when used as a readout file.
- Cannot be used as a write file.

DATA FORMAT

M1=nnn% M2=nnn% .... nnn%[cr/lf]
S1=nnn% S2=nnn% .... nnn%[cr/lf]
[cr/lf]

Values

nnn..Integer from 0 to 100
M?..Indicate the main robot group axes.

- A line containing only [cr/lf] is added at the end of the file, indicating the end of the file.

SAMPLE

SEND MRF TO CMU・・・Outputs all machine reference data from communication port.

Response:
M1=   55% M2=   23% M3=   33% M4=   26%[cr/lf]
[cr/lf]

8
9
10
11
12
13
14

25 EOF file

25.1 EOF data

Format
EOF

Meaning

- This file is a special file consisting only of a ^Z (=1Ah) code. When transmitting data to an external device through the communication port, the EOF data can be used to add a ^Z code at the end of file.
- When used as a readout file, ^Z (=1Ah) is read out.
- Cannot be used as a write file.

DATA FORMAT
^Z (=1Ah)

SAMPLE

SEND PGM TO CMU

SEND EOF TO CMU・・・・・・・・・・・・・・・・Outputs EOF data from communication port.

NAME=TEST1[cr/lf]
A=1[cr/lf]
:
HALT[cr/lf]
[cr/lf]
^Z

MEMO

- A "^Z" code may be required at the end of the transmitted file, depending on the specifications of the receiving device and application.

26 Serial port communication file

26.1 Serial port communication file

Format
CMU

Meaning
- Expresses the serial communication port.
- Depends on the various data formats.

SAMPLE

```
SEND PNT TO CMU・・・・・・・・・・・・・・・・Outputs all point data from communication port.
SEND CMU TO PNT ・・・・・・・・・・・・・・・・Inputs all point data from communication port.
```

27 SIW file

27.1 All SIW

Format

SIW()

Meaning
- Expresses all SIW (serial word input) data.
- Reads out all SIW information in hexadecimal digit when used as a readout file.
- Cannot be used as a write file.

DATA FORMAT

```
SIW( 0)=&Hnnnn [cr/lf]
SIW( 1)=&Hnnnn [cr/lf]
          :
SIW(15)=&Hnnnn [cr/lf]
[ cr/lf]
```

Values n...0 to 9, A to F: 4 digits (hexadecimal)

■ A line containing only [cr/lf] is added at the end of the file, indicating the end of the file.

SAMPLE

```
SEND SIW() TO CMU・・・・・・・・・・・・・・・・Outputs all SIW data from communication port.

Response:
SIW(0)=&H1001[cr/lf]
SIW(1)=&H0010[cr/lf]
SIW(2)=&H0000[cr/lf]
 :
SIW(15)=&H0000[cr/lf]
[cr/lf]
```

27.2 One SIW data

Format
SIW(m)

Meaning
- Expresses one SIW status.
- Reads out all SIW information in hexadecimal digit when used as a readout file.
- Cannot be used as a write file.

DATA FORMAT
SIW(m)=&Hnnnn [cr/lf]

Values
m..0 to 15
n..0 to 9, A to F: 4 digits (hexadecimal)

SAMPLE

```
SEND SIW(5) TO CMU・・・・・・・・・・・・・・・・Outputs SIW(5) from communication port.

Response:
SIW(5)=&H1001[cr/lf]
```

8
9
10
11
12
13
14

28 SOW file

28.1 All SIW

Format

```
SOW()
```

Meaning

- Expresses all SOW (serial word output) data.
- Reads out all SOW information in hexadecimal digit when used as a readout file.
- Cannot be used as a write file.

DATA FORMAT

```
SOW( 0)=&Hnnnn [cr/lf]
SOW( 1)=&Hnnnn [cr/lf]
            :
SOW(15)=&Hnnnn [cr/lf]
[ cr/lf]
```

Values n...0 to 9, A to F: 4 digits (hexadecimal)

- A line containing only [cr/lf] is added at the end of the file, indicating the end of the file.

SAMPLE

```
SEND SOW() TO CMU・・・・・・・・・・・・・・・・Outputs all SOW data from communication port.

Response:
SOW(0)=&H1001[cr/lf]
SOW(1)=&H0010[cr/lf]
SOW(2)=&H0000[cr/lf]
 :
SOW(15)=&H0000[cr/lf]
[cr/lf]
```

28.2 One SOW data

Format
SOW(m)

Meaning

- Expresses one SOW status.
- When used as a readout file, the specified SOW port status is read out.
- When used as a write file, the value is written to the specified SOW. However, writing to SOW0() and SOW1() is prohibited.

- Readout file

DATA FORMAT
SOW(m)=&Hnnnn [cr/lf]

- Write file

DATA FORMAT
&Hnnnn

Values

m..2 to 15
n...0 to 9, A to F: 4 digits (hexadecimal)

SAMPLE 1

SEND SOW(5) TO CMU・・・・・・・・・・・・・・・・Outputs SOW(5) from communication port.

Response:
SOW(5)=&H1001[cr/lf]

SAMPLE 2

SEND CMU TO SOW(5)・・・・・・・・・・・・・・・・Input to SOW(5) from communication port.
&H1001

Response:
OK[cr/lf]

8

9

10

11

12

13

14

29 Ethernet port communication file

29.1 Ethernet port communication file

Format
ETH

Meaning
- Expresses the Ethernet port.
- Depends on the various data formats.

SAMPLE

SEND PNT TO ETH・・・・・・・・・・・・・・・・Outputs all point data from the Ethernet port.
SEND ETH TO PNT・・・・・・・・・・・・・・・・Inputs all point data from the Ethernet port.

Chapter 11

User program examples

1 Basic operation..11-1
2 Application..11-8

1 Basic operation

1.1 Directly writing point data in program

■ **Overview**

The robot arm can be moved by PTP (point-to-point) motion by directly specifying point data in the program.

Processing flow

START

300.00	300.00	50.00	90.00	0.00	0.00	PTP movement
300.00	100.00	0.00	0.00	0.00	0.00	PTP movement
200.00	200.00	10.00	-90.00	0.00	0.00	PTP movement

STOP

SAMPLE

```
MOVE P,     300.00    300.00    50.00     90.00     0.00     0.00
MOVE P,     300.00    100.00     0.00      0.00     0.00     0.00
MOVE P,     200.00    200.00    10.00    -90.00     0.00     0.00
HALT
```

1.2 Using point numbers

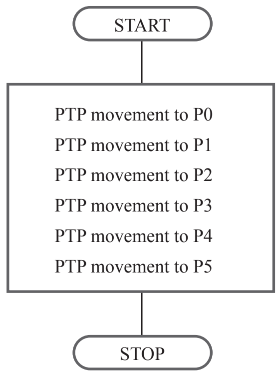
■ **Overview**

Coordinate data can be specified by using point numbers in a program. Coordinate data should be entered beforehand in "MANUAL>POINT" mode, for example as shown below.

POINT DATA

P0=	0.00	0.00	0.00	0.00	0.00	0.00
P1=	100.00	0.00	150.00	30.00	0.00	0.00
P2=	0.00	100.00	50.00	0.00	0.00	0.00
P3=	300.00	300.00	0.00	0.00	0.00	0.00
P4=	300.00	100.00	100.00	90.00	0.00	0.00
P5=	200.00	200.00	0.00	0.00	0.00	0.00

Processing flow



SAMPLE 1

```
MOVE P,P0
MOVE P,P1
MOVE P,P2
MOVE P,P3
MOVE P,P4
MOVE P,P5
HALT
```

SAMPLE 2

```
FOR J=0 TO 5
    MOVE P,P[J]
NEXT J
HALT
```

Although the same operation is executed by both SAMPLE 1 and SAMPLE 2, the program can be shortened by using point Nos. and the FOR statement.

1.3 Using shift coordinates

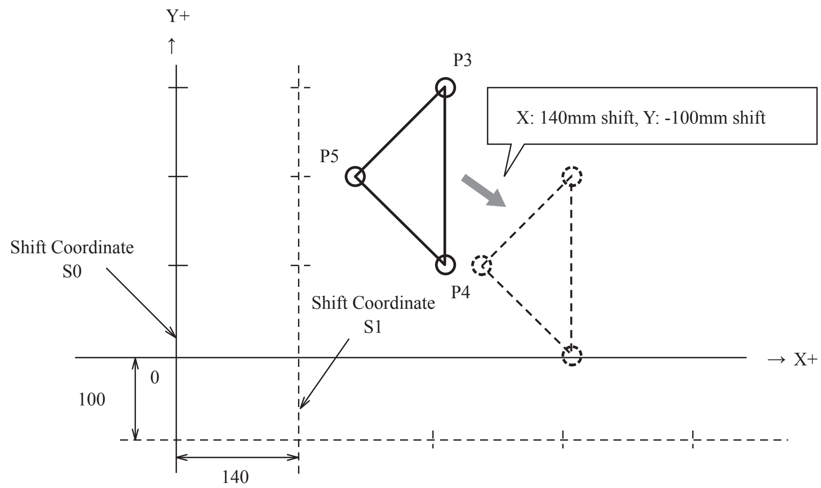
■ **Overview**

In the example shown below, after PTP movement from P3 to P5, the coordinate system is shifted +140mm along the X-axis and -100mm along the Y-axis, and the robot then moves from P3 to P5 again. The shift coordinate data is set in S1 and P3, P4, P5 are set as described in the previous section ("1.2 Using point numbers").

SHIFT DATA

S0=	0.00	0.00	0.00	0.00
S1=	140.00	-100.00	0.00	0.00

Shift Coordinate



SAMPLE

```
SHIFT     S0    ·················Shift 0.
FOR J=3 TO 5  Repeated movement from P3 to P5.
      MOVE P, P[J]
NEXT J
SHIFT S1       Changed to "shift 1".
      FOR K=3 TO 5·················Repeated movement occurs in the same manner from P3 to P5.
      MOVE P,P[K]
NEXT K
```

1.4 Palletizing

1.4.1 Calculating point coordinates

■ **Overview**

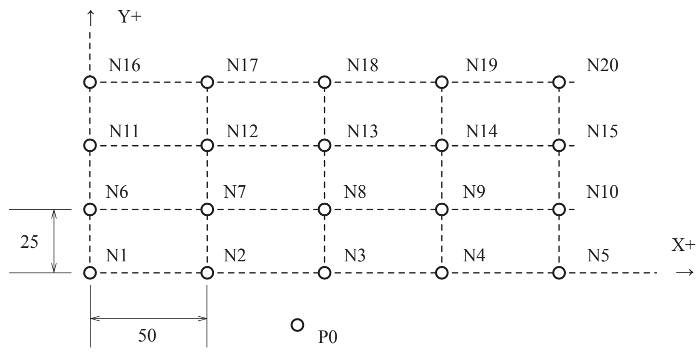
Repetitive movement between a fixed work supply position P0 and each of the equally spaced points on a pallet can be performed with the following program.

In the drawing below, points N1 to N20 are on Cartesian coordinates, consisting of 5 points positioned at a 50mm pitch in the X-axis direction and 4 points at a 25mm pitch in the Y-axis direction. The robot arm moves from point to point in the order of P0-N1-P0-N2...N5-P0-N6-P0... while repeatedly moving back and forth between point P0 and each pallet.

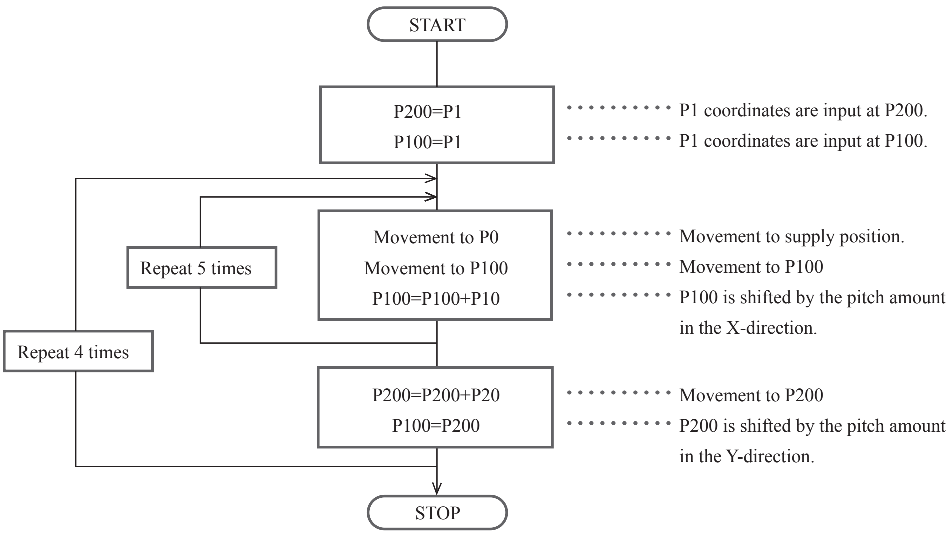
POINT DATA

Work supply position:						
P0=	0.0	0.0	0.0	0.0	0.0	0.0
X-axis pitch:						
P10=	50.0	0.0	0.0	0.0	0.0	0.0
Y-axis pitch:						
P20=	0.0	25.0	0.0	0.0	0.0	0.0
N1 position:						
P1 =	100.0	50.0	0.0	0.0	0.0	0.0

Calculating point coordinates



Processing flow



SAMPLE

```
P100=P1
P200=P1
FOR J=1 TO 4
    FOR K=1 TO 5
        MOVE P,P0
        MOVE P,P100
        P100=P100+P10
    NEXT K
    P200=P200+P20
    P100=P200
NEXT J
```

1.4.2 Utilizing pallet movement

■ **Overview**

Repetitive movement between a fixed work supply position P0 and each of the equally spaced points on a pallet can be performed with the following program. In the drawing below, points N1 to N24 are on Cartesian coordinates, consisting of 3 points positioned at a 50mm pitch in the X-axis direction, 4 points at a 50mm pitch in the Y-axis direction, and 2 points at 100mm pitch in the Z-axis direction. The robot arm moves from point to point in the order of P0-N1-P0-N2...-N5-P0-N6... while repeatedly moving back and forth between point P0 and each pallet.

POINT DATA

Work supply position:

P0=	0.00	0.00	100.00	0.00	0.00	0.00

Pallet definition:

PL0 (P3996 to P4000 are used)

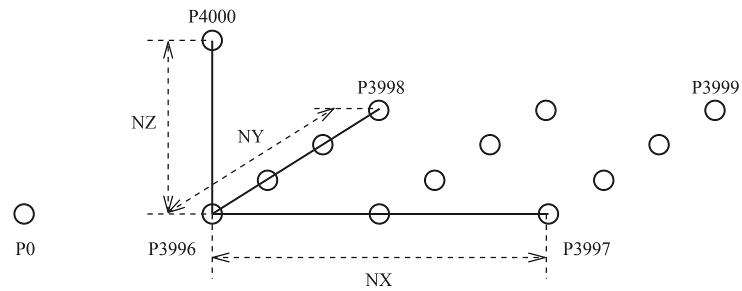
NX= 3

NY= 4

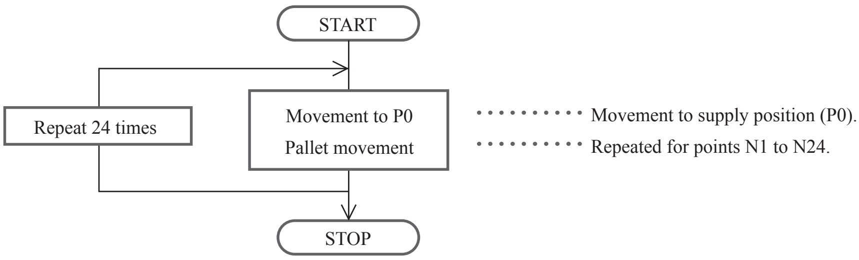
NZ= 2

P3996=	100.00	50.00	100.00	0.00	0.00	0.00
P3997=	200.00	50.00	100.00	0.00	0.00	0.00
P3998=	100.00	200.00	100.00	0.00	0.00	0.00
P3999=	200.00	200.00	100.00	0.00	0.00	0.00
P4000=	100.00	50.00	200.00	0.00	0.00	0.00

Utilizing pallet movement



Processing flow



SAMPLE

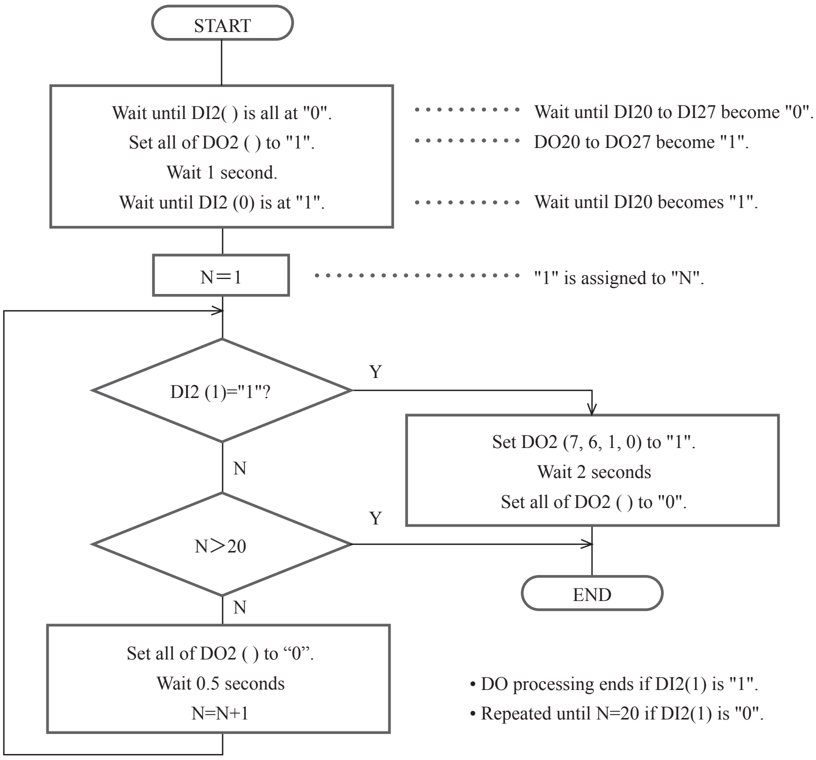
```
FOR I=1 TO 24 Repeated for I = 1 to 24.
    MOVE P,P0,Z=0.00 ················ Movement to supply position.
    PMOVE (0,I),Z=0.00 ·············· Movement to pallet point.
NEXT I
MOVE P,P0,Z=0.00
```

1.5 DI/DO (digital input and output) operation

▪ **Overview**

The following example shows general-purpose signal input and output operations through the STD.

Processing flow



SAMPLE

```
WAIT DI2( )=0 Wait until DI20 to DI27 become "0".

DO2( )=&B11111111・・・・・・・・・・・・・・・・・DO20 to DO27 become "1".
DELAY 1000
WAIT DI2(0)=1・・・・・・・・・・・・・・・・・・・・・Wait until DI21 becomes "1".
N=1
*LOOP1:
IF DI2(1)=1 THEN *PROGEND・・・・・・・・・Jumps to *PROGEND if DI21 = 1.
IF N>20 THEN *ALLEND・・・・・・・・・・・・Ended in N > 20 (jumps to *ALLEND).
DO2( )=0        DO20 to DO27 become "0".
DELAY 500
N=N+1
GOTO *LOOP1・・・・・・・・・・・・・・・・・・・・Loop is repeated.
'END ROUTINE
*PROGEND:   End processing.
DO2(7,6,1,0)=&B1111・・・・・・・・・・・・・・・Set DO27, 26, 21, 20 to "1".
DELAY 2000    Wait 2 seconds
DO2( )=0        Set DO20 to "0".
*ALLEND:
HALT
```

8
9
10
11
12
13
14

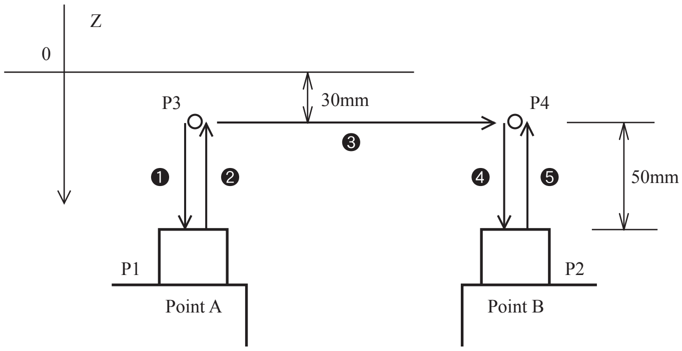
2 Application

2.1 Pick and place between 2 points

■ **Overview**

The following is an example for picking up a part at point A and placing it at point B.

