

mitsubishi

Mitsubishi Industrial Robot

CRnQ/CRnD Controller

INSTRUCTION MANUAL

Detailed explanations of functions and operations

MELFA

BFP-A8661-C



Safety Precautions

Always read the following precautions and the separate "Safety Manual" before starting use of the robot to learn the required measures to be taken.

⚠ CAUTION

All teaching work must be carried out by an operator who has received special training. (This also applies to maintenance work with the power source turned ON.)

Enforcement of safety training

⚠ CAUTION

For teaching work, prepare a work plan related to the methods and procedures of operating the robot, and to the measures to be taken when an error occurs or when restarting. Carry out work following this plan. (This also applies to maintenance work with the power source turned ON.)

Preparation of work plan

⚠ WARNING

Prepare a device that allows operation to be stopped immediately during teaching work. (This also applies to maintenance work with the power source turned ON.)

Setting of emergency stop switch

⚠ CAUTION

During teaching work, place a sign indicating that teaching work is in progress on the start switch, etc. (This also applies to maintenance work with the power source turned ON.)

Indication of teaching work in progress

⚠ WARNING

Provide a fence or enclosure during operation to prevent contact of the operator and robot.

Installation of safety fence

⚠ CAUTION

Establish a set signaling method to the related operators for starting work, and follow this method.

Signaling of operation start

⚠ CAUTION

As a principle turn the power OFF during maintenance work. Place a sign indicating that maintenance work is in progress on the start switch, etc.

Indication of maintenance work in progress

⚠ CAUTION

Before starting work, inspect the robot, emergency stop switch and other related devices, etc., and confirm that there are no errors.

Inspection before starting work

The points of the precautions given in the separate "Safety Manual" are given below.
Refer to the actual "Safety Manual" for details.

CAUTION

Use the robot within the environment given in the specifications. Failure to do so could lead to a drop or reliability or faults. (Temperature, humidity, atmosphere, noise environment, etc.)

CAUTION

Transport the robot with the designated transportation posture. Transporting the robot in a non-designated posture could lead to personal injuries or faults from dropping.

CAUTION

Always use the robot installed on a secure table. Use in an instable posture could lead to positional deviation and vibration.

CAUTION

Wire the cable as far away from noise sources as possible. If placed near a noise source, positional deviation or malfunction could occur.

CAUTION

Do not apply excessive force on the connector or excessively bend the cable. Failure to observe this could lead to contact defects or wire breakage.

CAUTION

Make sure that the workpiece weight, including the hand, does not exceed the rated load or tolerable torque. Exceeding these values could lead to alarms or faults.

WARNING

Securely install the hand and tool, and securely grasp the workpiece. Failure to observe this could lead to personal injuries or damage if the object comes off or flies off during operation.

WARNING

Securely ground the robot and controller. Failure to observe this could lead to malfunctioning by noise or to electric shock accidents.

CAUTION

Indicate the operation state during robot operation. Failure to indicate the state could lead to operators approaching the robot or to incorrect operation.

WARNING

When carrying out teaching work in the robot's movement range, always secure the priority right for the robot control. Failure to observe this could lead to personal injuries or damage if the robot is started with external commands.

CAUTION

Keep the jog speed as low as possible, and always watch the robot. Failure to do so could lead to interference with the workpiece or peripheral devices.

CAUTION

After editing the program, always confirm the operation with step operation before starting automatic operation. Failure to do so could lead to interference with peripheral devices because of programming mistakes, etc.

CAUTION

Make sure that if the safety fence entrance door is opened during automatic operation, the door is locked or that the robot will automatically stop. Failure to do so could lead to personal injuries.

CAUTION

Never carry out modifications based on personal judgments, or use non-designated maintenance parts.

Failure to observe this could lead to faults or failures.

WARNING

When the robot arm has to be moved by hand from an external area, do not place hands or fingers in the openings. Failure to observe this could lead to hands or fingers catching depending on the posture.

⚠ CAUTION

Do not stop the robot or apply emergency stop by turning the robot controller's main power OFF. If the robot controller main power is turned OFF during automatic operation, the robot accuracy could be adversely affected. Moreover, it may interfere with the peripheral device by drop or move by inertia of the arm.

⚠ CAUTION

Do not turn off the main power to the robot controller while rewriting the internal information of the robot controller such as the program or parameters. If the main power to the robot controller is turned off while in automatic operation or rewriting the program or parameters, the internal information of the robot controller may be damaged.

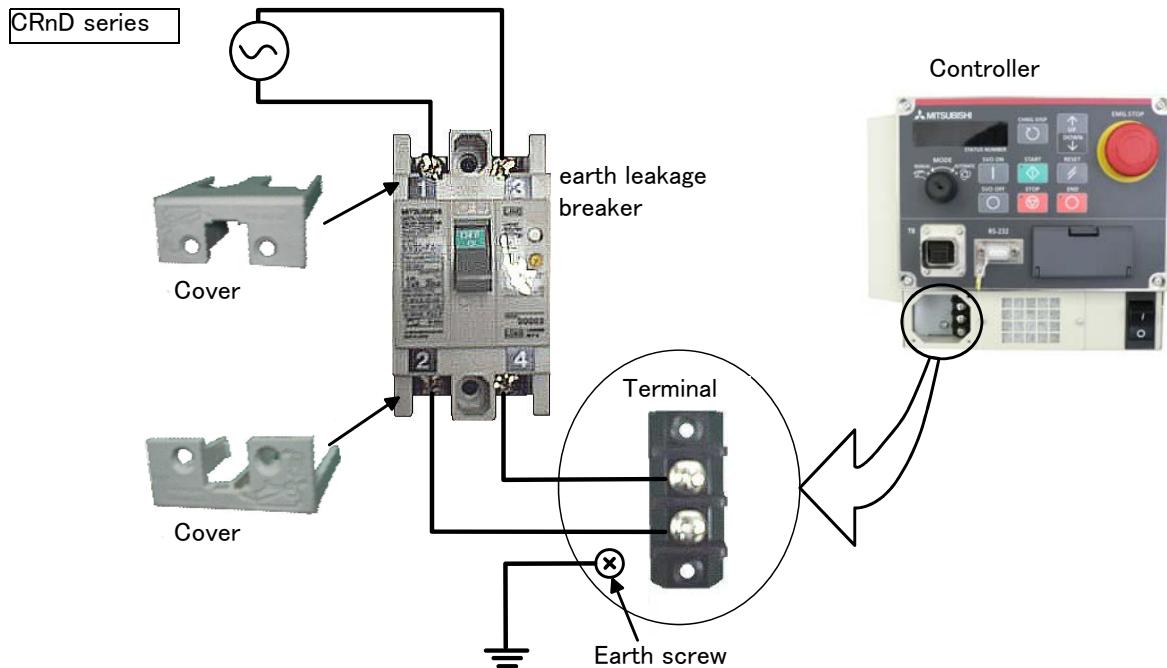
⚠ CAUTION

Security of operation and the maintenance of marketing of USB equipment cannot be done at our company. Care fully because the commercial item may not fit the problem of affinity with our equipment, and the FA environment (temperature, the noise, etc.). When using it, measures against the noise, such as measures against EMI and the addition of the ferrite core, may be necessary. Please fully confirm of operation of the customer

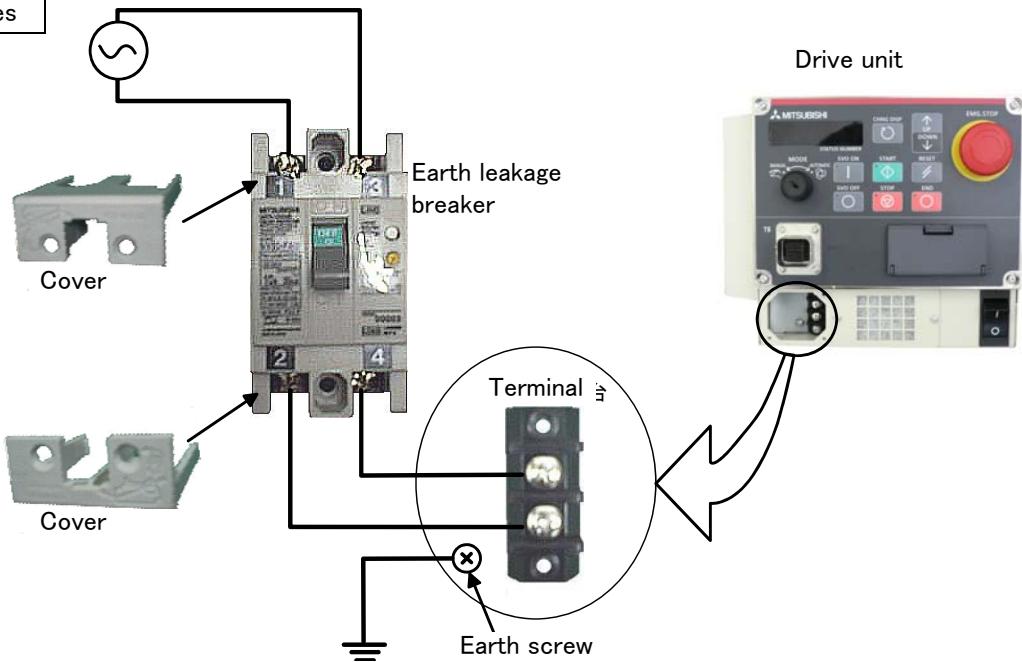
C. Notes of the basic component are shown. (CR1DDU1-700 series)

⚠ CAUTION

Please install the earth leakage breaker in the primary side supply power supply of the controller because of leakage protection.



CRnQ series



Revision history

Date	Specifications No.	Details of revisions
2008-05-12	BFP-A8661	First print
2008-09-01	BFP-A8661-A	<ul style="list-style-type: none"> Command were added (Def Long, Fine J, Fine P, MvTune) System variable was added (M_Uar32) Signal-parameter were added (ROBOTERR, QMLTCPUN, QMLTCPUn, QMLTCPUS) The output signal reset pattern of the sequencer link was added. The I/O function of the sequencer link was added. The connection method of display equipment (GOT) was added. The communications parameter (COMDEV, etc) and the communication setup were changed. Continuing function was corrected (parameter: CTN). The initial value of TB override operation rights (parameter: OvrdTB) was changed. The initial value of the dedicated I/O parameter of CRnQ series was added. Display unit (GOT1000 Series) connection was added (Reference)
2008-11-10	BFP-A8661-B	<ul style="list-style-type: none"> The parameter about the Ethernet interface and sample program were added. The details of CRnQ and CRnD were incorporated.
2008-11-12	BFP-A8661-C	<ul style="list-style-type: none"> Error in writing correction.

	Page
1 Before starting use	1-1
1.1 Using the instruction manuals	1-1
1.1.1 The details of each instruction manuals	1-1
1.1.2 Terminological definition	1-2
1.1.3 Symbols used in instruction manual	1-2
1.2 Safety Precautions	1-3
1.2.1 Precautions given in the separate Safety Manual	1-4
2 Explanation of functions	2-6
2.1 Operation panel (O/P) functions	2-6
2.2 Teaching pendant (T/B) functions	2-9
2.2.1 Operation rights	2-10
2.3 Functions Related to Movement and Control	2-11
3 Explanation of operation methods	3-13
3.1 Operation of the teaching pendant menu screens	3-13
(1) Screen tree	3-13
(2) Input of the number/character	3-16
(3) Selecting a menu	3-17
3.2 Jog Feed (Overview)	3-19
3.2.1 Types of jog feed	3-19
3.2.2 Speed of jog feed	3-20
3.2.3 JOINT jog	3-20
3.2.4 TOOL jog	3-21
3.2.5 XYZ jog	3-21
3.2.6 3-axis XYZ jog	3-22
3.2.7 CYLINDER jog	3-22
3.2.8 Switching Tool Data	3-23
3.2.9 Impact Detection during Jog Operation	3-24
(1) Impact Detection Level Adjustment during Jog Operation	3-25
3.3 Opening/Closing the Hands	3-26
3.4 Aligning the Hand	3-27
3.5 Programming	3-29
3.5.1 Creating a program	3-29
(1) Opening the program edit screen	3-29
(2) Creating a program	3-30
(3) Completion of program creation and saving programs	3-32
(4) Correcting a program	3-33
(5) Registering the current position data	3-35
(6) Deletion of the position variable	3-38
(7) Confirming the position data (Position jump)	3-39
(8) Correcting the MDI (Manual Data Input)	3-40
3.6 Debugging	3-41
(1) Step feed	3-41
(2) Step return	3-42
(3) Step feed in another slot	3-43
(4) Step jump	3-44
3.7 Automatic operation	3-46
(1) Setting the operation speed	3-46
(2) Selecting the program No.	3-46
(3) Starting automatic operation	3-47
(4) Stopping	3-48
(5) Resuming automatic operation from stopped state	3-48
(6) Resetting the program	3-49
3.8 Turning the servo ON/OFF	3-50
3.9 Error reset operation	3-51

Contents

	Page
3.10 Operation to Temporarily Reset an Error that Cannot Be Canceled	3-51
3.11 Operating the program control screen	3-52
(1) Program list display	3-52
(2) Copying programs	3-53
(3) Name change of the program (Rename)	3-54
(4) Deleting a program(Delete)	3-55
(5) Protection of the program (Protect)	3-56
3.12 Operation of operating screen	3-58
3.12.1 Display of the execution line	3-58
(1) Select the confirmation menu	3-58
(2) Step feed	3-58
(3) Step jump	3-61
(4) Step feed in another slot	3-61
(5) Finishing of the confirmation screen.	3-61
3.12.2 Test operation	3-62
(1) Select the test operation	3-62
3.13 Operating the monitor screen	3-63
(1) Input signal monitor	3-63
(2) Output signal monitor	3-65
(3) Input register monitor	3-67
(4) Output register monitor	3-68
(5) Variable monitor	3-71
(6) Error history	3-73
3.14 Operation of maintenance screen	3-74
3.15 Operation of the origin and the brake screen	3-76
(1) Origin	3-76
(2) Brake	3-76
3.16 Operation of setup / initialization screen	3-78
(1) Initialize the program	3-78
(2) Initialize the parameter	3-79
(3) Initialize the battery	3-80
(4) Operation	3-81
(5) Time setup	3-81
(6) Version	3-82
3.17 Operation of the initial-setting screen	3-83
(1) Set the display language	3-83
(2) Adjustment of contrast	3-85
4 MELFA-BASIC V	4-87
4.1 MELFA-BASIC V functions	4-87
4.1.1 Robot operation control	4-88
(1) Joint interpolation movement	4-88
(2) Linear interpolation movement	4-89
(3) Circular interpolation movement	4-90
(4) Continuous movement	4-92
(5) Acceleration/deceleration time and speed control	4-93
(6) Confirming that the target position is reached	4-95
(7) High path accuracy control	4-96
(8) Hand and tool control	4-97
4.1.2 Pallet operation	4-98
4.1.3 Program control	4-104
(1) Unconditional branching, conditional branching, waiting	4-104
(2) Repetition	4-106
(3) Interrupt	4-107
(4) Subroutine	4-108
(5) Timer	4-109
(6) Stopping	4-110

	Page
4.1.4 Inputting and outputting external signals	4-111
(1) Input signals	4-111
(2) Output signals	4-111
4.1.5 Communication	4-112
4.1.6 Expressions and operations	4-113
(1) List of operator	4-113
(2) Relative calculation of position data (multiplication)	4-115
(3) Relative calculation of position data (Addition)	4-115
4.1.7 Appended statement	4-116
4.2 Multitask function	4-117
4.2.1 What is multitasking?	4-117
4.2.2 Executing a multitask	4-118
4.2.3 Operation state of each slot	4-118
4.2.4 Precautions for creating multitask program	4-120
(1) Relationship between number of tasks and processing time	4-120
(2) Specification of the maximum number of programs executed concurrently	4-120
(3) How to pass data between programs via external variables	4-120
(4) Confirmation of operating status of programs via robot status variables	4-120
(5) The program that operates the robot is basically executed in slot 1.	4-120
(6) How to perform the initialization processing via constantly executed programs	4-120
4.2.5 Precautions for using a multitask program	4-121
(1) Starting the multitask	4-121
(2) Display of operation status	4-121
4.2.6 Example of using multitask	4-122
(1) Robot work details.	4-122
(2) Procedures to multitask execution	4-123
4.2.7 Program capacity	4-124
(1) Program save area	4-124
(2) Program edit area	4-124
(3) Program execution area	4-124
4.3 Detailed specifications of MELFA-BASIC V	4-126
(1) Program name	4-126
(2) Command statement	4-126
(3) Variable	4-127
4.3.1 Statement	4-128
4.3.2 Appended statement	4-128
4.3.3 Step	4-128
4.3.4 Step No.	4-128
4.3.5 Label	4-128
4.3.6 Types of characters that can be used in program	4-129
4.3.7 Characters having special meanings	4-130
(1) Uppercase and lowercase identification	4-130
(2) Underscore (_)	4-130
(3) Apostrophe (')	4-130
(4) Asterisk (*)	4-130
(5) Comma (,)	4-130
(6) Period (.)	4-130
(7) Space	4-130
4.3.8 Data type	4-131
4.3.9 Constants	4-131
4.3.10 Numeric value constants	4-131
(1) Decimal number	4-131
(2) Hexadecimal number	4-131
(3) Binary number	4-131
(4) Types of constant	4-131
4.3.11 Character string constants	4-131
4.3.12 Position constants	4-132
(1) Coordinate, posture and additional axis data types and meanings	4-132

Contents

	Page
(2) Meaning of structure flag data type and meanings	4-132
4.3.13 Joint constants	4-133
(1) Axis data format and meanings	4-133
4.3.14 Angle value	4-134
4.3.15 Variables	4-134
4.3.16 Numeric value variables	4-135
4.3.17 Character string variables	4-135
4.3.18 Position variables	4-135
4.3.19 Joint variables	4-136
4.3.20 Input/output variables	4-136
4.3.21 Array variables	4-136
4.3.22 External variables	4-138
4.3.23 Program external variables	4-138
4.3.24 User-defined external variables	4-139
4.3.25 Creating User Base Programs	4-140
4.3.26 Robot status variables	4-141
4.4 Logic numbers	4-145
4.5 Functions	4-145
(1) User-defined functions	4-145
(2) Built-in functions	4-145
4.6 List of Instructions	4-148
(1) Instructions related to movement control	4-148
(2) Instructions related to program control	4-148
(3) Definition instructions	4-149
(4) Multi-task related	4-149
(5) Others	4-150
4.7 Operators	4-151
4.8 Priority level of operations	4-152
4.9 Depth of program's control structure	4-152
4.10 Reserved words	4-152
4.11 Detailed explanation of command words	4-153
4.11.1 How to read the described items	4-153
4.11.2 Explanation of each command word	4-153
4.12 Detailed explanation of Robot Status Variable	4-266
4.12.1 How to Read Described items	4-266
4.12.2 Explanation of Each Robot Status Variable	4-266
4.13 Detailed Explanation of Functions	4-311
4.13.1 How to Read Described items	4-311
4.13.2 Explanation of Each Function	4-311
 5 Functions set with parameters	5-347
5.1 Movement parameter	5-347
5.2 Signal parameter	5-355
5.2.1 About multi CPU input offsets (CRnQ-700 controller only)	5-358
(1) Case (A)	5-358
(2) Case (B)	5-359
5.3 Operation parameter	5-360
5.4 Command parameter	5-363
5.5 Communication parameter	5-367
5.6 Standard Tool Coordinates	5-369
5.7 About Standard Base Coordinates	5-371
5.8 About user-defined area	5-372
5.9 Free plane limit	5-373
5.10 Automatic return setting after jog feed at pause	5-374
5.11 Automatic execution of program at power up	5-376

	Page
5.12 About the hand type	5-377
5.13 About default hand status	5-378
5.14 About the output signal reset pattern	5-379
5.15 About the communication setting(RS-232)	5-381
5.16 About the communication setting(Ethernet)	5-382
5.16.1 Details of parameters	5-382
(1) NETIP (IP address of robot controller)	5-382
(2) NETMSK (sub-net-mask)	5-382
(3) NETPORT (port No.)	5-382
(4) CRRCE11 to 19 (protocol)	5-382
(5) COMDEV (Definition of devices corresponding to COM1: to 8)	5-383
(6) NETMODE (server specification)	5-383
(7) NETHSTIP (The IP address of the server of the data communication point)	5-383
(8) MXTTOUT (Timeout setting for executing real-time external control command)	5-383
5.16.2 Example of setting of parameter 1 (When the Support Software is used)	5-384
5.16.3 Example of setting of parameter 2-1	5-385
5.16.4 Example of setting parameters 2-2	5-386
5.16.5 Example of setting parameters 3	5-387
5.17 Connection confirmation	5-388
5.17.1 Checking the connection with the Windows ping command	5-388
5.18 Hand and Workpiece Conditions (optimum acceleration/deceleration settings)	5-389
5.19 About the singular point adjacent alarm	5-391
5.20 About ROM operation/high-speed RAM operation function	5-392
5.21 Warm-Up Operation Mode	5-402
5.22 About singular point passage function	5-409
5.23 About the impact detection function	5-414
(1) Overview of the function	5-414
(2) Applicable models	5-415
(3) Related parameters	5-415
(4) How to use the impact detection function	5-416
6 External input/output functions	6-420
6.1 Types	6-420
6.2 Sequencer link I/O function	6-421
6.2.1 Parameter setting	6-421
(1) Sequencer CPU parameter setting	6-421
(2) Robot CPU parameter setting	6-421
6.2.2 CPU shared memory and robot I/O signal compatibility	6-422
6.2.3 Sequence ladder example	6-423
6.2.4 Assignment of the dedicated I/O signal. (at factory shipping)	6-425
6.2.5 Comparison of the I/O point of the CRnQ700 and the CRn500 series	6-427
6.3 Dedicated input/output	6-428
6.4 Enable/disable status of signals	6-435
6.5 External signal timing chart	6-436
6.5.1 Individual timing chart of each signal	6-436
6.5.2 Timing chart example	6-443
(1) External signal operation timing chart (Part 1)	6-443
(2) External signal operation timing chart (Part 2)	6-444
(3) Example of external operation timing chart (Part 3)	6-445
(4) Example of external operation timing chart (Part 4)	6-446
6.6 Emergency stop input	6-447
6.6.1 Robot Behavior upon Emergency Stop Input	6-447
6.7 Display unit (GOT1000 Series) connection (reference)	6-448
(1) Usage example	6-448
(2) Specifications	6-448

Contents

	Page
(3) Connection	6-449
(4) Settings	6-450
7 Appendix	Appendix-452
7.1 Real-time external control function	Appendix-452
7.1.1 Explanation of communication data packet	Appendix-454
7.1.2 Sample program	Appendix-457
(1) Sample program of data link	Appendix-457
(2) Sample program for real-time external control function	Appendix-463
7.2 Configuration flag	Appendix-474

*Introduction

Thank you for purchasing the Mitsubishi industrial robot.

This instruction manual explains the functions and operation methods of the robot controller and teaching pendant (R32TB), and the functions and specifications of the MELFA-BASIC V programming language.

Apply to both the CRnQ-700 series controller corresponding to iQPlatform, and the CRnD-700 series controller. Especially the function added individually is indicated to be "CRnQ" or "CRnQ-700" and "CRnD", or "CRnD-700."

Always read through this manual before starting use to ensure correct usage of the robot.

Note that this document is prepared for the following software versions.

Controller : Version

CRnQ: N1 or later

CRnD: P1 or later

T/B : Version 1.0 or later

Notice

*ONLY QUALIFIED SERVICE PERSONNEL MAY INSTALL OR SERVICE THE ROBOT SYSTEM.

*ANY PERSON WHO PROGRAM, TEACHES, OPERATE, MAINTENANCE OR REPAIRS THE ROBOT SYSTEM IS TRAINED AND DEMONSTRATES COMPETENCE TO SAFELY PERFORM THE ASSIGNED TASK.

*ENSURE COMPLIANCE WITH ALL LOCAL AND NATIONAL SAFETY AND ELECTRICAL CODES FOR THE INSTALLATION AND OPERATION OF THE ROBOT SYSTEM.

- No part of this manual may be reproduced by any means or in any form, without prior consent from Mitsubishi.
- The details of this manual are subject to change without notice.
- An effort has been made to make full descriptions in this manual. However, if any discrepancies or unclear points are found, please contact your dealer.
- The information contained in this document has been written to be accurate as much as possible. Please interpret that items not described in this document "cannot be performed.". Please contact your nearest dealer if you find any doubtful, wrong or skipped point.

1 Before starting use

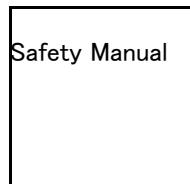
This chapter explains the details and usage methods of the instruction manuals, the basic terminology and the safety precautions.

1.1 Using the instruction manuals

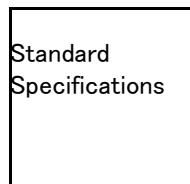
1.1.1 The details of each instruction manuals

The contents and purposes of the documents enclosed with this product are shown below. Use these documents according to the application.

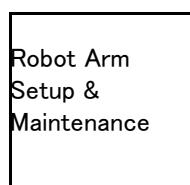
For special specifications, a separate instruction manual describing the special section may be enclosed.



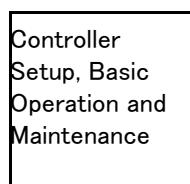
Safety Manual
Explains the common precautions and safety measures to be taken for robot handling, system design and manufacture to ensure safety of the operators involved with the robot.



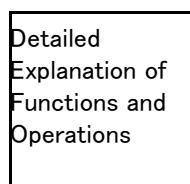
Standard Specifications
Explains the product's standard specifications, factory-set special specifications, option configuration and maintenance parts, etc. Precautions for safety and technology, when incorporating the robot, are also explained.



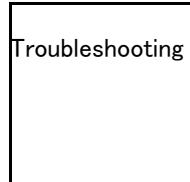
Robot Arm Setup & Maintenance
Explains the procedures required to operate the robot arm (unpacking, transportation, installation, confirmation of operation), and the maintenance and inspection procedures.



Controller Setup, Basic Operation and Maintenance
Explains the procedures required to operate the controller (unpacking, transportation, installation, confirmation of operation), basic operation from creating the program to automatic operation, and the maintenance and inspection procedures.



Detailed Explanation of Functions and Operations
Explains details on the functions and operations such as each function and operation, commands used in the program, connection with the external input/output device, and parameters, etc.



Troubleshooting
Explains the causes and remedies to be taken when an error occurs. Explanations are given for each error No.

1.1.2 Terminological definition

Explain the term currently used in this manual.

Robot controller The controller which controls the robot arm
It consists of the robot CPU system and the drive unit.(SQ series)

Robot CPU (Unit) The CPU unit for the robots which installed to the sequencer base unit.
(Q3*DB) (SQ series)

Robot CPU system Multi-CPU system. (SQ series)
It consists of MELSEC units, such as the sequencer base unit, the sequencer CPU unit, and the robot CPU unit, etc.

Drive unit The box which mounts the servo amplifier for the robots, the safety circuit, etc. (SQ series)

1.1.3 Symbols used in instruction manual

The symbols and expressions shown in [Table 1-1](#) are used throughout this instruction manual. Learn the meaning of these symbols before reading this instruction manual.

Table 1-1 : Symbols in instruction manual

Symbol	Meaning
 DANGER	Precaution indicating cases where there is a risk of operator fatality or serious injury if handling is mistaken. Always observe these precautions to safely use the robot.
 WARNING	Precaution indicating cases where the operator could be subject to fatalities or serious injuries if handling is mistaken. Always observe these precautions to safely use the robot.
 CAUTION	Precaution indicating cases where operator could be subject to injury or physical damage could occur if handling is mistaken. Always observe these precautions to safely use the robot.
[JOG]	If a word is enclosed in brackets or a box in the text, this refers to a key on the teaching pendant.
[RESET] + [EXE] (A) (B)	This indicates to press the (B) key while holding down the (A) key. In this example, the [RESET] key is pressed while holding down the [+EXE] key.
T/B	This indicates the teaching pendant.
O/P	This indicates the operating panel on the front of the controller(drive unit).

1.2 Safety Precautions

Always read the following precautions and the separate "Safety Manual" before starting use of the robot to learn the required measures to be taken.

⚠ CAUTION

All teaching work must be carried out by an operator who has received special training.
(This also applies to maintenance work with the power source turned ON.)

Enforcement of safety training

⚠ CAUTION

For teaching work, prepare a work plan related to the methods and procedures of operating the robot, and to the measures to be taken when an error occurs or when restarting. Carry out work following this plan. (This also applies to maintenance work with the power source turned ON.)

Preparation of work plan

⚠ WARNING

Prepare a device that allows operation to be stopped immediately during teaching work.
(This also applies to maintenance work with the power source turned ON.)

Setting of emergency stop switch

⚠ CAUTION

During teaching work, place a sign indicating that teaching work is in progress on the start switch, etc. (This also applies to maintenance work with the power source turned ON.)

Indication of teaching work in progress

⚠ DANGER

Provide a fence or enclosure during operation to prevent contact of the operator and robot.

Installation of safety fence

⚠ CAUTION

Establish a set signaling method to the related operators for starting work, and follow this method.

Signaling of operation start

⚠ CAUTION

As a principle turn the power OFF during maintenance work. Place a sign indicating that maintenance work is in progress on the start switch, etc.

Indication of maintenance work in progress

⚠ CAUTION

Before starting work, inspect the robot, emergency stop switch and other related devices, etc., and confirm that there are no errors.

Inspection before starting work

1.2.1 Precautions given in the separate Safety Manual

The points of the precautions given in the separate "Safety Manual" are given below. Refer to the actual "Safety Manual" for details.



If the automatic operation of the robot is operated by two or more control equipment, design the right management of operation of each equipment of the customer.