Pick and place between 2 points



■ **Precondition**

1. Set the robot movement path.

- Movement path: P3→P1→P3→P4→P2→P4
- Locate P3 and P4 respectively at a position 50mm above P1 and P2 and set the P1 and P2 positions by teaching.

2. I/O signal

DO (20)	Chuck (gripper) open/close = 0: open, 1: close

- A 0.1 second wait time is set during chuck open and close.

SAMPLE: When calculating to find P3 and P4

```
      P3=P1     ••••••••••••••••••••••••••P1 coordinates are assigned to P3.
      P4=P2     ••••••••••••••••••••••••••P2 coordinates are assigned to P4.
      LOCZ(P3)=LOCZ(P3)-50.0•••••••••••••• P3 is shifted 50mm in Z UP direction.
      LOCZ(P4)=LOCZ(P4)-50.0•••••••••••••• P4 is shifted 50mm in Z UP direction.
      MOVE P,P3
      GOSUB *OPEN
      MOVE P,P1
      GOSUB *CLOSE
      MOVE P,P3
      MOVE P,P4
      MOVE P,P2
      GOSUB *OPEN
      MOVE P,P4
      HALT
*OPEN:    Chuck OPEN routine.
      DO2(0)=0
      DELAY 100
      RETURN
*CLOSE:  Chuck CLOSE routine.
      DO2(0)=1
      DELAY 100
      RETURN
```

SAMPLE: When using arch motion

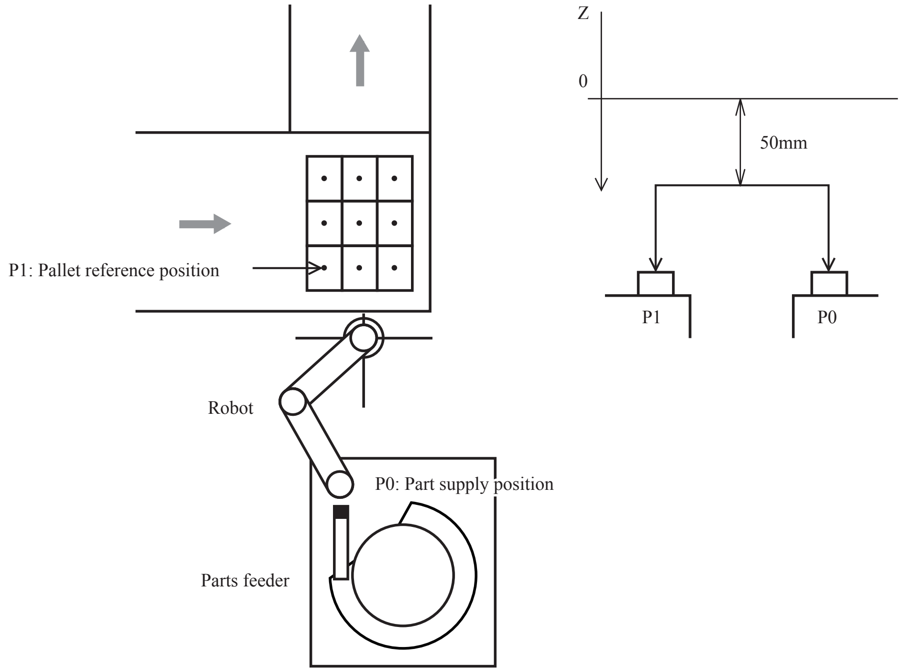
```
      P4=P2         ························P2 coordinates are assigned to P4.
      LOCZ(P4)=LOCZ(P4)-50.0··············· P4 is shifted 50mm in Z UP direction.
      GOSUB *OPEN
      MOVE P,P1,Z=30.0···················Arch motion at Z = 30mm.
      GOSUB *CLOSE
      MOVE P,P2,Z=30.0···················Arch motion at Z = 30mm.
      GOSUB *OPEN
      MOVE P,P4
      HALT
*OPEN:    Chuck OPEN routine.
      DO2(0)=0
      DELAY 100
      RETURN
*CLOSE:  Chuck CLOSE routine.
      DO2(0)=1
      DELAY 100
      RETURN
```

2.2 Palletizing

■ **Overview**

The following is an example for picking up parts supplied from the parts feeder and placing them on a pallet on the conveyor. The pallet is ejected when full.

Palletizing



■ **Precondition**

1. I/O signal

DI (30)	Component detection sensor	1: Parts are supplied
DI (31)	Pallet sensor	1: Pallet is loaded

DO (30)	Robot hand open/close	0: Open / 1: Close
DO (31)	Pallet eject	1: Eject

Robot hand open/close time is 0.1 second and pallet eject time is 0.5 seconds.

2. The points below should be input beforehand as point data.

P0	Part supply position
P1	Pallet reference position
P10	X direction pitch
P11	Y direction pitch

3. Vertical movement is performed to a position Z=50mm above the pallet and parts feeder.

SAMPLE 1: When point is calculated

```
WHILE -1        All repeated (-1 is always TRUE).
FOR A=0 TO 2
FOR B=0 TO 2

    WAIT DI(31)=1  ···················· Wait until a pallet "present" status occurs.
    WAIT DI(30)=1  ···················· Wait until the supplied component "present" status occurs.
    DO(30)=0 ························· Robot hand OPENS.
    DELAY 100
    MOVE P,P0,Z=50.0·················· Movement to supply position.
    DO(30)=1 ························· Robot hand CLOSES.
    DELAY 100
    P100=P1+P10*B+P11*A··············· Next point is calculated.
    MOVE P,P100,Z=50.0················ Movement to calculated point.
    DO(30)=0 ························· Robot hand OPENS.
    DELAY 100
NEXT
NEXT
DRIVE (3,0)     ························· Only Z-axis moves to 0.
DO(31)=1        ························· Pallet is ejected.
DELAY 500
DO(31)=0
WEND            ························· Loop is repeated.
HALT
```

SAMPLE 2: When using the palletizing function

```
* Precondition: Must be defined at pallet "0".
WHILE -1        All repeated.
FOR A=1 TO 9

    WAIT DI(31)=1  ···················· Wait until a pallet "present" status occurs.
    WAIT DI(30)=1  ···················· Wait until the supplied component "present" status occurs.
    DO(30)=0 ························· Robot hand OPENS.
    DELAY 100
    MOVE P,P0,Z=50.0·················· Movement to supply position.
    DO(30)=1 ························· Robot hand CLOSES.
    DELAY 100
    PMOVE(0,A),Z=50.0 ················ Movement to pallet point.
    DO(30)=0 ························· Robot hand OPENS.
    DELAY 100
NEXT
DRIVE(3,0)      ························· Only Z-axis moves to 0.
DO(31)=1        ························· Pallet is ejected.
DELAY 500
DO(31)=0
WEND            ························· Loop is repeated.
HALT
```

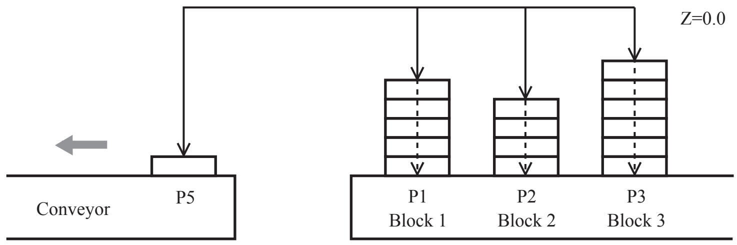
8
9
10
11
12
13
14

2.3 Pick and place of stacked parts

■ **Overview**

The following is an example for picking up parts stacked in a maximum of 6 layers and 3 blocks and placing them on the conveyor.
The number of parts per block may differ from others.
Parts are detected with a sensor installed on the robot hand.

Pick and place of stacked parts



■ **Precondition**

1. I/O signal

DI (30)	Component detection sensor	1: Parts are supplied
DI (31)	Robot hand open/close	0: Open / 1: Close

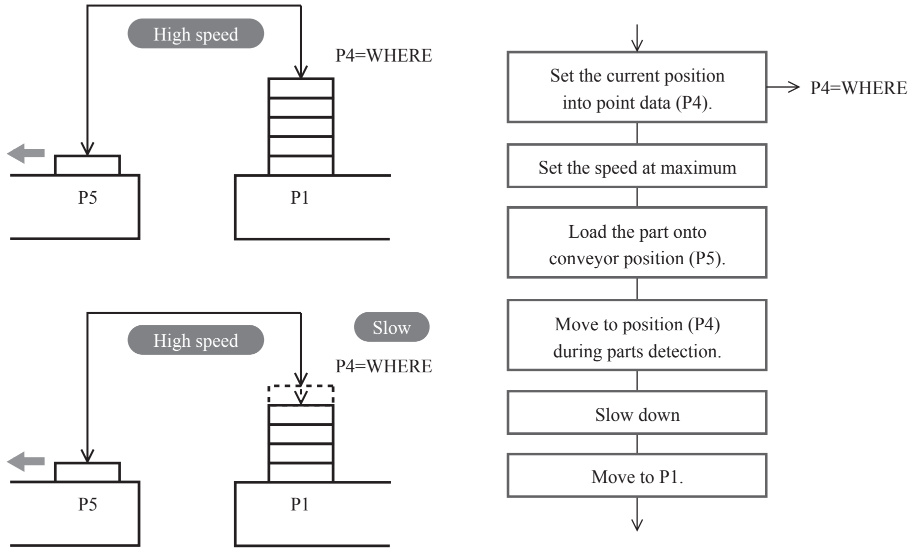
• Robot hand open/close time is 0.1 seconds.

2. The points below should be input beforehand as point data.

P1	Bottom of block 1
P2	Bottom of block 2
P3	Bottom of block 3
P5	Position on conveyor

3. Movement proceeds at maximum speeds but slows down when in proximity to the part.

Processing flow



4. Use a STOPON condition in the MOVE statement for sensor detection during movement.

SAMPLE

```
FOR A=1 TO 3
SPEED 100
GOSUB *OPEN
P6=P[A]
LOCZ(P6)=0.00
MOVE P,P6,Z=0.0
WHILE -1
    SPEED 20
    MOVE P,P[A],STOPON DI3(0)=1
    IF DI3(0)=0 THEN *L1
    'SENSOR ON
    P4=JTOXY(WHERE)
    GOSUB *CLOSE
    SPEED 100
    MOVE P,P5,Z=0.0
    GOSUB *OPEN
    MOVE P,P4,Z=0.0
WEND
*L1: 'SENSOR OFF
NEXT A
SPEED 100
DRIVE (3,0)
HALT
*OPEN:
DO3(0)=0
DELAY 100
RETURN
*CLOSE:
DO3(0)=1
DELAY 100
RETURN
```

2.4 Parts inspection (Multi-tasking example)

■ **Overview**

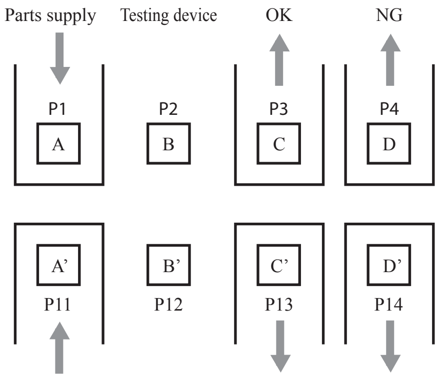
One robot is used to inspect two different parts and sort them according to the OK/NG results.

The robot picks up the part at point A and moves it to the testing device at point B. The testing device checks the part and sends it to point C if OK or to point D if NG.

The part at point A’ is picked up and moved to the testing device at point B' in the same way. The testing device checks the part and sends it to point C’ if OK or to point D’ if NG.

It is assumed that 10 to 15 seconds are required for the testing device to issue the OK/NG results.

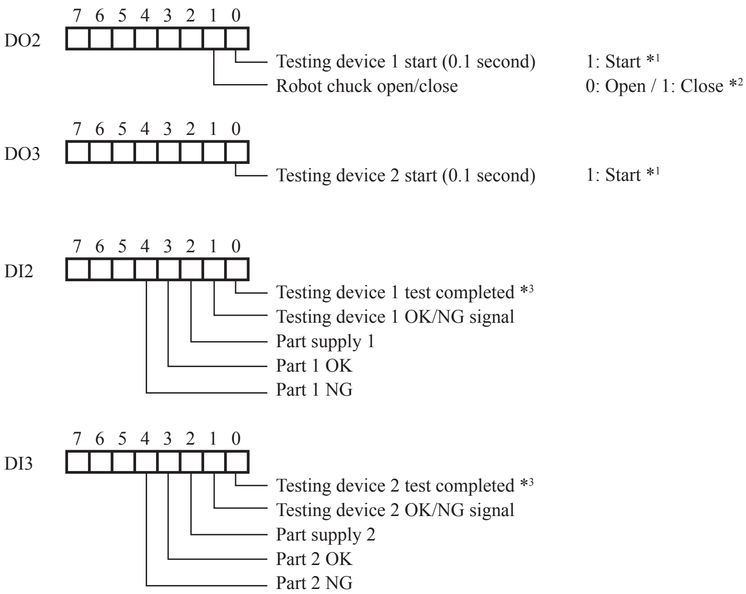
Parts inspection (Multi-tasking example)



■ **Precondition**

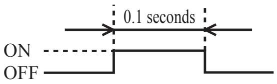
1. I/O signal

I/O signal



NOTE

- *1: As the start signal, supply a 0.1 second pulse signal to the testing device.



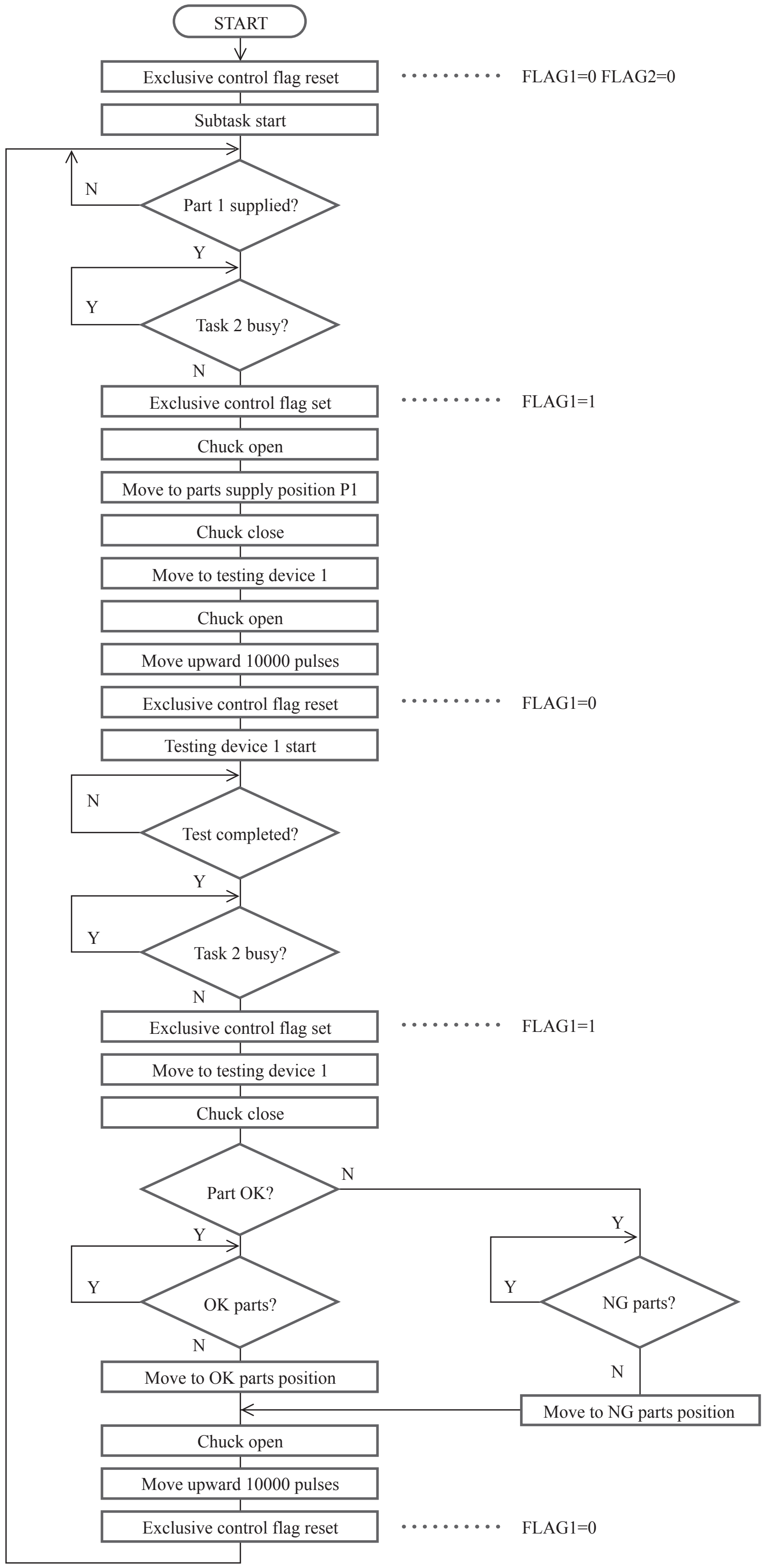
- *2: Chuck open and close time is 0.1 seconds.
- *3: Each time a test is finished, the test completion signal and OK/NG signal are sent from the testing device. After testing, the test completion signal turns ON (=1), and the OK/ NG signal turns ON (=1) when the result is OK and turns OFF (=0) when NG.

2. The main task (task 1) is used to test part 1 and the subtask (task 2) is used to test part 2.
3. An exclusive control flag is used to allow other tasks to run while waiting for the test completion signal from the testing device.

FLAG1	0: Executing Task 1	(Task 2 execution enabled)
	1: Task 1 standby	(Task 2 execution disabled)
FLAG2	0: Executing Task 2	(Task 1 execution enabled)
	1: Task 2 standby	(Task 1 execution disabled)

4. Flow chart

Processing flow



Task 2 (subtask) runs in the same flow.

- **Program example**

SAMPLE

```
<Main task>
FLAG1=0
FLAG2=0
UPPOS=0.0
START *S1,T2              Subtask Start
*L1:
WAIT DI2(2)=1             Part supply standby
WAIT FLAG2=0              Other tasks waiting for standby status.
FLAG1=1                   Exclusive control flag set
GOSUB *OPEN               Chuck open
MOVE P,P1,Z=UPPOS         Move to part supply position
GOSUB *CLOSE              Chuck close
MOVE P,P2,Z=UPPOS         Move to testing device
GOSUB *OPEN               Chuck open
DRIVEI (3,-10000)         Move Z-axis upward Z 10,000 pulses
FLAG1=0                   Exclusive control flag reset
DO2(0)=1                  Testing device start
DELAY 100
DO2(0)=0
WAIT DI2(0)=1             Test completion standby
WAIT FLAG2=0              Task completion standby
FLAG1=1                   Exclusive control flag set
MOVE P,P2,Z=UPPOS         Move to testing device
GOSUB *CLOSE              Chuck close
IF DI2(1)=1 THEN          Test
'GOOD
WAIT DI4(2)=0             Part movement standby
MOVE P,P3,Z=UPPOS         Move to OK parts position
ELSE
'NG
WAIT DI2(4)=0             Part movement standby
MOVE P,P4,Z=UPPOS         Move to NG parts position
ENDIF
GOSUB *OPEN               Chuck open
DRIVEI (3,-10000)         Move Z-axis upward Z 10,000 pulses
FLAG1=0                   Exclusive control flag reset
GOTO *L1
*OPEN:
DO2(1)=0
DELAY 100
RETURN
*CLOSE:
DO2(1)=1
DELAY 100
RETURN
```

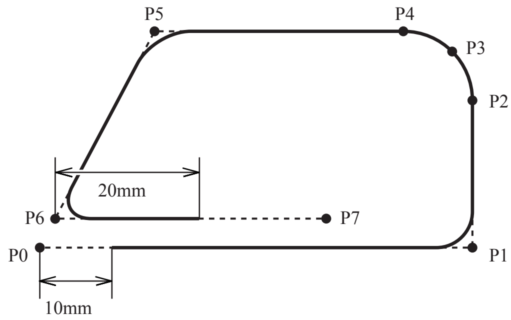
```
<Subtask>
*S1
WAIT DI3(2)=1
WAIT FLAG1=0
FLAG2=1
GOSUB *OPEN
MOVE P,P11,Z=UPPOS
GOSUB *CLOSE
MOVE P,P12,Z=UPPOS
GOSUB *OPEN
DRIVEI (3,-10000)
FLAG2=0
DO3(0)=1
DELAY 100
DO3(0)=0
WAIT DI3(0)=1
WAIT FLAG1=0
FLAG2=1
MOVE P,P12,Z=UPPOS
GOSUB *CLOSE
IF DI3(1)=1 THEN
'GOOD
WAIT DI3(3)=0
MOVE P,P13,Z=UPPOS
ELSE
'NG
WAIT DI3(4)=0
MOVE P,P14,Z=UPPOS
ENDIF
GOSUB *OPEN
DRIVEI (3,-10000)
FLAG2=0
GOTO *S1
```

2.5 Sealing

■ **Overview**

This section shows an example of the parts sealing operation.

Sealing



■ **Precondition**

1. I/O signal

DO (20)	Valve open/close	1: Open / 0: Close

2. Set P0 to P7 by teaching.

SAMPLE

```
MOVE P,P0,Z=0
SPEED 40
PATH SET        .......................................Start of path setting
PATH L,P1,DO(20)=1@10.0.......................Start of coating
                                               at 10 mm position
PATH L,P2                                                              Path setting
PATH C,P3,P4                                                           (Robot does not operate.)
PATH L,P5
PATH L,P6,S=30
PATH L,P7,DO(20)=0@20.0.......................End of coating at
                                               20 mm position
PATH END        .......................................End of path setting
PATH START      .......................................Execute PATH motion (Robot starts from P0 and stops at P7.)
HALT
```

8
9
10
11
12
13
14

2.6 Connection to an external device through RS-232C (example 1)

■ **Overview**

Point data can be written in a program by using an external device connected to the YRC series controller via the RS-232C port.

■ **Precondition**

1. Input to the external device from the controller
   SDATA/X/Y [cr/lf]

2. Output to the controller from the external device

NOTE

• [cr/lf] indicates CR code (=0Dh) + LF code (=0Ah).

POINT DATA							
P10=	156.42	243.91	0.00	0.00	0.00	0.00	[cr/lf]

SAMPLE

```
'INIT
      VCMD$="SDATA/X/Y"
      P0=      0.00      0.00      0.00      0.00      0.00      0.00
'MAIN ROUTINE
      MOVE P, P0
*ST:
      SEND VCMD$ TO CMU
      SEND CMU TO P10
      MOVE P, P10
GOTO *ST
```

MEMO

- "SEND xxx TO CMU" outputs the contents specified by "xxx" through the RS-232C.
- "SEND CMU TO xxx" sends data into the files specified by "xxx" through the RS-232C.

2.7 Connection to an external device through RS-232C (example 2)

■ **Overview**

Point data can be created from the desired character strings and written in a program by using an external device connected to the YRC series controller via the RS-232C port.

■ **Precondition**

1. Input to the external device from the controller

SDATA/X/Y [cr/lf]

NOTE

- [cr/lf] indicates CR code (=0Dh) + LF code (=0Ah).

2. Output to the controller from the external device

X=156.42, Y=243.91 [cr/lf]

MEMO

- "SEND xxx TO CMU" outputs the contents specified by "xxx" through the RS-232C.
- "SEND CMU TO xxx" sends data into the files specified by "xxx" through the RS-232C.
- The LEN ( ) function obtains the length of the character string.
- The MID$ ( ) function obtains the specified character string from among the character strings.
- The VAL ( ) function obtains the value from the character string.

8

9

10

11

12

13

14

SAMPLE

```
'INT
	VCMD$="SDATA/X/Y"
	VIN$=""
	VX$=""
	VY$=""
	P0=	0.00	0.00	0.00	0.00	0.00	0.00
	P11=	100.00	100.00	0.00	0.00	0.00	0.00
'MAIN ROUTINE
	MOVE P, P0
*ST:
	SEND VCMD$ TO CMU
	SEND CMU TO VIN$
	I=1
	VMAX=LEN(VIN$)
*LOOP:
	IF I>VMAX THEN GOTO *E_LOOP
	C$=MID$(VIN$,I ,1)
	IF C$="X" THEN
		I=I+2
		J=I
*X_LOOP:
		C$=MID$(VIN$, J, 1)
		IF C$="," THEN
*X1_LP:
			L=J-I
			VX$=MID$(VIN$, I, L)
			I=J+1
			GOTO *LOOP
		ENDIF
		J=J+1
		IF J>VMAX  THEN  GOTO  *X1_LP
		GOTO  *X_LOOP
	ENDIF
	IF C$="Y"  THEN
		I=I+2
		J=I
*Y_LOOP:
		C$=MID$(VIN$, J, 1)
		IF C$=","THEN
*Y1_LP:
			L=J-I
			VY$=MID$(VIN$, I, L)
			I=J+1
			GOTO  *LOOP
		ENDIF
		J=J+1
		IF  J>VMAX  THEN  GOTO  *Y1_LP
		GOTO *Y_LOOP
	END IF
	I=I+1
	GOTO *LOOP
*E_LOOP:
	WX=VAL(VX$)
	WY=VAL(VY$)
	LOCX(P11)=WX
	LOCY(P11)=WY
	MOVE P, P11
GOTO  *ST
```

Chapter 12

Online commands

1 Online Command List......................................12-1

2 Key operation...12-6

3 Utility operation...12-12

4 Data handling...12-24

5 Executing the robot language independently..12-44

6 Control codes..12-46

1 Online Command List

Online commands can be used to operate the controller via an RS-232C interface or via an Ethernet (option). This chapter explains the online commands which can be used. For details regarding the RS-232C connection method, refer to the "YRC Controller User's Manual". For details regarding Ethernet, refer to the "YRC Series Ethernet Manual".

About termination codes

During data transmission, the controller adds the following codes to the end of a line of transmission data.

- RS-232C
- CR (0Dh) and LF (0Ah) are added to the end of the line when the "Termination code" parameter of communication parameters is set to "CRLF".
- CR (0Dh) is added to the end of the line when the "Termination code" parameter of communication parameters is set to "CR".
- Ethernet (option)
- CR (0Dh) and LF (0Ah) are added to the end of the line.

When data is received, then the data up to CR (0Dh) is treated as one line regardless of the "Termination code" parameter setting, so LF (0Ah) is ignored.
The termination code is expressed as [cr/lf] in the detailed description of each online command in "12.2 Key operation" onwards.

1.1 Online command list: Function specific

Key operation

Operation type		Command	Option	Condition
Change mode	AUTO mode PROGRAM mode MANUAL mode SYSTEM mode	AUTO PROGRAM MANUAL SYSTEM		3
Program	Reset program Execute program Execute one line Skip one line Execute to next line Stop program	RESET RUN STEP SKIP NEXT STOP		4 2
Set break point		BREAK	m, n (m: break point No., n: line)	4
Switch execution task		CHGTSK		3
Change manual speed	Main robot	MSPEED	k (k : 1-100)	3
Move to absolute reset position	Main robot	ABSADJ	k, 0 or k, 1 (k : 1-6)	3
Absolute reset on each axis	Main robot	ABSRESET	k (k : 1-6)	
Return-to-origin	Main robot	ORGRTN	k (k : 1-6)	3
Manual movement (inching)	Main robot	INCH	k+ or k- (k : X, Y, Z, R, A, B)	3
Manual movement (jog)	Main robot	JOG	k+ or k- (k : X, Y, Z, R, A, B)	3
Point data teaching	Main robot	TEACH	m (m : point No.)	3

Utility

Operation type	Command	Option	Condition
Acquire program execution status	PADDR		4
Copy program 1 to program 2 Copy points "m - n" to point "k" Copy point comments "m - n" to point comment "k"	COPY	<program 1> TO <program 2> Pm-Pn TO Pk PCm-PCn TO PCk	3
Delete program Delete points "m - n" Delete point comments "m - n" Delete pallet "m"	ERA	<program> Pm-Pn PCm-PCn PLm	3
Rename "program 1" to "program 2"	REN	<program 1> TO <program 2>	3
Change program attribute	ATTR	<program> TO s (s : RW/RO)	3
Initialize data Program Point Shift Hand Pallet Point comment All data except parameters Parameter All data (MEM+PRM)	INIT	PGM PNT SFT HND PLT PCM MEM PRM ALL	3
Initialize data Communication parameter	INIT	CMU	3
Initialize data Error log	INIT	LOG	3
Setting Display language	LANGUAGE	k	3
Setting Point units Clear line message Setting Access level Setting Execution level Sequence execution flag	UNIT MSGCLR ACCESS EXELVL SEQUENCE	k k k k	3 1 3 3 3
Setting Main hand system	ARMTYP	m, k	3
Reset internal emergency stop flag	EMGRST		1
Check or set date Check or set time	DATE TIME		2 2

Conditions:
1. Always executable.
2. Not executable during inputs from the programming box.
3. Not executable during inputs from the programming box, and while the program is running.
4. Not executable during inputs from the programming box, while the program is running, and when specific restrictions apply.

Data handling

Operation type		Command	Option	Condition
Acquiring status	Display language	?	LANGUAGE	1
	Access level		ACCESS	
	Arm status		ARM	
	Break point status		BREAK	
	Controller configuration		CONFIG	
	Execution level		EXELVL	
	Mode indication		MOD	
	Error message		MSG [m, n]	
	Return-to-origin status		ORIGIN	
	Absolute reset status		ABSRST	
	Servo status		SERVO	
	Sequence execution flag status		SEQUENCE	
	AUTO/MANUAL speed status		SPEED	
	Point unit status		UNIT	
	Version information		VER	
	Current main robot position (pulse coordinate)		WHERE	
	Current main robot position (XY coordinate) (including extended setting)		WHRXY	
	Task number		TASKS	
	Task operation status		TSKMON	
	Selected shift status		SHIFT	
	Selected hand status		HAND	
	Remaining memory capacity		MEM	
	Emergency stop status		EMG	
	Error status by self-diagnosis		SELFCHK	
	Option slot status		OPSLOT	
	Main group current torque value		CHKTRQ	
	Numerical data		Numerical expression	
	Character string data		Character string expression	
	Point data		Point expression	
	Shift data		Shift expression	
Data readout		READ		2
Data write		WRITE		2

Robot language independent execution

Operation type	Command	Option	Condition
Program switching	SWI	<program>	4
Robot language executable independently			4

Control code

Operation type	Command	Option	Condition
Execution language interruption	^C(=03H)		1

Conditions:
1. Always executable.
2. Not executable during inputs from the programming box.
3. Not executable during inputs from the programming box, and while the program is running.
4. Not executable during inputs from the programming box, while the program is running, and when specific restrictions apply.

1.2 Online command list: In alphabetic order

Command	Option	Meaning	Condition
?	ABSRST	Acquire absolute reset status	1
	ACCESS	Acquire access level	1
	ARM	Acquire arm status	1
	BREAK	Acquire break point status	1
	CHKTRQ	Acquire main group current torque value	1
	CONFIG	Acquire controller configuration	1
	EMG	Acquire emergency stop status	1
	EXELVL	Acquire execution level	1
	HAND	Acquire selected hand status	1
	LANGUAGE	Acquire display language	1
	MEM	Acquire remaining memory capacity	1
	MOD	Acquire mode indication	1
	MSG [m, n]	Acquire error message	1
	OPSLOT	Acquire option slot status	1
	ORIGIN	Acquire return-to-origin status	1
	SELFCHK	Acquire error status by self-diagnosis	1
	SEQUENCE	Acquire sequence execution flag status	1
	SERVO	Acquire servo status	1
	SHIFT	Acquire selected shift status	1
	SPEED	Acquire AUTO/MANUAL speed status	1
	TASKS	Acquire task number	1
	TSKMON	Acquire task operation status	1
	UNIT	Acquire point position status	1
	VER	Acquire version	1
	WHERE	Acquire current main robot position (pulse coordinate)	1
	WHRXY	Acquire current main robot position (XY coordinate)	1
	Shift expression	Acquire shift data	1
	Point expression	Acquire point data	1
	Numeric expression	Acquire numeric data	1
	Character string expression	Acquire character string data	1
^C (=03H)		Execution language interruption	1
ABSADJ	k, 0 or k, 1 (k : 1-6)	Move to absolute reset position Main robot	3
ABSRESET	k (k : 1-6)	Absolute reset on each axis Main robot	3
ACCESS	k	Set access level	3
ARMTYP	m, k	Set main hand system	3
ATTR	<program> TO s (s : RW/RO)	Change program attribute	3
AUTO		Change mode: AUTO mode	3
BREAK	m, n (m: break point No., n: line)	Set break point	4
CHGTSK		Switch execution task	3
COPY	<program1> to <program2>	Copy program 1 to program 2	3
	PCm-PCn TO PCk	Copy point comments "m - n" to point comments "k"	3
	Pm-Pn TO Pk	Copy points "m - n" to points "k"	3
DATE		Check or set the date	2
EMGRST		Reset internal emergency stop flag	1
ERA	<program>	Delete program	3
	PCm-PCn	Delete point comments "m" to "n"	3
	PLm	Delete pallet "m"	3
	Pm-Pn	Delete points "m" to "n"	3
EXELVL	k	Execution level	3
INCH	k+ or k- (k : X, Y, Z, R, A, B)	Manual movement (inching) Main robot	3

Command	Option	Meaning	Condition
INIT	ALL	Initialize all data (MEM+PRM)	3
	CMU	Initialize communication parameter	3
	HND	Initialize hand data	3
	LOG	Initialize error history	3
	MEM	Initialize all memory data except parameters	3
	PCM	Initialize point comment data	3
	PGM	Initialize program data	3
	PLT	Initialize pallet data	3
	PNT	Initialize point data	3
	PRM	Initialize parameter data	3
	SFT	Initialize shift data	3
JOG	k+ or k- (k : X, Y, Z, R, A, B)	Manual movement (jog) Main robot	3
LANGUAGE	k	Set display language	3
MANUAL		Change mode: MANUAL mode	3
MSGCLR		Setting Clear line message	1
MSPEED	k (k : 1-100)	Change manual speed Main robot	3
NEXT		Execute program to next line	4
ORGRTN	k (k : 1-6)	Return-to-origin Main robot	3
PADDR		Acquire program execution status	4
PROGRAM		Change mode: PROGRAM mode	3
READ		Read data	2
REN	<program 1> TO <program 2>	Change program name from "1" to "2"	3
RESET		Reset program	4
RUN		Execute program	4
SEQUENCE	k	Set sequence execution flag	3
SKIP		Program: Skip one line	4
STEP		Program: Execute one line	4
STOP		Stop program	2
SWI	<program>	Switch programs	4
SYSTEM		Change mode: SYSTEM mode	3
TEACH	m (m: point number)	Point data teaching Main robot	3
TIME		Check or set time	2
UNIT	k	Set point unit system	3
WRITE		Write data	2
-		Robot language executable independently	4

Conditions:
1. Always executable.
2. Not executable during inputs from the programming box.
3. Not executable during inputs from the programming box, and while the program is running.
4. Not executable during inputs from the programming box, while the program is running, and when specific restrictions apply.

2 Key operation

2.1 Changing the mode

Command format

@AUTO[cr/lf]
@PROGRAM[cr/lf]
@MANUAL[cr/lf]
@SYSTEM[cr/lf]

Response format

OK[cr/lf]

NOTE

- Basically, a response "OK" appears when an instruction from key operation online command is received.
- An error message responds if online commands cannot be executed due to error.

Meaning Changes the mode.

AUTO..Changes to AUTO mode.
PROGRAM ....Changes to PROGRAM mode.
MANUAL ....Changes to MANUAL mode.
SYSTEM ....Changes to SYSTEM mode.

SAMPLE

Command: @AUTO[cr/lf]
Response: OK[cr/lf]

2.2 AUTO mode operation

1. Program execution

Command format

```
@RESET[cr/lf]
@RUN[cr/lf]
@STEP[cr/lf]
@SKIP[cr/lf]
@NEXT[cr/lf]
@STOP[cr/lf]
```

Response format

```
OK[cr/lf]
```

NOTE

- Programs can be executed only in AUTO mode.

Meaning Executes or stops the current program.

RESET..Resets the program.
RUN ....Executes the program.
STEP..Executes one line of the program. (Enters the subroutine.)
SKIP ....Skips one line of program. (Skips one line of the subroutine.)
NEXT ....Executes to the next line. (Executes the subroutine as one line.)
STOP ....Stops the program.

SAMPLE

Command:	@RUN[cr/lf]
Response:	OK[cr/lf]

2. Setting a break point

Command format

@BREAK m,n[cr/lf]

Response format

OK[cr/lf]

NOTE

- Programs can be executed only in AUTO mode.

Values
m...Break point number: 1 to 4
n...Line number to set a break point: 1 to 9999

Meaning Sets a break point used to temporarily stop execution of the program.
When setting a break point in the COMMON program, the line number should be +10000.
Break point is cleared when 0 is specified as the line number.

SAMPLE

Command:	@BREAK 1,28[cr/lf]
Response:	OK[cr/lf]

3. Switching the execution task

Command format

@CHGTSK[cr/lf]

Response format

OK[cr/lf]

NOTE

- Programs can be executed only in AUTO mode.

Meaning Switches the selected task while program execution is stopped.
The ongoing task is switched to another task not being executed in order from task 1 → 2 → ... → 8 → 1.

SAMPLE

Command:	@CHGTSK[cr/lf]
Response:	OK[cr/lf]

2.3 MANUAL mode operation

1. Changing the MANUAL mode speed

Command format
@MSPEED k[cr/lf]

Response format
OK[cr/lf]

NOTE

- The MANUAL mode speed can be changed only in MANUAL mode.

Values k..Manual movement speed: 1 to 100

Meaning Changes the MANUAL mode movement speed.
MSPEED: Changes manual movement speed for main robot.

SAMPLE	
Command:	@MSPEED 50[cr/lf]
Response:	OK[cr/lf]

2. Absolute reset

Command format
@ABSADJ k,f[cr/lf] @ABSRESET k[cr/lf]

Response format
OK[cr/lf]

NOTE

- The MANUAL mode speed can be changed only in MANUAL mode.

Values k..Designated axis: 1 to 6
f..Movement direction / 0: + direction, 1: - direction

Meaning Performs absolute reset.
ABSADJ....................................Moves the main robot axis to an absolute reset position.
ABSRESET ....Performs absolute reset on the main robot axis.

SAMPLE	
Command:	@ABSADJ 1,0[cr/lf]
Response:	OK[cr/lf]

MEMO

- ABSADJ can be used at mark format axes.

3. Return-to-origin operation

Command format
@ORGRTN k[cr/lf]

Response format
OK[cr/lf]

Values k..Specified axis: 1 to 6

NOTE

• The MANUAL mode speed can be changed only in MANUAL mode.

Meaning Performs return-to-origin on the specified axis.
Performs return-to-origin on an incremental mode axis when return-to-origin is executed.
Performs absolute search on a semi-absolute mode axis when return-to-origin is executed.
ORGRTN ....Performs return-to-origin on the specified main robot axis.

NOTE

• Response is transmitted after movement is complete.

SAMPLE	
Command:	@ORGRTN 1[cr/lf]
Response:	OK[cr/lf]

4. Manual movement: inching

Command format
@INCH km[cr/lf]

Response format
OK[cr/lf]

NOTE

• The MANUAL mode speed can be changed only in MANUAL mode.

Values k..Specified axis: X, Y, Z, R, A, B
m...Movement direction / +, -

Meaning Manually moves (inching motion) the specified axis. The robot performs the same motion as when moved manually in inching mode with the programming box jog keys (moves a fixed distance each time a jog key is pressed).
INCH ....Moves the specified main robot axis in inching mode.

SAMPLE	
Command:	@INCH X+[cr/lf]
Response:	OK[cr/lf]

5. Manual movement: jog

Command format
@JOG km[cr/lf]

Response format
OK[cr/lf]

NOTE
• Response is transmitted after movement is complete.

Values k..Specified axis: X, Y, Z, R, A, B
m..Movement direction / +, -

Meaning Manually moves (jog motion) the specified axis. The robot performs the same motion as when holding down the programming box jog keys in manual mode.
After the robot has started moving, it will stop when any of the following occurs.
• When software limit was reached.
• When interlock signal was turned off.
• When STOP key on the programming box was pressed.
• When an online command (^C (=03H)) to interrupt execution was input.
JOG...Moves the specified main robot axis in jog mode.