Use the robot within the environment given in the specifications. Failure to do so could lead to a drop or reliability or faults. (Temperature, humidity, atmosphere, noise environment, etc.)



Transport the robot with the designated transportation posture. Transporting the robot in a non-designated posture could lead to personal injuries or faults from dropping.



Always use the robot installed on a secure table. Use in an instable posture could lead to positional deviation and vibration.



Wire the cable as far away from noise sources as possible. If placed near a noise source, positional deviation or malfunction could occur.



Do not apply excessive force on the connector or excessively bend the cable. Failure to observe this could lead to contact defects or wire breakage.



Make sure that the workpiece weight, including the hand, does not exceed the rated load or tolerable torque. Exceeding these values could lead to alarms or faults.



Securely install the hand and tool, and securely grasp the workpiece. Failure to observe this could lead to personal injuries or damage if the object comes off or flies off during operation.



Securely ground the robot and controller. Failure to observe this could lead to malfunctioning by noise or to electric shock accidents.



Indicate the operation state during robot operation. Failure to indicate the state could lead to operators approaching the robot or to incorrect operation.



When carrying out teaching work in the robot's movement range, always secure the priority right for the robot control. Failure to observe this could lead to personal injuries or damage if the robot is started with external commands.



Keep the jog speed as low as possible, and always watch the robot. Failure to do so could lead to interference with the workpiece or peripheral devices.



After editing the program, always confirm the operation with step operation before starting automatic operation. Failure to do so could lead to interference with peripheral devices because of programming mistakes, etc.

Make sure that if the safety fence entrance door is opened during automatic operation, the door is locked or that the robot will automatically stop. Failure to do so could lead to personal injuries.



Never carry out modifications based on personal judgments, or use non-designated maintenance parts.

Failure to observe this could lead to faults or failures.



When the robot arm has to be moved by hand from an external area, do not place hands or fingers in the openings. Failure to observe this could lead to hands or fingers catching depending on the posture.

CAUTION

Do not stop the robot or apply emergency stop by turning the robot controller's main power OFF.

If the robot controller main power is turned OFF during automatic operation, the robot accuracy could be adversely affected.

CAUTION

Do not turn off the main power to the robot controller while rewriting the internal information of the robot controller such as the program or parameters. If the main power to the robot controller is turned off while in automatic operation or rewriting the program or parameters, the internal information of the robot controller may be damaged.

DANGER

When the SSCNETIII cable is removed, install the cap in the connector.

If the cap is not installed, there is a possibility of malfunctioning by adhesion of the dust etc.

DANGER

Don't remove the SSCNETIII cable, when the power supply of the robot controller is turned on. Don't face squarely the light emitted from the tip of the SSCNETIII connector or the cable. If light strikes the eyes, there is a possibility of feeling the sense of incongruity for the eyes. (The light source of SSCNETIII is equivalent to the class 1 specified to JISC6802 and IEC60825-1.)

2 Explanation of functions

2.1 Operation panel (O/P) functions

(1) Description of the operation panel button

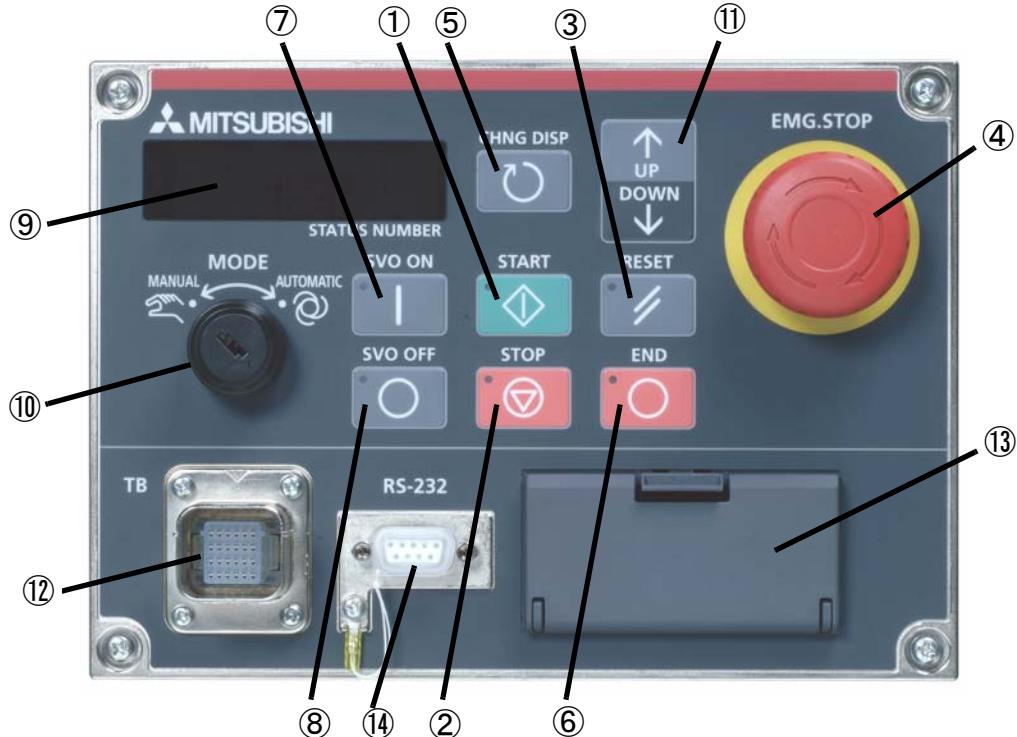


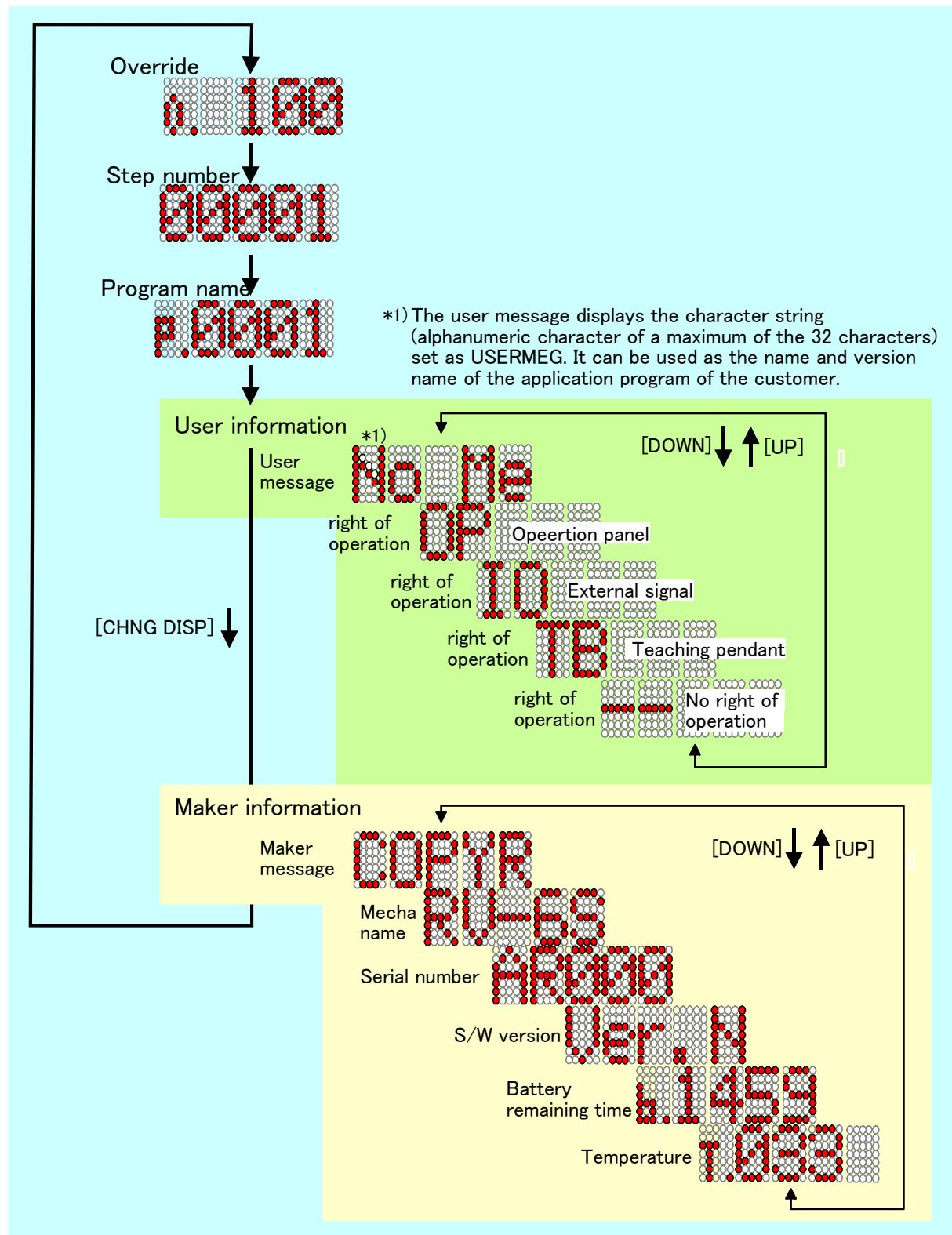
Fig.2-1:Operation panel

- ① START button..... This executes the program and operates the robot. The program is run continuously.
- ② STOP button..... This stops the robot immediately. The servo does not turn OFF.
- ③ RESET button..... This resets the error. This also resets the program's halted state and resets the program.
- ④ Emergency stop switch This stops the robot in an emergency state. The servo turns OFF.
- ⑤ CHNGDISP button..... This changes the details displayed on the display panel in the order of "Override" → "Program No." → "Line No.".
- ⑥ END button..... This stops the program being executed at the last line or END statement.
- ⑦ SVO.ON button..... This turns ON the servo power. (The servo turns ON.)
- ⑧ SVO.OFF button..... This turns OFF the servo power. (The servo turns OFF.)
- ⑨ STATUS NUMBER
(display panel)..... The alarm No., program No., override value (%), etc., are displayed.
- ⑩ MODE key switch..... This changes the robot's operation mode.
AUTOMATIC..... operations from the controller or external equipment are valid. Operations for which the operation mode must be at the external device or T/B are not possible. It is necessary to set the parameter for the rights of operation to connection between the operation panel and external equipment. For details, please refer to "INSTRUCTION MANUAL/Detailed explanations of functions and operations" of the separate volume.
- MANUAL..... When the T/B is valid, only operations from the T/B are valid. Operations for which the operation mode must be at the external device or controller are not possible.
- ⑪ UP/DOWN button..... This scrolls up or down the details displayed on the "STATUS. NUMBER" display panel.
- ⑫ T/B connection connector This is a dedicated connector for connecting the T/B. When not using T/B, connect the attached dummy connector.
- ⑬ Interface cover USB interface and battery are mounted. Unused in CRnQ-700 series
- ⑭ RS-232 connector This is an RS-232C specification connector for connecting the personal computer. Not installed in the CRnQ-700 series

(2) Description of the STATUS NUMBER

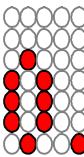
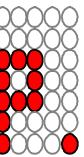
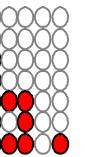
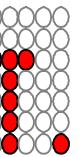
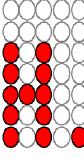
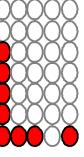
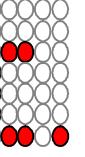
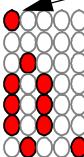
1) Display change of STATUS NUMBER

The display of the display panel can be changed by [CHNG DISP], [\uparrow UP], and [\downarrow DOWN] key.



2) The various status displays

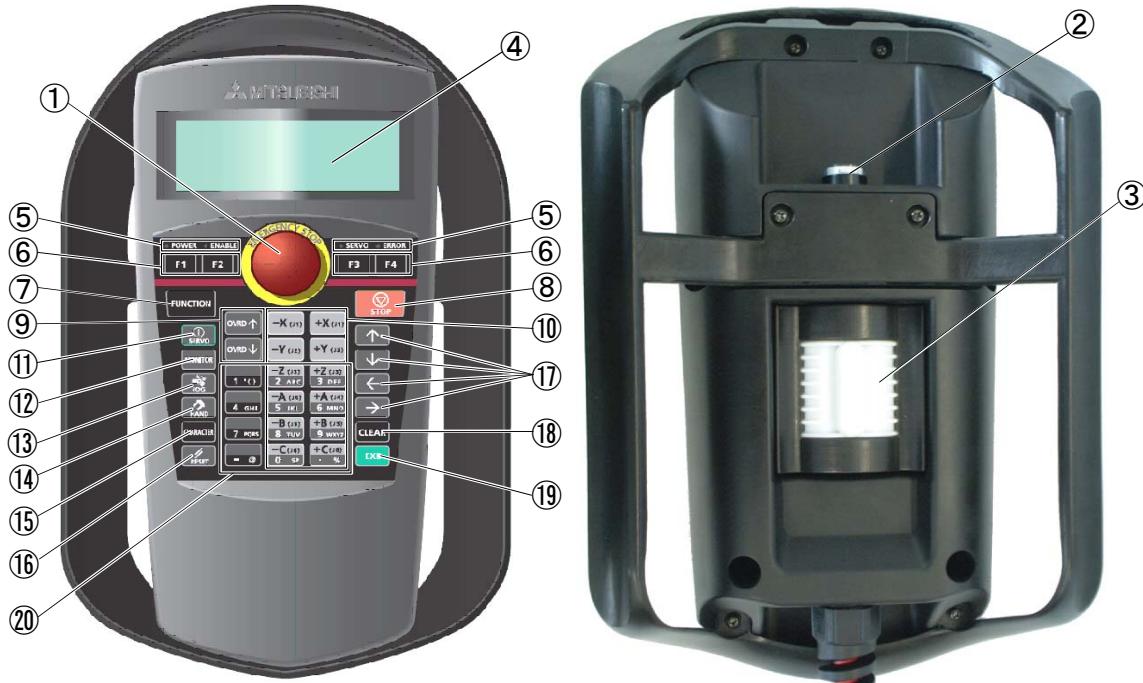
The various states are indicated by the character at left end.

Override	Program name	Battery remaining time	Temperature
			
High level error	Low level error	Warning	
			
O/P operation rights	If the operation panel has the right of operation, the upper left dot turns on.		
			

2.2 Teaching pendant (T/B) functions

This chapter explains the functions of R32TB (optional).

(1) Function of each key



- 1) : [Emergency stop] switch The robot servo turns OFF and the operation stops immediately.
- 2) : [Enable/Disable] switch This switch changes the T/B key operation between enable and disable.
- 3) : [Enable] switch When the [Enable/Disable] switch "2)" is enabled, and this key is released or pressed with force, the servo will turn OFF, and the operating robot will stop immediately.
- 4) : LCD display panel The robot status and various menus are displayed.
- 5) : Status display lamp Display the state of the robot or T/B.
- 6) : [F1], [F2], [F3], [F4] Execute the function corresponding to each function currently displayed on LCD.
- 7) : [FUNCTION] Change the function display of LCD.
- 8) : [STOP] key This stops the program and decelerates the robot to a stop.
- 9) : [OVRD ↑][OVRD ↓] key Change moving speed. Speed goes up by [OVRD ↑] key. Speed goes down by [OVRD ↓] key
- 10) : JOG operation key Move the robot according to jog mode. And, input the numerical value.
- 11) : [SERVO] key Press this key with holding [Enable] switch lightly, then servo power will turn on.
- 12) : [MONITOR] key It becomes monitor mode and display the monitor menu.
- 13) : [JOG] key It becomes jog mode and display the jog operation.
- 14) : [HAND] key It becomes hand mode and display the hand operation.
- 15) : [CHAR] key This changes the edit screen, and changes between numbers and alphabetic characters.
- 16) : [RESET] key This resets the error. The program reset will execute, if this key and the EXE key are pressed.
- 17) : [↑][↓][←][→] key Moves the cursor each direction .
- 18) : [CLEAR] key Erase the one character on the cursor position .
- 19) : [EXE] key Input operation is fixed. And, while pressing this key, the robot moves when direct mode.
- 20) : Number/Character key Erase the one character on the cursor position . And, inputs the number or character

Fig.2-2:General-view

2.2.1 Operation rights

Only one device is allowed to operate the controller (i.e., send commands for operation and servo on, etc.) at the same time, even if several devices, such as T/Bs or PCs, are connected to the controller. This limited device "has the operation rights".

Operations that start the robot, such as program start and error reset, and operations that can cause starting require the operation rights. Conversely, operation that stop the robot, such as stopping and servo OFF, can be used without the operation rights for safety purposes.

Table 2-1: Relation of setting switches and operation rights ○ :Has operation rights, X:Does not have operation rights

Setting switch	T/B [ENABLE/DISBLE]	DISABLE		ENABLE	
		Controller [MODE]	AUTOMATIC	MANUAL	AUTOMATIC
Operation rights	T / B	X	X	X ^{Note 2)}	○
	Controller operation panel	○ Note 1)	X	X ^{Note 2)}	X
	Personal computer	○ Note 1)	X	X ^{Note 2)}	X
	External signal	○ Note 1)	X	X ^{Note 2)}	X

Note 1) When the "operation right input signal (IOENA)" is input from an external device, the external signal has the operation rights, and the personal computer's operation rights are disabled.

Note 2) If the [MODE] switch is set to "AUTOMATIC" when the T/B is set to "ENABLE", the error 5000 will occur.

Table 2-2: Operations requiring operation rights Operation item: ○ =Requires operation rights, X= Does not require operation rights

Class	Operation rights	Operation
Operation	○	Servo ON
	X	Servo OFF
	○	Program stop/cycle stop
	X	Slot initialization (program reset)
	○	Error reset
	X	Override change. Note this is always possible from the T/B.
	○	Override read
	X	Program No. change
	○	Program No./line No. read
	X	Program stop/cycle stop
Input/output signal	X	Input/output signal read
	X	Output signal write
	○	Dedicated input start/reset/servo ON/brake ON/OFF/manual mode changeover/general-purpose output reset/program No. designation/line No. designation/override designation
	X	Dedicated input stop/servo OFF/continuous cycle/ operation rights input signal/ program No.output request/line No. output request/override output request/error No. request, numeric input
	X	Hand input/output signal read
	○	Hand output signal write
Program editing Note1)	X	Line registration/read/call; Position addition/correction/read; Variable write/read
	○	Step feed/return, execution
	X	Step up/down
	○	Step jump, direct execution, jog
File operation	X	Program list read/protection setting/copy/delete/rename/ initialization
Maintenance operation	X	Parameter read, clock setting/read, operation hour meter read, alarm history read
	○	Origin setting, parameter change

Note1) When one device is being used for editing on-line, editing from other devices is not possible.

2.3 Functions Related to Movement and Control

This controller has the following characteristic functions.

Function	Explanation	Explanation page
Optimum speed control	This function prevents over-speed errors as much as possible by limiting the speed while the robot is tracking a path, if there are postures of the robot that require the speed to be limited while moving between two points. However, the speed of the hand tip of the robot will not be constant if this function is enabled.	Page 251, "Spd (Speed)"
Optimum acceleration/deceleration control	This function automatically determines the optimum acceleration/deceleration time when the robot starts to move or stops, according to the weight and center of gravity settings of the hand, and the presence of a workpiece. The cycle time improves normally, although the cycle time decreases by the condition..	Page 230, "Mxt (Move External)" , Page 214, "Loadset (Load Set)"
XYZ compliance	With this function, it is possible to control the robot in a pliable manner based on feedback data from the servo. This function is particularly effective for fitting or placing workpieces. Teaching along the robot's orthogonal coordinate system is possible. However, depending on the workpiece conditions, there are cases where this function may not be used.	Page 168, "Cmp Tool (Composition Tool)"
Impact Detection	The robot stops immediately if the robot's tool or arm interferes with a peripheral device, minimizing damage. This function can be activated during automatic operation as well as during jog operation. Note) Please note that this function cannot be used together with the multi-mechanism control function.	Page 175, "ColChk (Col Check)" Refer to "COL" parameter in Page 347, "5 Functions set with parameters" .
Maintenance Forecast	The maintenance forecast function forecasts the robot's battery, belt and grease maintenance information based on the robot's operating status. This function makes it possible to check maintenance information using the optional Personal Computer Support software. Note) Please note that this function cannot be used together with the multi-mechanism control function.	Use optional Personal Computer Support software.
Position Restoration Support	The position restoration support function calculates the correction values of OP data, tools and the robot base by only correcting a maximum of several 10 points if a deviation in the joint axis, motor replacement, hand deformation or a deviation in the robot base occurs, and corrects position deviation. This function is implemented by optional Personal Computer Support software.	Use optional Personal Computer Support software. Vertical multi-joint robot:
Continuous path control	This function is used to operate the robot between multiple positions continuously without acceleration or deceleration. This function is effective to improvement of the cycle time.	Page 92, "(4) Continuous movement" , Page 172, "Cnt (Continuous)"
Multitask program operation	With this function, it is possible to execute programs concurrently by grouping between programs for the robot movement, programs for communication with external devices, etc. It is effective to shorten input/output processing. In addition, it is possible to construct a PLC-less system by creating a program for controlling peripheral jigs.	Refer to X*** instructions such as Page 117, "4.2.1 What is multitasking?" , Page 262, "XRun (X Run)" .
Program constant execution function	With this function, it is possible to execute a program all the time after the controller's power is turned on. This function is effective when using the multitask functions to make the robot program serve as a PLC.	Refer to "SLTn" parameter start attribute (ALWAYS) in Page 347, "5 Functions set with parameters" .
Continuity function	With this function, it is possible to store the status at power off and resume from the same status when the power is turned on again.	Refer to "CTN" parameter in Page 347, "5 Functions set with parameters" .
Additional axis control	With this function, it is possible to control up to two axes as additional axes of the robot. Since the positions of these additional axes are stored in the robot's teaching data as well, it is possible to perform completely synchronous control. In addition, arc interpolation while moving additional axes (travelling axes) is also possible. The additional axis interface card optional is required of CR1/CR2 series controller.	Separate manual "ADDITIONAL AXIS INTERFACE".
Multi-mechanism control	With this function, it is possible to control up to two (excluding the standard robots) robots (user mechanism) driven by servo motors, besides the standard robots.	Separate manual "ADDITIONAL AXIS INTERFACE".

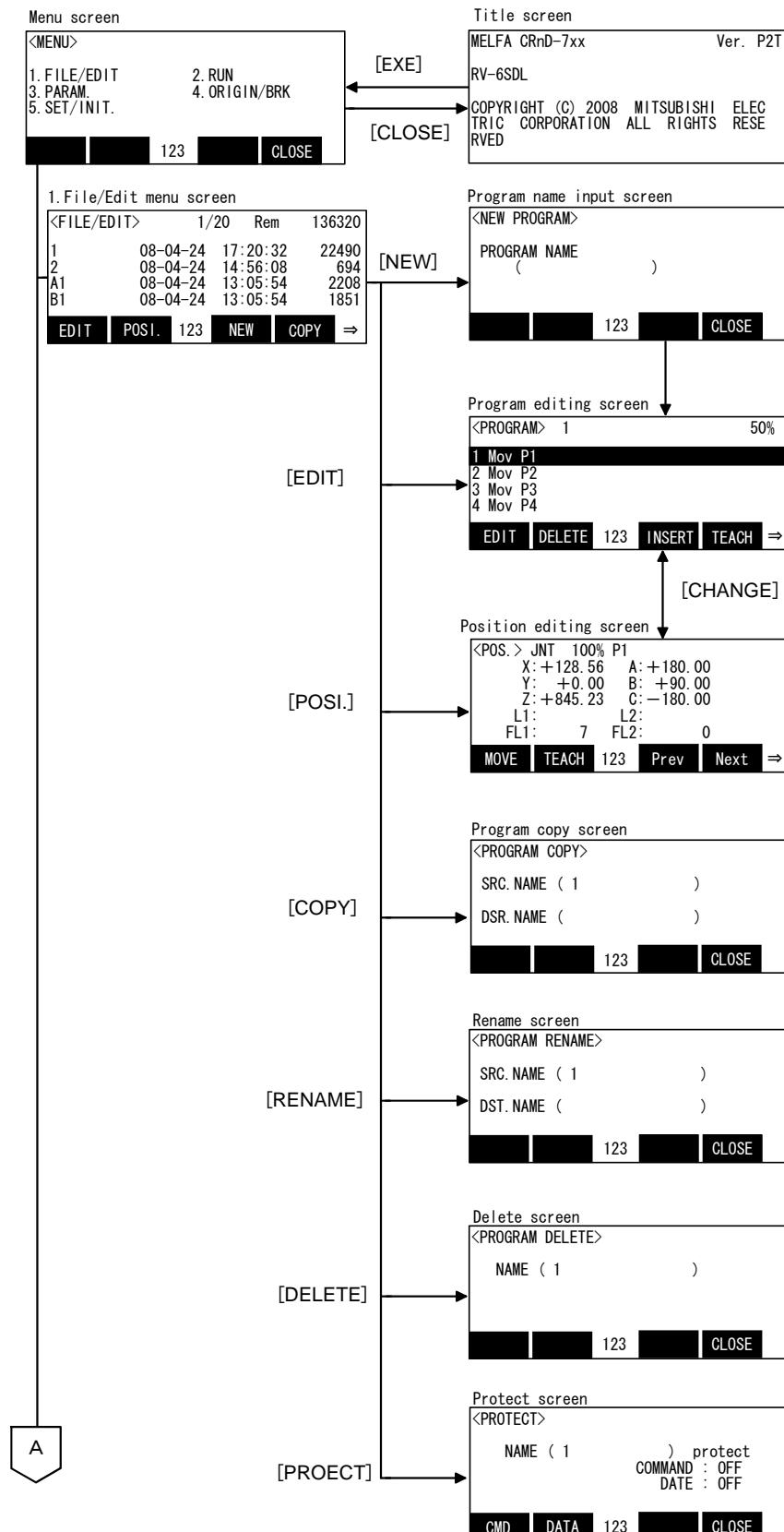
Function	Explanation	Explanation page
External device communication function	<p>The following methods are available for communicating with the external devices</p> <p>For controlling the controller and for interlock within a program</p> <ol style="list-style-type: none"> 1) Via input/output signals (CRnQ: PLC link input/output : 8192/8192 max.) (CRnD: Parallel input/output : 256/256 max.) 2) Via CC-Link (optional) <p>As a data link with an external device</p> <ol style="list-style-type: none"> 3) Communication via RS-232C (1 standard port) 4) Communication via Ethernet <p>The data link refers to a given function in order to exchange data, for instance amount of compensation, with external devices (e.g., vision sensors).</p>	<p>Refer to Page 285, "M_In/M_Inb/M_Inw", Page 292, "M_Out/M_Outb/M_Outw".</p> <p>Page 381, "5.15 About the communication setting(RS-232)" Separate manual "Ethernet Interface".</p>
Interrupt monitoring function	With this function, it is possible to monitor signals, etc. during program operation, and pause the current processing in order to execute an interrupt routine if certain conditions are met. It is effective for monitoring that workpieces are not dropped during transport.	Page 180, "Def Act (Define act)" , Page 156, "Act (Act)"
Inter-program jump function	With this function, it is possible to call a program from within another program using the CallP instruction.	Page 159, "CallP (Call P)"
Pallet calculation function	This function calculates the positions of workpieces arranged in the grid and glass circuit boards in the cassette. It helps to reduce the required teaching amount. The positions can be given in row-by-column format, single row format, or arc format.	Page 98, "4.1.2 Pallet operation" , Page 190, "Def Plt (Define pallet)" , Page 238, "Plt (Pallet)"
User-defined area function	With this function, it is possible to specify an arbitrary space consisting of up to 32 areas, monitor whether the robot's hand tip is within these areas in real time, output the status to an external device, and check the status with a program, or use it to generate an error. Moreover, two functions (Zone and Zone2) that have a similar function are available for use in a robot program.	<p>Page 372, "5.8 About user-defined area",Page 300, "M_Uar".</p> <p>Page 344, "Zone", Page 345, "Zone 2" Page 346, "Zone3"</p>
JOINT movement range XYZ operation range Free plane limit	<p>It is possible to restrict the robot movement range in the following three ways</p> <p>JOINT movement range: It is possible to restrict the movement range of each axis.</p> <p>XYZ operation range: It is possible to restrict the movement range using the robot's XYZ coordinate system.</p> <p>Free plane limit: It is possible to define an arbitrary plane and restrict the movement range of the robot to be only in front of or only behind the plane.</p>	<p>Refer to "MEJAR" and "MEPAR" parameter in Page 347, "5 Functions set with parameters"</p> <p>Refer to Page 373, "5.9 Free plane limit"</p>