SAMPLE	
Command:	@JOG X+[cr/lf]
Response:	OK[cr/lf]

6. Point data teaching

NOTE
• The MANUAL mode speed can be changed only in MANUAL mode.

Command format
@TEACH mmmm[cr/lf]

Response format
OK[cr/lf]

Values mmmm ....Point number for registering point data: 0 to 9999

Meaning Registers the current robot position as point data for the specified point number. If point data is already registered in the specified point number, then that point data will be overwritten. Point data is registered in the same format as the currently selected unit system.
TEACH.....................................Registers the current position of the main group as point data for the specified point number.

SAMPLE	
Command:	@TEACH 100[cr/lf]
Response:	OK[cr/lf]

8
9
10
11
12
13
14

3 Utility operation

3.1 Acquiring the program execution status

Command format

@PADDR[cr/lf]

Response format

<program name> Tn, m, k[cr/lf]

NOTE
- The current program execution status can be acquired only when the program is stopped during AUTO mode.

Values
<program name> ....Currently selected program name
Tn ....Current task number: 1 to 8
m..Current program line number: 1 to 9999
k...Current task priority level:17 to 47

Meaning Acquires the current program execution status.

MEMO
- If the COMMON program is used, the response format might become as follows.
<COMMON>/<program name>, Tn,m,k[cr/lf]

SAMPLE

Command:	@PADDR[cr/lf]
Response:	<TEST>,T3,134,32[cr/lf]

3.2 Copy

1. Copying a program

Command format

COPY <program name 1> TO <program name 2> [cr/lf]

Response format

@COPY <program name 1> TO <program name 2> [cr/lf]

Values
<program name 1> ....Program name in copy source (8 characters or less consisting of alphanumeric characters and underscore)
<program name 2> ....Program name in copy destination (8 characters or less consisting of alphanumeric characters and underscore)

Meaning Copies the contents of program name 1 under program name 2.

SAMPLE

Command:	@COPY <TEST1> TO <TEST2>[cr/lf]
Response:	OK[cr/lf]

2. Copying point data

Command format
@COPY Pmmmm-Pnnnn TO Pkkkk[cr/lf]

Response format
OK[cr/lf]

Values
mmmm ....Top point number in copy source: 0 to 9999
nnnn ....Last point number in copy source: 0 to 9999
kkkk ....Top point number in copy destination: 0 to 9999

Meaning Copies the point data between Pmmmm and Pnnnn to Pkkkk.

SAMPLE

Command:	@COPY P101-P200 TO P1101[cr/lf]
Response:	OK[cr/lf]

3. Copying point comments

Command format
@COPY PCmmmm-PCnnnn TO PCkkkk[cr/lf]

Response format
OK[cr/lf]

Values
mmmm ....Top point comment number in copy source: 0 to 9999
nnnn ....Last point comment number in copy source: 0 to 9999
kkkk ....Top point comment number in copy destination: 0 to 9999

Meaning Copies the point comments between PCmmmm and PCnnnn to PCkkkk.

SAMPLE

Command:	@COPY PC101-PC200 TO PC1101[cr/lf]
Response:	OK[cr/lf]

3.3 Erase

1. Erasing a program

Command format

@ERA <program name> [cr/lf]

Response format

OK[cr/lf]

Values <program name> ....Program name to be erased (8 characters or less consisting of alphanumeric characters and underscore)

Meaning Erases the designated program.

SAMPLE

Command:	@ERA <TEST1>[cr/lf]
Response:	OK[cr/lf]

2. Erasing point data

Command format

@ERA Pmmmm-Pnnnn[cr/lf]

Response format

OK[cr/lf]

Values mmmm ....Top point number to be erased: 0 to 9999
nnnn..Last point number to be erased: 0 to 9999

Meaning Erases the point data between Pmmmm and Pnnnn.

SAMPLE

Command:	@ERA P101-P200[cr/lf]
Response:	OK[cr/lf]

3. Erasing point comments

Command format

@ERA PCmmmm-PCnnnn[cr/lf]

Response format

OK[cr/lf]

Values mmmm ....Top point comment number to be erased: 0 to 9999
nnnn...Last point comment number to be erased: 0 to 9999

Meaning Erases the point comments between PCmmmm and PCnnnn.

SAMPLE

Command: @ERA PC101-PC200[cr/lf]
Response: OK[cr/lf]

4. Erasing pallet data

Command format

@ERA PLm[cr/lf]

Response format

OK[cr/lf]

Values m..Pallet number to be erased: 0 to 19

Meaning Erases the PLm pallet data.

SAMPLE

Command: @ERA PL1[cr/lf]
Response: OK[cr/lf]

8
9
10
11
12
13
14

3.4 Rename program name

Command format

@REN <program name 1> TO <program name 2> [cr/lf]

Response format

OK[cr/lf]

Values

<program name 1> ....Program name before renaming (8 characters or less consisting of alphanumeric characters and underscore)

<program name 2> ....Program name after renaming (8 characters or less consisting of alphanumeric characters and underscore)

Meaning Changes the name of the specified program.

SAMPLE

Command: @REN <TEST1> TO <TEST2>[cr/lf]
Response: OK[cr/lf]

3.5 Changing the program attribute

Command format

@ATTR <program name> TO s[cr/lf]

Response format

OK[cr/lf]

Values

<program name> ....Program name to change the attribute (8 characters or less consisting of alphanumeric characters and underscore)

s ....Attribute / RW: read & write, RO: read only

Meaning Changes the attribute of the designated program.

SAMPLE

Command: @ATTR <TEST1> TO RO[cr/lf]
Response: OK[cr/lf]

3.6 Initialize

1. Initializing the memory

Command format
@INIT <memory area>[cr/lf]

Response format
OK[cr/lf]

Values <memory area> ....One of the following memory areas is specified.

PGM ....Initializes the program area.
PNT ....Initializes the point data area.
SFT ....Initializes the shift data area.
HND ....Initializes the hand data area.
PLT ....Initializes the pallet data area.
PCM ....Initializes the point comment area.
MEM ....Initializes the above areas (PGM ... all data up to PCM).
PRM ....Initializes the parameter area.
ALL ....Initializes all areas (MEM+PRM).

Meaning Initializes the memory.

SAMPLE

Command:	@INIT PGM[cr/lf]
Response:	OK[cr/lf]

2. Initializing the communication port

Command format
@INIT CMU[cr/lf]

Response format
OK[cr/lf]

Meaning Initializes the communication port parameters.
For information about the communication port initial settings, refer to the Controller user's manual.

SAMPLE

Command:	@INIT CMU[cr/lf]
Response:	OK[cr/lf]

3. Initializing the error log

Command format
@INIT LOG[cr/lf]

Response format
OK[cr/lf]

Meaning Initializes the error log.

SAMPLE	
Command:	@INIT LOG[cr/lf]
Response:	OK[cr/lf]

3.7 Setting the display language

Command format
@LANGUAGE k[cr/lf]

Response format
OK[cr/lf]

Values k...Display language / 0: Japanese, 1: English

Meaning Sets the controller display language.

SAMPLE	
Command:	@ LANGUAGE 1[cr/lf]
Response:	OK[cr/lf]

3.8 Setting the coordinates and units in MANUAL mode

Command format

@UNIT k[cr/lf]

Response format

OK[cr/lf]

Values k..Unit definition
0 : pulses
1 : mm or degrees
2 : mm or degrees in tool coordinate mode

Meaning Select the display unit to indicate current position.
k=2 (tool coordinate mode) can be selected only when the hand attached to the R-axis of a SCARA robot is selected.

SAMPLE

Command:	@UNIT 1[cr/lf]
Response:	OK[cr/lf]

3.9 Clearing the programming box error message

Command format

@MSGCLR[cr/lf]

Response format

OK[cr/lf]

Meaning Clears the error messages displayed on the programming box.

SAMPLE

Command:	@MSGCLR[cr/lf]
Response:	OK[cr/lf]

3.10 Setting the UTILITY mode

1. Setting the access level

Command format
@ACCESS k[cr/lf]

Response format
OK[cr/lf]

REFERENCE

• For details regarding the access level, refer to the Controller user's manual.

Values k..Access level: 0 to 3

Meaning Sets the access level.

SAMPLE	
Command:	@ ACCESS 1[cr/lf]
Response:	OK[cr/lf]

2. Setting the execution level

Command format
@EXELVL k[cr/lf]

Response format
OK[cr/lf]

REFERENCE

• For details regarding the execution level, refer to the Controller user's manual.

Values k..Execution level: 0 to 8

Meaning Sets the execution level.

SAMPLE	
Command:	@ EXELVL 1[cr/lf]
Response:	OK[cr/lf]

3. Setting the sequence program execution flag

Command format
@SEQUENCE k[cr/lf]

Response format
OK[cr/lf]

Values k...Execution flag / 0: disable, 1: enable, 3: enable (DO reset)

Meaning Sets the sequence program execution flag.

SAMPLE	
Command:	@ SEQUENCE 1[cr/lf]
Response:	OK[cr/lf]

4. Setting the SCARA robot hand system

Command format
@ARMTYP m,k[cr/lf]

Response format
OK[cr/lf]

Values m...................................Current hand system / 0: right-handed system, 1: left-handed system
k....................................Hand system at program reset / 0: right-handed system, 1: left-handed system

Meaning Sets the SCARA robot hand system.
ARMTYP ....Changes the main robot hand system.

SAMPLE	
Command:	@ ARMTYP 0, 0 [cr/lf]
Response:	OK[cr/lf]

5. Resetting the internal emergency stop flag

Command format
@EMGRST[cr/lf]

Response format
OK[cr/lf]

Meaning Resets the internal emergency stop flag.

SAMPLE	
Command:	@ EMGRST[cr/lf]
Response:	OK[cr/lf]

8 9 10 11 12 13 14

3.11 Checking and setting the date

Command format
@DATE[cr/lf]

Response format
current date: yy/mm/dd[cr/lf] enter new date: (YY/MM/DD)[cr/lf]

Values yy/mm/dd ....Current date (year, month, day)
yy...Lower 2 digits of the year (00 to 99)
mm...Month (01 to 12)
dd...Day (01 to 31)

Meaning Acquires the current date in the controller and sets a new date.

Command format
aa/bb/cc[cr/lf]

Response format
OK[cr/lf] or "error message" [cr/lf]

NOTE
- To change only the year or month, the slash (/) following it can be omitted.

Example:

To set the year to 2007, enter 07[cr/lf].

To set the month to June, enter /06[cr/lf].

Values aa/bb/cc......................................Date to be set. (year/month/day)
aa ....Lower 2 digits of the year (00 to 99) *This can be omitted.
bb...Month (01 to 12) *This can be omitted.
cc ....Day (01 to 31) *This can be omitted.

MEMO
- The currently set values are used for the omitted items.
- If only [cr/lf] is transmitted, then the date remains unchanged.
- If an improbable date is entered, then "5.2: Data error" occurs.

SAMPLE 1

To change only the day,
//15[cr/lf] Day is set to 15th.

SAMPLE 2

Command:	@DATE[cr/lf]
Response:	current date: 07/05/10[cr/lf]
	enter new date: (YY/MM/DD)[cr/lf]
Transmission:	07/05/11[cr/lf]
Response:	OK[cr/lf]

3.12 Checking and setting the time

8
9
10
11
12
13
14

Command format

@TIME[cr/lf]

Response format

current time: hh:mm:ss[cr/lf]
enter new time: (HH:MM:SS)[cr/lf]

Values
hh:mm:ss ....Current time
hh ....hour (00 to 23)
mm ....minute (00 to 59)
ss ....second (00 to 59)

Meaning Acquires the current time in the controller and sets a new time.

Command format

aa:bb:cc[cr/lf]

Response format

OK[cr/lf] or "error message" [cr/lf]

Values
aa:bb:cc......................................Time to be set.
aa ....hour (00 to 23) *This can be omitted.
bb ....minute (00 to 59) *This can be omitted.
cc ....second (00 to 59) *This can be omitted.

MEMO

- The currently set values are used for the omitted items.
- If only [cr/lf] is transmitted, then the time remains unchanged.
- If an improbable time is entered, then "5.2: Data error" occurs.

SAMPLE 1

To change only the minute,
:20:[cr/lf] Minute is set to 20 minutes.

SAMPLE 2

Command:	@TIME[cr/lf]
Response: current time:	10:21:35[cr/lf]
	enter new time:(HH/MM/SS)[cr/lf]
Transmission:	10:25:00[cr/lf]
Response:	OK[cr/lf]

4 Data handling

4.1 Acquiring the display language

Command format
@?LANGUAGE[cr/lf]

Response format
m[cr/lf]

Values m...Display language / JAPANESE or ENGLISH

Meaning Acquires the language for displaying messages.

SAMPLE

Command	@?LANGUAGE[cr/lf]
Response	JAPANESE[cr/lf]

4.2 Acquiring the access level

Command format
@?ACCESS[cr/lf]

Response format
LEVELk[cr/lf]

Values k...Access level: 0 to 3

Meaning Acquires the access level.

REFERENCE

- For a detailed description of the access level, refer to the Controller user's manual.

SAMPLE

Command	@?ACCESS[cr/lf]
Response	LEVEL2[cr/lf]

4.3 Acquiring the arm status

Command format

@?ARM[cr/lf]

Response format

m1,m2[cr/lf]

Values Main robot

m1 .... Current arm setting status / RIGHTY: right-handed system, LEFTY: left-handed system

m2 .... Arm setting status at program reset / RIGHTY: right-handed system, LEFTY: left-handed system

Meaning Acquires the arm setting status.

SAMPLE

Command	@?ARM[cr/lf]
Response	RIGHTY,RIGHTY[cr/lf]

4.4 Acquiring the break point status

Command format

@?BREAK[cr/lf]

Response format

k1,k2,k3,k4[cr/lf]

Values kn .... Line number on which break point "n" is set: 1 to 9999

Meaning Acquires the break point status.
When kn is 0, this means no break point is set.
When a break point is set in the COMMON program, the line number shows +10000.

SAMPLE

Command	@?BREAK[cr/lf]
Response	12,35,0,0[cr/lf]

4.5 Acquiring the controller configuration status

Command format

@?CONFIG[cr/lf]

Response format

mr-ma-r-o1-o2[cr/lf]

Values

mr ....Main robot name
ma ....Main group axis setting (Auxiliary axes are shown separated by "+".)
r..Standard interface unit
o1 ....Option unit
o2..Other setting

Meaning Acquires the controller configuration status.

SAMPLE

Command	@?CONFIG[cr/lf]
Response	R6YXH250-XYZR-SRAM/196kB,DIO_N-DIO_N(1/2)[cr/lf]

4.6 Acquiring the execution level

Command format

@?EXELVL[cr/lf]

Response format

LEVELk[cr/lf]

Values

k ....Execution level: 0 to 8

REFERENCE

• For a detailed description of the execution level, refer to the Controller user's manual.

Meaning Acquires the execution level.

SAMPLE

Command	@?EXELVL[cr/lf]
Response	LEVEL2[cr/lf]

4.7 Acquiring the mode status

Command format

@?MOD[cr/lf]

Response format

s[cr/lf]

Values s ....Mode status

s		Meaning
English	Japanese	
AUTO	ジドウ	AUTO mode
PROGRAM	プログラム	PROGRAM mode
MANUAL	シュドウ	MANUAL mode
SYSTEM	システム	SYSTEM mode

Meaning Acquires the controller mode status.

SAMPLE

Command @?MOD[cr/lf]
Response AUTO[cr/lf]

4.8 Acquiring the message

Command format 1
@?MSG[c/r]

Response format 1
gg,bb: msg[c/r]or OK[c/r]

Command format 2
@?MSG m,n[cr/lf]

Response format 2

yy/mm/dd,hh:mm:ss gg.bb:msg[cr/lf]
yy/mm/dd,hh:mm:ss gg.bb:msg[cr/lf]
:
OK[cr/lf]

Values

gg..Error group
bb..Error category
msg ....Error message
m...Top number to be acquired: 1 to 500
n..Last number to be acquired: 1 to 500
yy/mm/dd ....Date (year/month/day) when error occurred
hh:mm:ss ....Time (hour:minute:second) when error occurred

Meaning Command format 1 acquires information on the message line displayed on the programming box.
Command format 2 acquires error history message.

SAMPLE 1

Command	@?MSG[cr/lf]
Response	5.30: Undefined identifier[cr/lf] or OK[cr/lf]

SAMPLE 2

Command @?MSG 1,5[cr/lf]
Response 01/10/28,14:20:20 5.30: Undefined identifier[cr/lf]
01/10/28,14:18:34 5.1: Syntax error[cr/lf]
01/10/28,14:10:54 5.30: Undefined identifier[cr/lf]
01/10/28,14:05:40 14.22: No start code(@)[cr/lf]
01/10/28,14:05:00 5.52: Command doesn’t exist[cr/lf]
OK[cr/lf]

4.9 Acquiring return-to-origin status

Command format

@?ORIGIN[cr/lf]

Response format

COMPLETE[cr/lf] or INCOMPLETE[cr/lf]

Meaning Acquires return-to-origin status.

Response format ....COMPLETE: Return-to-origin is complete.
INCOMPLETE: Return-to-origin is incomplete.

SAMPLE

Command	@?ORIGIN[cr/lf]
Response	COMPLETE[cr/lf]

4.10 Acquiring the absolute reset status

Command format

@?ABSRST[cr/lf]

Response format

COMPLETE[cr/lf] or INCOMPLETE, xxxxxxxx[cr/lf]

Values xxxxxxxx ....Absolute reset status of each axis (axis 8 to axis 1 from the left)

0: Incomplete
1: Complete
9: Not applicable

Meaning Acquires the absolute reset status.

Response format ....COMPLETE: Return-to-origin is complete.
INCOMPLETE: Return-to-origin is incomplete.

SAMPLE

Command	@?ABSRST[cr/lf]
Response	INCOMPLETE,99991011[cr/lf]

4.11 Acquiring the servo status

Command format

@?SERVO[cr/lf]

Response format

OFF,xxxxxxxx [cr/lf] or ON,xxxxxxxx[cr/lf]

Values

xxxxxxxx ....Status of each axis (axis 8 to axis 1 from the left)
0: Mechanical break ON + dynamic break ON
1: Servo ON
2: Mechanical break OFF + dynamic break OFF
9: Not applicable

Meaning

Acquires the servo status.
Response outputs are defined as follows:
ON ....Motor power is ON.
OFF...Motor power is OFF.

SAMPLE

Command @?SERVO[cr/lf]
Response ON,99991011[cr/lf]

4.12 Acquiring the sequence program execution status

Command format

@?SEQUENCE[cr/lf]

Response format

1. ENABLE,s[cr/lf]
2. ENABLE(RST.DO),s[cr/lf]
3. DISABLE[cr/lf]

Values

s ....The sequence program's execution status is indicated as "RUNNING" or "STOP".
RUNNING ....Program execution is in progress.
STOP ....Program execution is stopped.

Meaning

Acquires the sequence program execution status.
Response output means as follows:
ENABLE ....Enabled
ENABLE(RST.DO) ....Enabled and output is cleared at emergency stop
DISABLE ....Disabled

SAMPLE

Command @? SEQUENCE[cr/lf]
Response DISABLE[cr/lf]

4.13 Acquiring the speed setting status

Command format

@?SPEED[cr/lf]

Response format

ma,mm[cr/lf]

Values Main group

ma ....Automatic movement speed setting status: 1 to 100

mm..Manual movement speed setting status: 1 to 100

Meaning Acquires the speed setting status.

SAMPLE

Command @?SPEED[cr/lf]

Response 100,50[cr/lf]

4.14 Acquiring the point coordinates and units

Command format

@?UNIT[cr/lf]

Response format

s[cr/lf]

Values s: Coordinates and units ....PULSE: joint coordinate in "pulse" units

MM: Cartesian coordinate in "mm" or “deg.” units

Meaning Acquires the coordinates and units for point data.

SAMPLE

Command @?UNIT[cr/lf]

Response PULSE[cr/lf]

4.15 Acquiring the version information

Command format

@?VER[cr/lf]

Response format

cv,cr-mv-d1/d2/d3/d4/d5/d6/d7/d8{-ov}[cr/lf]

Values

cv ....Host version number (YRC, RCX)
cr...Host revision number (Rxxxx)
mv..Programming box version number (Vx.xx)
d?(?:1-8) ....Driver version number (Vx.xx)
ov...Option unit version number (Vx.xx)

Meaning Acquires the version information.

SAMPLE

Command @?VER[cr/lf]
Response V8.02,R1021-V5.10-V1.01/V1.01/V1.01/V1.01/-----/-----/-----/-----/[cr/lf]

4.16 Acquiring the current positions

1. Acquiring the current positions on pulse unit coordinates

Command format

@?WHERE[cr/lf]

Response format

[POS]xxxxxx yyyyyy zzzzzz rrrrrr aaaaaa bbbbbb[cr/lf]

Values

xxxxxx......................................Current position of axis 1 in "pulse" units
yyyyyy......................................Current position of axis 2 in "pulse" units
:
bbbbbb......................................Current position of axis 6 in "pulse" units

Meaning Acquires the current positions.
WHERE: Acquires the current positions of main group axes.

SAMPLE

Command @?WHERE[cr/lf]
Response [POS] 1000 2000 3000 -40000 0 0[cr/lf]

2. Acquiring the current positions on XY coordinates

Command format

@?WHRXY[cr/lf]

Response format

[POS]xxxxxx yyyyyy zzzzzz rrrrrr aaaaaa bbbbbb[cr/lf]

Values

xxxxxx.......................................Current position of axis 1 in "mm" or "deg" units
yyyyyy.......................................Current position of axis 2 in "mm" or "deg" units
:
bbbbbb.......................................Current position of axis 6 in "mm" or "deg" units

Meaning Acquires the current positions.
WHRXY: Acquires the current positions of main group axes.

SAMPLE

Command @?WHRXY[cr/lf]
Response [POS] 100.00 200.00 300.00 -40.00 0.00 0.00[cr/lf]

3. XY coordinate system current position (including extended setting) acquisition

NOTE

- The "XY coordinate system current position (including extended setting) acquisition" function is only available in software version 1.66M or higher.

Command format

@?WHRXYEX[cr/lf]

Response format

[POS]xxxxxx yyyyyy zzzzzz rrrrrr aaaaaa bbbbbb n xr yr[cr/lf]

Values

xxxxxx...................... Axis 1 current position in "mm" units.
yyyyyy...................... Axis 2 current position in "mm" units.
:
bbbbbb...................... Axis 6 current position in "mm" units.
n................................ SCARA robot extended hand system flag (*1)
1: Right-handed system; 2: Left-handed system
xr............................... Extended setting's X-arm rotation information (*2).
0: The "mm → pulse" converted angle data x (*3) range is –180.00° < x < = 180.00°.
1: The "mm → pulse" converted angle data x (*3) range is 180.00° < x < = 540.00°.
-1: The "mm → pulse" converted angle data x (*3) range is –540.00° < x < = -180.00°.
yr............................... Extended setting's Y-arm rotation information (*2).
0: The "mm → pulse" converted angle data y (*3) range is –180.00° < y < = 180.00°.
1: The "mm → pulse" converted angle data y (*3) range is 180.00° < y < = 540.00°.
-1: The "mm → pulse" converted angle data y (*3) range is –540.00° < y < = -180.00°.

*1: The hand system flag is "0" on all robots other than the SCARA robot.

*2: The arm rotation information is "0" on all robots other than the R6YXTW500 robot.

*3: The joint-coordinates-converted pulse data represents each arm's distance (converted to angular data) from its mechanical origin point.

Meaning • The acquired current position data includes additional dedicated R6YXTW500 information. WHRXYEX: Acquires the current position of a main group axis.

NOTE

- The "XY coordinate system current position (including extended setting) acquisition" function is only available in software version 1.66M or higher.

SAMPLE

Command @?WHRXYEX
Response [POS] 13.44 206.06 0.00 83.24 0.00 0.00 1 0 -1[cr/lf]

4.17 Acquiring the tasks in RUN or SUSPEND status

Command format

@?TASKS[cr/lf]

Response format

n{,n{,{...}}}[cr/lf]

Values n: Task number...........................1 to 8 (Task currently run or suspended)

Meaning Acquires the tasks in RUN or SUSPEND status.

SAMPLE

Command @?TASKS[cr/lf]
Response 1,3,4,6[cr/lf]

4.18 Acquiring the tasks operation status

Command format

@?TSKMON[cr/lf]

Response format

nfp,{nfp},{nfp},{nfp},{nfp},{nfp},{nfp},{nfp}[cr/lf]

Values

n : Line number being executed in each task .... 1 to 9999
f : Status of each task .... R: RUN
U: SUSPEND
S: STOP
p : Priority level of each task .... 17 to 47

Meaning Acquires the status of each task in order from Task 1 to Task 8.

SAMPLE

Command @?TSKMON[cr/lf]
Response 11R32,,43U32,,,,129R31,[cr/lf]

4.19 Acquiring the shift status

Command format

@?SHIFT[cr/lf]

Response format

m[cr/lf]

Values m...Shift number selected for main robot: 0 to 9

Meaning Acquires the shift status.

SAMPLE

Command @?SHIFT[cr/lf]
Response 1[cr/lf]

4.20 Acquiring the hand status

Command format

@?HAND[cr/lf]

Response format

m[cr/lf]

Values m..Hand number selected for main robot: 0 to 3

Meaning Acquires the hand status.

SAMPLE

Command @?HAND[cr/lf]
Response 1[cr/lf]

4.21 Acquiring the remaining memory capacity

Command format

@?MEM[cr/lf]

Response format

k/m[cr/lf]

Values k..Remaining source area (unit: bytes)
m..Remaining object area (unit: bytes)

Meaning Acquires the remaining memory capacity.

SAMPLE

Command @?MEM[cr/lf]
Response 102543/1342[cr/lf]

4.22 Acquiring the emergency stop status

Command format
@?EMG[cr/lf]

Response format
k[cr/lf]

Values k...Emergency stop status / 0: normal operation, 1: emergency stop

Meaning Acquires the emergency stop status by checking the internal emergency stop flag.

SAMPLE
Command @?EMG[cr/lf] Response 1[cr/lf]

4.23 Acquiring the error status by self-diagnosis

Command format
@?SELFCHK[cr/lf]

- When no error was found

Response format
OK[cr/lf]

- If an error occurred

Response format
m.n: "message" [cr/lf] : END [cr/lf]

Values m...Error group
n...Error category
"message" ....Show error message.

Meaning Acquires the error status by self-diagnosis that checks for errors inside the controller.

SAMPLE
Command @?SELFCHK[cr/lf] Response 12.1: Emg.stop on[cr/lf] END[cr/lf]

8 9 10 11 12 13 14

4.24 Acquiring the option slot status

Command format

@?OPSLOT[cr/lf]

Response format

OP.1 : <option board name> [cr/lf]
OP.2 : <option board name> [cr/lf]
OP.3 : <option board name> [cr/lf]
OP.4 : <option board name> [cr/lf]

Values

<option board name> ....Name of option board installed in the controller.
DIO_Nm....................................DIO board with NPN specifications (m: board ID)
DIO_Pm .... DIO board with PNP specifications (m: board ID)
CCLnk .... CC-Link compatible board
D_Net .... DeviceNet compatible board
Profi .... Profibus compatible board
E_Net.. Ethernet compatible board
no board...................................... No option board is installed.
illegal board................................Incompatible board is installed.

Meaning Acquires the option slot status by checking the option boards.

SAMPLE

Command: @?OPSLOT[cr/lf]
Response: OP.1: DIO_N2[cr/lf]
OP.2: DIO_N1[cr/lf]
OP.3: no board[cr/lf]
OP.4: CCLnk [cr/lf]

4.25 Acquiring various values

1. Acquiring the value of a numerical expression

Command format
@? "numerical expression" [cr/lf]

Response format
"numerical value" [cr/lf]

Meaning Acquires the value of the specified numerical expression.
The numerical expression's value format is "decimal" or "real number".

SAMPLE 1

Command:	@?SQR(100*5)[cr/lf]
Response:	2.23606E01[c/lf]

SAMPLE 2

Command:	@?LOCX(WHERE)[cr/lf]
Response:	102054[cr/lf]

2. Acquiring the value of a character string expression

Command format
@? "character string expression" [cr/lf]

Response format
"character string " [cr/lf]

Meaning Acquires the value (character string) of the specified character string expression.

SAMPLE

If A$="ABC" and B$="DEF"

Command:	@?A$+B$+"123"[cr/lf]
Response:	ABCDEF123[cr/lf]

8 9 10 11 12 13 14

8
9
10
11
12
13
14

3. Acquiring the value of a point expression

Command format
@? "point data expression" [cr/lf]

Response format
"point data" [cr/lf]

Meaning Acquires the value (point data) of the specified point expression.

SAMPLE	
Command:	@?P1+WHRXY[cr/lf]
Response:	10.41 -1.60 52.15 3.00 0.00 0.00 0[cr/lf]

4. Acquiring the value of a shift expression

Command format
@? "shift expression" [cr/lf]

Response format
"shift data" [cr/lf]

Meaning Acquires the value (shift data) of the specified shift expression.

SAMPLE	
Command:	@?s1[cr/lf]
Response:	25.00 12.60 10.00 0.00[cr/lf]

4.26 Data readout processing

Command format

@READ <readout file> [cr/lf]

Response format

Response output depends on the designated readout file.

NOTE

- For more information about files, refer to the earlier Chapter 11 "Data file description".

Values <readout file> .... Designate a readout file name.

Meaning Reads out the data from the designated file.

Online commands that are input through the RS-232C port have the same meaning as the following command.

- SEND <readout file> TO CMU

Commands via Ethernet have the same meaning as the following command.

- SEND <readout file> TO ETH

Type	Readout file name	Definition format	
		All	Separate file
User memory	All files	ALL	————
	Program	PGM	<bbbbbbbb>
	Point data	PNT	Pn
	Point comment	PCM	PCn
	Parameter	PRM	/cccccc/
	Shift definition	SFT	Sn
	Hand definition	HND	Hn
	Pallet definition	PLT	PLn
Variable, constant	Variable	VAR	ab...by
	Array variable	ARY	ab...by(x)
	Constant		"cc...c"
Status	Program directory	DIR	<<bbbbbbbb>>
	Parameter directory	DPM	————
	Machine reference	MRF	————
	Error history (log)	LOG	————
	Memory size	MEM	————
Device	DI port	DI()	DIn()
	DO port	DO()	DOn()
	MO port	MO()	MOn()
	TO port	TO()	TOn()
	LO port	LO()	LOn()
	SI port	SI()	SIn()
	SO port	SO()	SOn()
	SIW port	SIW()	SIWn()
	SOW port	SOW()	SOWn()
Others	File end code	EOF	————

a: Alphabetic character n: Number

b: Alphanumeric character or underscore (_) x: Expression (Array argument)

c: Alphanumeric character or symbol y: variable type

SAMPLE

Command: @READ PGM[cr/lf]
 @READ P100[cr/lf]

4.27 Data write processing

Command format

@WRITE <write file> [cr/lf]

Response format

Input request display ▶ ***Input the data! [cr/lf]
After input is completed ▶ OK [cr/lf]

NOTE

- For more information about files, refer to the earlier Chapter 10 "Data file description".

Values <write file> .... Designate a write file name.

Meaning Writes the data in the designated file.

Online commands that are input through the RS-232C port have the same meaning as the following command.

• SEND CMU TO <write file>

Commands via Ethernet have the same meaning as the following command.

• SEND ETH TO <write file>

MEMO

- At the DO, MO, TO, LO, SO, SOW ports, an entire port (DO(), MO(), etc.) cannot be designated as a WRITE file.
- Some separate files (DOn(), MOn(), etc.) cannot be designated as a WRITE file. For details, see Chapter 11 "Data File Details".

Type	Write file name	Definition format	
		All	Separate file
User memory	All files	ALL	———
	Program	PGM	<bbbbbbbb>
	Point data	PNT	Pn
	Point comment	PCM	PCn
	Parameter	PRM	/cccccc/
	Shift definition	SFT	Sn
	Hand definition	HND	Hn
	Pallet definition	PLT	PLn
Variable, constant	Variable	VAR	ab...by
	Array variable	ARY	ab...by(x)
Device	DO port	———	DOn()
	MO port	———	MOn()
	TO port	———	TOn()
	LO port	———	LOn()
	SO port	———	SOn()
	SOW port	———	SOWn()

a: Alphabetic character
b: Alphanumeric character or underscore (_)
c: Alphanumeric character or symbol
n: Number
x: Expression (Array argument)
y: variable type

SAMPLE 1

Command: @WRITE PRM[cr/lf]
@WRITE P100[cr/lf]

4.28 Current torque value acquisition

Command format

@?CHKTRQ k[cr/lf]

Values k ... Axis setting (k = 1 to 6).

Response format

n[cr/lf]

Values n..Current torque value (n = -100 to 100) of the specified axis.
* The plus/minus sign indicates the direction.

Meaning • Acquires the current torque value of the specified axis.
CHKTRQ: Acquires the current torque value of a main group axis.

NOTE

• If the specified axis has been set to "no axis" in the system generation, or if that axis uses the YC-Link or a power gripper, a "5.37: Specification mismatch" error message displays and command execution is stopped.

SAMPLE

Command:
Response:

5 Executing the robot language independently

5.1 Switching the program

Command format

@SWI <program name>[cr/lf]

Response format

OK[cr/lf] or LINEx,m,n: "message"

Values

OK ....Program was switched correctly.
LINEx,m,n : "message"..............Error occurred during compiling.
x ....Line number on which error occurred
m...Error group
n..Error category
"message" ....Error message

Meaning

Switches the program.
• In AUTO mode, the program that is switched to will be compiled.
• In other modes, the program is only switched.
However, when "SEQUENCE" program is designated, a sequence object is created.

SAMPLE 1

In AUTO mode:
Command @SWI <TEST1>[cr/lf]
Response Line2,5.39:Illegal identifier[cr/lf]

SAMPLE 2

In other modes:
Command @SWI <TEST1>[cr/lf]
Response OK[cr/lf]

5.2 Other robot language command processing

Command format

@"robot language"[cr/lf]

Response format

OK[cr/lf] or ***Aborted

Values

OK ....Command ended correctly.
***Aborted.................................An error occurred.

Meaning

Robot language commands can be executed.
- Independently executable commands can only be executed.
- Command format depends on each command to be executed.

SAMPLE 1

Command @SET DO(20) [cr/lf]
Response OK[cr/lf]

SAMPLE 2

Command @MOVE P,P100,S=20[cr/lf]
Response OK[cr/lf]

6 Control codes

6.1 Interrupting the command execution

Command format

^C (=03H)

Response format

***Aborted

Meaning Interrupts execution of the current command.

SAMPLE

Command:	@MOVE P,P100,S=20[cr/lf]
	^C
Response:	***Aborted[cr/lf]

Chapter 13

IO commands

1 Overview..13-1
2 IO command format..13-1
3 Sending and receiving IO commands 13-2
4 IO command list...13-4
5 IO command description..................................13-5

1 Overview

Using bit information (DI/DO port) for general-purpose input/output allows issuing commands directly from the PLC. It is now possible to execute commands such as the MOVE command that were impossible to execute up until now without using the robot program or RS-232C port.

2 IO command format

When using the IO command, the following functions are assigned to each IO.

Output: Controller → PLC

	Output port	Contents
YRC	DO26	Execution check output
	DO27	Execution in-progress output

Input: PLC → Controller

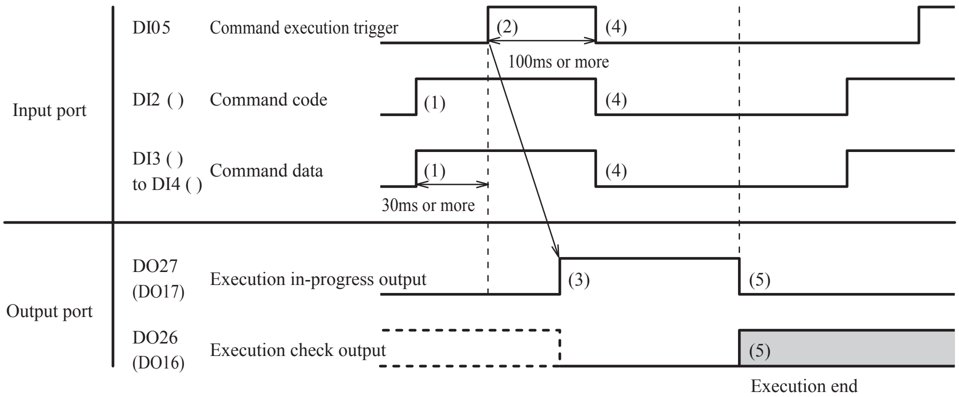
Input port	Contents
DI05	IO command execution trigger input
DI2 ( )	Command code
DI3 ( )	Command data
DI4 ( )	

- IO commands can be executed by using part of the general-purpose input and output. When no connection is made to the option DIO, then DI4() is always recognized as being OFF.
- IO commands cannot be executed while program execution is in progress (DO13 is ON).
- IO commands cannot be executed simultaneously with online commands.
- IO commands **assign command codes to be executed to DI2(), and command data to DI3() and DI4(). These are executed when the DI05 is changed from OFF to ON.** The controller processes the IO commands when they are received and sends execution check results and execution in-progress information to the PLC via DO26 and DO27.
- Command data added to the IO commands will differ according to the IO command. For details, refer to Chapter 14 "5 IO command description".
- Data is set in binary code. If the data size is greater than 8 bits, set the upper bit data into the higher address. (little endian)
  • For example, to set 0x0F9F [hexadecimal] (=3999) in the DI3 () and DI4 () ports, set 0x0F [hexadecimal] in DI4 () and set 0x9F [hexadecimal] in DI3 ().
- The IO command execution trigger input is not accepted when the execution in-progress output is ON.
- The execution in-progress output is ON in the following cases.
  • When an IO command is running after receiving an IO command execution trigger input.
  • When an IO command is terminated after receiving an IO command execution trigger input yet a maximum of 100ms state is maintained when IO command trigger input is ON.
- **The IO command trigger input pulse must always be maintained for 100ms or more during input.** Commands cannot be accepted unless this statement is maintained.
- Sometimes 20ms or more is needed for the execution in-progress output to turn ON after startup (rising edge) of the IO command trigger input pulse. The IO command trigger input might not be accepted during this period.
- After inputting the IO command trigger input pulse and the in-progress output turns OFF, at least a 100ms time period must always elapse before executing the next command. If this elapsed time period is too small, the IO command execution trigger input might not be accepted.
- The execution check output turns OFF when an IO command is received.
- The execution check output turns ON when an IO command ends normally, but it remains OFF if the IO command ends abnormally.

3 Sending and receiving IO commands

Sending and receiving is performed in the IO register as shown below.

Sending and receiving IO commands



(1) Set command code and command data (Time interval between (1) and (2): 30ms or more)
(2) Set IO command execution trigger input (Pulse width: 100ms or more)
(3) Transition to execute
(4) Clear the IO command execution trigger input and command code and command data
(5) Set termination of IO command and execution check output

Example: Follow these steps when sending and receiving IO commands to execute the PTP movement command to point 19.

(1) Set the following values in the register to execute the PTP movement command by designating a point.
- DI2( ): Command code (0x01)
- DI3( ): Lower point setting (0x13= point 19)
- DI4( ): Upper point setting (0x00=point 19)

(2) Set DI05 from ON to OFF.

(3) The controller receives the IO command and executes it if the command and command data are acceptable. The execution in-progress output turns ON and the execution check output turns OFF at this time. The robot moves to the position specified by point 19.

(4) Clear DI2( ) through DI4( ) after checking that execution in-progress output is ON.

(5) When the command ends, the output being executed turns OFF, and if execution ended normally, the execution judgment output turns ON. If execution ended in error, the execution judgment output remains OFF.

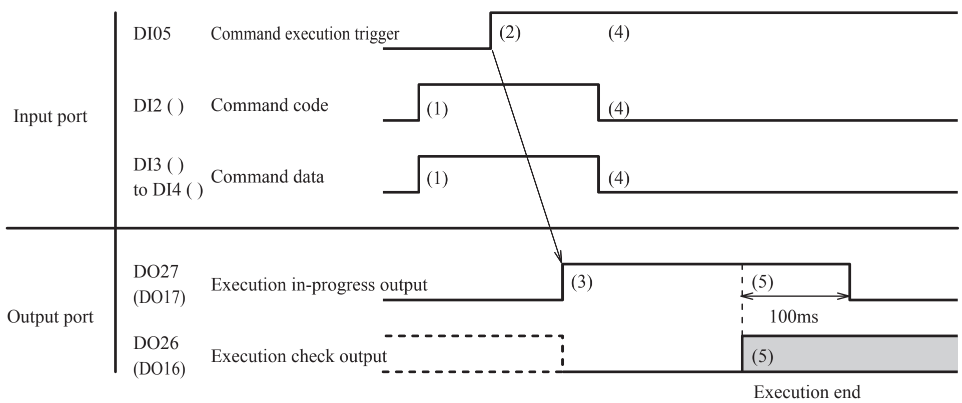
MEMO

- If DI05 was not set to OFF in (4), the execution in-progress output remains ON for a maximum of 100ms from the timing in (5).