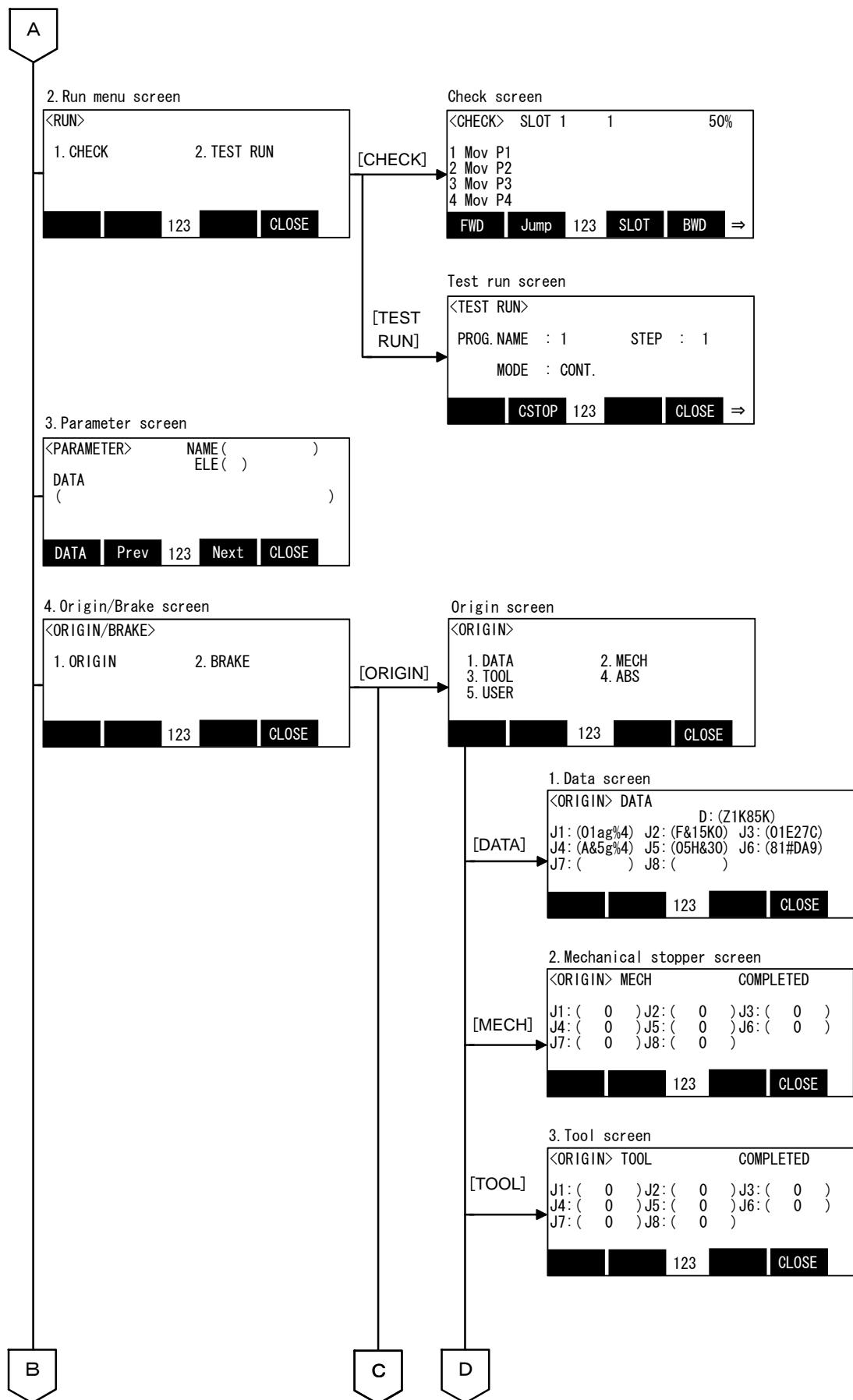
3 Explanation of operation methods

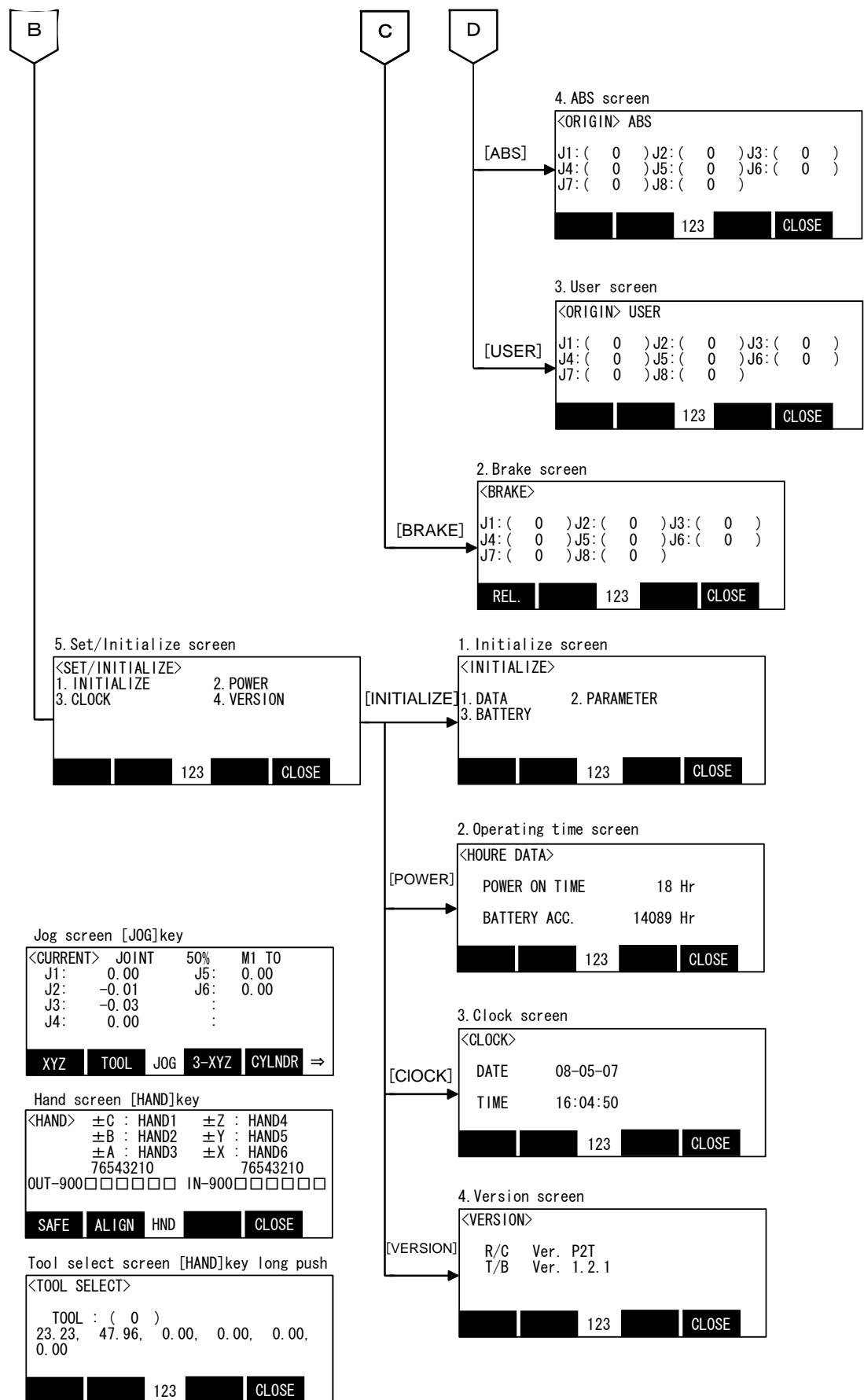
This chapter describes how to operate R32TB (optional)

3.1 Operation of the teaching pendant menu screens

(1) Screen tree







(2) Input of the number/character

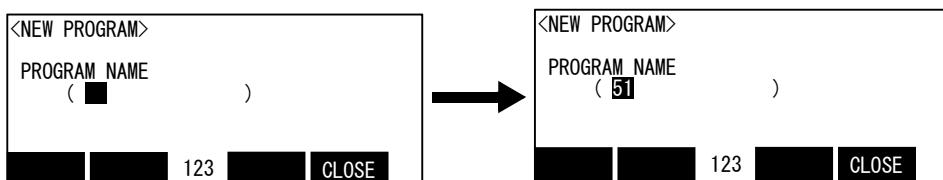
Each time the [CHARACTER] key is pressed, the number input mode and the character input mode change. The current input mode is displayed in the center under the screen, and the display of "123" shows that the number input mode and "ABC" is the character input mode.

1) Input the number

The number ("–" (minus) and ".") (decimal point) are included can be inputted if the key currently displayed on the lower left of each key is pressed.

Press the [CHARACTER] key, and in the condition that "123" is displayed on the screen lower side, press the number key.

Ex.) If "51" is inputted into the program name.



Input the number [CHARACTER] [5] [1]

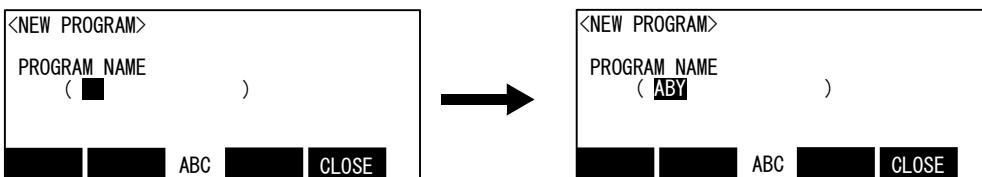
2) Input the character

The character is displayed on the lower right of each key. The character can be inputted if the key is pressed. Press the [CHARACTER] key, and in the condition that "ABC" is displayed on the screen lower side, press the character key. Whenever the key as which two or more characters are displayed presses the key, it changes the input character.

Ex.) The [ABC] key : "A" "B" "C" "a" "b" "c" · · · · · It repeats.

If it continues and inputs the character currently displayed on the same key, once press the [→] key and advance the cursor.

Ex.) If it inputs "ABY", push the [ABC], [→], [ABC] twice, [WXYZ] 3 times.



Input the character [CHARACTER] [ABC] [→] [ABC] [ABC] [WXYZ] [WXYZ] [WXYZ]

It comes out to input the character which is not displayed on the key. The character currently assigned to the key is shown below.

a) [' ()] key ' → (→) → " → ^ → : → ; → ¥ → ?

b) [@ =] key @ → = → + → - → * → / → < → >

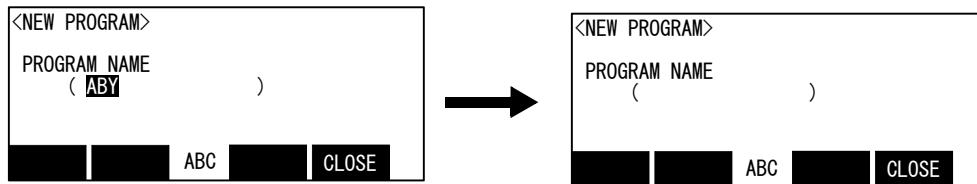
c) [, %] key , → % → # → \$ → ! → & → _ → .

3) Delete the character

The character mistaken and inputted will delete the character in the position of the cursor, if the [CLEAR] key is pressed.

Ex.) If "B" of "ABY" is changed into "M" and it is made "AMY".

Move the cursor to character "B", and input "M" and "Y" after pressing and deleting the [CLEAR] key.



Correction of the input character [←] [CLEAR] [MNO] [WXYZ] [WXYZ] [WXYZ]

If the long pushing [CLEAR] key, all the data in the parenthesis can be deleted.

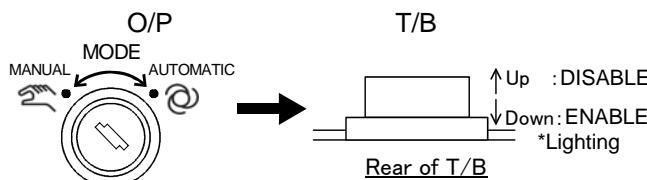
(3) Selecting a menu

A menu can be selected with either of the following two methods.

*Press the number key for the item to be selected.

*Move the cursor to the item to be selected, and press the [EXE] key.

How to select the Management/edit screen ("1. Management/edit") from the menu screen with each method is shown below.

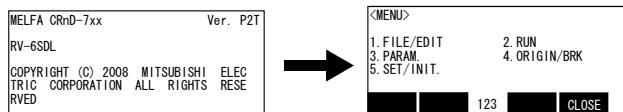


1) Set the controller [MODE] switch to "MANUAL".

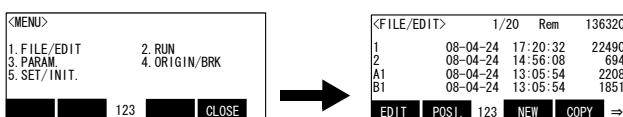
2) Set the T/B to "ENABLE".

3) Press one of the keys (example, [EXE] key) while the <TITLE> screen is displayed. The <MENU> screen will appear.

Display the MENU screen from the title screen.

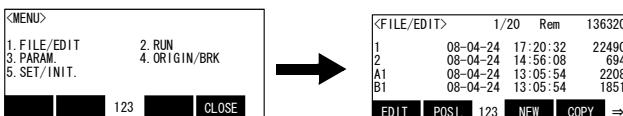


*Press the number key method



1) Press the [1] key. The <Management/edit> screen will appear.

*Use the arrow key method



1) Press the arrow keys and move the cursor to "1. Management/edit", and then press the [EXE] key. The <Management/edit> screen will appear.

Move the cursor - set [↑][↓][←][→]+EXE

The same operations can be used on the other menu screens.

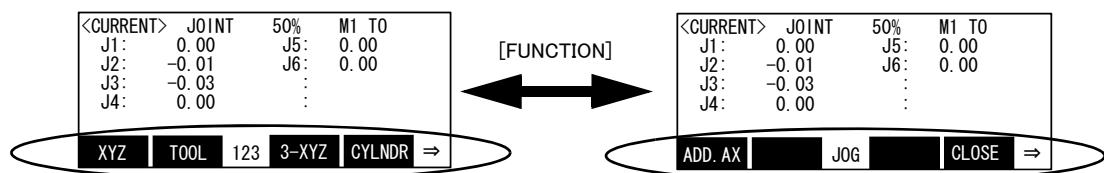
◆◆◆ Using the T/B ◆◆◆

Unless the controller [MODE] switch is set to "MANUAL", operations other than specific operations (current position display on JOG screen, changing of override, monitoring of input/output, error history) cannot be carried out from the T/B.

◆◆◆ Function key ◆◆◆

There is the menu displayed on the lowest stage of the screen in the white character. These are assigned to [F1], [F2], [F3], and [F4] key sequentially from the left. The menu currently displayed by pressing the corresponding function key can be selected.

And, if "=>" is displayed at the right end of the menu, it is shown that there is still the menu other than the current display, and whenever it presses the [FUNCTION] key, the display menu changes.



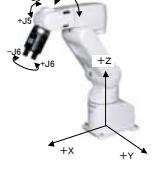
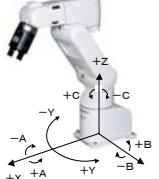
3.2 Jog Feed (Overview)

Jog feed refers to a mode of operation in which the position of the robot is adjusted manually. Here, an overview of this operation is given, using the vertical multi-joint type robot as an example. The axes are configured differently depending on the type of robot. For each individual type of robot, please refer to separate manual: "ROBOT ARM SETUP & MAINTENANCE," which provides more detailed explanations.

3.2.1 Types of jog feed

The following five types of jog feed are available

Table 3-1:Types of jog feed

Type	Operation	Explanation
JOINT jog	 <ol style="list-style-type: none"> 1) Set the key switch to the [ENABLE] position. 2) Hold the enable lightly. 3) Press the [SERVO] key. (The servo is turned on.) 4) Press the [JOG], [F1] key to change to the JOINT jog mode. 5) Press the key corresponding to each of the axes from J1 to J6. 	<p>In this mode, each of the axes can be adjusted independently. It is possible to adjust the coordinates of the axes J1 to J6 as well as the additional axes J7 and J8 independently. Note that the exact number of axes may be different depending on the type of robot, however.</p> <p>The additional axis keys [J1] and [J2] correspond to axes J7 and J8, respectively.</p>
TOOL jog	 <ol style="list-style-type: none"> 1) Perform steps 1) to 3) above. 4) Press the function key to change to the TOOL jog mode. 5) Press the key corresponding to each of the axes from X,Y,Z,A,B,C. 	<p>The position can be adjusted forward/backward, left/right, or upward/downward relative to the direction of the hand tip of the robot (the Tool coordinate system). The tip moves linearly. The posture can be rotated around the X, Y, and Z axes of the Tool coordinate system of the hand tip by pressing the A, B, and C keys, without changing the actual position of the hand tip. It is necessary to specify the tool length in advance using the MEXTL parameter.</p> <p>The Tool coordinate system, in which the hand tip position is defined, depends on the type of robot. In the case of a vertical multi-joint type robot, the direction from the mechanical interface plane to the hand tip is +Z.</p> <p>In the case of a horizontal multi-joint type robot, the upward direction from the mechanical interface plane is +Z.</p>
XYZ jog	 <ol style="list-style-type: none"> 1) Perform steps 1) to 3) above. 4) Press the function key to change to the XYZ jog mode. 	<p>The axes are adjusted linearly with respect to the robot coordinate system.</p> <p>The posture rotates around the X, Y, and Z axes of the robot coordinate system by pressing the A, B, and C keys, without changing the actual position of the hand tip. It is necessary to specify the tool length in advance using the MEXTL parameter.</p>
3-axis XYZ jog	 <ol style="list-style-type: none"> 1) Perform steps 1) to 3) above. 4) Press the function key twice to switch to the 3-axis XYZ jog mode. 	<p>The axes are adjusted linearly with respect to the robot coordinate system.</p> <p>Unlike in the case of XYZ jog, the posture will be the same as in the case of the J4, J5, and J6 axes JOINT jog feed. While the position of the hand tip remains fixed, the posture is interpolated by X, Y, Z, J4, J5, and J6; i.e., a constant posture is not maintained. It is necessary to specify the tool length in advance using the MEXTL parameter.</p>
CYLINDER jog	 <ol style="list-style-type: none"> 1) Perform steps 1) to 3) above. 4) Press the function key twice to switch to the CYLINDER jog mode. 	<p>Use the cylindrical jog when moving the hand in the cylindrical direction with respect to the robot's origin. Adjusting the X-axis coordinate moves the hand in the radial direction from the center of the robot. Adjusting the Y-axis coordinate moves the hand in the same way as in JOINT jog feed around the J1 axis. Adjusting the Z-axis coordinate moves the hand in the Z direction in the same way as in XYZ jog feed.</p> <p>Adjusting the coordinates of the A, B, and C axes rotates the hand in the same way as in XYZ jog feed. They may be valid in horizontal 4-axis (or 5-axis) RH type robots.</p>

If the robot's control point comes near a singular point during the operation of TOOL jog, XYZ jog or CYLINDER jog mode among the types of jog feed listed in [Table 3-1](#), a warning mark is displayed on the T/B

screen together with the sound of buzzer to warn the operator. It is possible to set this function valid or invalid by parameter MESNGLSW. (Refer to [Page 347, "5 Functions set with parameters".](#)) Please refer to [Page 391, "5.19 About the singular point adjacent alarm"](#) for details of this function.

3.2.2 Speed of jog feed

The current speed (%) is displayed on the screen. To change these values, press either the [OVRD ↑] or [OVRD ↓] key. The following types of jog feed speed are available.

[[OVRD ↑] key ----- [OVRD ↓] key			
LOW	HIGH	3%	5%

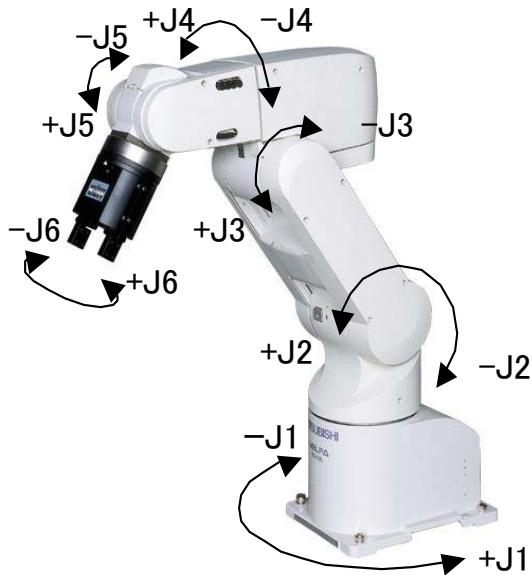
LOW and HIGH are fixed-dimension feed. In fixed-dimension feed, the robot moves a fixed amount every time the key is pressed. The amount of movement depends on the individual robot.

Table 3-2:Fixed-dimension of RV-6SD

	JOINT jog	TOOL, XYZ jog
LOW	0.01 deg.	0.01 mm
HIGH	0.10 deg.	0.10 mm

3.2.3 JOINT jog

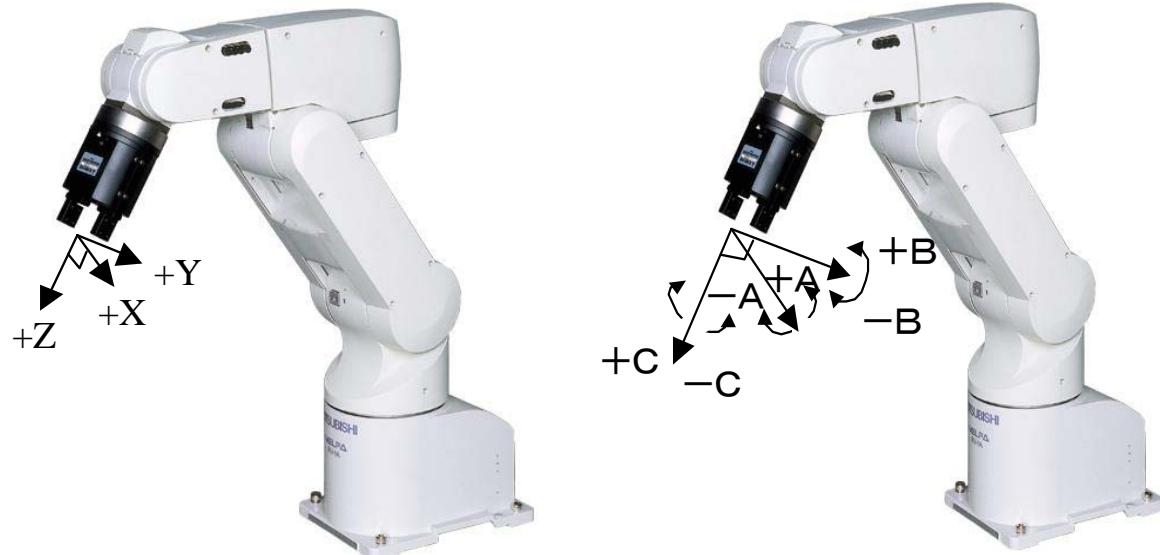
Adjusts the coordinates of each axis independently in angle units.



3.2.4 TOOL jog

Adjusts the coordinates of each axes along the direction of the hand tip.

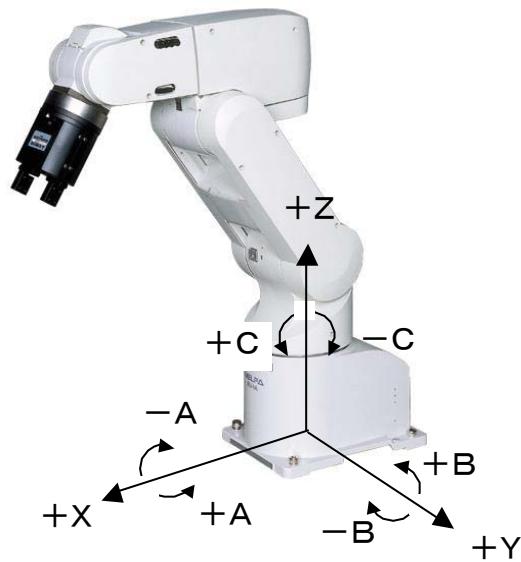
The X, Y, and Z axis coordinates are adjusted in mm units. The A, B, and C axis coordinates are adjusted in angle units.



3.2.5 XYZ jog

Adjusts the axis coordinates along the direction of the robot coordinate system.

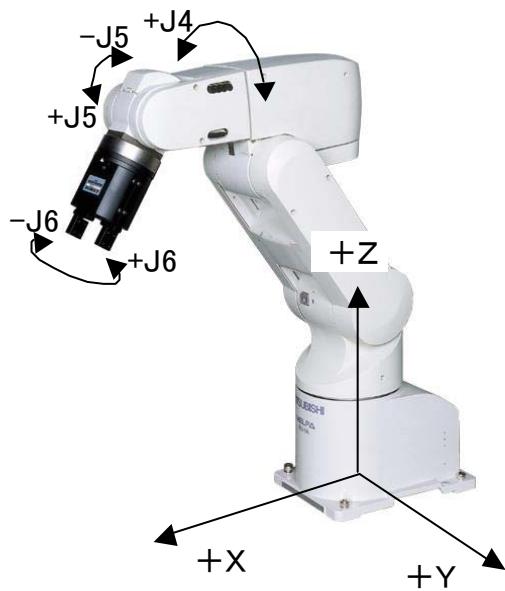
The X, Y, and Z axis coordinates are adjusted in mm units. The A, B, and C axis coordinates are adjusted in angle units.



3.2.6 3-axis XYZ jog

Adjusts the X, Y, and Z axis coordinates along the direction of the robot coordinate system in the same way as in XYZ jog feed. The J4, J5 and J6 axes perform the same operation as in JOINT jog feed, but the posture changes in order to maintain the position of the control point (X, Y and Z values).

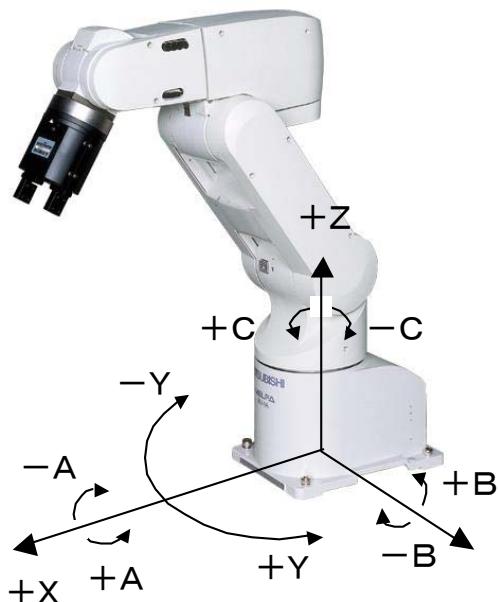
The X, Y, and Z axis coordinates are adjusted in mm units. The J4, J5, and J6 axis coordinates are adjusted in angle units.



3.2.7 CYLINDER jog

Adjusting the X-axis coordinate moves the hand in the radial direction away from the robot's origin. Adjusting the Y-axis coordinate rotates the arm around the J1 axis. Adjusting the Z-axis coordinate moves the hand in the Z direction of the robot coordinate system. Adjusting coordinates of the A, B, and C axes moves the hand in the same way as in XYZ jog feed.

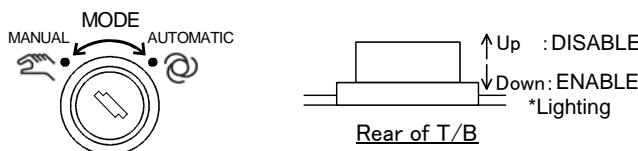
The X and Z axis coordinates are adjusted in mm units. The Y, A, B, and C axis coordinates are adjusted in angle units.



3.2.8 Switching Tool Data

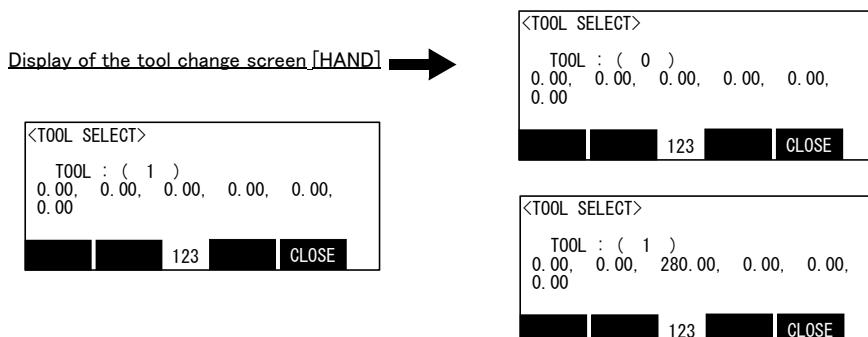
Set the tool data you want to use in the MEXTL1 to 4 parameters, and select the number of the tool you want to use according to the following operation.

- 1) Push the [ENABLE] switch of T/B and enable T/B.



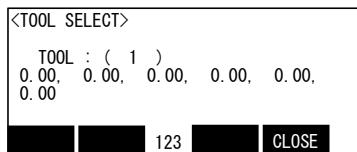
- 2) Long press the [HAND] key, and display the <tool change> screen.

3) If the number key to wish is pressed and the [EXE] key is pressed, tool data will change. MEXTL1~4 of the parameter corresponds to 1~4 of the number.



The change of tool data [1] ~ [4] [EXE]

- 4) Press the function key assigned for "closing" and finish.



Completed [F4]

- 5) The current tool number (T1~T4) is displayed on the upper right of the jog screen.

⚠ CAUTION

To move the robot to the position where teaching was performed while switching tool data (MEXTL1 to 4 parameters) during the automatic operation of the program, substitute the M_Tool variable by a tool number when needed, and operate the robot by switching tool data. Exercise caution as the robot moves to an unexpected direction if the tool data during teaching does not match the tool number during operation.

⚠ CAUTION

To move the robot while switching tool data during the step operation of the program, exercise caution as the robot moves to an unexpected direction if the tool data at the time of teaching does not match the tool number during step operation.

◇◆◇ Verifying the Tool Number ◇◆◇

The current tool number can be checked on the <TOOL SETTING> screen, JOG screen, or with the M_Tool variable.

◇◆◇ Related Information ◇◆◇

MEXTL, MEXTL1, MEXTL2, MEXTL3 and MEXTL4 parameters

Tool instruction, M_Tool variable

The MEXTL parameter holds tool data at that point. When using the MEXTL1 to 4 parameters, be careful as the MEXTL parameter is overwritten once a tool number is selected.

Execute the Tool instruction to return the tool number to 0.

3.2.9 Impact Detection during Jog Operation

This function can be enabled and disabled with a parameter. If the controller detects an impact, an error numbered 101n will be generated (the least significant digit, n, is the axis number). This function can also be enabled during jog operation; initial setting differs depending on the type.

Table 3-3:Impact detection parameters

Parameter	Name	No. of elements	Description	Initial value
Impact detection Note1)	COL	Integer 3	<p>Define whether the impact detection function can/cannot be used, and whether it is enabled/disabled immediately after power ON.</p> <p>Element 1: The impact detection function can (1)/cannot (0) be used.</p> <p>Element 2: It is enabled (1)/disabled (0) as the initial state during operation.</p> <p>Element 3: Enable (1)/disable (0)/NOERR mode (2) during jog operation</p> <p>The NOERR mode does not issue an error even if impact is detected. It only turns off the servo. Use the NOERR mode if it is difficult to operate because of frequently occurring errors when an impact is detected.</p> <p>The specification depends on the settings for jog operation (element 3) in cases other than program operation (including position jump and step feed).</p>	RV-SQ/SD series 0,0,1
Detection level during jog operation	COLLVLJG	Integer 8	<p>Set the detection level (sensitivity) during jog operation (including pause status) for each joint axis. Unit: %</p> <p>Make the setting value smaller to increase the detection level (sensitivity). If an impact error occurs even when no impact occurs during jog operation, increase a numeric value.</p> <p>Setting range: 1 to 500 (%)</p>	The standard is 200,200,200,200,200,200,200,200, but it varies with models.
Hand condition	HNDDAT0	Real value 7	<p>Set the initial condition of the hand. (Specify with the tool coordinate system.)</p> <p>Immediately after power ON, this set value is used during jog operation. To use the impact detection function during jog operation, set the actual hand condition before using. If it is not set, erroneous detection may occur. (Weight, size X, size Y, size Z, center of gravity X, center of gravity Y, center of gravity Z)</p> <p>Unit: Kg, mm</p>	<p>The value may vary with models. The maximum load is set as the load.</p> <p><Example of RV-3SD></p> <p>3.5,284.0,284.0,286.0,0.0,0.0,0.7,5.0</p>
Workpiece condition	WRKDAT0	Real value 7	<p>Set the initial condition of the workpiece. (Specify with the tool coordinate system.)</p> <p>Immediately after power ON, this setting value is used during jog operation.</p> <p>(Weight, size X, size Y, size Z, center of gravity X, center of gravity Y, center of gravity Z)</p> <p>Unit: Kg, mm</p>	<p>It is only released with the RV-SD and RH-SDH series.</p> <p>0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0</p>

Note1)This function cannot be used together with the multi-mechanism control function.

(1) Impact Detection Level Adjustment during Jog Operation

The sensitivity of impact detection during jog operation is set to a lower value. If higher impact sensitivity is required, adjust the COLVLJG parameter before use. Also, be sure to set the HNDDAT0 and WRKDAT0 parameters correctly before use. If a jog operation is carried out without setting these parameters correctly, erroneous detection may occur depending on the posture of the robot.

◆◆◆ Precaution for the Impact Detection Function ◆◆◆

Enabling the impact detection function does not completely prevent the robot, hand, workpiece and others from being damaged, which may be caused by interference with peripheral devices. In principle, operate the robot by paying attention not to interfere with peripheral devices.

◆◆◆ Operation after Impact ◆◆◆

If the servo is turned ON while the hand and/or arm is interfering with peripheral devices, the impact detection state occurs again, preventing the servo from being turned ON. If an error persists even after repeatedly turning ON the servo, release the arm by a brake release operation once and then turn ON the servo again. Or, release the arm by turning ON the servo according to the [Page 51, "Operation to Temporarily Reset an Error that Cannot Be Canceled"](#).

◆◆◆ Relationship with impact detection for automatic operation ◆◆◆

Settings of the impact detection function for jog operation and the impact detection function for automatic operation are independent. The setting for jog operation is used when the robot is not performing program operation. Even if the impact detection function for automatic operation is disabled in a program when the setting for jog operation is enabled, the setting is switched to that for jog operation (impact detection enabled) when the operation is paused.

3.3 Opening/Closing the Hands

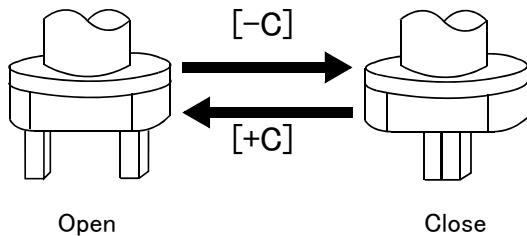
The open/close operation of the hands attached to on the robot is explained below.

Hands 1 to 6 can be opened and closed with the T/B.

<HAND>	$\pm C$: HAND1	$\pm Z$: HAND4
	$\pm B$: HAND2	$\pm Y$: HAND5
	$\pm A$: HAND3	$\pm X$: HAND6
	76543210	76543210
OUT-900	□□□□□□	IN-900□□□□□□

SAFE	ALIGN	HND	CLOSE
------	-------	-----	-------

Press the [HAND] key, and display the hand screen.



Opening and closing hand 1

Open: Press [+C] key

Close: Press [-C] key

Opening and closing hand 2

Open: Press [+B] key

Close: Press [-B] key

Opening and closing hand 3

Open: Press [+A] key

Close: Press [-A] key

Opening and closing hand 4

Open: Press [+Z] key

Close: Press [-Z] key

Opening and closing hand 5

Open: Press [+Y] key

Close: Press [-Y] key

Opening and closing hand 6

Open: Press [+X] key

Close: Press [-X] key

OUT-900 ~ OUT-907	7	6	5	4	3	2	1	0
Open/Close	Close	Open	Close	Open	Close	Open	Close	Open
Hand number	4		3		2		1	

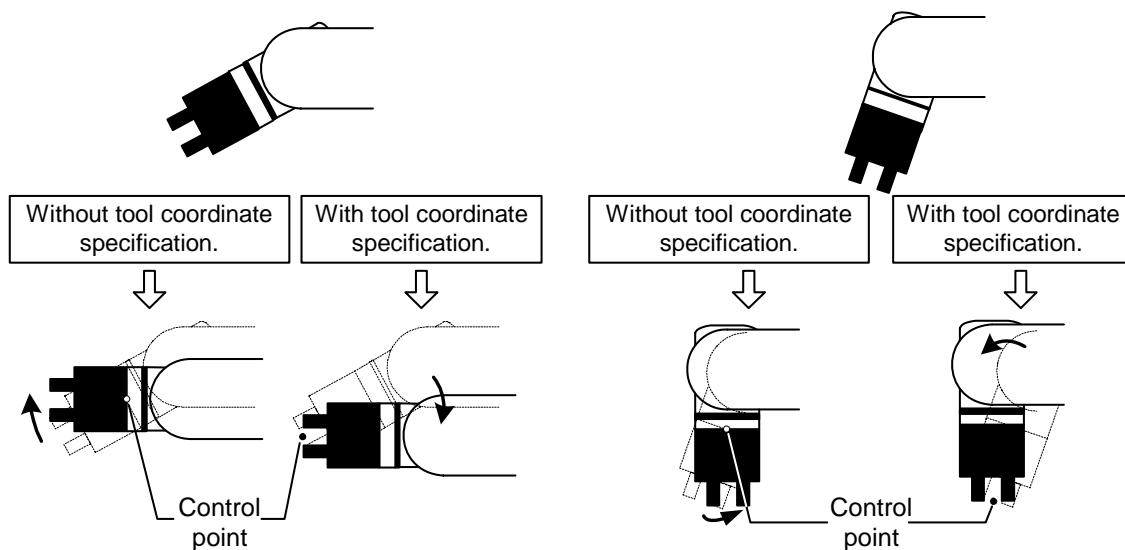
IN-900 ~ IN-907	7	6	5	4	3	2	1	0
Input signal	907	906	905	904	903	902	901	900

It is possible to mount various tools on the robot's hand area. In the case of pneumatic control, where the solenoid valve (at double solenoid) is used, two bits of the hand signal is controlled by the open/close operation of the hand. For more information about the hand signal, please refer to [Page 377, "5.12 About the hand type"](#) and [Page 378, "5.13 About default hand status"](#).

3.4 Aligning the Hand

The posture of the hand attached to the robot can be aligned in units of 90 degrees.

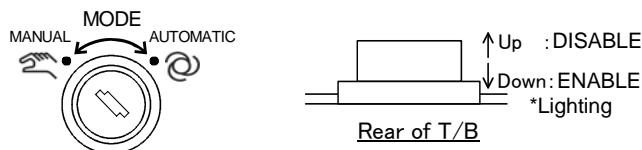
This feature moves the robot to the position where the A, B and C components of the current position are set at the closest values in units of 90 degrees.



If the tool coordinates are specified by the Tool instruction or parameters, the hand is aligned at the specified tool coordinates. If the tool coordinates are not specified, the hand is aligned at the center of the mechanical interface. The above illustration shows an example of a small vertical robot. [With Tool Coordinate Specification] indicates when the tool coordinates are specified at the tip of the hand. For more information about the tool coordinates, refer to [Page 369, "5.6 Standard Tool Coordinates"](#).

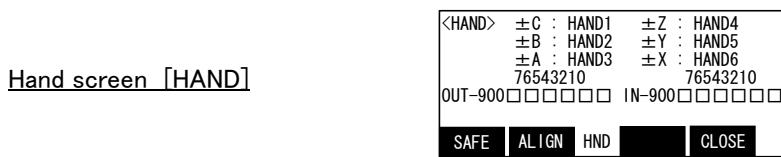
The hand alignment procedure is as follows:

- 1) Push the [ENABLE] switch of T/B and enable T/B.



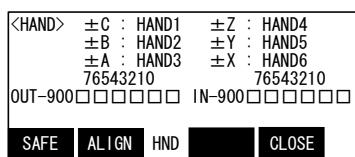
- 2) Press down the enabling switch (3 position switch), press the [SERVO] key and carry out servo-on.

- 3) Press the "HAND" key and display the <hand> screen.



- 4) Pressing the function key currently assigned to "alignment" is kept with the enabling switch (3 position switch) pressed down. While keeping pushing, the robot does hand alignment movement and [START] LED of the controller unit turns on during movement.

If either is detached in the middle of movement, the robot will stop.



Execution of hand alignment [Align]

 **CAUTION**

If any posture components (A, B and C) become 180 degrees as a result of aligning the hand, the component values can be either +180 degrees or -180 degrees even if the posture is the same. This is due to internal operation errors, and there is no consistency in which sign is employed. If the position is used as position data for the pallet definition instruction (Def Plt) and the same posture component values include both +180 degrees and -180 degrees, the hand will rotate and move in unexpected ways because the pallet operation calculates positions by dividing the distance between -180 degrees and +180 degrees. When using position data whose posture component values include 180 degrees for pallet definitions, use either + or - consistently for the sign of 180 degrees. Note that if the position data is used directly as the target position in an interpolation instruction, the hand moves without problem regardless of the sign.

3.5 Programming

MELFA-BASIC V used with this controller allows advanced work to be described with ample operation functions. The programming methods using the T/B are explained in this section. The functions shown in **Table 3-4** are used to input one line. (Refer to [Page 153, "4.11 Detailed explanation of command words"](#) in this manual for details on the MELFA-BASIC V commands and description methods.)

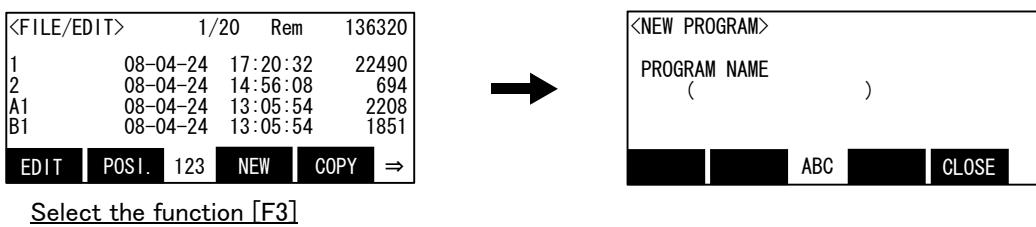
Table 3-4:Process for inputting one line

Input details	Function
Step No. and command statement (Example: 10 Mov P1)	Input as one line of the program
Only step No. (Example: 10)	Deletes designated line from program
Only command statement (Example: Mov P1)	Immediately executes that command (Direct execution)

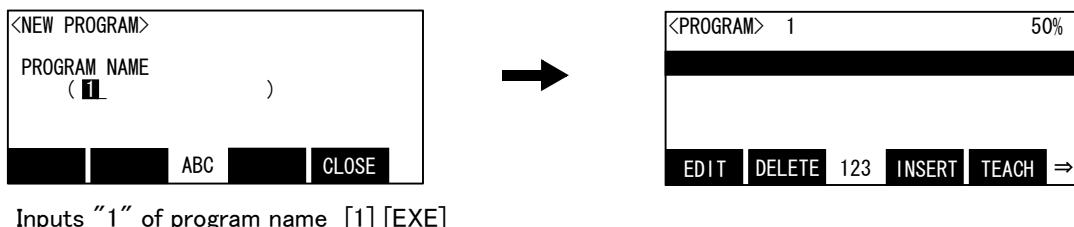
3.5.1 Creating a program

(1) Opening the program edit screen

- 1) Select "1. management / edit" screen on the <menu> screen.
- 2) Press the function key corresponding to "new." Display the program name input screen.



- 3) Input the program name. Display the command edit screen.
(Open the existing program, if the existing program name is inputted)

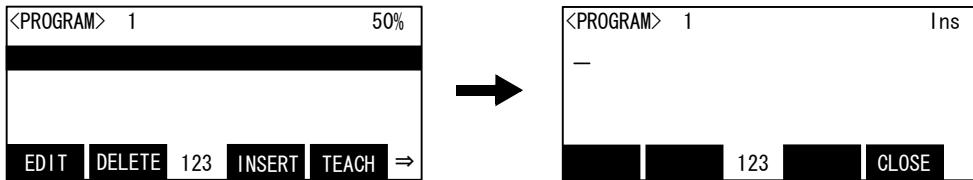


(2) Creating a program

The key operation in the case of inputting the program of the following and the three steps is shown.

- 1 Mov P1
- 2 Mov P2
- 3 End

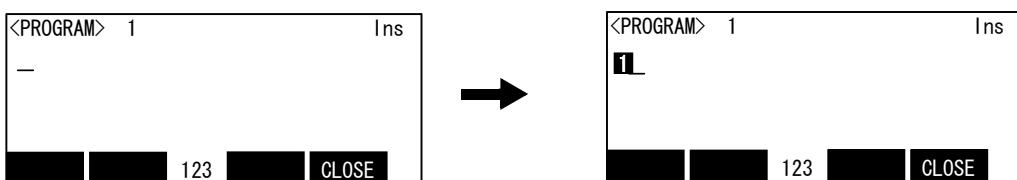
1) Press the function key ([F3]) corresponding to "insertion" in the command edit screen.



Step insertion [F3]

2) Input of step number "1."

Press the [CHARACTER] key, set it in the number input mode and press the [1] key.
The space between step number and command is ommissible.

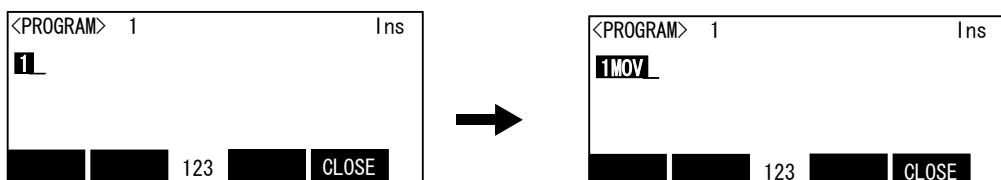


Step number input [1]

3) Input "Mov."

Press the [CHARACTER] key, set it in the character input mode

Press the [MNO] ("M"), [→], [MNO] ("o") 3 times, and [TUV] ("v") 3 times in order.



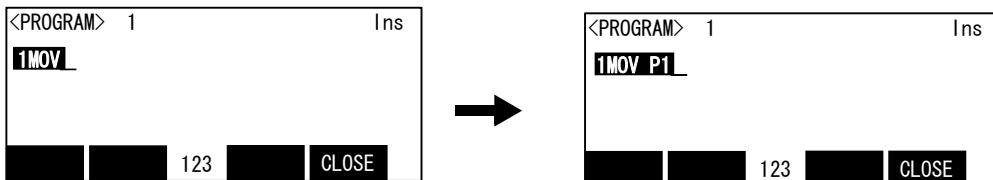
Input "MOV" [CHARACTER] [MNO] [→] [MNO] [MNO] [MNO] [TUV] [TUV] [TUV]

4) Input "P1."

Press the [SP] ("space"), [PQRS] ("P").

Press the [CHARACTER] key, set it in the number input mode and press the [1] key.

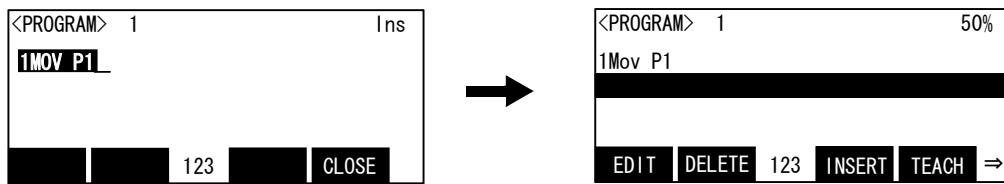
For the instruction word and the data which accompanies the command, the space is required.



Input "P1" [SP] [PQRS] [CHARACTER] [1]

5) Registration of Step 1

Press the [EXE] key and register the step 1.

Registration of Step 1 [EXE]

6) Hereafter, input Steps 2 and 3 in the same way.



The input of the program was completed above.

◆◆◆◆ Displaying the previous and next command step ◆◆◆◆

Display the four lines on the screen of T/B. For moving the cursor to the front line, the [↑] key is pressed, for moving the cursor to the next line, press the [↓] key, and select.

◆◆◆◆ Displaying a specific line ◆◆◆◆

Press the [FUNCTION] key, and change the function display, and press the [F2] key. The display changes to the JUMP screen. The specification line can be displayed, if the step number to display in the parenthesis is inputted and the [EXE] key is pressed.

◆◆◆◆ The step number can be omitted when inserting. ◆◆◆◆

It is inserted in the next of the cursor line if it omits.

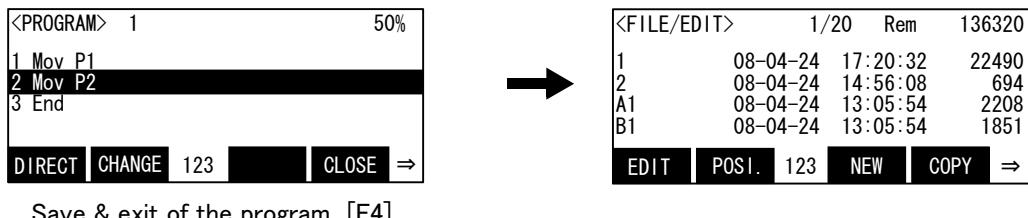
◆◆◆◆ The capital letter and the small letter are changed automatically. ◆◆◆◆

Display the reserved word and the variable name in MELFA BASIC V combining the capital letter and the small letter. Change automatically at the time of confirmation of the line also with the capital letter (with the small letter) at the time of the input from TB.

(3) Completion of program creation and saving programs

If the function key which corresponds for "closing" is pressed, the program will be saved and creation will be finished.

If the "close" is not indicated, press the [FUNCTION] key, and display it.



Left Screen (PROGRAM):

<PROGRAM> 1		50%
1	Mov P1	
2	Mov P2	
3	End	
DIRECT CHANGE 123		CLOSE ⇒

Right Screen (FILE/EDIT):

<FILE/EDIT>		1/20	Rem	136320
1	08-04-24	17:20:32	22490	
2	08-04-24	14:56:08	694	
A1	08-04-24	13:05:54	2208	
B1	08-04-24	13:05:54	1851	
EDIT POSI. 123		NEW	COPY	⇒

Save & exit of the program [F4]

◆◆◆◆ Precautions when saving programs ◆◆◆◆

Make sure to perform the operation above. The edited data will not be updated if the power is turned off without doing so after modifying a program on the program edit screen. Moreover, as much as possible, try to save programs not only on the controller but also on a PC in order to make backup copies of your work. It is recommended to manage programs using RT ToolBox 2 (optional).

(4) Correcting a program

Before correcting a program, refer to [Page 29, "3.5.1 Creating a program" in "\(1\)Opening the program edit screen"](#), and open the program edit screen.

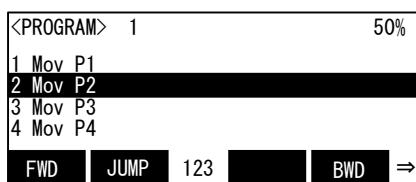
An example, change "5 Mov P5" to "5 Mvs P5".



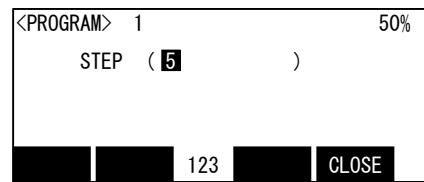
1) Display the step 5

Press the [FUNCTION] key and change the function display. Press the [F2](Jump) key and display the command edit screen. Press the [5], [EXE] key and display the 5th step.

Step 5 can be called even if it moves the cursor to Step 5 by the [↑] or [↓] key.



Call the step 5 [F2]



Call the step 5 [5] [EXE]

2) Correction of the instruction word.

Press the function key corresponding to "edit."

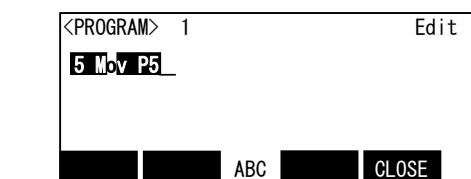
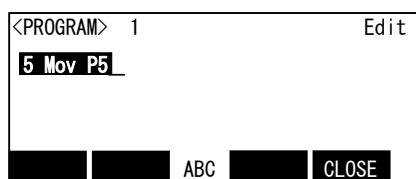


Correct the command 命令語の修正 [F1]

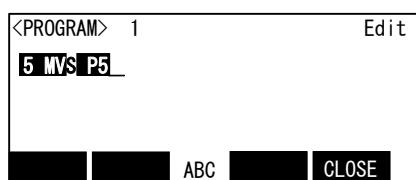


3) Press the [->] key 3 times. Move the cursor to "o."

Press the [CLEAR] key twice and delete "ov". Leave "M." Press the [TUV] key 3 times (input "v"), the [->] key, the [PQRS] key 4 times (input "s"). Then, 5 step is "Mvs P5". Press the [EXE] key, and register step 5.



Correct the command [TUV] [TUV] [TUV] [→]
[PQRS] [PQRS] [PQRS] [PQRS]



Correct the command [EXE]

◇◆◇ Select and correct the line. ◇◆◇

[↑] By the [↓] key, the cursor can be moved to Step 5, and the function key corresponding to "edit" can also be pressed and corrected to it.

◇◆◇ Cancel correction. ◇◆◇

Correction can be canceled if the function key which corresponded for "closing" is pressed in the middle of correction.

◇◆◇ Correction of the character ◇◆◇

Move the cursor to up to the mistaken character, and input the correct character after pressing the [CLEAR] key and deleting leftward.

◇◆◇ If the program is corrected ◇◆◇

If the program is corrected, certainly save. (Function key [F4 which correspond for "closing" are pushed, or push the [ENABLE] switch on the back of T/B, and disable T/B.) Please check that it has been correctly corrected by step operation about the details.

(5) Registering the current position data

Teach the position variable which moves the robot to the movement position by jog operation etc., and is using the position by the program (registration). It is overwritten if already taught (correction). There are the teaching in the command edit screen and the teaching in the position edit screen.

(a)Teaching in the command edit screen

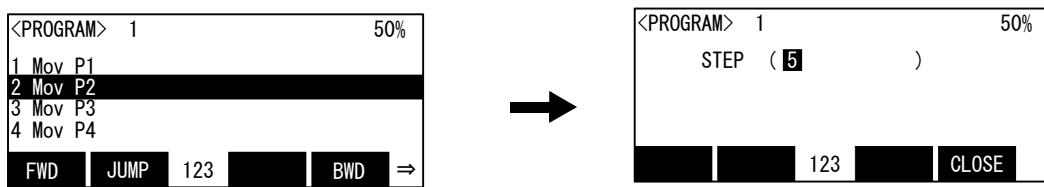
Call the step which is using the position variable to teach.

The operating procedure in the case of teaching the current position to the below to the position variable P5 of step 5 "Mov P5" is shown. Move the robot to the movement position by jog operation etc. beforehand.

1) Call the step 5

Press the function key corresponding to "JUMP", then step number input screen is displayed. Press the [5], [EXE] key, move cursor to step 5.

Step 5 can be called even if it moves the cursor to Step 5 by the [↑], [↓] key.

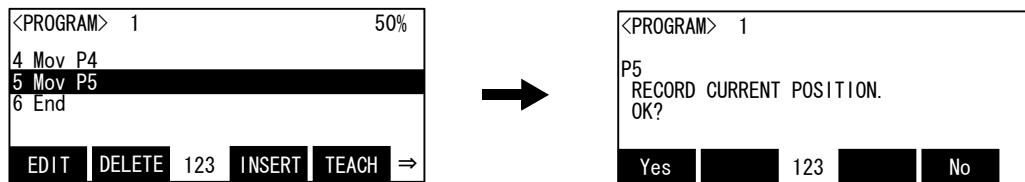


Call the step 5 [F2]

Call the step 5 [5] [EXE]

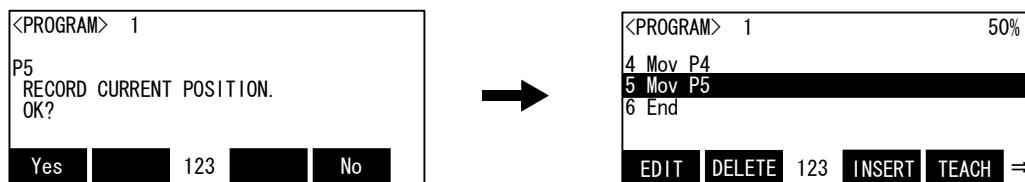
2) Teaching of the current position

Press the function key corresponding to "Teach"([F4]), then the confirmation screen is displayed.



Register the current position [F4]

3) Press the function key corresponding to "Yes", then the robot's current position data will be taught to P5, and display will return to the original command edit screen. The teaching can be canceled if the function key corresponding to "No" is pressed.



Register the current position [F1]

The teaching of the current position was completed above.

◆◆◆ Only one position variable is the target. ◆◆◆

If the read step is using two or more position variables, such as "Mov P1+P2" and "P1=P10", the position variable of most left-hand side is the target of the teaching.

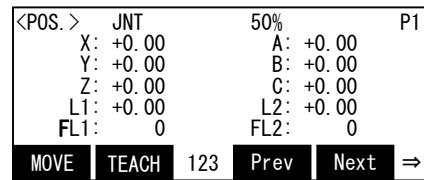
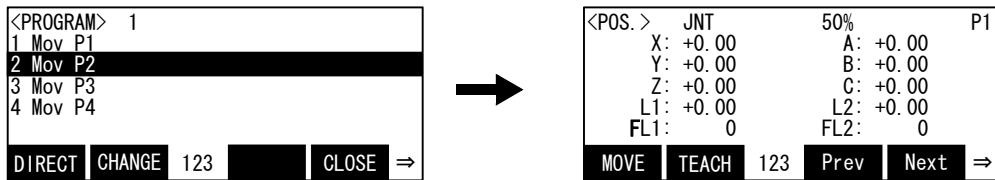
It is the following page if it teaches other variables. Refer to "(b)Teaching in the position edit screen."

(b)Teaching in the position edit screen

The operating procedure in the case of teaching the current position to the below to the position variable P5 is shown. Move the robot to the movement position by jog operation etc. beforehand.

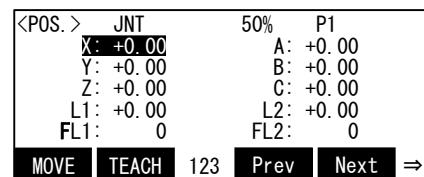
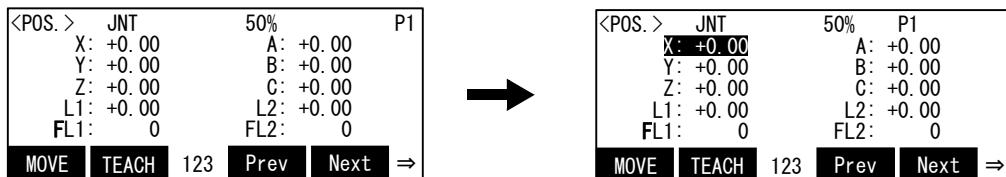
1) Teaching in the position edit screen

Press the function key ([F2]) corresponding to "the change", and display the position edit screen.



Display the current position [F2]

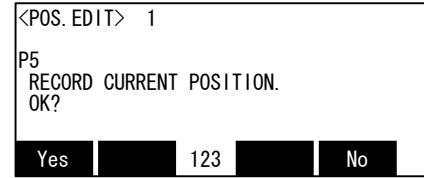
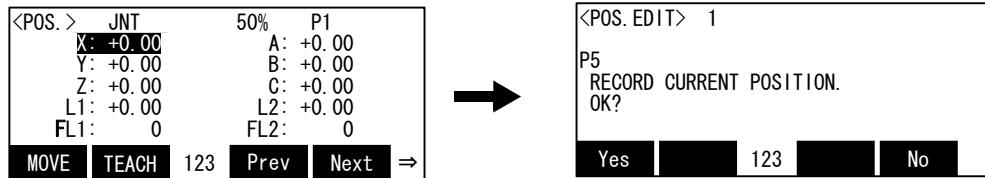
2) Press the function key corresponding to "Prev" and "Next", and call "P5".