NOTE

- When a SCARA robot is used and a hand system flag is set for the point data you specify, that hand system has priority over the current arm type.

Sending and receiving IO commands (2)



4 IO command list

IO commands are expressed with hexadecimal codes.

No.	Command contents		Command code (DI2( )) Main robot
1	MOVE command	PTP designation	0x01
		Linear interpolation	0x03
2	MOVEI command	PTP designation	0x09
3	Pallet movement command	PTP designation at pallet 0	0x18
4	Jog movement command		0x20
5	Inching movement command		0x24
6	Point teaching command		0x28
7	Absolute reset movement command		0x30
8	Absolute reset command		0x31
9	Return-to-origin command		0x32
10	Servo command	On	0x34
		Off	0x35
		Free	0x36
		Power-on	0x37
11	Manual movement speed change command		0x38
12	Auto movement speed change command		0x39
13	Program speed change command		0x3A
14	Shift designation change command		0x3B
15	Hand designation change command		0x3C
16	Arm designation change command		0x3D
17	Point display unit designation command		0x3E

Remarks

*3 The pallet movement command is only valid for pallet 0.

*4 The movement methods for the JOG, INCHING, and POINT TEACHING commands differ according to the point units that were specified.

*6 The point teaching command uses different point units according to the point units that were specified.

*8 If no axis is specified, the absolute reset command is executed on all axes in either case of command code 0x31.

*9 If no axis is specified, the return-to-origin command is executed on all axes in either case of command code 0x32.

*17 The point display unit designation command is for use on the controller.

5 IO command description

5.1 MOVE command

Moves the robot by the absolute position method

1. PTP designation

This command moves the robot to a target position in PTP motion by specifying the point number.

Command

DI port	Contents		Value
DI2 ()	Command code	For main robot	0x01
DI3 ()	Point number		0xpppp
DI4 ()			

Values pppp: Point number ....Specify in 16 bits.
Specified range ....0 (=0x0000) to 9999(=0x270F)

- The point number setting range is 0 to 255 when there is no DI4().
- When a SCARA robot is used and a hand system flag is set for the point data you specify, that hand system has priority over the current arm type.

CAUTION

- When performing linear interpolation with a hand system flag specified, be sure that the same hand system is used at the current position and target position. If the same hand system is not used, an error will occur and robot movement will be disabled.
- When performing a linear interpolation, the current position's X-arm and Y-arm rotation information must be the same as the movement destination's X-arm and Y-arm rotation information. If the two are different, an error will occur and movement will be disabled.

2. Linear interpolation

This command moves the robot to a target position by linear interpolation by specifying the point number.

Command

DI port	Contents		Value
DI2 ()	Command code	For main robot	0x03
DI3 ()	Point number		0xpppp
DI4 ()			

Values pppp: Point number ....Specify in 16 bits.
Specified range ....0 (=0x0000) to 9999(=0x270F)

- The point number setting range is 0 to 255 when there is no DI4().
- When a SCARA robot is used and a hand system flag is set for the point data you specify, that hand system has priority over the current arm type.

NOTE

- X-arm and Y-arm rotation information is only available in software Ver.1.66M or higher.

5.2 MOVEI command

Moves the robot by the relative position method

1. PTP designation

This command moves the robot a specified distance in PTP motion by specifying the point number.

Command

DI port	Contents		Value
DI2 ()	Command code	For main robot	0x09
DI3 ()	Point number		0xpppp
DI4 ()			

Values ppppp..Specify the point number in 16 bits.
Specified range: 0 (=0x0000) to 9999 (=0x270F)

MEMO

- When MOVEI motion to the original target position is interrupted and then restarted, the target position for the resumed movement can be selected as the "MOVEI/DRIVEI start position" in the controller's "other parameters". For details, refer to the controller user's manual.
  1) KEEP (default setting) .... Continues the previous (before interruption) movement. The original target position remains unchanged.
  2) RESET................................ Relative movement begins anew from the current position. The new target position is different from the original one (before interruption). (Backward compatibility)
- The point number setting range is 0 to 255 when there is no DI4().
- When a SCARA robot is used and a hand system flag is set for the point data you specify, that hand system has priority over the current arm type.

NOTE

- In versions prior to those shown above, a "RESET" must be performed at the controller.

5.3 Pallet movement command

Moves the robot to a position relative to pallet 0

1. PTP designation

This command moves the robot to a target position in PTP motion by specifying the work position number.

Command

DI port	Contents		Value
DI2 ()	Command code	For main robot	0x18
DI3 ()	Work position number		0xwwww
DI4 ()			

Values wwww ....Specify the work position number in 16 bits.
Specified range: 1 (=0

MEMO

- The work position number setting range is 0 to 255 when there is no DI4().

5.4 Jog movement command

Performs robot JOG movement in the MANUAL mode.

This command is only valid in MANUAL mode.
This command is linked with the controller point display units. The robot axis moves in PTP motion when display units are in pulses, and moves by linear interpolation on Cartesian coordinates when units are in millimeters.
Jog speed is determined by the manual movement speed.
To stop the jog movement command, set the dedicated input interlock signal to OFF. After checking that jog movement has stopped, set the interlock signal back to ON.

Command

DI port	Contents			Value
DI2 ()	Command code		For main robot	0x20
DI3 ()	Axis to move and direction	bit 0	Axis 1	tt
		bit 1	Axis 2	
		bit 2	Axis 3	
		bit 3	Axis 4	
		bit 4	Axis 5	
		bit 5	Axis 6	
		bit 6	(0:Fixed)	0
		bit 7	Direction	d
DI4 ()	Not used			0x00

Values tt: Axis setting ....Bits 0 to 3 select 1 axis
d: Movement direction ....Bit 1 / 0: plus direction; 1: minus direction

5.5 Inching movement command

Performs robot INCHING movement in the MANUAL mode.

Inching movement distance is linked to the manual movement speed. The inching command can only be executed in MANUAL mode.
This command is linked with the controller's point display unit system. So when display units are in pulses, the axis moves a certain number of pulses at the manual speed setting. When display units are in millimeters, the axis moves on Cartesian coordinates by linear interpolation at the manual speed setting divided by 100.

Command

DI port	Contents			Value
DI2 ()	Command code		For main robot	0x24
DI3 ()	Axis to move and direction	bit 0	Axis 1	tt
		bit 1	Axis 2	
		bit 2	Axis 3	
		bit 3	Axis 4	
		bit 4	Axis 5	
		bit 5	Axis 6	
		bit 6	(0:Fixed)	0
		bit 7	Direction	d
DI4 ()	Not used			0x00

Values tt: Axis setting ....Bits 0 to 3 select 1 axis
d: Movement direction ....Bit 1 / 0: plus direction; 1: minus direction

8
9
10
11
12
13
14

5.6 Point teaching command

Teaches the robot's current position to a specified point No.

Point data units of this command are linked to the controller's point display unit system.

Command

DI port	Contents		Value
DI2 ()	Command code	For main robot	0x28
DI3 ()	Point number		0xpppp
DI4 ()			

Values pppp: Point number ....Specify in 16 bits.
Specified range ....0 (=0x0000) to 9999(=0x270F)

- The point number setting range is 0 to 255 when there is no DI4().

5.7 Absolute reset movement command

Moves the nearest position where an absolute reset is possible

When absolute reset of the specified axis uses the mark method, this command moves the axis to the nearest position where absolute reset can be executed.
Positions capable of absolute reset are located at every 1/4 rotation of the motor.

Command

DI port	Contents			Value
DI2 ()	Command code		For main robot	0x30
DI3 ()	Axis to move and direction	bit 0	Axis 1	tt
		bit 1	Axis 2	
		bit 2	Axis 3	
		bit 3	Axis 4	
		bit 4	Axis 5	
		bit 5	Axis 6	
		bit 6	(0:Fixed)	0
		bit 7	Direction	d
DI4 ()	Not used			0x00

Values tt: Axis setting ....Bits 0 to 3 select 1 axis
d: Movement direction ....Bit 1 / 0: plus direction; 1: minus direction

5.8 Absolute reset command

Executes absolute reset of the specified axis.

If no axis is specified, the absolute reset command is executed on all axes in either case of command code 0x31.
However, this command cannot be executed if return-to-origin is not yet complete on the axis using the mark method. In this case, perform absolute reset individually on each axis.

Command

DI port	Contents			Value
DI2 ()	Command code		For main robot	0x31
DI3 ()	Axis specification	bit 0	Axis 1	tt
		bit 1	Axis 2	
		bit 2	Axis 3	
		bit 3	Axis 4	
		bit 4	Axis 5	
		bit 5	Axis 6	
		bit 7 - bit 6	(0:Fixed)	0
DI4 ()	Not used			0x00

Values tt: Axis specification..................Bits 0 to 3 select 1 axis

5.9 Return-to-origin command

Executes an incremental axis return-to-origin

When this command is executed on an incremental mode axis, that axis returns to its origin. When executed on a semi-absolute mode axis, an absolute search is performed on that axis.
If no axis is specified (DI3() is 0), this command is executed on all axes in either case of 0x32.

Command

DI port	Contents			Value
DI2 ()	Command code		For main robot	0x32
DI3 ()	Axis specification	bit 0	Axis 1	tt
		bit 1	Axis 2	
		bit 2	Axis 3	
		bit 3	Axis 4	
		bit 4	Axis 5	
		bit 5	Axis 6	
		bit 7 - bit 6	(0:Fixed)	0
DI4 ()	Not used			0x00

Values tt: Axis specification..................Bits 0 to 3 select 1 axis

5.10 Servo command

Operates the robot servo

These commands operate the robot servos, and switch the servos between their ON, OFF, and FREE settings in an axis-specific manner.

Operation	Contents
Servo ON	Execute this command to turn on the servo of a specified axis. The motor power must be turned on when specifying the axis. All controller servos are turned on if no axis is specified.
Servo OFF	Execute this command to turn off the servo of a specified axis. All controller servos are turned off if no axis is specified.
Servo Free	Execute this command to turn off the mechanical brake and dynamic brake after turning off the servo of a specified axis. Servo OFF and Free are repeated when this command is consecutively executed. All controller servos will be free if no axis is specified.
Power ON	Execute this command to turn on the motor power. No axis can be specified.

Command

DI port	Contents			Value
DI2 ()	Command code	Servo ON	For main robot	0x34
		Servo OFF	For main robot	0x35
		Servo Free	For main robot	0x36
		Power ON	All controller servos	0x37
DI3 ()	Axis specification	bit 0	Axis 1	tt
		bit 1	Axis 2	
		bit 2	Axis 3	
		bit 3	Axis 4	
		bit 4	Axis 5	
		bit 5	Axis 6	
		bit 7 - bit 6	(0:Fixed)	0
DI4 ()	Not used			0x00

Values tt: Axis specification..................Bits 0 to 3 select 1 axis (if not specified, the command applies to all controller processing).
No axis can be specified when executing Power ON.

5.11 Manual movement speed change command

Changes the MANUAL mode's manual movement speed

This command can only be executed in MANUAL mode.

Command

DI port	Contents		Value
DI2 ()	Command code	For main robot	0x38
DI3 ()	Specified speed		0xss
DI4 ()	Not used		0x00

Values ss: Manual movement speed ......Specify in 8 bits.
Specified range ....1 (=0x01) to 100 (=0x64)

5.12 Auto movement speed change command

Changes the AUTO mode's automatic movement speed

This command can only be executed in AUTO mode.

Command

DI port	Contents		Value
DI2 ()	Command code	For main robot	0x39
DI3 ()	Specified speed		0xss
DI4 ()	Not used		0x00

Values ss: Auto movement speed...........Specify in 8 bits.
Specified range ....1 (=0x01) to 100 (=0x64)

5.13 Program speed change command

Changes the AUTO mode's program movement speed

The program speed changed with this command is reset to 100% when the program is reset or changed.

Command

DI port	Contents		Value
DI2 ()	Command code	For main robot	0x3A
DI3 ()	Specified speed		0xss
DI4 ()	Not used		0x00

Values ss: Program speed.......................Specify in 8 bits.
Specified range ....1 (=0x01) to 100 (=0x64)

5.14 Shift designation change command

Changes a selected shift to the specified shift No.

Command

DI port	Contents		Value
DI2 ()	Command code	For main robot	0x3B
DI3 ()	Specified shift number		0xss
DI4 ()	Not used		0x00

Values ss: Shift number..........................Specify in 8 bits.
Specified range ....0 (=0x00) to 9 (=0x09)

5.15 Hand designation change command

Changes a selected hand to the specified hand No.

Command

DI port	Contents		Value
DI2 ()	Command code	For main robot	0x3C
DI3 ()	Specified hand number		0xss
DI4 ()	Not used		0x00

Values ss: Hand number.........................Specify in 8 bits.
Specified range ....For main robot: 0 (=0x00) to 3 (=0x03)

5.16 Arm designation change command

Changes the arm setting status

Command

DI port	Contents		Value
DI2 ()	Command code	For main robot	0x3D
DI3 ()	Status of specified arm		0xss
DI4 ()	Not used		0x00

Values ss: Arm designation status..........Specify in 8 bits / 0x00: Right-handed system, 0x01: Left-handed system

5.17 Point display unit designation command

Changes the point display unit system

This command is for the controller.

Command

DI port	Contents		Value
DI2 ()	Command code	For controller	0x3E
DI3 ()	Display units for specified point		0xss
DI4 ()	Not used		0x00

Values ss: Point display unit ....Specify in 8 bits / 0x00: Pulse units, 0x01: Millimeter units

Chapter 14

Appendix

1	Reserved word list	15-1
2	Robot Language Lists: Command list in alphabetic order	15-3
3	Robot Language Lists: Function Specific	7
4	Functions: in alphabetic order	13
5	Functions: operation-specific	15
6	Execution Level	17

1 Reserved word list

The words shown below are reserved for robot language and cannot be used as identifiers (variables, etc.).

A	DEC	GOTO	MSGCLR
ABSADJ	DECEL	H	MSPEED
ABSRESET	DECLARE	HALT	N
ABSRPOS	DEF	HAND	NAME
ABSRST	DEFIO	HND	NEXT
ABOVE	DEFPOS	HEX	NONFLIP
ABS	DEGRAD	HOME	NOT
ABSINIT	DELAY	HOLD	O
ACC	DI	I	OFFLINE
ACCEL	DIM	IF	ON
ACCESS	DIR	IN	ONLINE
ALL	DIST	INCH	OR
AND	DO	INIT	ORD
ARCH	DPM	INPUT	ORGORD
ARM	DRIVE	INT	ORGRTN
ARMTYPE	DRIVEI	INTEGER	ORIGIN
ARY	DS	IRET	OUT
ASPEED	DSPEED	J	OUTPOS
ATN	E	JTOXY	P
ATTR	ELSE	JOG	P
AUTO	ELSEIF	L	PASS
AXWGHT	EMGRST	LANGUAGE	PADDR
B	EMG	LEFT	PATH
BELOW	END	LEFTY	PC
BIN	ENDIF	LEN	PCM
BIT	EOF	LET	PDEF
BREAK	ERA	LOCA	PLN
BYTE	ERL	LOCB	PLT
C	ERR	LOCF	PMOVE
CALL	ERROR	LOCR	PRINT
CASE	ETH	LOCX	PROGRAM
CHANGE	EXELVL	LOCY	PGM
CHGPRI	EXIT	LOCZ	PNT
CHGTSK	EXITTASK	LOG	POS
CHKTRQ	F	LOOP	PPNT
CHR	FDD	LSHIFT	PRM
CMU	FLIP	M	PTP
CMU1	FN	MANUAL	PWR
CONFIG	FOR	MCHREF	R
CONT	FREE	MEM	RADDEG
COO	FUNCTION	MID	READ
COPY	G	MIRROR	REF
COS	GASP	MOD	REN
CURTRQ	GEN	MOVE	RELESE
CUT	GO	MOVEI	REM
D	GOHOME	MRF	REMOTE
DATE	GOSUB	MSG	RESET

RESTART	STEP	WAIT
RESUME	STOP	WEIGHT
RETURN	STOPON	WEND
RIGHT	STR	WHERE
RIGHTY	SUB	WHILE
RO	SUSPEND	WHRXY
ROTATE	SWI	WHRXYEX
RSHIFT	SYS	WORD
RUN	SYSTEM	WRITE
RW	T	X
S	TAN	XOR
S	TASK	XY
SELECT	TASKS	XYTOJ
SEND	TCOUNTER	Y
SEQUENCE	TEACH	YZ
SERVO	THEN	Z
SET	TIME	ZX
SFT	TIMER	SYMBOL
SGI	TO	_SYSFLG
SGR	TOLE	
SHARED	TORQUE	
SHIFT	TRQSTS	
SI	TRQTIME	
SID	TSKMON	
SIN	U	
SIW	UNIT	
SKIP	UNTIL	
SO	V	
SOD	VAL	
SOW	VAR	
SPEED	VER	
SQR	VEL	
START	W	

Because the following names are used as system variable names, they cannot be used at the beginning of other variable names (n: numeric value).

DIn	Hn	Pn	SOn
DOn	LOn	SIn	TOn
FN	MOn	Sn	

Variable name usage examples

- Although keywords which are reserved as robot language words cannot be used as they are, **they can be used as variable names if alphanumeric characters are added to them.**

Example: "ABS" cannot be used, but "ABS1" or "ABSX" can be used.

- Keywords reserved as system variables cannot be used at the beginning of other variable names, even if alphanumeric characters are added to them.

Example: "FN" cannot be used. "FNA" and "FN123" also cannot be used.