Call the position 5 [F3] [F4]

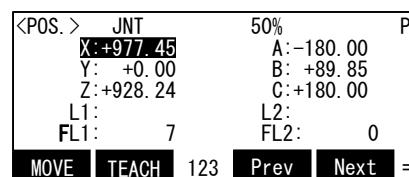
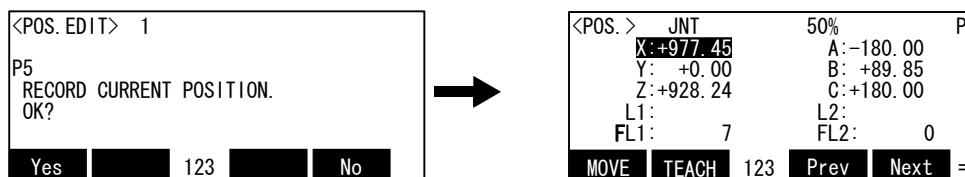
3) Teaching of the current position

Press the function key corresponding to "Teach"([F4]), then the confirmation screen is displayed.



Call the position 5 [F2]

4) Press the function key corresponding to "Yes", then the robot's current position data will be taught to P5, and display will return to the original position edit screen. The teaching can be canceled if the function key corresponding to "No" is pressed.



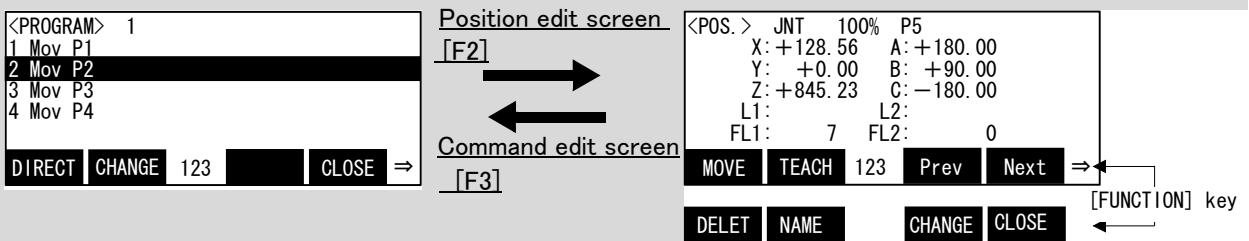
Register the current position [F1]

The teaching of the current position was completed above.

◆◆◆ Change of the command edit screen and the position edit screen ◆◆◆

If the function key corresponding to "the change" is pressed, the command edit screen and the position edit screen can be changed each other.

If the "change" is not displayed on the screen, it is displayed that the [FUNCTION] key is pressed. If "=>" is displayed at the right end of the menu, the state of changing the menu by pressing the [FUNCTION] key is shown.



The position variable of order can be called one by one by Prev (F3) and Next (F4). Usually, although it is the call of only the position variable, change the function key and the call can do the joint variable by the name (F2). After calling the joint variable, the joint variable of order can be called one by one by Prev (F3) and If it displays to the head or the last by the joint variable, it will return to the position variable by the next display.

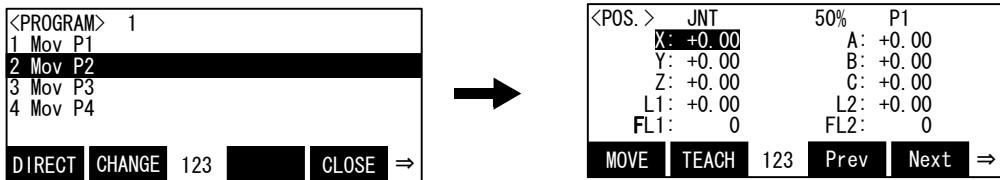
(6) Deletion of the position variable

The operating procedure which deletes the position variable is shown.

Restrict to the variable which is not used by the program and it can delete.

1) Display the position edit screen.

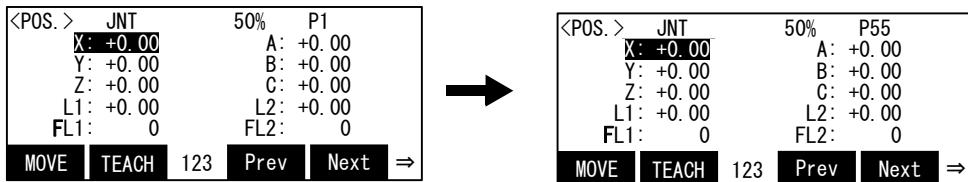
Press the function key corresponding to "Cange", and display the position edit screen.



Display the position edit screen [F2]

2) Display the position variable to delete.

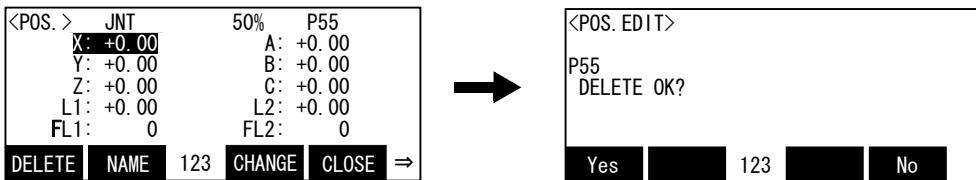
Press the function key corresponding to "Prev" and "Next", and display the position variable to delete.



Call the position 55 [F3] [F4]

3) Deletion of the position variable

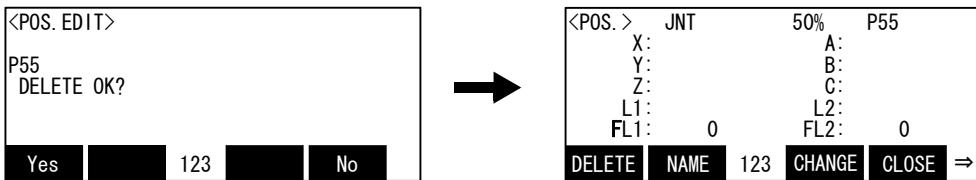
Press the function key corresponding to "Delete", then the confirmation screen is displayed.
(When "Delete" is not displayed, it is displayed that the [FUNCTION] key is pressed)



Delete the P55

4) Deletion of the position variable

Press the function key corresponding to "Yes", then the position variable is deleted.



(7) Confirming the position data (Position jump)

Move the robot to the registered position data place.

The robot can be moved with the "joint mode" or "XYZ mode" method.

Perform a servo ON operation while lightly holding the deadman switch before moving positions.

Table 3-5:Moving to designated position data

Name	Movement method
Joint mode	The robot moves with joint interpolation to the designated position data place. This moving method is used when the jog mode is JOINT jog. The axes are adjusted in the same way as with the Mov instruction.
XYZ mode	The robot moves with linear interpolation to the designated position data place. Thus, the robot will not move if the structure flag for the current position and designated position differ. This moving method is used when the jog mode is XYZ, 3-axis XYZ, CYLINDER or TOOL jog. The axes are adjusted in the same way as with the Mvs instruction.

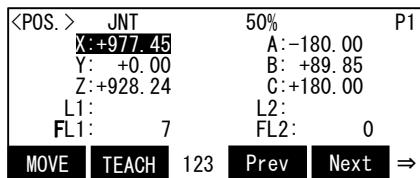
The operation method is shown in the following.

Do this operation by maintaining the servo-on state, carrying out servo-on and holding the enabling switch (3 position switch) lightly.

1) Display the position variable to make it move beforehand.

Press the function key corresponding to "Move", then move the robot to position which currently displayed variable, only while keeping pressing the key.

If the function key corresponding to "Move" is detached, the robot will stop. And, if the enabling switch (3 position switch) is detached or it presses down still more strongly, servo-off will be carried out and the robot will stop.



CAUTION

The robot moves by this operation.

When the robot moves, confirm not interfering with peripheral equipment etc. beforehand.

We recommend you to lower speed at first. And, also important to predicting the trajectory of the robot by moving mode (the joint, the XYZ) of operation.

(8) Correcting the MDI (Manual Data Input)

MDI is the method of inputting the numerical value into each axial element data of position data directly, and registering into it.

This is a good registration method for registration of the position variable which adds position data and is used as an amount of relative displacement from a reference position (difference), if it tunes registered position data finely.

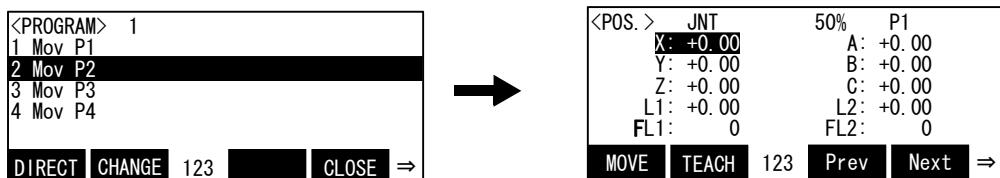
Reference)Position data as an amount of relative displacement

Ex.) In the case of move by joint interpolation to over 50mm from P1 of reference position, the P1 is registered by teaching.. And set "50.00" into Z-axis element, and set "0.00" to the other element by MDI. Then, executing the Mov P1+P50 is possible.

The operation method in the case of registering P50 of the above-mentioned example by MDI is shown.

1) Display the position edit screen.

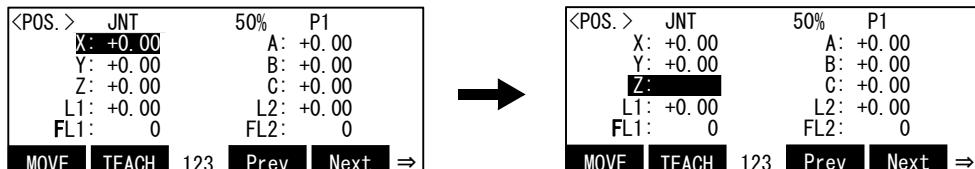
Press the function key corresponding to "Cange", and display the position edit screen.



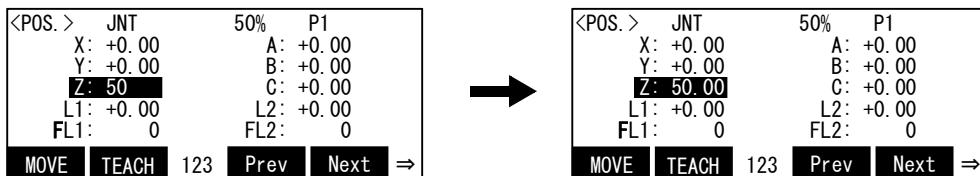
Display the position edit screen [F2]

2) Input "50.00" into Z-axis element

Press the [↓] key twice and move the cursor to the Z-axis. Press the [CLEAR] key, and delete "+0.00" currently displayed. Press [5], [0], and the [EXE] key. As for the position variable P50, only the value of the Z-axis is registered as the 50mm.



Clear the value [↓] [CLEAR]



Input 50 [5] [0] [EXE]

3.6 Debugging

Debugging refers to testing that the created program operates correctly, and to correcting an errors if an abnormality is found. These can be carried out by using the T/B's debugging function. The debugging functions that can be used are shown below. Always carry out debugging after creating a program, and confirm that the program runs without error.

(1) Step feed

The program is run one line at a time in the feed direction. The program is run in line order from the head or the designated line.

Confirm that the program runs correctly with this process.

Using the T/B execute the program line by line (step operation), and confirm the operation.

Display the edit screen of the program which is the target of debugging. Perform the following operations while pressing lightly on the enabling switch of the T/B after the servo has been turned on.

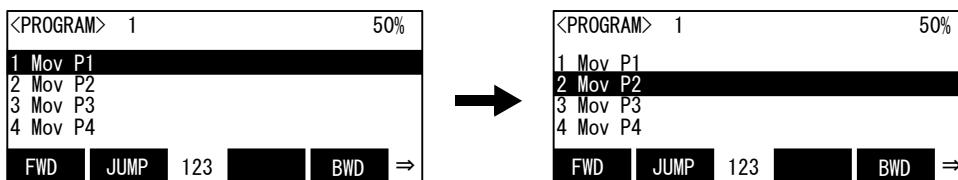
1) Execution of step feed

Press the [FUNCTION] key and change the function display. Pressing the [F1] (FWD) key is kept, and the robot will start moving.

When the execution of one line is completed, the robot will stop, and the next line will appear on the screen.

If [F1] (FWD) is released during this step, the robot will stop. And, detach the enabling switch (3 position switch), or push in still more strongly -- thing servo-off can be carried out and execution can be stopped.

During execution, the lamp on the controller's [START] switch will light. If execution of the one step is completed, LED of the [START] switch will go out and LED of the [STOP] switch will turn it on. If the [F1] key is detached, the cursor of the T/B screen will move to the following step.



Step feed [F1]

Whenever it presses the function key corresponding to "FWD", step to the following step.



CAUTION

Take special care to the robot movements during automatic operation. If any abnormality occurs, press the [EMG.STOP] switch and immediately stop the robot.

◆◆◆ About step operation ◆◆◆

"Step operation" executes the program line by line. The operation speed is slow, and the robot stops after each line, so the program and operation position can be confirmed.

During execution, the lamp on the controller's [START] switch will light. Execution of the End command or the Hlt command will not step feed any more.

◆◆◆ Change of the execution step ◆◆◆

The execution step can be changed by cursor movement by the arrow key, and jump operation ("JUMP").

◆◆◆ Immediately stopping the robot during operation ◆◆◆

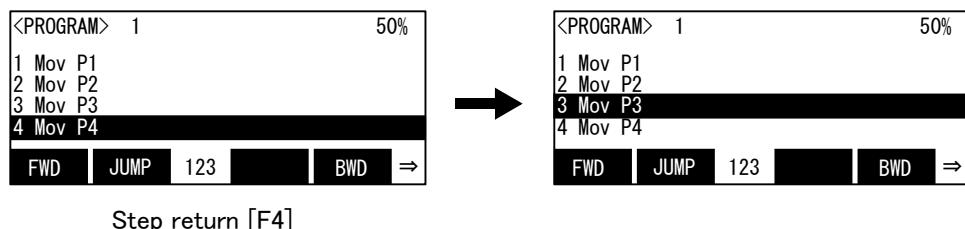
- Press the [EMG.STOP] (emergency stop) switch.
The servo will turn OFF, and the moving robot will immediately stop.
To resume operation, reset the alarm, turn the servo ON, and start step operation.
- Release or firmly press the "enable" switch.
The servo will turn OFF, and the moving robot will immediately stop.
To resume operation, lightly press the "enable" switch, and start step operation.
- Release the [F1] (FWD)key.
The step execution will be stopped. The servo will not turn OFF.
To resume operation, press the [F1] (FWD)key.

(2) Step return

The line of a program that has been stopped with step feed or normal operation is returned one line at a time and executed. This can be used only for the interpolation commands. Note that only up to four lines can be returned.

1) Execution of step return

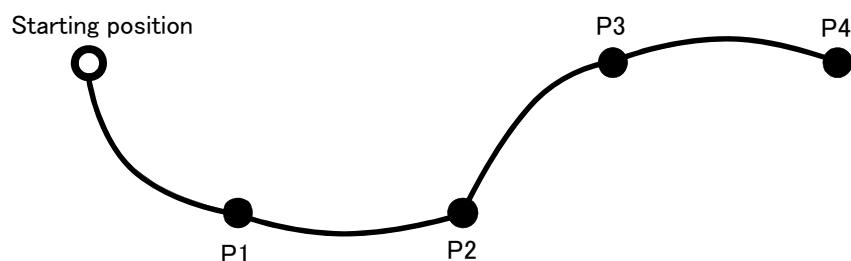
If the function key corresponding to "BWD" is pressed, only while keeping pushing, only the one step will be executed in the return direction of the step.
If the function key is released during this step, the robot will stop. And, release the enabling switch (3 position switch), or push in still more strongly, then the servo power off, and execution can be stopped. During execution, the lamp on the controller's [START] switch will light. If execution of the one step is completed, LED of the [START] switch will go out and LED of the [STOP] switch will turn it on. The cursor of the T/B screen moves to the step of the next interpolation command in the return direction of the step.



Whenever it presses the function key corresponding to "BWD", it returns to the front step.

[Supplement] If it does step return after carrying out the step feed of the following program to Step 4 and step return is further done after returning to P1, it will return to the position at the time of the start which did step feed.(The position at the time of the start is the position which began to execute Step 1.)

Program
1 Mov P1
2 Mov P2
3 Mov P3
4 Mov P4
:
:



CAUTION

Take special care to the robot movements during automatic operation. If any abnormality occurs, press the [EMG.STOP] switch and immediately stop the robot.

◆◆◆ Immediately stopping the robot during operation ◆◆◆

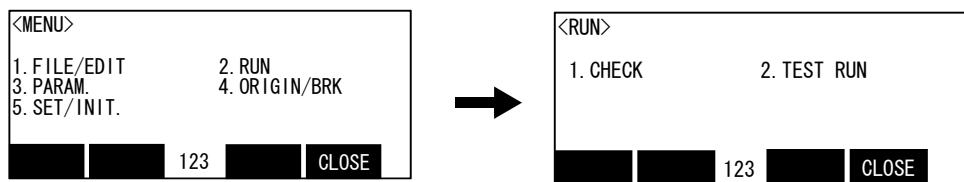
- Press the [EMG.STOP] (emergency stop) switch.
The servo will turn OFF, and the moving robot will immediately stop.
To resume operation, reset the alarm, turn the servo ON, and start step operation.
- Release or firmly press the "enable" switch.
The servo will turn OFF, and the moving robot will immediately stop.
To resume operation, lightly press the "enable" switch, and start step operation.
- Release the [F1] (FWD)key.
The step execution will be stopped. The servo will not turn OFF.
To resume operation, press the [F1] (FWD)key.

(3) Step feed in another slot

When checking a multitask program, it is possible to perform step feed in the confirmation screen of the operation menu, not in the edit screen.

1) Selection of the operation menu

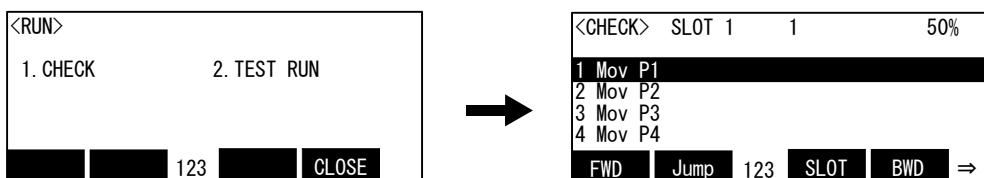
Press the [2] keys in the menu screen and select "2. Operation."



2) Selection of the confirmation screen

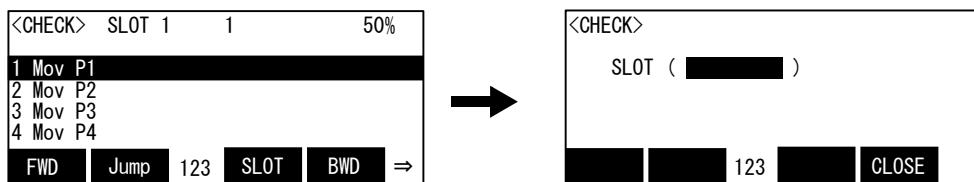
Press the [1] keys in the menu screen and select "1. Confirm."

Display the program set as the slot 1. The program name is displayed following the slot number.



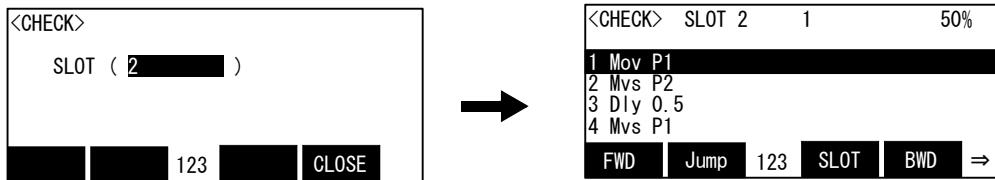
3) Change of the slot

Press the function key ([F3]) corresponding to the "slot" will display the slot number specified screen.



Input the slot number to wish and press the [EXE] key.

Display the inputted program of the slot number. (The following example specifies the slot 2)



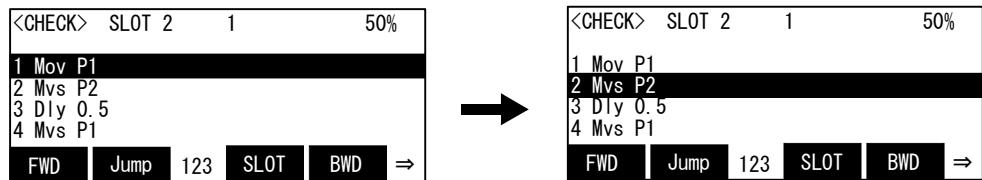
4) Execution of step operation

Step feed and step return can be executed like the step operation in the command edit screen.

Only while keeping pressing the function key, execute step feed and step return separately.

If the function key is released during this step, the robot will stop. And, detach the enabling switch (3 position switch), or push in still more strongly -- thing servo-off can be carried out and execution can be stopped.

During execution, the lamp on the controller's [START] switch will light. If execution of the one step is completed, LED of the [START] switch will go out and LED of the [STOP] switch will turn it on. If the [F1] key is detached, the cursor of the T/B screen will move to the following step.



CAUTION

Take special care to the robot movements during automatic operation. If any abnormality occurs, press the [EMG.STOP] switch and immediately stop the robot.

◆◆◆ Change of the execution step ◆◆◆

The execution step can be changed by cursor movement by the arrow key, and jump operation ("JUMP").

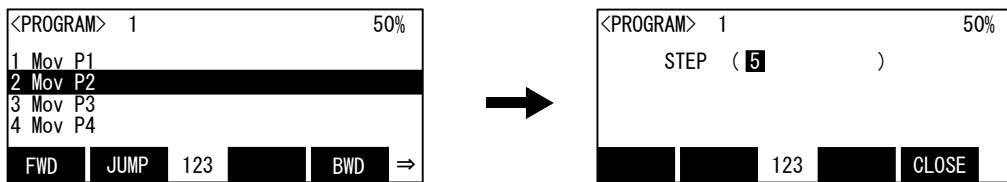
(4) Step jump

It is possible to change the currently displayed step or line.

The operation in the case of doing step operation from Step 5 as an example is shown.

1) Call Step 5.

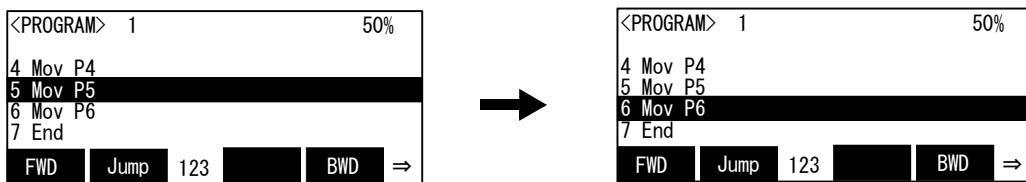
Press the function key corresponding to "JUMP", and press the [5], [EXE] key. The cursor moves to Step 5.



Step 5 can be called even if it moves the cursor to Step 5 by the [↑], [↓] key.

2) Execution of step feed

If the function key corresponding to "FWD" is pressed, step feed can be done from Step 5.



CAUTION

Take special care to the robot movements during automatic operation. If any abnormality occurs, press the [EMG.STOP] switch and immediately stop the robot.

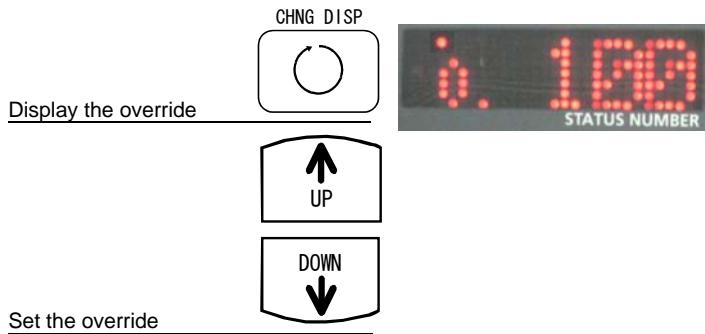
3.7 Automatic operation

(1) Setting the operation speed

The operation speed is set with the controller or T/B.

The actual speed during automatic operation will be the operation speed = (controller (T/B) setting value) x (program setting value).

*Operating with the controller



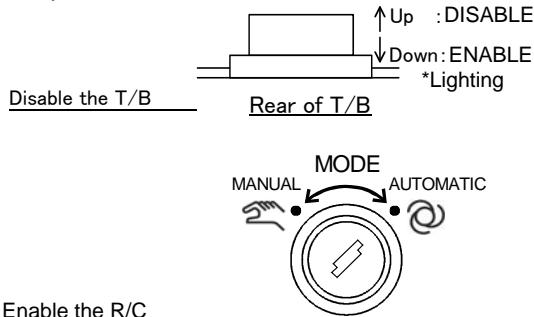
- 1) Press the controller [CHNG DISP] switch twice, and display the "OVERRIDE" on the STATUS NUMBER display panel.
- 2) Each time the [UP] key is pressed, the override will increase in the order of (10 - 20 - 30 - 40 - 50 - 60 - 70 - 80 - 90 - 100%). The speed will decrease in reverse each time the [DOWN] key is pressed.

*Operating with the T/B

Each time the [OVRD ↑] keys are pressed, the override will increase in the order of (LOW - HIGH - 3 - 5 - 10 - 30 - 50 - 70 - 100%). The speed will decrease in reverse each time the [OVRD ↓] keys are pressed.

(2) Selecting the program No.

Prepare the control



- 1) Set the T/B [ENABLE] switch to "DISABLE".

- 2) Set the controller [MODE] switch to "AUTOMATIC".

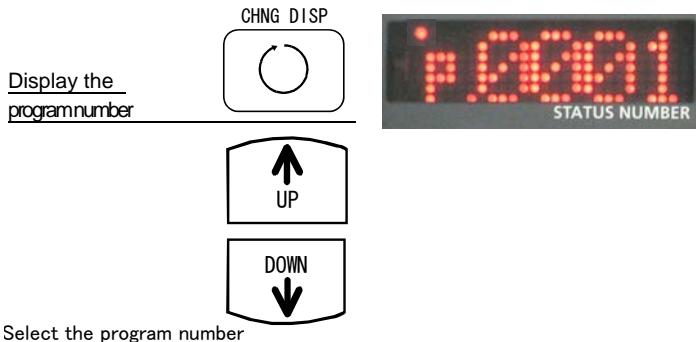
- 3) Press the [CHNG DISP] switch and display "PROGRAM NO." on the STATUS NUMBER display.

When the [UP] switch is pressed, the registered program Nos. will scroll up, and then the [DOWN] switch is pressed, the program Nos. will scroll down.

Display the program No. to be used for automatic operation.

*They are not displayed if a program name consisting of five or more characters is specified. If these are selected from an external device, "P ----" is displayed.

Select the program number

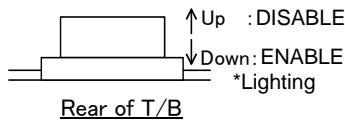


(3) Starting automatic operation

**CAUTION**

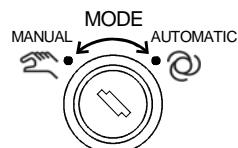
Before starting automatic operation, always confirm the following item. Starting automatic operation without confirming these items could lead to property damage or physical injury.

- Make sure that there are no operators near the robot.
- Make sure that the safety fence is locked, and operators cannot enter unintentionally.
- Make sure that there are no unnecessary items, such as tools, inside the robot operation range.
- Make sure that the workpiece is correctly placed at the designated position.
- Confirm that the program operates correctly with step operation.

Prepare the control

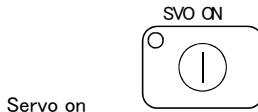
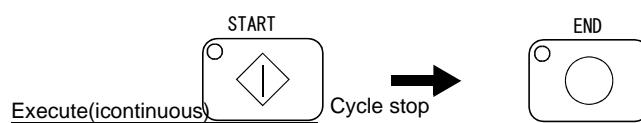
Disable the T/B

- 1) Set the T/B [ENABLE] switch to "DISABLE".



Enable the R/C

- 2) Set the controller [MODE] switch to "AUTOMATIC".

Servo onStart of automatic operation

- 3) Push the [SVO ON] switch of the controller, and servo power turn on.

- 4) Automatic operation will start when the controller [START] switch is pressed. (Continuous operation)
If the [END] switch is pressed during the continuous operation, the program will stop after one cycle. The LED blinks during the cycle stop.

**CAUTION**

Before starting automatic operation, always confirm that the target program No. is selected.

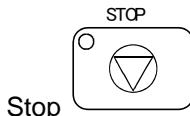
**CAUTION**

Take special care to the robot movements during automatic operation. If any abnormality occurs, press the [EMG. STOP] switch and immediately stop the robot.

(4) Stopping

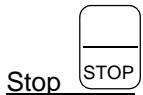
The running program is immediately stopped, and the moving robot is decelerated to a stop.

*Operating with the controller



1) Press the [STOP] switch.

*Operating with the T/B



1) Press the [STOP] key.

Operation rights not required

The stopping operation is always valid regardless of the operation rights.

(5) Resuming automatic operation from stopped state

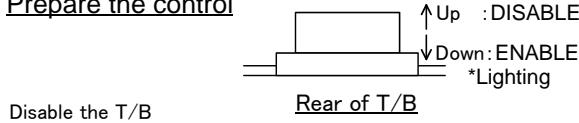


CAUTION

Before starting automatic operation, always confirm the following item. Starting automatic operation without confirming these items could lead to property damage or physical injury.

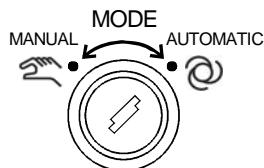
- Make sure that there are no operators near the robot.
- Make sure that the safety fence is locked, and operators cannot enter unintentionally.
- Make sure that there are no unnecessary items, such as tools, inside the robot operation range.
- Make sure that the workpiece is correctly placed at the designated position.
- Confirm that the program operates correctly with step operation.

Prepare the control



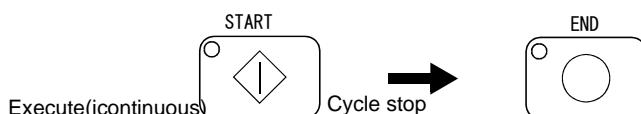
1) Set the T/B [ENABLE] switch to "DISABLE".

Enable the R/C



2) Set the controller [MODE] switch to "AUTOMATIC".

Restart

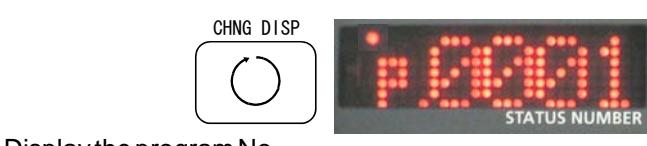
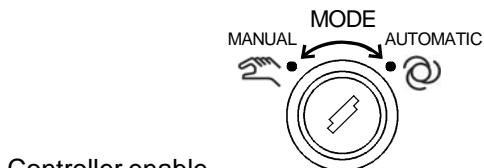
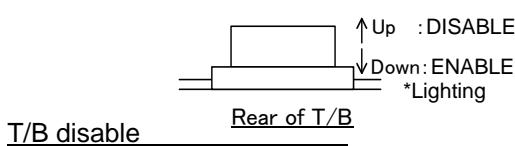


3) Automatic operation will start when the controller [START] switch is pressed.
Continuation operation / 1 cycle operation holds the former state.

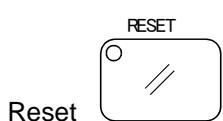
(6) Resetting the program

The program's stopped state is canceled, and the execution line is returned to the head.

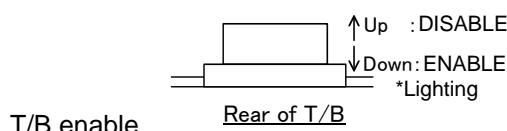
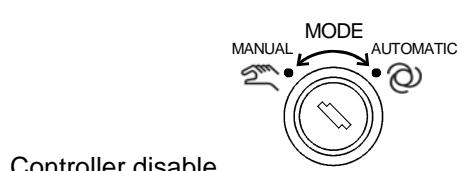
*Operating with the controller



Execute of program reset



*Operating with the T/B



Execute of program reset

Program reset [RESET] + [EXE]

Valid only while program is stopped

The program cannot be reset while the program is running. Always carry out this step while the program is stopped.

When resetting the program from the controller operation panel, display the "program No." on the STATUS NUMBER display, and then reset.

1) Set the T/B [ENABLE] switch to "DISABLE".

2) Set the controller [MODE] switch to "AUTOMATIC".

3) Press the controller [CHG DISP] switch, and display the program No.

4) Press the controller [RESET] switch.
The STOP lamp will turn OFF, and the program's stopped state will be canceled.

1) Set the [Mode selection switch] on the front of the controller to "MANUAL".

2) Set the T/B [ENABLE] switch to "ENABLE".

3) Press the [EXE] key while holding down the [RESET] key. The execution line will return to the head, and the program will be reset.

STOP lamp turns OFF

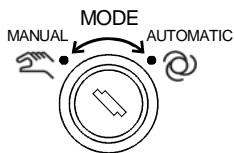
The STOP lamp will turn OFF when the program is reset.

3.8 Turning the servo ON/OFF

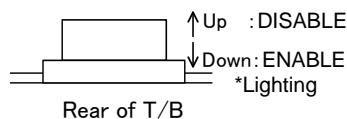
For safety purposes, the servo power can be turned ON during the teaching mode only while the enable switch on the back of the T/B is lightly pressed. Carry out this operation with the T/B while lightly pressing the deadman switch.

*Turning servo ON with T/B

Prepare the T/B



Controller disable



T/B enable

Execute servo ON

Servo ON operation [SERVO]

Execute servo OFF

Servo OFF operation [enabling]

1) Set the [Mode selection switch] on the front of the controller to "MANUAL".

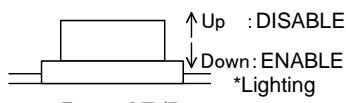
2) Set the T/B [ENABLE] switch to "ENABLE".

3) The servo will turn ON when the [SERVO] key is pressed.

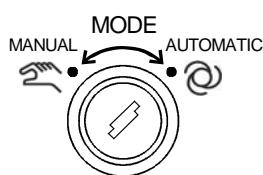
4) Servo-off will be carried out, if the enabling switch (3 position switch) is detached or it pushes in still more strongly.

*Operating with the controller

Prepare the controller

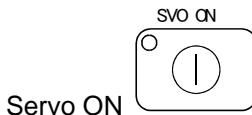


T/B disable



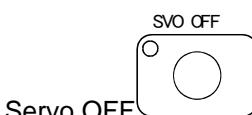
Controller enable

Execute servo ON



Servo ON

Execute servo OFF



Servo OFF

1) Set the T/B [ENABLE] switch to "DISABLE".

2) Set the controller [MODE] switch to "AUTOMATIC".

3) When the [SVO ON] switch is pressed, the servo will turn ON, and the SVO ON lamp will light.

4) When the [SVO OFF] switch is pressed, the servo will turn OFF, and the SVO OFF lamp will light.

Brakes will activate

The brakes will automatically activate when the servo is turned OFF. Depending on the type of robot, some axes may not have brakes.

3.9 Error reset operation

*Error reset operation from the operation panel

Cancel errors

- 1) Press the [RESET] key.
If the error by the side of T/B is not reset, do reset operation from T/B.

Error reset [RESET]

*Error reset operation from the T/B

Cancel errors

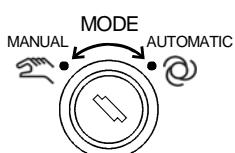
- 1) Press the [RESET] key.

Error reset [RESET]

3.10 Operation to Temporarily Reset an Error that Cannot Be Canceled

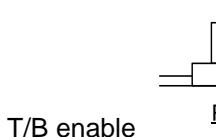
Depending on the type of robot, errors that cannot be cancelled may occur when axis coordinates are outside the movement range, etc. In this case, it is not possible to turn the servo on and perform jog operations with the normal operations. The following procedure can be used to cancel such errors temporarily. For instance, if the axes are outside the movement range, perform a jog operation to adjust the axes while the error is canceled temporarily.

*Operation to cancel errors temporarily from the T/B



Controller disable

- 1) Set the [Mode selection switch] on the front of the controller to "MANUAL".



Cancel errors temporarily

- 2) Set the T/B [ENABLE/] switch to "ENABLE".

- 3) Hold the enabling switch lightly, hold down the [SERVO] key and keep on pressing the [RESET] key.

Error reset [SERVO] + [RESET]

The operation above will reset errors temporarily. Do not release the key; if it is released the error occurs again. Perform a jog operation as well while keeping the [RESET] key pressed.

3.11 Operating the program control screen

Here, explain the operation method of the following related with program management.

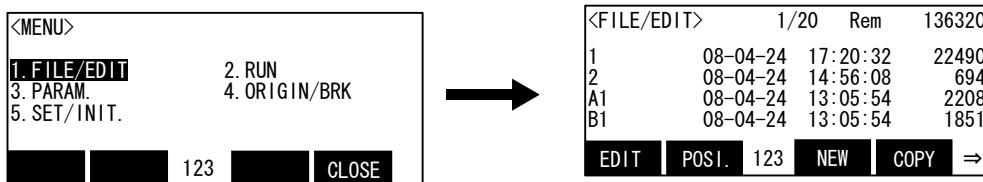
- (1) Program list display
- (2) Copying programs (Copy)
- (3) Name change of the program (Renaming).
- (4) Deleting a program (Delete).
- (5) Protection of the program (Protect).

(1) Program list display

This function allows the status of the programs registered in the controller to be confirmed.

- 1) Select the Management/edit menu

Press the [1] key in the menu screen. "1. Management and edit" are selected and display the list of the programs.



Same operation can be done, even if the cursor is moved to "1. management and edit" by the [↑] or [↓] key and it presses the [EXE] key.

And, the program which is the target of each operation can also be selected.

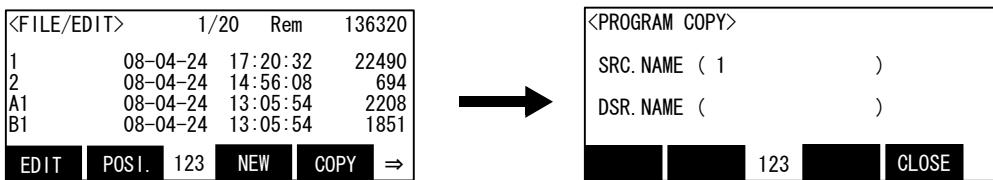
The menu ("Edit", "Position", "New", "Copy") corresponding to the function key is displayed under the screen.

Press the [FUNCTION] key, then display the "Renaming", "Deletion", "Protection", "Close".

(2) Copying programs

1) Select the copy menu

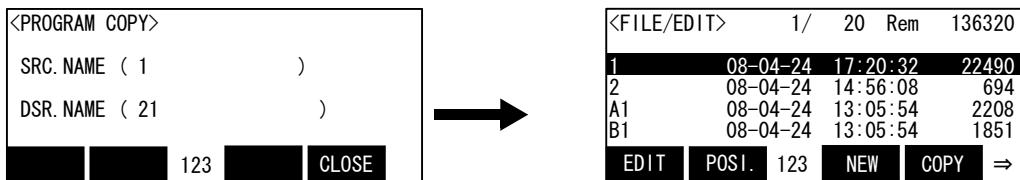
Press the function key corresponding to the "copy" by program list display. Display the copy screen.



2) Specification and execution of the program to copy.

In the parenthesis of the copied source, the program name beforehand selected by the program list screen is displayed. (The figure the program name "1") If it changes, move the cursor by the arrow key.

Input the program name copied in the parenthesis of the copy destination, and press the [EXE] key.



◆◆◆◆ Protected information is not copied ◆◆◆◆

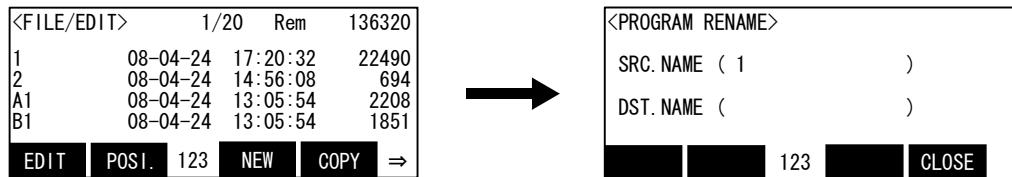
The program protection information and variable protection information is not copied with the copy operation.

Reset this information as necessary.

(3) Name change of the program (Rename)

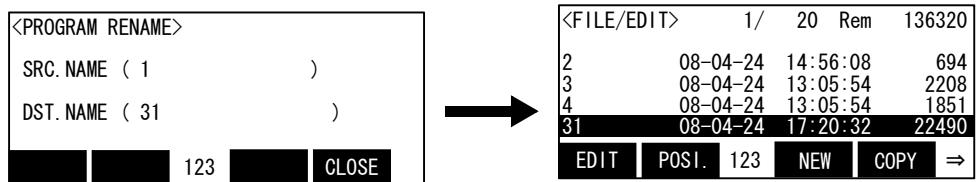
1) Select the rename menu

Press the function key corresponding to the "Rename" by program list display. Display the rename screen if the "renaming" menu is not displayed, press and display the [FUNCTION] key.



2) Specification of the program which changes the name.

In the parenthesis of the renaming source, the program name beforehand selected by the program list screen is displayed. (The figure the program name "1") If it changes, move the cursor by the arrow key. Into the parenthesis of the renaming destination, input the new program name and press the [EXE] key.



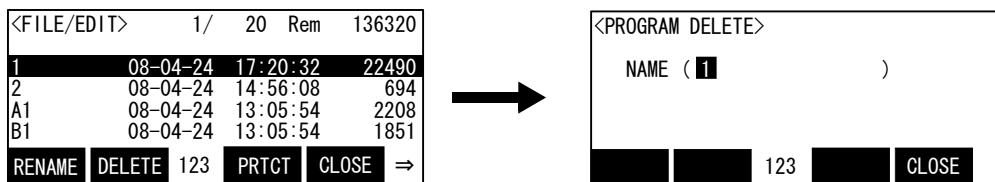
◇◆◇ The program name protected cannot be changed. ◇◆◇

The program name with which command protection is set up cannot be changed. Please execute after removing command protection.

(4) Deleting a program(Delete)

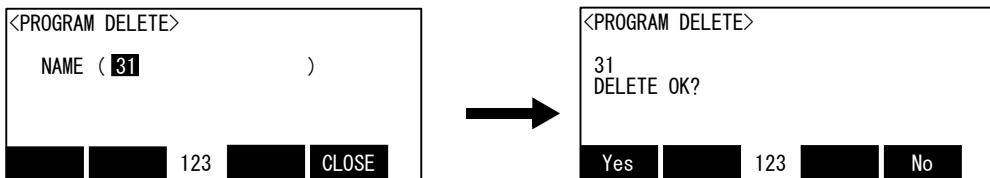
1) Select the delete menu

Press the function key corresponding to the "Delete" by program list display. Display the delete screen. If the "Delete" menu is not displayed, press and display the [FUNCTION] key



2) Specification of the program which delete.

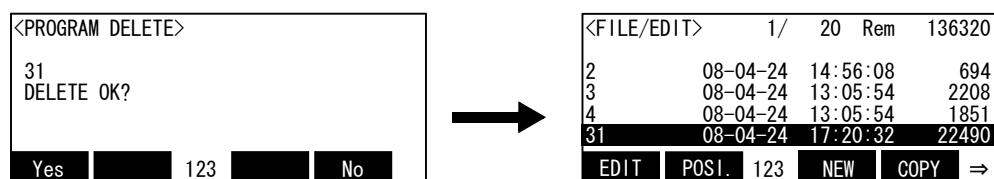
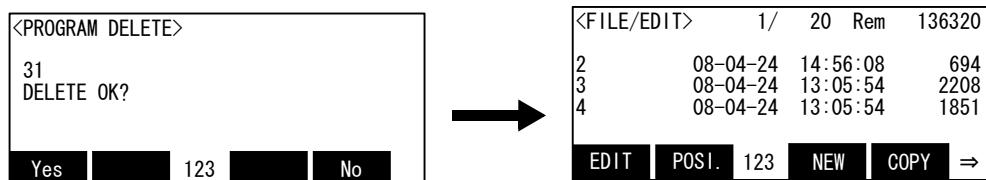
In the parenthesis of the deleteing source, the program name beforehand selected by the program list screen is displayed. (The figure the program name "1") If it changes, input the correct program name. Press the [EXE] key, and display the confirmation screen.



3) Delete the program

If the function key corresponding to "Yes" is pressed, it will delete the specification program and will return to the program list display.

If it does not delete, press the function key corresponding to "No." It returns to the deletion screen.



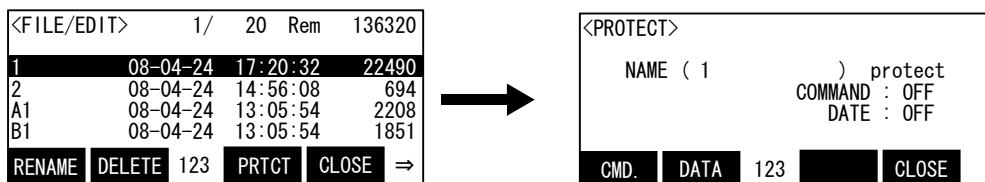
◆◆◆ The program name protected cannot be deleted. ◆◆◆

The program name with which command protection is set up cannot be deleted. Please execute after removing command protection.

(5) Protection of the program (Protect)

1) Select the protect menu

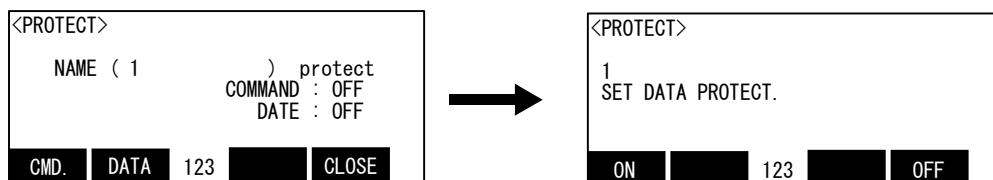
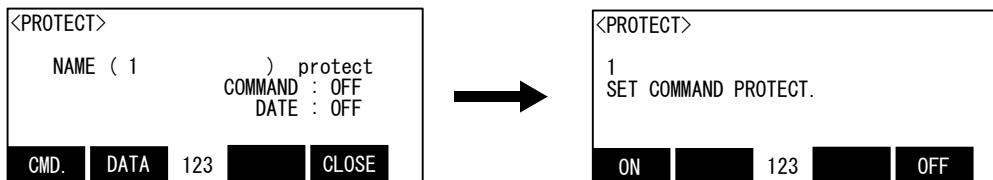
Press the function key corresponding to the "Protect" by program list display. Display the protect screen. If the "Protect" menu is not displayed, press and display the [FUNCTION] key



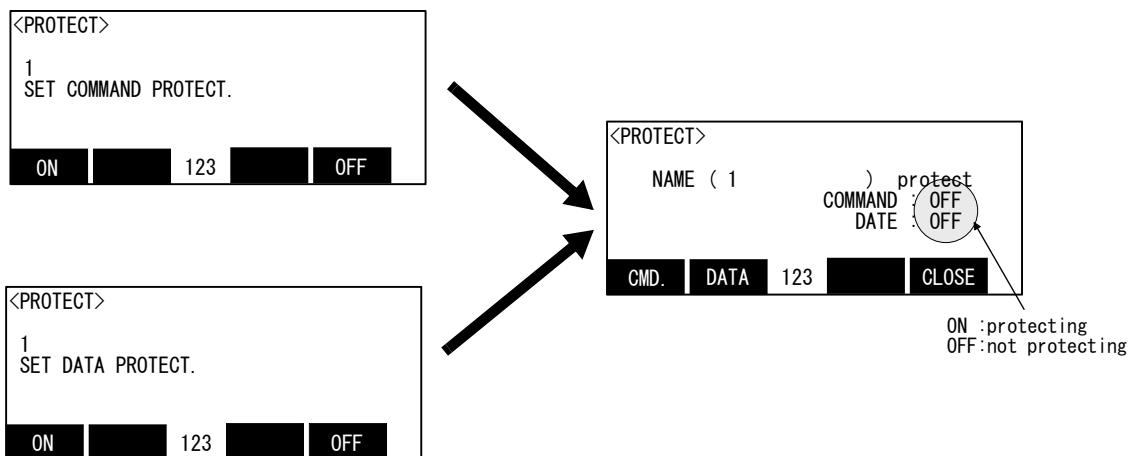
2) Setup of the protection.

The protection of the program can specify the command and data (variable value) separately.

If it sets up protection of the command, press the function key corresponding to "Command." If it sets up protection of the data, press the function key corresponding to "Data."



If the function key corresponding to "ON" is pressed, it will be set up for "protecting." If the function key corresponding to "OFF" is pressed, it will be set up for "not protecting."



◇◆◇ About command protection ◇◆◇

It is the function which protects deletion of the program, name change, and change of the command from the operation mistake.

- Protection information is not copied in copy operation.
- In initialization operation, protection information is disregarded and execute initialization.

◇◆◇ About data protection ◇◆◇

It is the function which protects the variable from the substitution to each variable by registration of the position data based on the operation mistake, change, and the mistaken execution of the program.

- Protection information is not copied in copy operation.
- In initialization operation, protection information is disregarded and execute initialization.

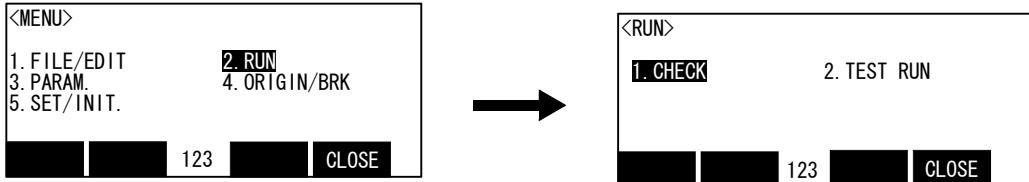
3.12 Operation of operating screen

- (1)Display of the execution line1.Confirmation: Display the executing program line, or execute step feed
- (2)Display of the test execution line2.Test execution: Display the name of the program selected, and the executing step number. And, change the continuation mode of operation to cycle stop mode.

3.12.1 Display of the execution line

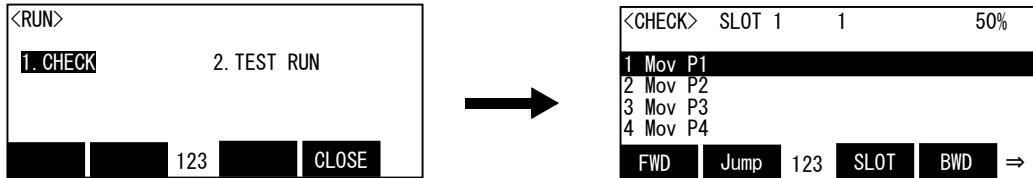
(1) Select the confirmation menu

- 1) Press the [2] key in the menu screen, and display the operation menu screen.



- 2) Press the [1] key, and display the confirmation screen.

Display the program set as the slot 1 on the screen. The program name is displayed following the slot number.



The cursor moves to the execution line during program execution.

(2) Step feed

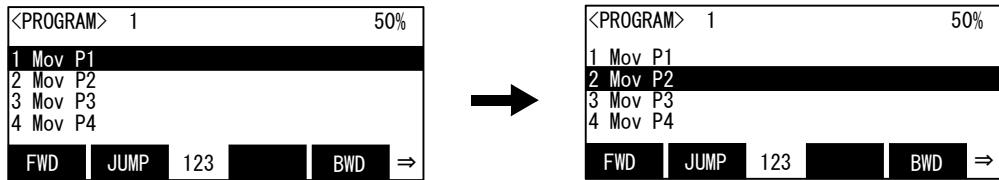
The same operation as above-mentioned step feed and step return can be done.

1) Step feed

Pressing the [F1] (FWD) key is kept, and the robot will start moving.

If [F1] (FWD) is released during this step, the robot will stop. And, detach the enabling switch (3 position switch), or push in still more strongly -- thing servo-off can be carried out and execution can be stopped.

During execution, the lamp on the controller's [START] switch will light.1 If execution of the one step is completed, LED of the [START] switch will go out and LED of the [STOP] switch will turn it on. If the [F1] key is detached, the cursor of the T/B screen will move to the following step.



Whenever it presses the function key corresponding to "FWD", step to the following step.



CAUTION

Take special care to the robot movements during automatic operation. If any abnormality occurs, press the [EMG.STOP] switch and immediately stop the robot.

◆◆◆ About step operation ◆◆◆

“Step operation” executes the program line by line. The operation speed is slow, and the robot stops after each line, so the program and operation position can be confirmed.

During execution, the lamp on the controller’s [START] switch will light. Execution of the End command or the Hlt command will not step feed any more.

◆◆◆ Change of the execution step ◆◆◆

The execution step can be changed by cursor movement by the arrow key, and jump operation (“JUMP”).

◆◆◆ Immediately stopping the robot during operation ◆◆◆

- Press the [EMG.STOP] (emergency stop) switch.
The servo will turn OFF, and the moving robot will immediately stop.
To resume operation, reset the alarm, turn the servo ON, and start step operation.
- Release or forcibly press the “enable” switch.
The servo will turn OFF, and the moving robot will immediately stop.
To resume operation, lightly press the “enable” switch, and start step operation.
- Release the [F1] (FWD)key.
The step execution will be stopped. The servo will not turn OFF.
To resume operation, press the [F1] (FWD)key.

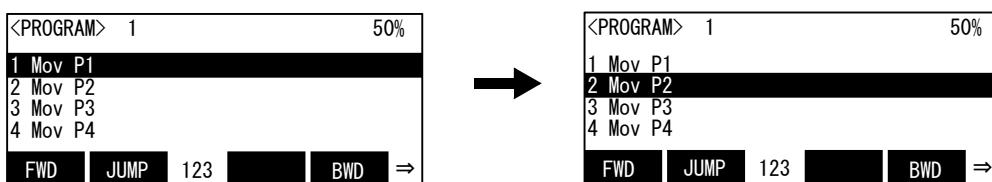
2) Step return

The line of a program that has been stopped with step feed or normal operation is returned one line at a time and executed. This can be used only for the interpolation commands. Note that only up to four lines can be returned.

If the function key corresponding to “BWD” is pressed, only while keeping pushing, only the one step will be executed in the return direction of the step.

If the function key is released during this step, the robot will stop. And, detach the enabling switch (3 position switch), or push in still more strongly — thing servo-off can be carried out and execution can be stopped.

During execution, the lamp on the controller’s [START] switch will light. If execution of the one step is completed, LED of the [START] switch will go out and LED of the [STOP] switch will turn it on. The cursor of the T/B screen moves to the step of the next interpolation command in the return direction of the step.



Whenever it presses the function key corresponding to “BWD”, it returns to the front step.



CAUTION

Take special care to the robot movements during automatic operation. If any abnormality occurs, press the [EMG.STOP] switch and immediately stop the robot.

◆◆◆ Immediately stopping the robot during operation ◆◆◆

- Press the [EMG.STOP] (emergency stop) switch.
The servo will turn OFF, and the moving robot will immediately stop.
To resume operation, reset the alarm, turn the servo ON, and start step operation.
- Release or for cibly press the "enable" switch.
The servo will turn OFF, and the moving robot will immediately stop.
To resume operation, lightly press the "enable" switch, and start step operation.
- Release the [F1] (FWD)key.
The step execution will be stopped. The servo will not turn OFF.
To resume operation, press the [F1] (FWD)key.

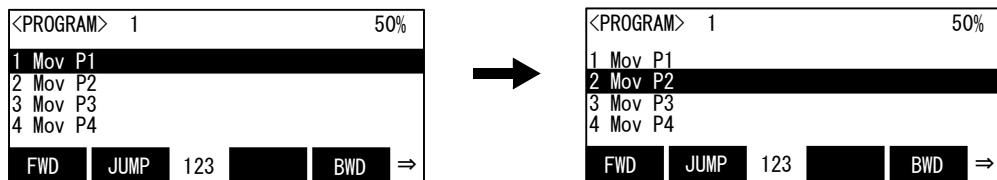
(3) Step jump

It is possible to change the currently displayed step or line.

The operation in the case of doing step operation from Step 5 as an example is shown.

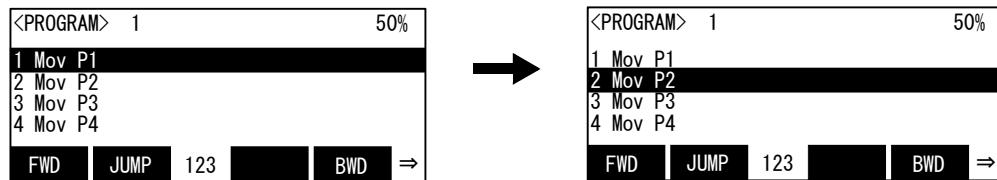
1) Call Step 5.

Press the function key corresponding to "JUMP", and press the [5], [EXE] key. The cursor moves to Step 5.



2) Execution of step feed

If the function key corresponding to "FWD" is pressed, step feed can be done from Step 5.



CAUTION

Take special care to the robot movements during automatic operation. If any abnormality occurs, press the [EMG.STOP] switch and immediately stop the robot.

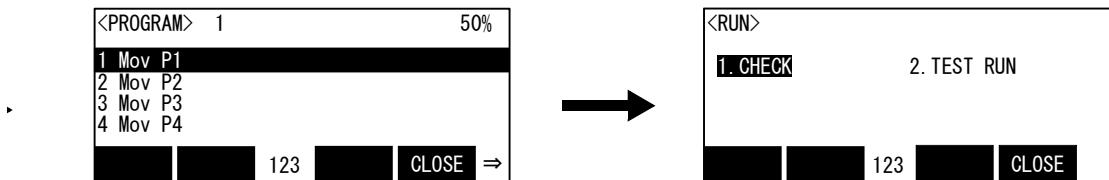
(4) Step feed in another slot

When checking a multitask program, it is possible to perform step feed in the confirmation screen of the operation menu, not in the edit screen.

Refer to "[Page 43, "\(3\) Step feed in another slot"](#)" for operation method.

(5) Finishing of the confirmation screen.

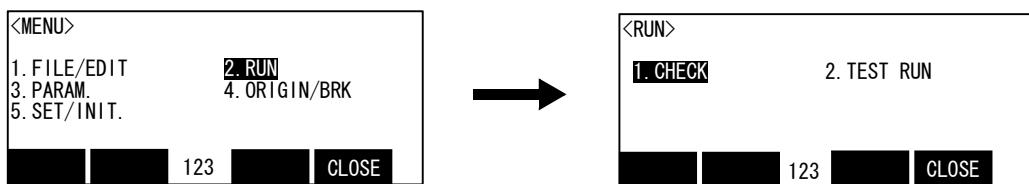
1) Press the function key corresponding to "Close", and return to the operation menu screen.



3.12.2 Test operation

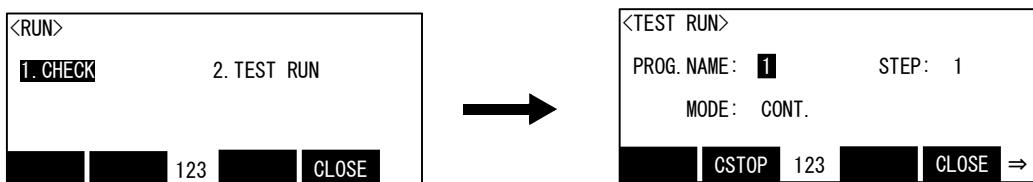
(1) Select the test operation

- 1) Press the [2] key in the menu screen, and display the operation menu screen.

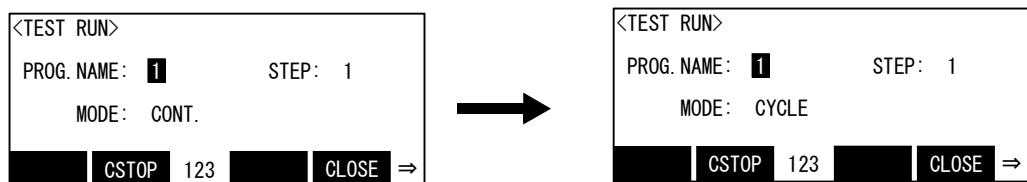


- 2) Press the [2] key, and display the test operation screen.

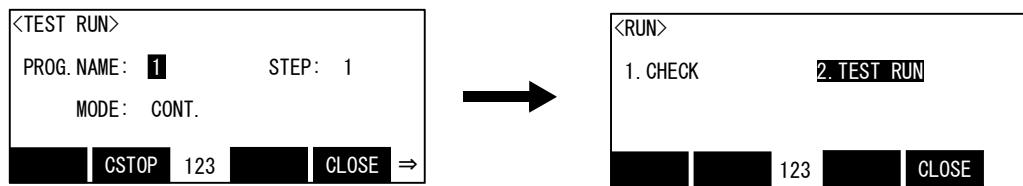
The program name, execution step number, and operating mode is displayed.



- 3) When the function key ([F2]) corresponding to "CSTOP" is pressed during program execution, it is change to the cycle mode of operation. The "cycle" is displayed after the mode and the [END] button of the operation panel blinks. Finish operation after executing the last line of the End command or the program.



- 4) Press the function key corresponding to "Close", and return to the operation menu screen.



◆◆◆ If execution of the program is stopped, it will become the continuation mode of operation. ◆◆◆
 If the [STOP] key is pressed in the cycle mode of operation and execution of the program is stopped, it changes to the continuation mode of operation. If it continues execution of the program by the cycle mode of operation, please press the [F4] key again after pushing the [START] button. (It also becomes the cycle mode of operation to push the [END] button of the controller)

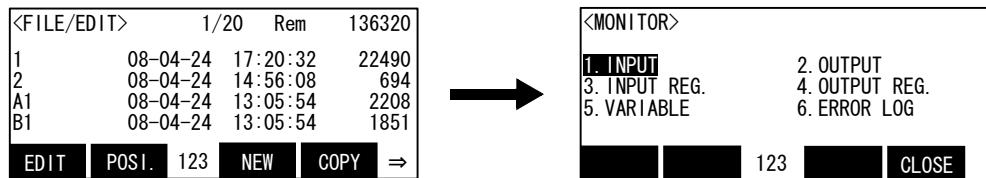
3.13 Operating the monitor screen

Here, explain the operation method of the following functions.

- (1)Input signal monitor 1.Input : Parallel input signal monitor
- (2)Output signal monitor 2.Output : Parallel output signal monitor. Setup of ON/OFF
- (3)Input register monitor 3.Input register: Input register of CC-Link
- (4)Output register monitor 4.Output register: Output register of CC-Link
- (5)Variable monitor 5.Variable : Variable value monitor & set up
- (6)Error history display 6.Error history: History of the occurrence error

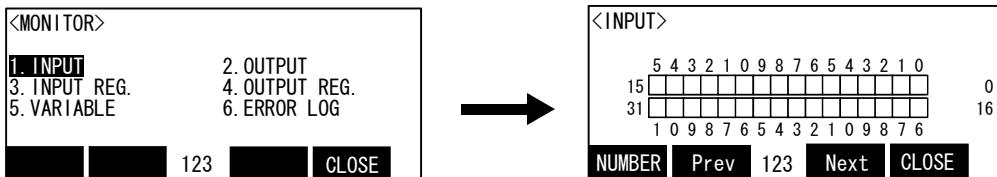
All of the above press the [MONITOR] key of T/B. It operates, even when T/B is invalid.

Although the screen currently displayed may be free, the variable monitor does not operate in the program (command) edit screen.



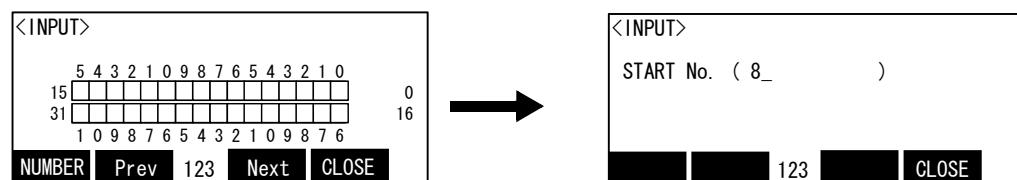
(1) Input signal monitor

- 1) Press the [1] key in the monitor menu screen, and display the input signal screen. The input signal of the 32 points can be monitored on the one screen.

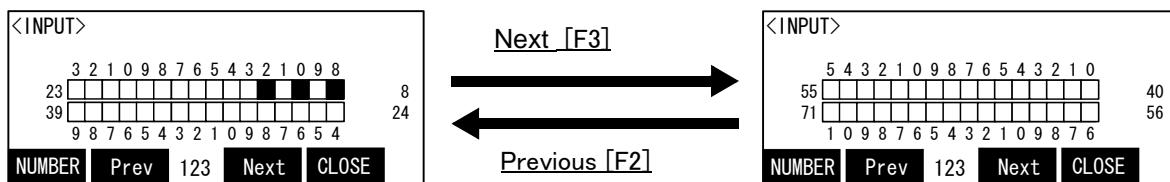


The case where the state of the input signals 8–15 is confirmed is shown in the following.

- 2) Press the function key corresponding to "Number".
Set "8" as the start number.

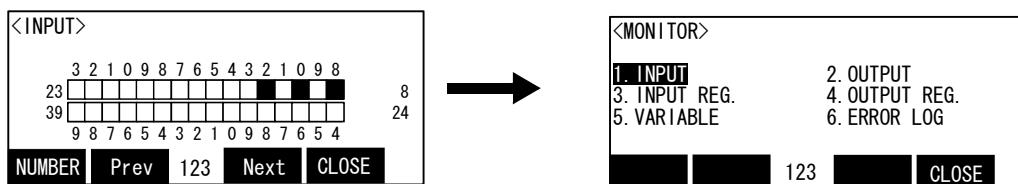


3) Display the ON/OFF state of the 32 points at the head for the input signal No. 8. Black painting indicates ON and white indicates OFF.

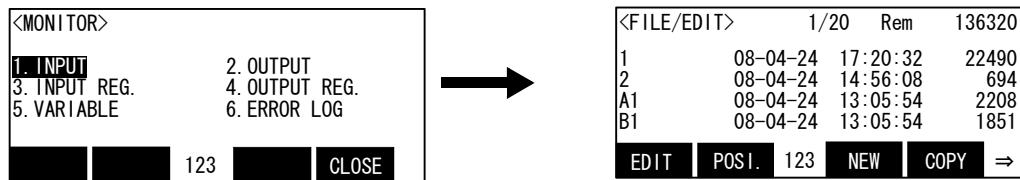


Press the function key corresponding to "Next", then display the next input signal screen. Press the function key corresponding to "Prev", then display the previous input screen

4) Press the function key corresponding to "Close", and return to the monitor menu screen.



5) Press the function key corresponding to "Close" in monitor menu screen is pressed, finish the monitor, and return to the original screen.



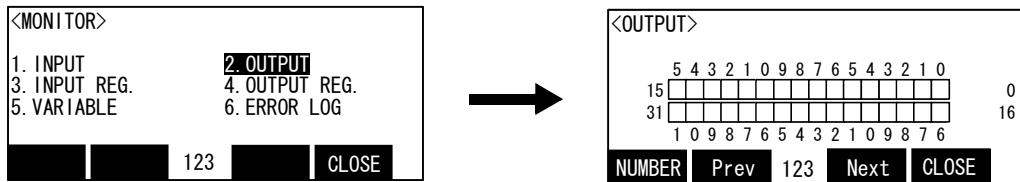
◆◆◆ Finish the monitor ◆◆◆

If the [MONITOR] key is pressed, the monitor will be finished always and it will return to the original screen.

(2) Output signal monitor

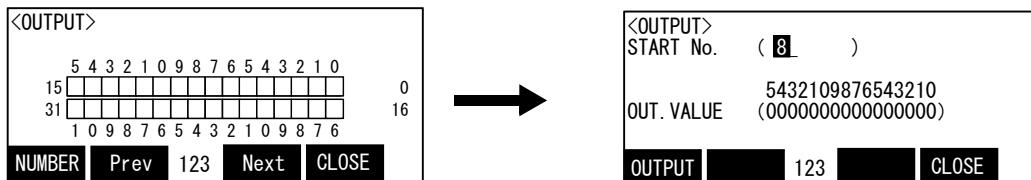
There are the function which always makes the ON/OFF state of the output signal the monitor, and the function outputted compulsorily.

- 1) Press the [2] key in the monitor menu screen, and display the output signal screen. The output signal of the 32 points can be monitored on the one screen.



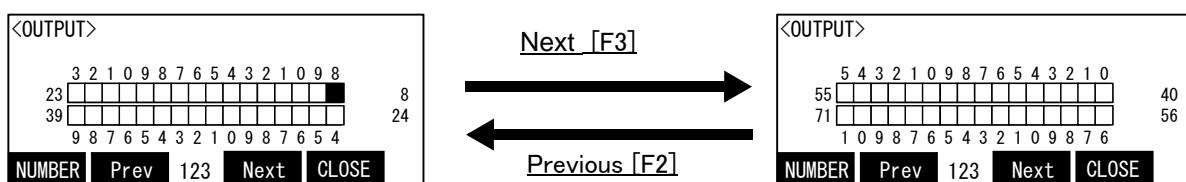
The case where the state of the output signals 8–15 is confirmed is shown in the following.

- 2) Press the function key corresponding to "Number".
 Set "8" as the start number.



Although the state of the current output signal is displayed on the output value on the display, it is not always the display here in the section which sets up the compulsive output value of the signal.

Press the function key corresponding to "Close". Display the ON/OFF state of the 32 points at the head for the output signal No. 8. Black painting indicates ON and white indicates OFF.



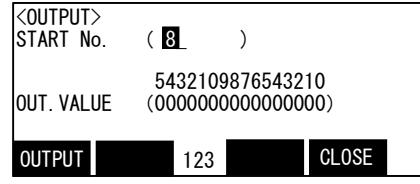
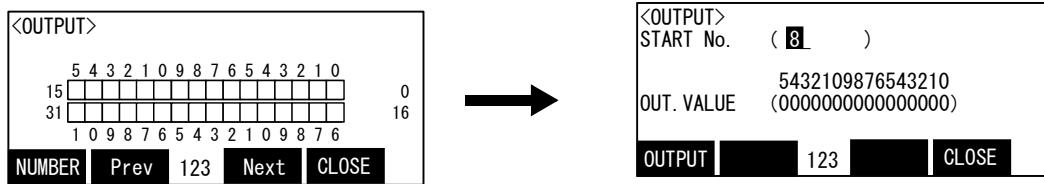
Press the function key corresponding to "Next", then display the next output signal screen. Press the function key corresponding to "Prev", then display the previous output screen

3) The compulsive output of the output signal.

In the following, the operation method in the case of turning off the output signal No. 8 compulsorily is shown.

Press the function key corresponding to "Number".

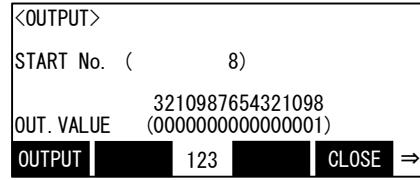
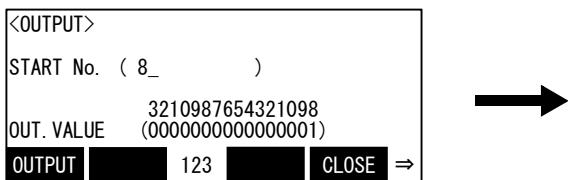
Set "8" as the start number. (Press [8], and [EXE] key)



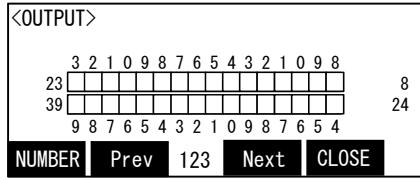
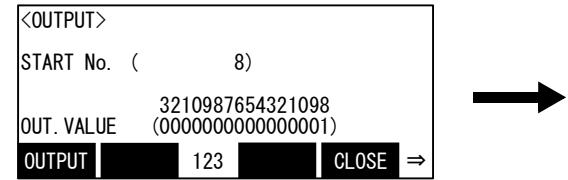
4) Move the cursor to the position of "8" of the output value by the arrow key.

Since the output signal 8 number is turned on now, value "1" is displayed.

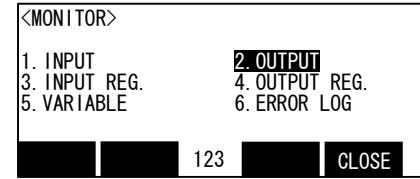
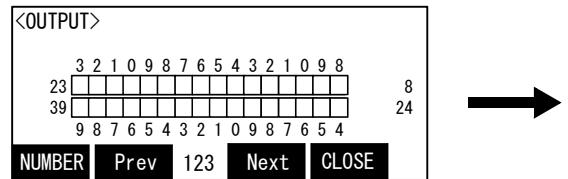
If the value is changed into "0" which shows OFF and the function key ([F1]) corresponding to the "Output" is pressed, this output signal will actually be off.



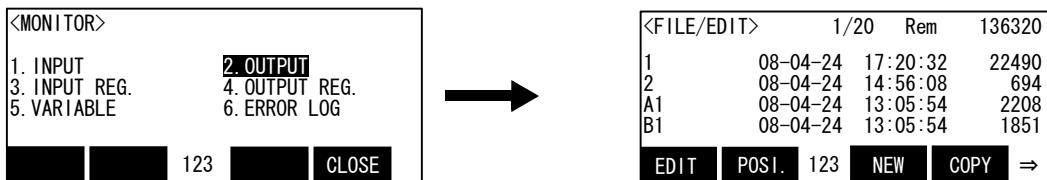
5) Press the function key corresponding to "Close", and return to the output monitor screen.



6) Press the function key corresponding to "Close", and return to the monitor menu screen.



- 7) Press the function key corresponding to "Close" in monitor menu screen is pressed, finish the monitor, and return to the original screen.

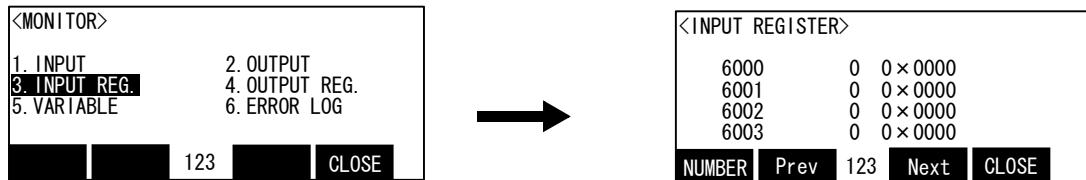


(3) Input register monitor

If CC-Link is used, it is the function which always monitors the value of the input register.

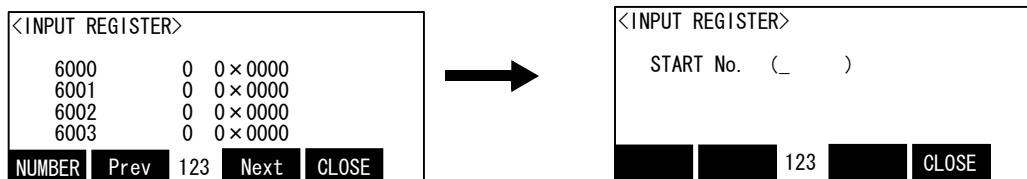
Note) Since there is no CC-Link option in the CRnQ-700 series, this function can not be used.

- 1) Press the [3] key in the monitor menu screen, and display the input register screen. The input register of the 4 registers can be monitored on the one screen.

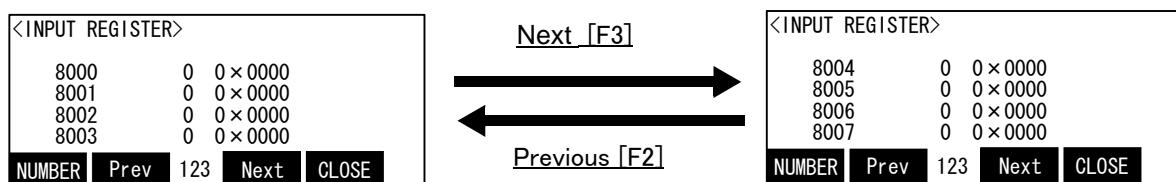


The case where the state of the input register 8000 is confirmed is shown in the following.

- 2) Press the function key corresponding to "Number".
3) Set "8000" as the start number.

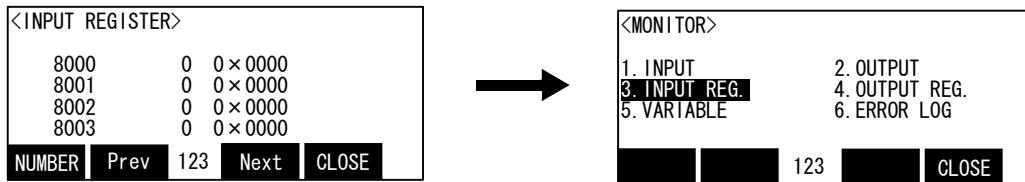


- 4) Display the ON/OFF state of the 4 input register at the head for the input register No. 8000.

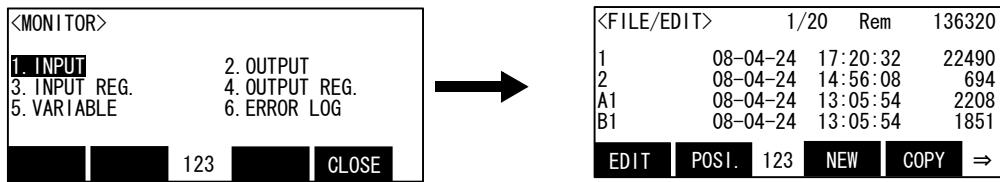


Press the function key corresponding to "Next", then display the next input register screen. Press the function key corresponding to "Prev", then display the previous input register screen.

5) Press the function key corresponding to "Close", and return to the monitor menu screen.



6) Press the function key corresponding to "Close" in monitor menu screen is pressed, finish the monitor, and return to the original screen.



◆◆◆ Finish the monitor ◆◆◆

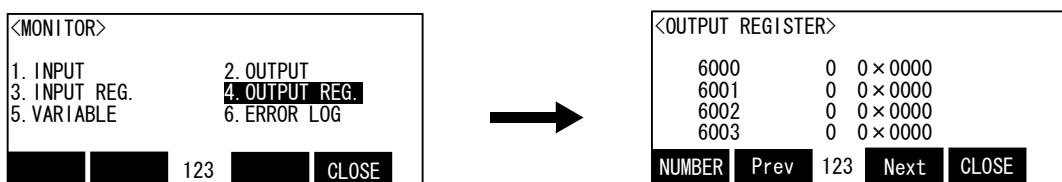
If the [MONITOR] key is pressed, the monitor will be finished always and it will return to the original screen.

(4) Output register monitor

If CC-Link is used, it is the function which always monitors the value of the output register.

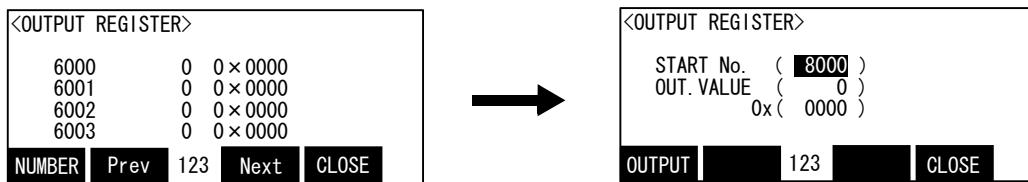
Note) Since there is no CC-Link option in the CRnQ-700 series, this function can not be used.

1) Press the [4] key in the monitor menu screen, and display the output register screen. The output register of the 4 registers can be monitored on the one screen.

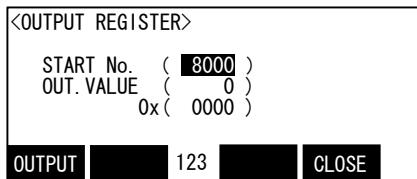


The case where the state of the output register 8000 is confirmed is shown in the following.

- 2) Press the function key corresponding to "Number".
Set "8000" as the start number.

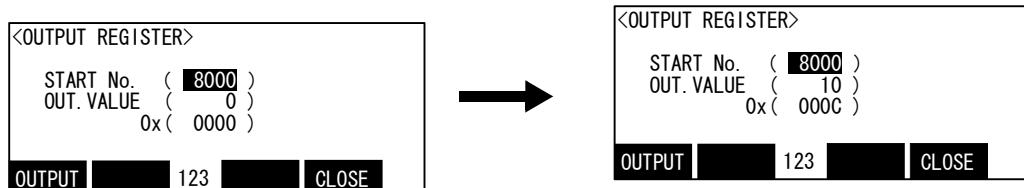


- 3) The current output value of No. 8000 is displayed by the decimal number in the parenthesis following the output value. The value in the parenthesis following lower 0x is the hexadecimal number.

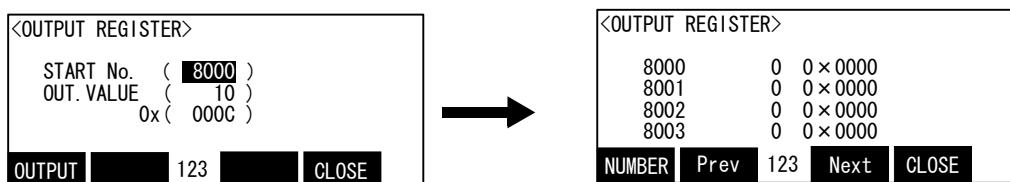


If the function key which corresponds for "Close" is pressed, it can also return to the monitoring screen on the basis of No. 8000 of the output register, but the output value can be changed on the current screen. The case where the value of the output register No. 8000 is set as 12 (decimal number) is shown in the following.

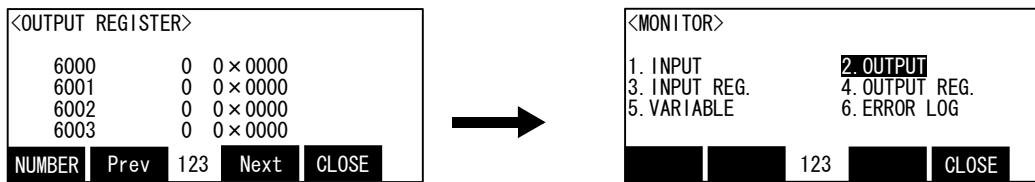
- 4) The setup of the value can be set up by the decimal number or the hexadecimal number.
If it sets up by the decimal number, move the cursor to the output value by the arrow key, and input "10." The unnecessary character should press and erase the [CLEAR] key.
If it sets up by the hexadecimal number, move the cursor to 0x by the arrow key, and input "C." The unnecessary character should press and erase the [CLEAR] key.
Press the function key ([F1]) corresponding to the "Output", then will actually output the set-up value.



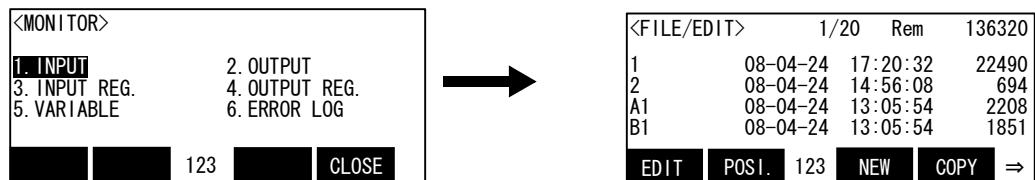
- 5) Press the function key corresponding to "Close", and return to the output register monitor screen.



6) Press the function key corresponding to "Close", and return to the output menu screen



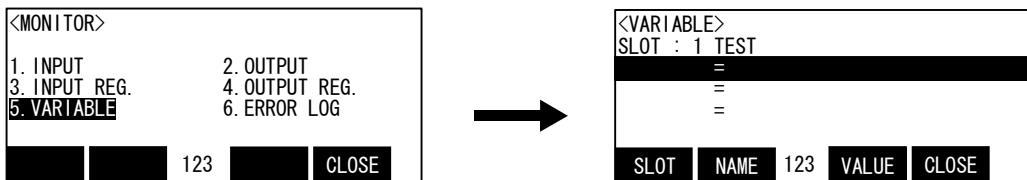
7) Press the function key corresponding to "Close", and return to the monitor menu screen.



(5) Variable monitor

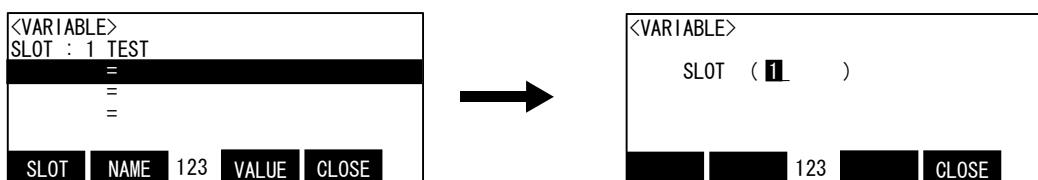
It is the function to display or change the details of the variable currently used by the program.

- 1) Press the [5] key in the monitor menu screen, and display the variable monitor screen.



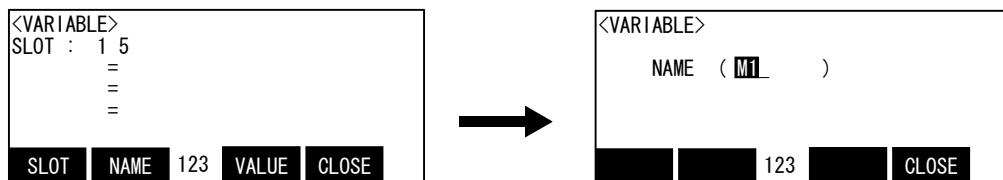
- 2) Specify the target program of the monitor with the slot number.

Press the function key corresponding to "SLOT", and input the slot number.
Set up "1", if the multitasking function is not being used.



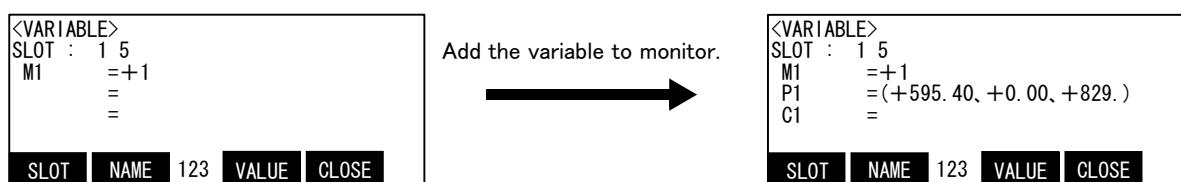
- 3) "Display the slot number and the program name after "slot::"

Press the function key corresponding to the "Name", and input the variable name to monitor.



- 4) Display the value of the numeric variable M1 on the screen.

The variable which will be monitored if the cursor is moved to the line which is vacant in the arrow key and operation of the above 3 is repeated can be added. The variable which can be monitored simultaneously is to the three pieces.

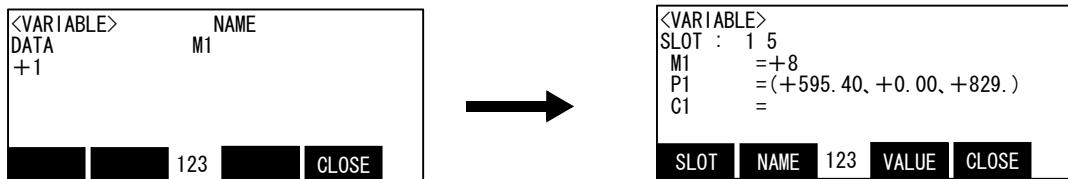


5) Change the variable value.

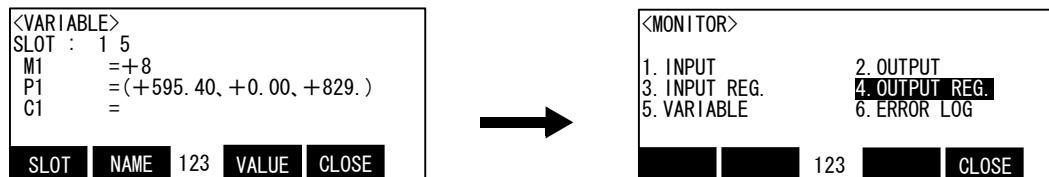
The value of the variable currently displayed can be changed.

Move the cursor to the variable name changed by the arrow key, and press the function key corresponding to the "Value."

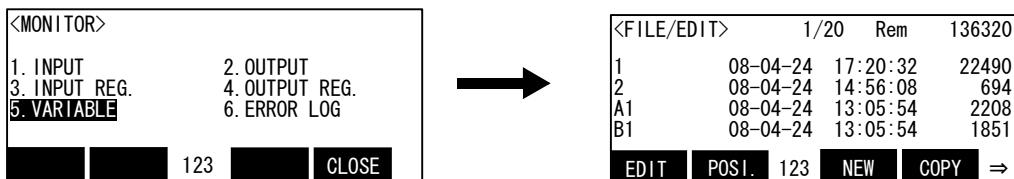
Although the current value (data) is displayed, it can input and change.



6) Press the function key corresponding to "Close", and return to the monitor menu screen.



7) Press the function key corresponding to "Close" in monitor menu screen is pressed, finish the monitor, and return to the original screen.



◆◆◆ The right of operation is unnecessary. ◆◆◆

It operates, even when T/B is invalid.

And, the value (data) of the variable can be changed also in automatic operation.