8 9 10 11 12 13 14

No.	Command	Function	Condition	Direct	Type
A					
1	ABS	Acquires the absolute value of a specified value.	-	-	Functions
2	ABSINIT	Resets the current position of a specified main group axis.	4	○	Command Statements
3	ABSRPOS	Acquires the machine reference of the specified main group axis. (Valid only for axes where the return-to-origin method is set as "mark method".)	-	-	Functions
4	ABSRST	Executes a return-to-origin at the robot absolute motor axes.	4	○	Command Statements
5	ACCEL	Specifies/acquires the acceleration coefficient parameter of the main group.	4/-	○	Command Statements/ Functions
6	ARCH	Specifies/acquires the arch position parameter of the main group.	4/-	○	Command Statements/ Functions
7	ARMCND	Acquires the current arm status of the main robot.	-	-	Functions
8	ARMTYPE	Acquires the current "hand system" setting of the main robot.	-	-	Functions
10	ASPEED	Changes the AUTO movement speed of the main group.	4	○	Command Statements
9	ATN	Acquires the arctangent of the specified value.	-	-	Functions
11	AXWGHT	Specifies/acquires the axis tip weight parameter of the main group.	4/-	○	Command Statements/ Functions
C					
12	CALL	Executes (calls) another program.	6	×	Command Statements
13	CHANGE	Switches the main robot hand.	4	○	Command Statements
14	CHGPRI	Changes the priority ranking of a specified task.	6	×	Command Statements
15	CHR$	Acquires a character with the specified character code.	-	-	Functions
16	COS	Acquires the cosine value of a specified value.	-	-	Functions
17	CURTRQ	Acquires the current torque value of the specified main group axis.	-	×	Functions
18	CUT	Terminates a task currently being executed or temporarily stopped.	6	×	Command Statements
D					
19	DATE$	Acquires the date as a "yy/mm/dd" format character string.	-	-	Functions
20	DECEL	Specifies/acquires the deceleration rate parameter of the main group.	4/-	○	Command Statements/ Functions
23	DEGRAD	Converts a specified value to radians (↔RADDEG).	-	-	Functions
24	DELAY	Waits for the specified period (units: ms).	6	×	Command Statements
27	DIM	Declares the array variable name and the number of elements.	6	×	Command Statements
26	DIST	Acquires the distance between 2 specified points.	-	-	Functions
28	DO	Outputs a specified value to the DO port.	1	○	Command Statements
29	DRIVE	Moves a specified main group axis to an absolute position.	4	○	Command Statements
29	DRIVE	(With T-option) Executes an absolute movement command for a specified axis.	4	○	Command Statements
30	DRIVEI	Moves a specified main group axis to a relative position.	4	○	Command Statements
E					
33	ERL	Gives the line No. where an error occurred.	-	-	Functions
33	ERR	Gives the error code number of an error which has occurred.	-	-	Functions
34	EXIT FOR	Terminates the FOR to NEXT statement loop.	6	×	Command Statements
36	EXIT TASK	Terminates its own task which is in progress.	6	×	Command Statements

No.	Command	Function	Condition	Direct	Type
F					
37	FOR to NEXT	Controls repetitive operations. Executes the FOR to NEXT statement repeatedly until a specified value is reached.	6	×	Command Statements
G					
38	GOSUB to RETURN	Jumps to a subroutine with the label specified by a GOSUB statement, and executes that subroutine.	6	×	Command Statements
39	GOTO	Unconditionally jumps to the line specified by a label.	6	×	Command Statements
H					
40	HALT	Stops the program and performs a reset.	6	×	Command Statements
41	HAND	Defines the main robot hand.	4	○	Command Statements
42	HOLD	Temporarily stops the program.	6	×	Command Statements
I					
43	IF	Allows control flow to branch according to conditions.	6	×	Command Statements
44	INPUT	Assigns a value to a variable specified from the programming box.	1	○	Command Statements
45	INT	Acquires an integer for a specified value by truncating all decimal fractions.	-	-	Functions
J					
46	JTOXY	Converts joint coordinate data to main group Cartesian coordinate data. (↔XYTOJ)	-	-	Functions
L					
48	LEFT$	Extracts a character string comprising a specified number of digits from the left end of a specified character string.	-	-	Functions
49	LEFTY	Sets the main robot hand system to "Left".	4	○	Command Statements
50	LEN	Acquires the length (number of bytes) of a specified character string.	-	-	Functions
51	LET	Executes a specified assignment statement.	1	○	Command Statements
52	LO	Outputs a specified value to the LO port to enable/disable axis movement.	1	○	Command Statements
53	LOCx	Specifies/acquires point data or shift data for a specified axis.	-	-	Command Statements/ Functions
54	LSHIFT	Shifts a value to the left by the specified number of bits. (↔RSHIFT)	-	-	Functions
M					
55	MCHREF	Acquires the return-to-origin or absolute-search machine reference for a specified main group axis.	-	-	Functions
56	MID$	Extracts a character string of a desired length from a specified character string.	-	-	Functions
57	MO	Outputs a specified value to the MO port.	1	○	Command Statements
58	MOVE	Performs absolute movement of all main robot axes.	5	○	Command Statements
59	MOVEI	Performs relative movement of all main robot axes.	4	○	Command Statements
O					
60	OFFLINE	Sets a specified communication port to the "offline" mode.	1	○	Command Statements
62	ON ERROR GOTO	If an error occurs during program execution, this command allows the program to jump to the error processing routine specified by the label without stopping the program, or it stops the program and displays the error message.	6	×	Command Statements
63	ON to GOSUB	Jumps to a subroutine with labels specified by a GOSUB statement in accordance with the conditions, and executes that subroutine.	6	×	Command Statements

No.	Command	Function	Condition	Direct	Type
64	ON to GOTO	Jumps to label-specified lines in accordance with the conditions.	6	×	Command Statements
65	ONLINE	Sets the specified communication port to the "online" mode.	1	○	Command Statements
61	ORD	Acquires the character code of the first character in a specified character string.	-	-	Functions
66	ORGORD	Specifies/acquires the axis sequence parameter for performing return-to-origin and absolute search operations in the main group.	4/-	○	Command Statements/ Functions
67	ORIGIN	Executes a return-to-origin for incremental specs. axes.	4	○	Command Statements
68	OUT	Turns ON the bits of the specified output ports and the command statement ends.	6	×	Command Statements
69	OUTPOS	Specifies/acquires the OUT enable position parameter of the main group.	4/-	○	Command Statements/ Functions
P					
70	PATH	Sets the movement path.	6	×	Command Statements
71	PATH END	Ends the movement path setting.	6	×	Command Statements
72	PATH SET	Starts the movement path setting.	6	×	Command Statements
73	PATH START	Starts the PATH motion.	6	×	Command Statements
74	PDEF	Defines the pallet used to execute pallet movement commands.	1	○	Command Statements
75	PMOVE	Executes the main robot pallet movement command.	4	○	Command Statements
76	Pn	Defines points within a program.	1	○	Command Statements
77	PPNT	Creates point data specified by a pallet definition number and pallet position number.	-	-	Functions
78	PRINT	Displays a character string at the programming box screen.	1	○	Command Statements
R					
79	RADDEG	Converts a specified value to degrees. (↔DEGRAD)	-	-	Functions
80	REM	Expresses a comment statement.	6	×	Command Statements
81	RESET	Turns the bit of a specified output port OFF.	1	○	Command Statements
82	RESTART	Restarts another task during a temporary stop.	6	×	Command Statements
83	RESUME	Resumes program execution after error recovery processing.	6	×	Command Statements
85	RIGHT$	Extracts a character string comprising a specified number of digits from the right end of a specified character string.	-	-	Functions
86	RIGHTY	Sets the main robot hand system to "Right".	4	○	Command Statements
87	RSHIFT	Shifts a value to the right by the specified number of bits. (↔LSHIFT)	-	-	Functions
S					
88	Sn	Defines the shift coordinates within the program.	4	○	Command Statements
89	SELECT CASE to END SELECT	Allows control flow to branch according to conditions.	6	×	Command Statements
90	SEND	Sends a file.	1	○	Command Statements
91	SERVO	Controls the servo ON/OFF of specified main group axes or all main group axes.	4	○	Command Statements
92	SET	Turns the bit at the specified output port ON.	3	In part ×	Command Statements
94	SHIFT	Sets the shift coordinates for the main robot by using the shift data specified by a shift variable.	4	○	Command Statements
95	SIN	Acquires the sine value for a specified value.	-	-	Functions
96	SO	Outputs a specified value to the SO port.	1	○	Command Statements
97	SPEED	Changes the main group's program movement speed.	4	○	Command Statements

No.	Command	Function	Condition	Direct	Type
98	START	Specifies the task number and priority ranking of a specified task, and starts that task.	6	×	Command Statements
99	STR$	Converts a specified value to a character string (↔VAL)	-	-	Functions
100	SQR	Acquires the square root of a specified value.	-	-	Functions
102	SUSPEND	Temporarily stops another task which is being executed.	6	×	Command Statements
103	SWI	Switches the program being executed, performs compiling, then begins execution from the first line.	2	○	Command Statements
T					
104	TAN	Acquires the tangent value for a specified value.	-	-	Functions
105	TCOUNTER	Outputs count-up values at 10ms intervals starting from the point when the TCOUNTER variable is reset.	-	-	Functions
106	TIME$	Acquires the current time as an "hh:mm:ss" format character string.	-	-	Functions
107	TIMER	Acquires the current time in seconds, counting from 12:00 midnight.	-	-	Functions
108	TO	Outputs a specified value to the TO port.	1	○	Command Statements
109	TOLE	Specifies/acquires the main group tolerance parameter.	4/-	○	Command Statements/ Functions
110	TORQUE	Specifies/acquires the maximum torque command value which can be set for a specified main group axis.	4/-	○	Command Statements/ Functions
111	TRQSTS	Acquires the command end status for the DRIVE command with torque limit option executed at the main group.	-	-	Functions
112	TRQTIME	Specifies/acquires the current limit time-out period at the specified main group axis when using a torque limit option in the DRIVE statement.	1/-	○	Command Statements/ Functions
V					
113	VAL	Converts the numeric value of a specified character string to an actual numeric value. (↔STR$)	-	-	Functions
W					
114	WAIT	Waits until the conditions of the DI/DO conditional expression are met (with time-out).	6	×	Command Statements
115	WAIT ARM	Waits until the main group robot axis operation is completed.	6	×	Command Statements
116	WEIGHT	Specifies/acquires the main robot tip weight parameter.	4/-	○	Command Statements/ Functions
118	WHERE	Reads out the current position of the main group robot arm in joint coordinates (pulses).	-	-	Functions
119	WHILE to WEND	Controls repeated operations.	6	×	Command Statements
120	WHRXY	Reads out the current position of the main group arm as Cartesian coordinates (mm, degrees).	-	-	Functions
X					
121	XYTOJ	Converts the point variable Cartesian coordinate data to the main group's joint coordinate data (↔JTOXY).	-	-	Functions
122	_SYSFLG	Axis status monitoring flag.	-	-	Functions

3 Robot Language Lists: Function Specific

Program commands

General commands

No.	Command	Function	Condition	Direct	Type
27	DIM	Declares the array variable name and the number of elements.	6	×	Command Statements
51	LET	Executes a specified assignment statement.	1	○	Command Statements
80	REM	Expresses a comment statement.	6	×	Command Statements

Arithmetic commands

No.	Command	Function	Condition	Direct	Type
1	ABS	Acquires the absolute value of a specified value.	-	-	Functions
2	ABSINIT	Resets the current position of a specified main group axis.	4	○	Command Statements
9	ATN	Acquires the arctangent of the specified value.	-	-	Functions
16	COS	Acquires the cosine value of a specified value.	-	-	Functions
23	DEGRAD	Converts a specified value to radians (↔RADDEG).	-	-	Functions
26	DIST	Acquires the distance between 2 specified points.	-	-	Functions
45	INT	Acquires an integer for a specified value by truncating all decimal fractions.	-	-	Functions
54	LSHIFT	Shifts a value to the left by the specified number of bits. (↔RSHIFT)	-	-	Functions
79	RADDEG	Converts a specified value to degrees. (↔DEGRAD)	-	-	Functions
87	RSHIFT	Shifts a value to the right by the specified number of bits. (↔LSHIFT)	-	-	Functions
95	SIN	Acquires the sine value for a specified value.	-	-	Functions
100	SQR	Acquires the square root of a specified value.	-	-	Functions
104	TAN	Acquires the tangent value for a specified value.	-	-	Functions

Date / time

No.	Command	Function	Condition	Direct	Type
19	DATE $	Acquires the date as a "yy/mm/dd" format character string.	-	-	Functions
105	TCOUNTER	Outputs count-up values at 10ms intervals starting from the point when the TCOUNTER variable is reset.	-	-	Functions
106	TIME $	Acquires the current time as an "hh:mm:ss" format character string.	-	-	Functions
107	TIMER	Acquires the current time in seconds, counting from 12:00 midnight.	-	-	Functions

8

9

10

11

12

13

14

Character string operation

No.	Command	Function	Condition	Direct	Type
15	CHR $	Acquires a character with the specified character code.	-	-	Functions
48	LEFT $	Extracts a character string comprising a specified number of digits from the left end of a specified character string.	-	-	Functions
50	LEN	Acquires the length (number of bytes) of a specified character string.	-	-	Functions
56	MID $	Extracts a character string of a desired length from a specified character string.	-	-	Functions
61	ORD	Acquires the character code of the first character in a specified character string.	-	-	Functions
85	RIGHT $	Extracts a character string comprising a specified number of digits from the right end of a specified character string.	-	-	Functions
99	STR $	Converts a specified value to a character string (↔VAL)	-	-	Functions
113	VAL	Converts the numeric value of a specified character string to an actual numeric value. (↔STR$)	-	-	Functions

Point, coordinates, shift coordinates

No.	Command	Function	Condition	Direct	Type
13	CHANGE	Switches the main robot hand.	4	○	Command Statements
41	HAND	Defines the main robot hand.	4	○	Command Statements
46	JTOXY	Converts joint coordinate data to main group Cartesian coordinate data. (↔XYTOJ)	-	-	Functions
49	LEFTY	Sets the main robot hand system to "Left".	4	○	Command Statements
76	Pn	Defines points within a program.	1	○	Command Statements
77	PPNT	Creates point data specified by a pallet definition number and pallet position number.	-	-	Functions
86	RIGHTY	Sets the main robot hand system to "Right".	4	○	Command Statements
88	Sn	Defines the shift coordinates in the program.	4	○	Command Statements
94	SHIFT	Sets the shift coordinates for the main robot by using the shift data specified by a shift variable.	4	○	Command Statements
121	XYTOJ	Converts the point variable Cartesian coordinate data to the main group's joint coordinate data (↔JTOXY).	-	-	Functions
53	LOCx	Specifies/acquires point data or shift data for a specified axis.	-	-	Command Statements/ Functions

Branching commands

No.	Command	Function	Condition	Direct	Type
34	EXIT FOR	Terminates the FOR to NEXT statement loop.	6	×	Command Statements
37	FOR to NEXT	Controls repetitive operations. Executes the FOR to NEXT statement repeatedly until a specified value is reached.	6	×	Command Statements
38	GOSUB to RETURN	Jumps to a subroutine with the label specified by a GOSUB statement, and executes that subroutine.	6	×	Command Statements
39	GOTO	Unconditionally jumps to the line specified by a label.	6	×	Command Statements
43	IF	Allows control flow to branch according to conditions.	6	×	Command Statements
63	ON to GOSUB	Jumps to a subroutine with labels specified by a GOSUB statement in accordance with the conditions, and executes that subroutine.	6	×	Command Statements
64	ON to GOTO	Jumps to label-specified lines in accordance with the conditions.	6	×	Command Statements
89	SELECT CASE to END SELECT	Allows control flow to branch according to conditions.	6	×	Command Statements
119	WHILE to WEND	Controls repeated operations.	6	×	Command Statements

Error control

No.	Command	Function	Condition	Direct	Type
62	ON ERROR GOTO	If an error occurs during program execution, this command allows the program to jump to the error processing routine specified by the label without stopping the program, or it stops the program and displays the error message.	6	×	Command Statements
83	RESUME	Resumes program execution after error recovery processing.	6	×	Command Statements
33	ERL	Gives the line No. where an error occurred.	-	-	Functions
33	ERR	Gives the error code number of an error which has occurred.	-	-	Functions

Program & task control

Program control

No.	Command	Function	Condition	Direct	Type
12	CALL	Executes (calls) another program.	6	×	Command Statements
40	HALT	Stops the program and performs a reset.	6	×	Command Statements
42	HOLD	Temporarily stops the program.	6	×	Command Statements
103	SWI	Switches the program being executed, performs compiling, then begins execution from the first line.	2	○	Command Statements

Task control

No.	Command	Function	Condition	Direct	Type
14	CHGPRI	Changes the priority ranking of a specified task.	6	×	Command Statements
18	CUT	Terminates a task currently being executed or temporarily stopped.	6	×	Command Statements
36	EXIT TASK	Terminates its own task which is in progress.	6	×	Command Statements
82	RESTART	Restarts another task during a temporary stop.	6	×	Command Statements
98	START	Specifies the task number and priority ranking of a specified task, and starts that task.	6	×	Command Statements
102	SUSPEND	Temporarily stops another task which is being executed.	6	×	Command Statements

Robot control

Robot operations

No.	Command	Function	Condition	Direct	Type
4	ABSRST	Executes a return-to-origin at the robot absolute motor axes.	4	○	Command Statements
13	CHANGE	Switches the main robot hand.	4	○	Command Statements
29	DRIVE	Moves a specified main group axis to an absolute position.	4	○	Command Statements
30	DRIVEI	Moves a specified main group axis to a relative position.	4	○	Command Statements
41	HAND	Defines the main robot hand.	4	○	Command Statements
49	LEFTY	Sets the main robot hand system to "Left".	4	○	Command Statements
58	MOVE	Performs absolute movement of all main robot axes.	5	○	Command Statements
59	MOVEI	Performs relative movement of all main robot axes.	4	○	Command Statements
67	ORIGIN	Executes a return-to-origin for incremental specs. axes.	4	○	Command Statements
75	PMOVE	Executes the main robot pallet movement command.	4	○	Command Statements
86	RIGHTY	Sets the main robot hand system to "Right".	4	○	Command Statements
91	SERVO	Controls the servo ON/OFF of specified main group axes or all main group axes.	4	○	Command Statements

Status acquisition

No.	Command	Function	Condition	Direct	Type
3	ABSRPOS	Acquires the machine reference of the specified main group axis. (Valid only for axes where the return-to-origin method is set as "mark method".)	-	-	Functions
7	ARMCND	Acquires the current arm status of the main robot.	-	-	Functions
8	ARMTYPE	Acquires the current "hand system" setting of the main robot.	-	-	Functions
55	MCHREF	Acquires the return-to-origin or absolute-search machine reference for a specified main group axis.	-	-	Functions
111	TRQSTS	Acquires the command end status for the DRIVE command with torque limit option executed at the main group.	-	-	Functions
118	WHERE	Reads out the current position of the main group robot arm in joint coordinates (pulses).	-	-	Functions
120	WHRXY	Reads out the current position of the main group arm as Cartesian coordinates (mm, degrees).	-	-	Functions
115	WAIT ARM	Waits until the main group robot axis operation is completed.			

Status change

No.	Command	Function	Condition	Direct	Type
5	ACCEL	Specifies/acquires the acceleration coefficient parameter of the main group.	4/-	○	Command Statements/ Functions
6	ARCH	Specifies/acquires the arch position parameter of the main group.	4/-	○	Command Statements/ Functions
10	ASPEED	Changes the AUTO movement speed of the main group.	4	○	Command Statements
11	AXWGHT	Specifies/acquires the axis tip weight parameter of the main group.	4/-	○	Command Statements/ Functions
20	DECEL	Specifies/acquires the deceleration rate parameter of the main group.	4/-	○	Command Statements/ Functions
66	ORGORD	Specifies/acquires the axis sequence parameter for performing return-to-origin and absolute search operations in the main group.	4/-	○	Command Statements/ Functions
69	OUTPOS	Specifies/acquires the OUT enable position parameter of the main group.	4/-	○	Command Statements/ Functions
74	PDEF	Defines the pallet used to execute pallet movement commands.	1	○	Command Statements
97	SPEED	Changes the main group's program movement speed.	4	○	Command Statements
109	TOLE	Specifies/acquires the main group tolerance parameter.	4/-	○	Command Statements/ Functions
116	WEIGHT	Specifies/acquires the main robot tip weight parameter.	4/-	○	Command Statements/ Functions

Path control

No.	Command	Function	Condition	Direct	Type
70	PATH	Sets the movement path.	6	×	Command Statements
71	PATH END	Ends the movement path setting.	6	×	Command Statements
72	PATH SET	Starts the movement path setting.	6	×	Command Statements
73	PATH START	Starts the PATH motion.	6	×	Command Statements

Torque control

No.	Command	Function	Condition	Direct	Type
17	CURTRQ	Acquires the current torque value of the specified main group axis.	-	×	Functions
29	DRIVE	(With T-option) Executes an absolute movement command for a specified axis.	4	○	Command Statements
110	TORQUE	Specifies/acquires the maximum torque command value which can be set for a specified main group axis.	4/-	○	Command Statements/ Functions
112	TRQTIME	Specifies/acquires the current limit time-out period at the specified main group axis when using a torque limit option in the DRIVE statement.	1/-	○	Command Statements/ Functions

8
9
10
11
12
13
14

Input/output & communication control

Input/output control

No.	Command	Function	Condition	Direct	Type
24	DELAY	Waits for the specified period (units: ms).	6	×	Command Statements
28	DO	Outputs a specified value to the DO port.	1	○	Command Statements
52	LO	Outputs a specified value to the LO port to enable/disable axis movement.	1	○	Command Statements
57	MO	Outputs a specified value to the MO port.	1	○	Command Statements
68	OUT	Turns ON the bits of the specified output ports and the command statement ends.	6	×	Command Statements
81	RESET	Turns the bit of a specified output port OFF.	1	○	Command Statements
92	SET	Turns the bit at the specified output port ON.	3	In part ×	Command Statements
96	SO	Outputs a specified value to the SO port.	1	○	Command Statements
108	TO	Outputs a specified value to the TO port.	1	○	Command Statements
114	WAIT	Waits until the conditions of the DI/DO conditional expression are met (with time-out).	6	×	Command Statements

Programming box

No.	Command	Function	Condition	Direct	Type
44	INPUT	Assigns a value to a variable specified from the programming box.	1	○	Command Statements
78	PRINT	Displays a character string at the programming box screen.	1	○	Command Statements

Communication control

No.	Command	Function	Condition	Direct	Type
65	ONLINE	Sets the specified communication port to the "online" mode.	1	○	Command Statements
60	OFFLINE	Sets a specified communication port to the "offline" mode.	1	○	Command Statements
90	SEND	Sends a file.	1	○	Command Statements

Other

Other

No.	Command	Function	Condition	Direct	Type
122	_SYSFLG	Axis status monitoring flag.	-	-	Functions

4 Functions: in alphabetic order

No.	Function	Type	Function
A			
1	ABS	Arithmetic function	Acquires the absolute value of a specified value.
3	ABSRPOS	Arithmetic function	Acquires the machine reference of the specified main group axis. (Valid only for axes where the return-to-origin method is set as "mark method".)
5	ACCEL	Arithmetic function	Acquires the acceleration coefficient parameter of the main group.
6	ARCH	Arithmetic function	Acquires the arch position parameter of the main group.
7	ARMCND	Arithmetic function	Acquires the current arm status of the main robot.
8	ARMTYPE	Arithmetic function	Acquires the current "hand system" setting of the main robot.
9	ATN	Arithmetic function	Acquires the arctangent of the specified value.
11	AXWGHT	Arithmetic function	Acquires the axis tip weight parameter of the main group.
C			
15	CHR$	Character string function	Acquires a character with the specified character code.
16	COS	Arithmetic function	Acquires the cosine value of a specified value.
17	CURTRQ	Arithmetic function	Acquires the current torque value of the specified main group axis.
D			
19	DATE$	Character string function	Acquires the date as a "yy/mm/dd" format character string.
20	DECEL	Arithmetic function	Acquires the deceleration rate parameter of the main group.
23	DEGRAD	Arithmetic function	Converts a specified value to radians (↔RADDEG).
26	DIST	Arithmetic function	Acquires the distance between 2 specified points.
E			
33	ERL	Arithmetic function	Gives the line No. where an error occurred.
33	ERR	Arithmetic function	Gives the error code number of an error which has occurred.
I			
45	INT	Arithmetic function	Acquires an integer for a specified value by truncating all decimal fractions.
J			
46	JTOXY	Point function	Converts joint coordinate data to main group Cartesian coordinate data. (↔XYTOJ)
L			
48	LEFT$	Character string function	Extracts a character string comprising a specified number of digits from the left end of a specified character string.
50	LEN	Arithmetic function	Acquires the length (number of bytes) of a specified character string.
53	LOCx	Point function	Acquires point data or shift data for a specified axis.
54	LSHIFT	Arithmetic function	Shifts a value to the left by the specified number of bits. (↔RSHIFT)
M			
55	MCHREF	Arithmetic function	Acquires the return-to-origin or absolute-search machine reference for a specified main group axis.
56	MID$	Character string function	Extracts a character string of a desired length from a specified character string.
O			
61	ORD	Arithmetic function	Acquires the character code of the first character in a specified character string.