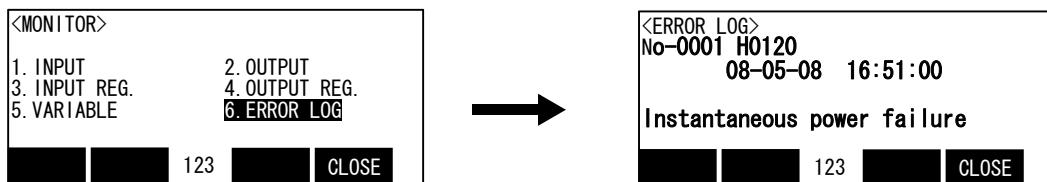
◆◆◆ Finish the monitor ◆◆◆

If the [MONITOR] key is pressed, the monitor will be finished always and it will return to the original screen.

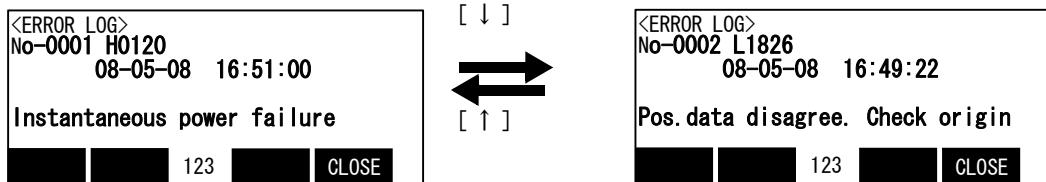
(6) Error history

Display the error history. Please use reference at the time of trouble occurrence.

- 1) Press the [6] key in the monitor menu screen, and display the error history.



Display error history before and after by the arrow key.



◆◆◆ The right of operation is unnecessary. ◆◆◆

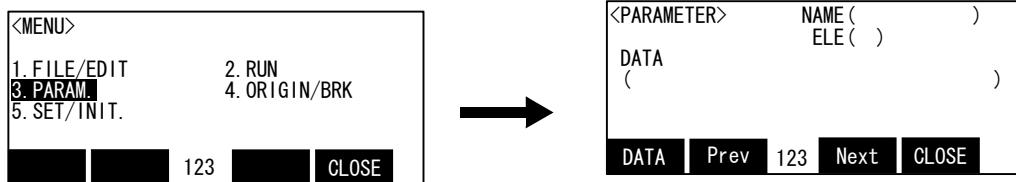
It operates, even when T/B is invalid.

And, the value (data) of the variable can be changed also in automatic operation.

3.14 Operation of maintenance screen

The parallel I/O designated input/output settings and settings for the tool length, etc., are registered as parameters. The robot moves based on the values set in each parameter. This function allows each parameter setting value to be displayed and registered.

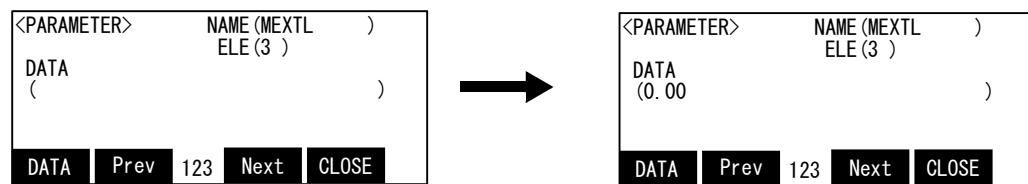
- 1) Press the [3] key in the menu screen, and display the parameter screen.



An example of changing the parameter "MEXTL (tool data)" Z axis (3rd element) setting value from 0 to 100mm is shown below.

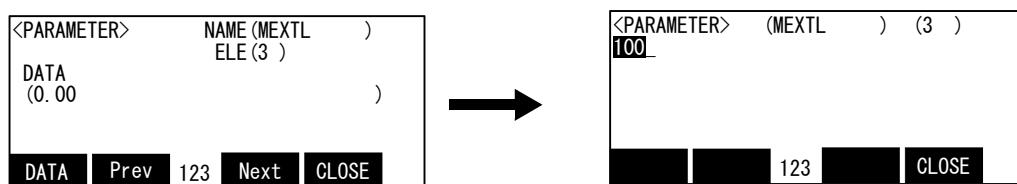
- 2) Input "MEXTL" into the name and input "3" into the element.

- 3) The data set up now is displayed.



- 4) Press the function key corresponding to the "Data", and input new preset value "100."

Delete the unnecessary number by the [CLEAR] key.



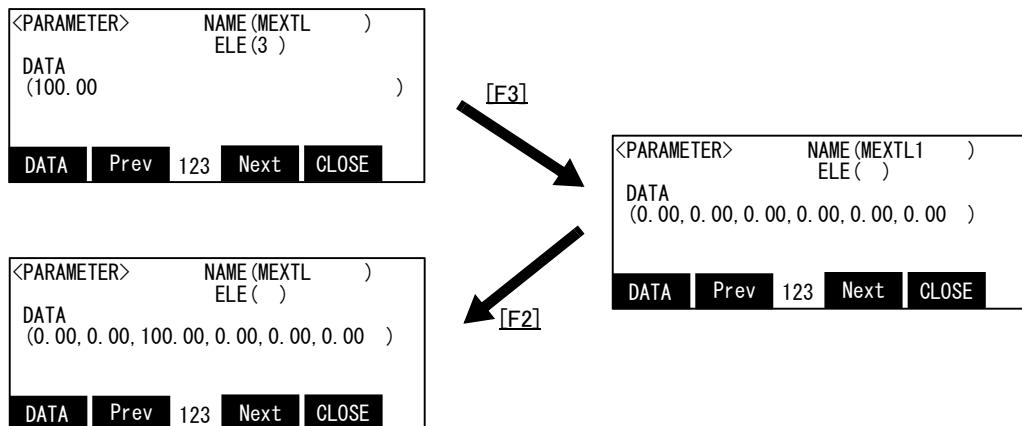
If the [EXE] key is pressed, the buzzer will sound, the value will be fixed and it will return to the screen of the parameter.

If the function key corresponding to the "Close" is pressed also after inputting the new preset value, change can be canceled and it can return to the parameter screen.

And, press the function key corresponding to "Next" will display the next parameter.

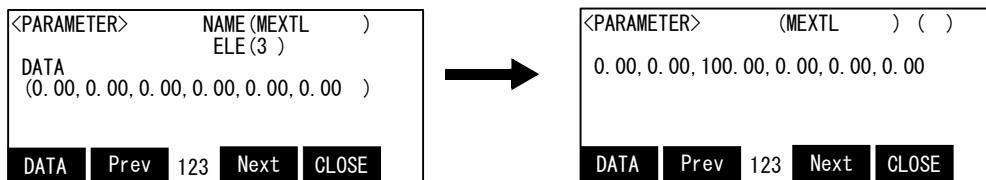
Display that the previous parameter presses the function key corresponding to "Prev".

In this case, because of to display all the elements of the parameter shown by the name, delete specification of the element number.



The value can be changed also in this state.

Press the function key corresponding to the "Data", make it move to the position of the element number which changes the cursor by the arrow key, and input the new preset value. Delete the unnecessary number by the [CLEAR] key.



If the [EXE] key is pressed, the buzzer will sound, the value will be fixed and it will return to the screen of the parameter.

If the function key corresponding to the "Close" is pressed also after inputting the new preset value, change can be canceled and it can return to the parameter screen.

◆◆◆ Power must be turned ON again ◆◆◆

The changed parameter will be validated only after the controller power has been turned OFF and ON once.

◆◆◆ Only display is valid during program execution. ◆◆◆

If the setting value of the parameter is changed during execution of the program, the error will occur. (Even if the error occurs, execution of the program does not stop)

◆◆◆ Display the parameter near the name of the inputted parameter. ◆◆◆

Even if the name of the parameter does not input all characters correctly, it displays the parameter near the inputted name automatically.

3.15 Operation of the origin and the brake screen

(1) Origin

If the origin position has been lost or deviated when the parameters are lost or due to robot interference, etc., the robot origin must be set again using this function.

Refer to the separate manual: "Robot arm setup & maintenance" for details on the operation.

(2) Brake

In the state of servo off, it is the function to release the brake of the servo motor. Refer to the [Page 50, "3.8 Turning the servo ON/OFF"](#) for servo off operation.

Use it, if it moves the robot arm directly by hand.



CAUTION

Due to the robot configuration, when the brakes are released, the robot arm will drop with its own weight depending on the released axis.

Always assign an operator other than the T/B operator to prevent the arm from dropping. This operation must be carried out with the T/B operator giving signals.

Refer to [Table 3-6](#) and accurately designate the axis for which the brakes are to be released.

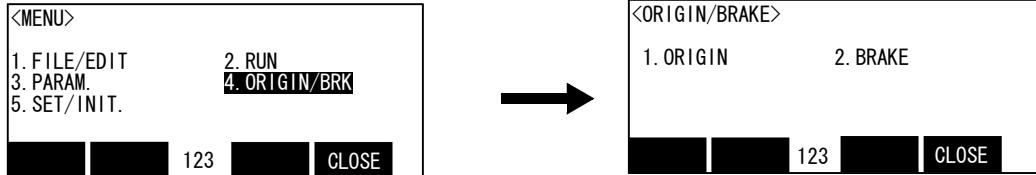
Table 3-6: The brake release axis unit classified by type

Type	Axis								Remarks
	1	2	3	4	5	6	7	8	
RV-SQ/SD series	■	■	■	■	■	■	■	■	

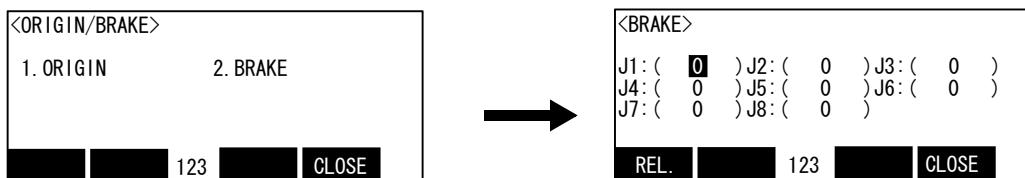
Note) "■" mark means that brake release with the independent axis is possible.

The operation method is shown in the following. Perform this operation, in the condition that the enabling switch (3 position switch) is pushed lightly.

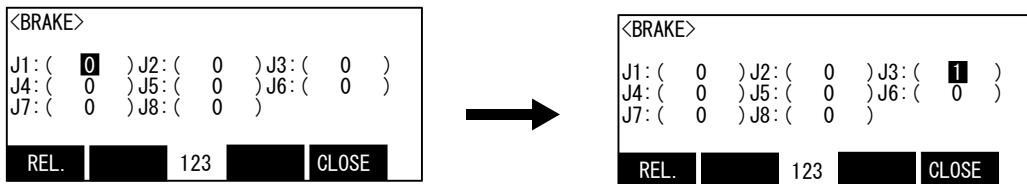
1) Press the [4] key in the monitor menu screen, and display the origin/brake screen.



2) Press the [2] key in the origin/brake screen, and display the break release screen.



3) Input "1" into the axis which release the brake.



⚠ CAUTION

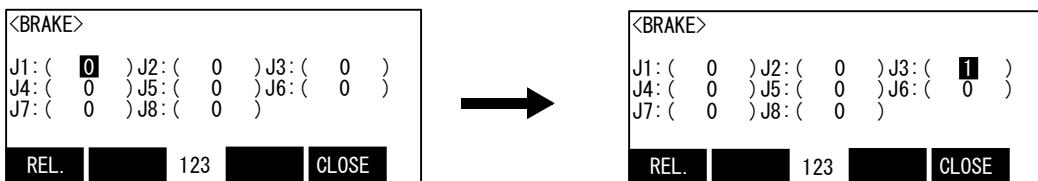
Due to the robot configuration, when the brakes are released, the robot arm will drop with its own weight depending on the released axis.

Always assign an operator other than the T/B operator to prevent the arm from dropping. This operation must be carried out with the T/B operator giving signals.

Refer to [Table 3-6](#) and accurately designate the axis for which the brakes are to be released.

4) Press function key continuously corresponding to "Release" to release the brake of the specified axis only while the keys are pressed.

The brakes will activate when the function key or enabling switch is released.



3.16 Operation of setup / initialization screen

Here, explain the operation method of the following functions.

- (1)Initialization.....1. Programs : Delete all the programs
2. Parameter: Return the parameter to the setup at the time of shipment.

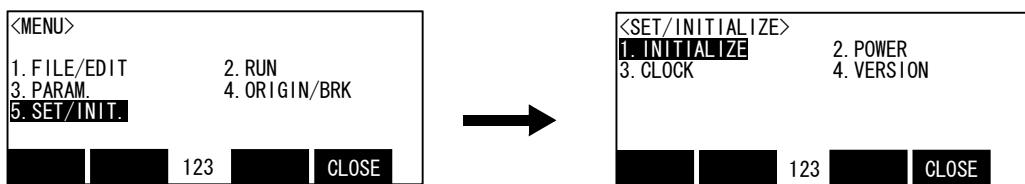
3. Battery : Reset the expended hours of the battery.

- (2)OperationDisplay the accumulation time of the power supply ON, and the remaining time of the battery.

- (3)Time.....Display of the date and time, the setup

- (4)VersionDisplay the software version of the controller and the teaching pendant.

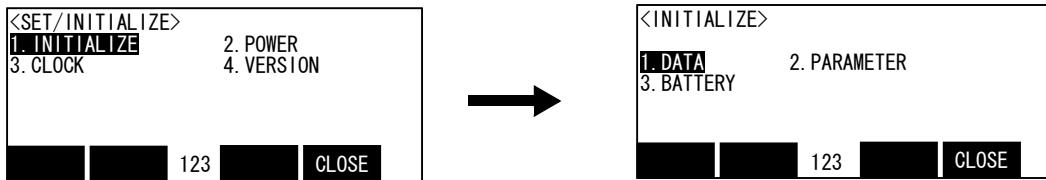
Press the [5] key in the menu screen, and display the set/initial screen.



(1) Initialize the program

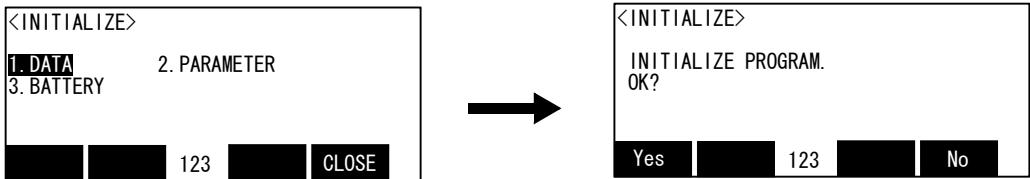
Delete all the programs.

- 1) Press the [1] key in the set/initial screen, and display the initial menu screen

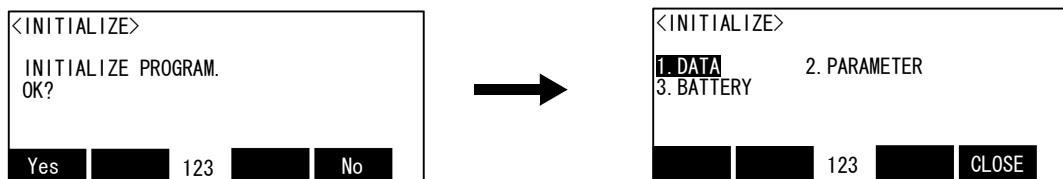


- 2) Press the [1] key in the initial menu screen, and select the program.

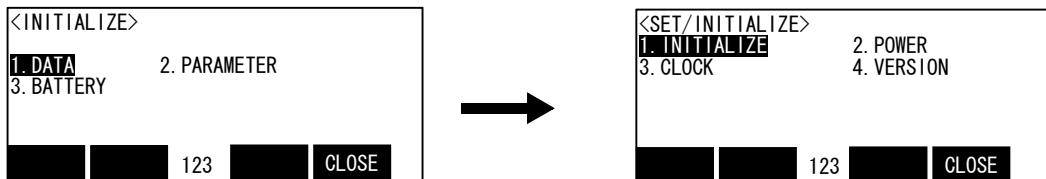
Display the screen of confirmation.



- 3) If it initializes, press the function key corresponding to "Yes". If it does not initialize, press the function key corresponding to "no". The screen returns to initiali menu screen.



4) Press the function key corresponding to "Close", and return to the set/initial screen.



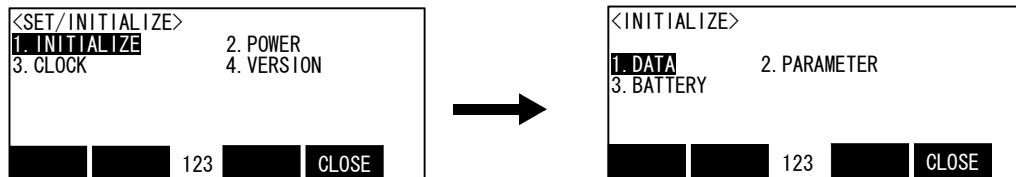
◆◆◆ Executed even when protected ◆◆◆

The program will be initialized even if the program protection or variable protection is set to ON.

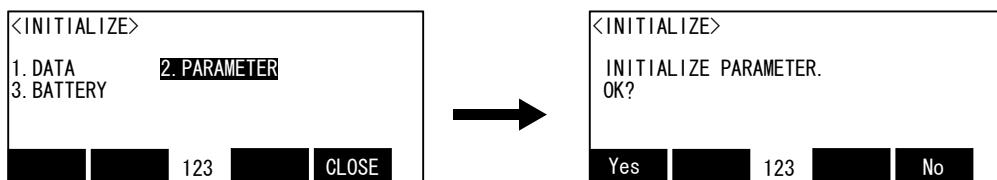
(2) Initialize the parameter

Return the parameter to the setup at the time of shipment.

1) Press the [1] key in the set/initial screen, and display the initial menu screen.

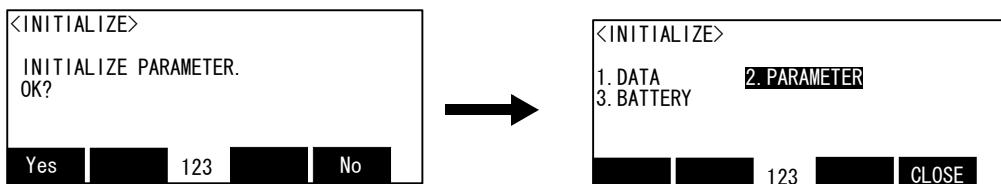


2) Press the [2] key in the initial menu screen, and select the parameter. Display the screen of confirmation.

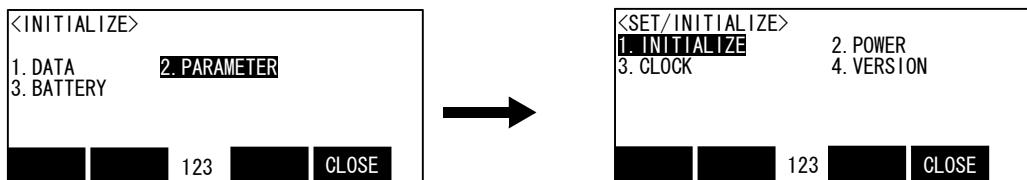


3) If it initializes, press the function key corresponding to "Yes". If it does not initialize, press the function key corresponding to "no".

The screen returns to initiali menu screen.



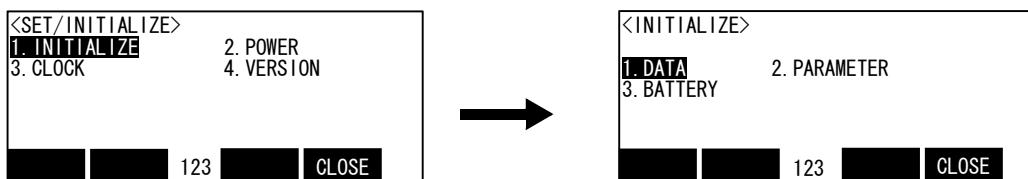
4) Press the function key corresponding to "Close", and return to the set/initial screen.



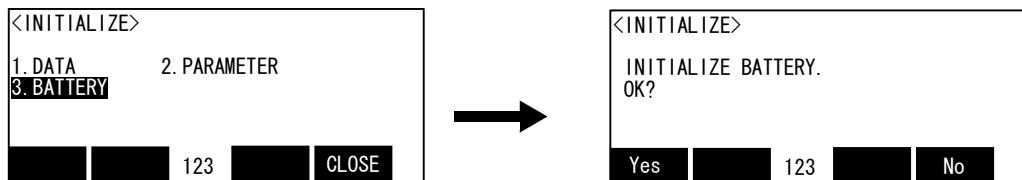
(3) Initialize the battery

Reset the expended hours of the battery

1) Press the [1] key in the set/initial screen, and display the initial menu screen.

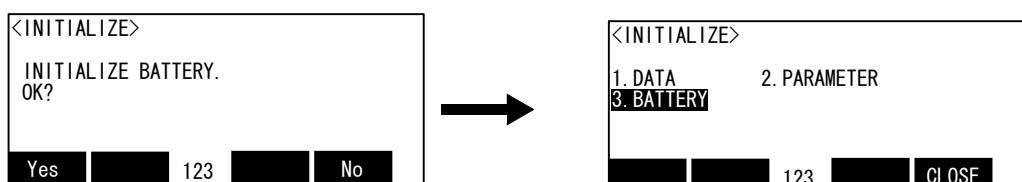


2) Press the [3] key in the initial menu screen, and select the battery. Display the screen of confirmation

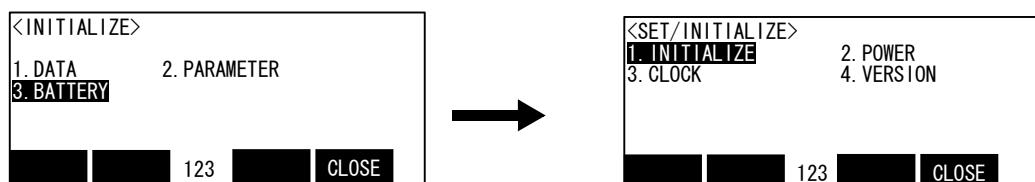


3) If it initializes, press the function key corresponding to "Yes". If it does not initialize, press the function key corresponding to "no".

The screen returns to initiali menu screen.



4) Press the function key corresponding to "Close", and return to the set/initial screen.



◇◆◇ Always initialize after battery replacement ◇◆◇

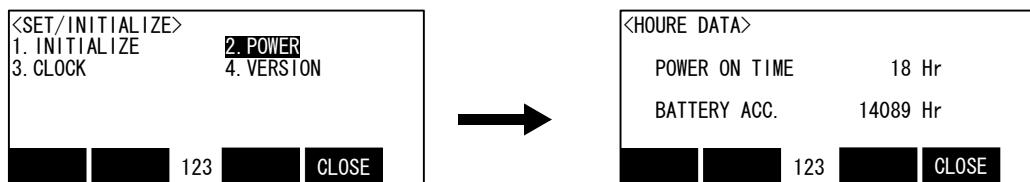
The battery usage time is calculated in the controller, and a caution message is displayed when the battery is spent. Always initialize the battery consumption time after replacing the battery to ensure that the caution message is displayed correctly.

If this initialization is carried out when the battery has not been replaced, the display timing of the caution message will deviate. Thus, carry this step out only when the battery has been replaced.

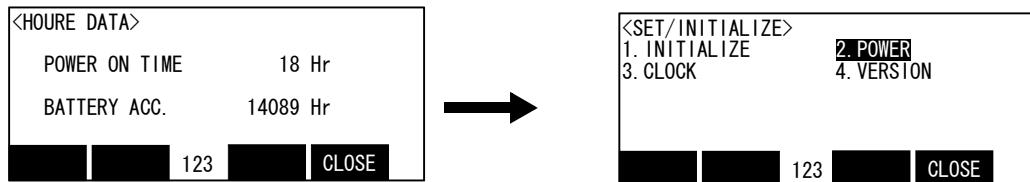
(4) Operation

Display the accumulation time of the power supply ON, and the remaining time of the battery.

- 1) Press the [2] key in the set/initial screen, and display the operation screen.



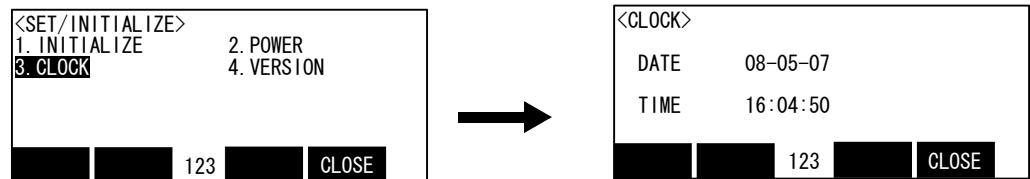
- 2) Press the function key corresponding to "Close", and return to the set/initial screen.



(5) Time setup

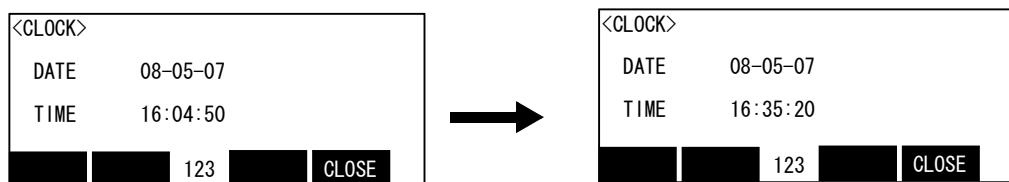
Display of the date and time, the setup

- 1) Press the [3] key in the set/initial screen, and display the time screen.

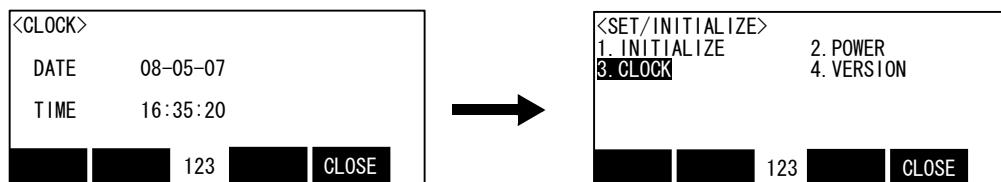


- 2) Date and time can be setup on the time screen.

Move the cursor by the arrow key and input the current date and time.



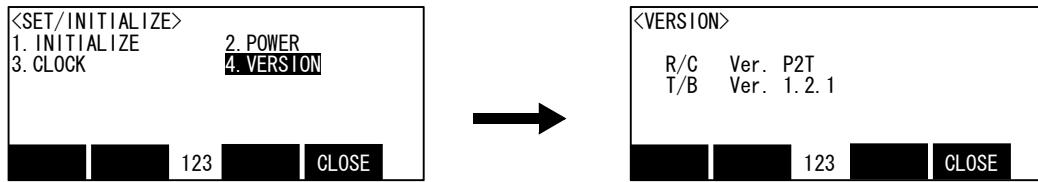
3) Press the function key corresponding to "Close", and return to the set/initial screen.



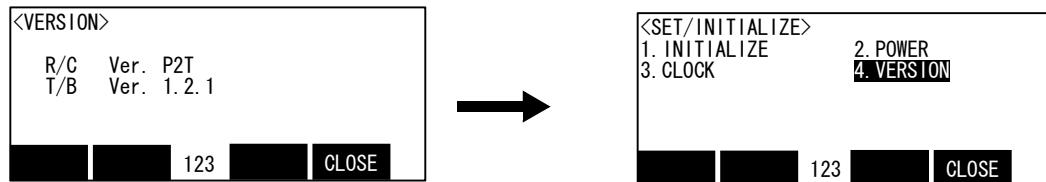
(6) Version

Display the software version of the controller and the teaching pendant

1) Press the [4] key in the set/initial screen, and display the version screen.



2) Press the function key corresponding to "Close", and return to the set/initial screen.



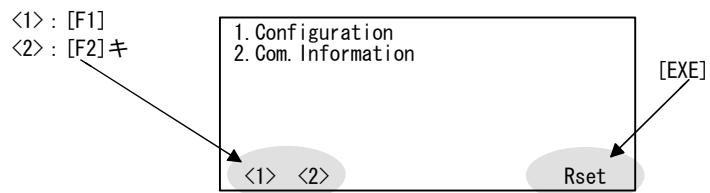
3.17 Operation of the initial-setting screen

There is the function of initial setting shown in the following.

(1)Setup of the display language The character displayed on the T/B can be set to either Japanese or English.

(2)Adjustment of contrast The brightness of the screen of T/B can be adjusted in the 16 steps.

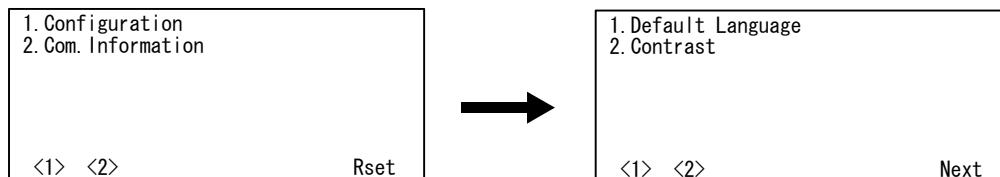
Operate this operation on the initial-setting screen displayed at turning on the control power in the condition of pushing both of [F1] key and [F3] key of T/B.



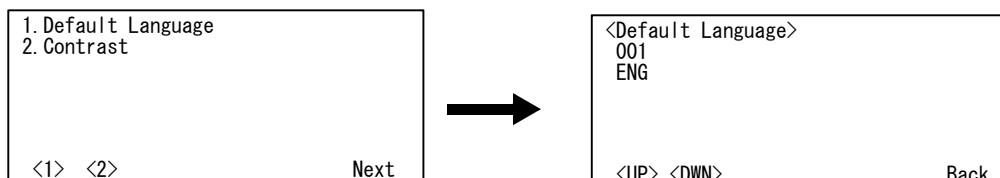
(1) Set the display language

The character displayed on the T/B can be set to either Japanese or English.