8
9
10
11
12
13
14

No.	Function	Type	Function
66	ORGORD	Arithmetic function	Acquires the axis sequence parameter for performing return-to-origin and absolute search operations in the main group.
69	OUTPOS	Arithmetic function	Acquires the OUT enable position parameter of the main group.
P			
77	PPNT	Point function	Creates point data specified by a pallet definition number and pallet position number.
R			
79	RADDEG	Arithmetic function	Converts a specified value to degrees. (↔DEGRAD)
85	RIGHT$	Character string function	Extracts a character string comprising a specified number of digits from the right end of a specified character string.
87	RSHIFT	Arithmetic function	Shifts a value to the right by the specified number of bits. (↔LSHIFT)
S			
95	SIN	Arithmetic function	Acquires the sine value for a specified value.
100	SQR	Arithmetic function	Acquires the square root of a specified value.
99	STR$	Character string function	Converts a specified value to a character string (↔VAL)
T			
104	TAN	Arithmetic function	Acquires the tangent value for a specified value.
105	TCOUNTER	Arithmetic function	Outputs count-up values at 10ms intervals starting from the point when the TCOUNTER variable is reset.
106	TIME$	Character string function	Acquires the current time as an "hh:mm:ss" format character string.
107	TIMER	Arithmetic function	Acquires the current time in seconds, counting from 12:00 midnight.
109	TOLE	Arithmetic function	Acquires the main group tolerance parameter.
110	TORQUE	Arithmetic function	Acquires the maximum torque command value which can be set for a specified main group axis.
111	TRQSTS	Arithmetic function	Acquires the command end status for the DRIVE command with torque limit option executed at the main group.
112	TRQTIME	Arithmetic function	Acquires the current limit time-out period at the specified main group axis when using a torque limit option in the DRIVE statement.
V			
113	VAL	Arithmetic function	Converts the numeric value of a specified character string to an actual numeric value. (↔STR$)
W			
116	WEIGHT	Arithmetic function	Acquires the main robot tip weight parameter.
118	WHERE	Point function	Reads out the current position of the main group robot arm in joint coordinates (pulses).
120	WHRXY	Point function	Reads out the current position of the main group arm as Cartesian coordinates (mm, degrees).
X			
121	XYTOJ	Point function	Converts the point variable Cartesian coordinate data to the main group's joint coordinate data (↔JTOXY).
122	_SYSFLG	Arithmetic function	Axis status monitoring flag.

8
9
10
11
12
13
14

5 Functions: operation-specific

Point related functions

No.	Function name	Function
46	JTOXY	Converts joint coordinate data to main group Cartesian coordinate data. (↔XYTOJ)
53	LOCx	Acquires point data or shift data for a specified axis.
77	PPNT	Creates point data specified by a pallet definition number and pallet position number.
118	WHERE	Reads out the current position of the main group robot arm in joint coordinates (pulses).
120	WHRXY	Reads out the current position of the main group arm as Cartesian coordinates (mm, degrees).
121	XYTOJ	Converts the point variable Cartesian coordinate data to the main group's joint coordinate data (↔JTOXY).

Parameter related functions

No.	Function name	Function
3	ABSRPOS	Acquires the machine reference of the specified main group axis. (Valid only for axes where the return-to-origin method is set as "mark method".)
5	ACCEL	Acquires the acceleration coefficient parameter of the main group.
6	ARCH	Acquires the arch position parameter of the main group.
7	ARMCND	Acquires the current arm status of the main robot.
8	ARMTYPE	Acquires the current "hand system" setting of the main robot.
11	AXWGHT	Acquires the axis tip weight parameter of the main group.
17	CURTRQ	Acquires the current torque value of the specified main group axis.
20	DECEL	Acquires the deceleration rate parameter of the main group.
50	LEN	Acquires the length (number of bytes) of a specified character string.
55	MCHREF	Acquires the return-to-origin or absolute-search machine reference for a specified main group axis.
61	ORD	Acquires the character code of the first character in a specified character string.
66	ORGORD	Acquires the axis sequence parameter for performing return-to-origin and absolute search operations in the main group.
69	OUTPOS	Acquires the OUT enable position parameter of the main group.
109	TOLE	Acquires the main group tolerance parameter.
110	TORQUE	Acquires the maximum torque command value which can be set for a specified main group axis.
111	TRQSTS	Acquires the command end status for the DRIVE command with torque limit option executed at the main group.
112	TRQTIME	Acquires the current limit time-out period at the specified main group axis when using a torque limit option in the DRIVE statement.
116	WEIGHT	Acquires the main robot tip weight parameter.

Numeric calculation related functions

No.	Function name	Function
1	ABS	Acquires the absolute value of a specified value.
9	ATN	Acquires the arctangent of the specified value.
16	COS	Acquires the cosine value of a specified value.
23	DEGRAD	Converts a specified value to radians (↔RADDEG).
26	DIST	Acquires the distance between 2 specified points.
45	INT	Acquires an integer for a specified value by truncating all decimal fractions.
54	LSHIFT	Shifts a value to the left by the specified number of bits. (↔RSHIFT)
79	RADDEG	Converts a specified value to degrees. (↔DEGRAD)
87	RSHIFT	Shifts a value to the right by the specified number of bits. (↔LSHIFT)
95	SIN	Acquires the sine value for a specified value.
100	SQR	Acquires the square root of a specified value.
104	TAN	Acquires the tangent value for a specified value.
113	VAL	Converts the numeric value of a specified character string to an actual numeric value. (↔STR$)

Character string calculation related functions

No.	Function name	Function
15	CHR $	Acquires a character with the specified character code.
19	DATE $	Acquires the date as a "yy/mm/dd" format character string.
48	LEFT $	Extracts a character string comprising a specified number of digits from the left end of a specified character string.
56	MID $	Extracts a character string of a desired length from a specified character string.
85	RIGHT $	Extracts a character string comprising a specified number of digits from the right end of a specified character string.
99	STR $	Converts a specified value to a character string (↔VAL)

Parameter related functions

No.	Function name	Function
122	_SYSFLG	Axis status monitoring flag.
33	ERL	Gives the line No. where an error occurred.
33	ERR	Gives the error code number of an error which has occurred.
105	TCOUNTER	Outputs count-up values at 10ms intervals starting from the point when the TCOUNTER variable is reset.
106	TIME $	Acquires the current time as an "hh:mm:ss" format character string.
107	TIMER	Acquires the current time in seconds, counting from 12:00 midnight.

6 Execution Level

The level used to execute a program can be set as shown below.
However, the following commands can be executed only when in a "return-to-origin completion" condition.

Movement commands: MOVE, MOVEI, DRIVE, DRIVEI, PMOVE, PATH START
Position acquisition commands: WHERE, WHRXY

Level	Content									
	Program execution at return-to-origin incompletion		At power ON				Program reset at program START		Return-to-origin signal input in AUTO mode	
			Mode		Program reset					
	Possible	Not possible	MANUAL	AUTO	Yes	No	Yes	No	Enabled	Disabled
0		○	○			○		○		○
1	○		○			○		○		○
2	○		○		○			○		○
3	○			○		○		○		○
4	○			○	○			○		○
5	○		○		○		○			○
6	○			○	○		○			○
7	○			○		○		○	○ *1	
8	○			○	○		○		○ *1	

*1: When the AUTO mode absolute reset signal input (DI17) is enabled, the "robot program running" (DO13) signal switches ON during the processing operation executed by the AUTO mode absolute reset signal input.

REFERENCE For execution level details, refer the user's manuals for each controller.

Index

Index

Symbole

_SELECT 1-5

A

Absolute reset 12-9
Absolute reset command 13-9
Absolute reset movement command 13-8
Acceleration coefficient 7-22, 7-23
Acceleration setting 7-106
Acquiring return-to-origin status 12-29
Acquiring the absolute reset status 12-29
Acquiring the access level 12-24
Acquiring the arm status 12-25
Acquiring the break point status 12-25
Acquiring the controller configuration status 12-26
Acquiring the current positions 12-32
Acquiring the current positions on pulse unit coordinates 12-32
Acquiring the current positions on XY coordinates 12-33
Acquiring the display language 12-24
Acquiring the emergency stop status 12-37
Acquiring the error status by self-diagnosis 12-37
Acquiring the execution level 12-26
Acquiring the message 12-28
Acquiring the mode status 12-27
Acquiring the option slot status 12-38
Acquiring the point coordinates and units 12-31
Acquiring the program execution status 12-12
Acquiring the remaining memory capacity 12-36
Acquiring the servo status 12-30
Acquiring the shift status 12-35
Acquiring the speed setting status 12-31
Acquiring the tasks in RUN status 12-34
Acquiring the tasks in SUSPEND status 12-34
Acquiring the tasks operation status 12-35
Acquiring the version information 12-32
All file 10-23
Arch motion setting 7-103, 7-138
Arithmetic assignment statement 7-83
Arithmetic operations 4-1
Arm designation change command 13-12
Arm lock output 7-86
Arm lock output variable 3-13
Arm lock output variables 6-6
Array subscript 7-45
Array variable file 10-31
Array variables 3-5
Assignment statement 7-83
Automatic movement speed 7-28
AUTO mode operation 12-7
Auto movement speed change command 13-11
Axis tip weight 7-29

B

Bit Settings 3-19

C

Cartesian coordinate format 4-5
CASE 7-154
Changing the MANUAL mode speed 12-9
Changing the mode 12-6
Changing the program attribute 12-16
Character constants 2-2
Character string
  Comparison 4-4
  Connection 4-4
  Link 7-84
  Operations 4-4
Character string assignment statement 7-84
Circular interpolation 7-97, 7-125
Clearing the error message 12-19
Command Statement Format 1-8
Comment 1-7, 7-145
COMMON 1-6
Communication port 7-114, 7-119
Compiling 7-170
Constant file 10-30
Control codes 12-46
CONT setting 7-93, 7-105
Coordinate plane setting 7-107, 7-129
Copying point comments 12-13
Copying point data 12-13

D

Data file 10-1
  Data file types 10-1
Data format conversion 4-3
Data handling 12-24
Data readout processing 12-41
Data write processing 12-42
Deceleration rate 7-37
Deceleration setting 7-106
Declares array variable 7-45
Define point 7-140
Defines functions which can be used by the user 7-40
DI/DO conditional expressions 4-6
DI file 10-33

DO file 10-35
Dummy argument 7-167
Dynamic variables 3-20

E

EOF file 10-50
Erasing point comments 12-15
Erasing point data 12-14
Error code number 7-116
Error message history file 10-47
Error processing 7-116, 7-148
Error recovery processing 7-148
Ethernet port communication file 10-56
Executes absolute movement of specified axes 7-47
Execution level 1-5
Execution Level 14-17
External program 7-39

F

FUNCTION 1-2
Functions: in alphabetic order 7-13, 14-13
Functions: operation-specific 7-15, 14-15

G

Global variable 10-27
Global variables 3-20

H

Hand
- Acquiring the status 12-36
- Define 7-70
- Definition file 10-15
- Designation change command 13-12
- Switche 7-31

Hand system flag 4-5, 7-100, 7-111, 7-140, 10-4, 10-18
Hand system to "Right" 7-151

I

IF 7-74
- Block IF statement 7-75
- Simple IF statement 7-74

Inching movement command 13-7
Incremental mode 7-121
Initialize 12-17
- Communication port 12-17
- Memory 12-17

Integer constants 2-1
Internal output 7-92
Internal output variable 3-12
Interrupting the command execution 12-46
IO command
- Description 13-5
- Execution trigger input 13-1, 13-2
- Format 13-1
- List 13-4
- Sending and receiving 13-2

IO register 13-2

J

Jog movement command 13-7
Joint coordinate format 4-5

L

Label 7-79
LABEL Statement 7-79
Left-hand system 7-81
Linear interpolation 7-125
Linear interpolation movement 7-95
Line number where error occurred 7-116
Local variable 7-167
Local variables 3-20
LO file 10-39
Logic operations 4-2

M

Machine reference file 10-49
Main task 7-32
MANUAL mode operation 12-9
Manual movement speed change command 13-11
MO file 10-37
Motor power 7-157
MOVE command 13-5
MOVEI command 13-6
Movement direction setting. 7-53
Movement speed 7-163
Moves the specified robot axes in a relative manner 7-55
Multi-task 5-1

N

Numeric constants 2-1

O

Online Command List 12-1
Online commands
- Acquiring the value of a character string expression 12-39
- Acquiring the value of a numerical expression 12-39

Acquiring the value of a point expression 12-40
Acquiring the value of a shift expression 12-40
Operation speed 7-28
OUT enable position 7-123

P

Pallet
Define 7-135
Definition file 10-17
Definition number 7-135, 7-136
Erasing 12-15
Movement 7-136
Movement command 13-6
Position number 7-136
Palletizing 11-4, 11-10
Parallel input variable 3-10
Parallel output variable 3-11
Parallel port...7-43, 7-46
Parameter directory file 10-26
Parameter file 10-10
PATH 7-125, 8-1
Cautions when using this function 8-2
Ends the movement setting................................... 7-131
Features 8-1
How to use 8-1
Specifies the motion path 7-125
Starts the movement setting 7-132
Starts the PATH motion 7-134
Performs absolute movement 7-93, 7-109
Pick and place 11-12
Point assignment statement.................................. 7-84
Point comment file 10-8
Point data
For a specified axis 7-87
Format 4-5
Point data variable 3-7
Point display unit designation command 13-12
Point element variable....................................... 3-9
Point file 10-4
Point teaching command..................................... 13-8
Port output setting 7-108, 7-130
Priority of arithmetic operation 4-3
Program
Copy 12-12
Erase 12-14
Selected 1-5
Stop 7-69
Switch between the programs 1-5
Switche 7-170
Temporarily stop 7-73
Program directory file 10-24
Program execution wait 7-42
Program file.. 10-2
Program level 7-159
Programming box 12-19
Program Names 1-2
Program speed change command 13-11
PTP movement of specified axis............................ 7-47

R

Read file 7-155
Ready queues 5-3
Real constants 2-1
Registering function keys 1-3
Relational operators 4-1
Rename program name 12-16
Reserved word list 14-1
Resetting the internal emergency stop flag 12-21
Return-to-origin 7-21
Return-to-origin command 13-9
Return-to-origin sequence 7-120
Robot Language Lists: Command list in alphabetic order 14-3
Robot Language Lists: Function Specific 14-7
RS-232C 11-18, 11-19

S

Sealing... 11-17
Sequence function 6-1
Sequence program 6-1
Acquiring the execution status 12-30
Cannot be used 6-3
Compiling 6-2
Creating 6-5
Executing 6-4
Priority of logic operations 6-8
Program capacity 6-8
Programming method 6-1
Scan time.. 6-8
Setting the execution flag 12-21
Specifications 6-8
STEP execution 6-4
SEQUENCE program 1-4
Serial double word input.................................... 3-17
Serial double word output 3-18
Serial input variable 3-15
Serial output variable 3-16
Serial port 7-162
Serial port communication file 10-51
Serial word input.. 3-17
Serial word output 3-18
Servo
FREE 7-157
OFF 7-157

ON 7-157
Servo command 13-10
Servo status 7-157
Setting a break point 12-8
Setting the coordinates and units in MANUAL mode 12-19
Setting the display language 12-18
Setting the execution level 12-20
Setting the hand system 12-21
Setting the SCARA robot hand system 12-21
Setting the sequence program execution flag 12-21
Setting the UTILITY mode 12-20
Shift assignment statement 7-85
Shift coordinate 7-153, 7-160
Definition file 10-13
Shift coordinate variable 3-8
Shift designation change command 13-12
Shift element variable 3-10
SI file 10-43
SO file 10-45
SOW file 10-54
Static variables 3-20
STOPON condition setting 7-139
STOPON conditions setting 7-50, 7-59
Sub-procedure 7-30, 7-159, 7-167
Sub-routine 7-67
Subroutine 7-117
Switching the execution task 12-8
Switching the program 12-44
System Variables 3-2, 3-7

T

Task
Condition wait 5-4
Definition 5-1
Deleting 5-6
Directly terminate 7-35
Number 7-164
Priority order 5-1
Priority ranking 7-32, 7-164
Program example 5-8
Restart 7-147
Restarting 5-5
Scheduling 5-3
Sharing the data 5-8
Start 7-164
Starting 5-2
Status and transition 5-2
Stopping 5-7
Suspending 5-5
Temporarily stop 7-169
Terminate 7-65
Task status
NON EXISTEN 5-2
READY 5-2
RUN 5-2
STOP 5-2
SUSPEND 5-2
WAIT 5-2
Timer output variable 3-14
Tip weight 7-185
TO file 10-41
Tolerance 7-176
TO port 7-175
Torque command value 7-177
Torque limit setting 7-50, 7-180
Torque limit value 7-50
Torque offset value 7-50
Type Conversions 3-6

U

User program examples
Application 11-8
Basic operation 11-1
User Variables 3-2
Using point numbers 11-2
Using shift coordinates 11-3
Utility operation 12-12

V

Valid range of dynamic array variables 3-20
Valid range of variables 3-20
Value Pass-Along & Reference Pass-Along 3-6
Variable file 10-27
Variable Names 3-3
Variable Types 3-4

W

WAIT status 5-4
Write file 7-155

X

XY setting 7-52

Revision history

A manual revision code appears as a suffix to the catalog number on the front cover manual.

Cat. No. I139E-EN-02

Revision code

The following table outlines the changes made to the manual during each revision.

Revision code	Date	Description
01	February 2010	Original production
02	February 2014	ABSINIT and CURTRQ commands were added. Chapter 9 "Limitless motion" was added. Parameters related to R6YXTW500 model were added.

Authorized Distributor: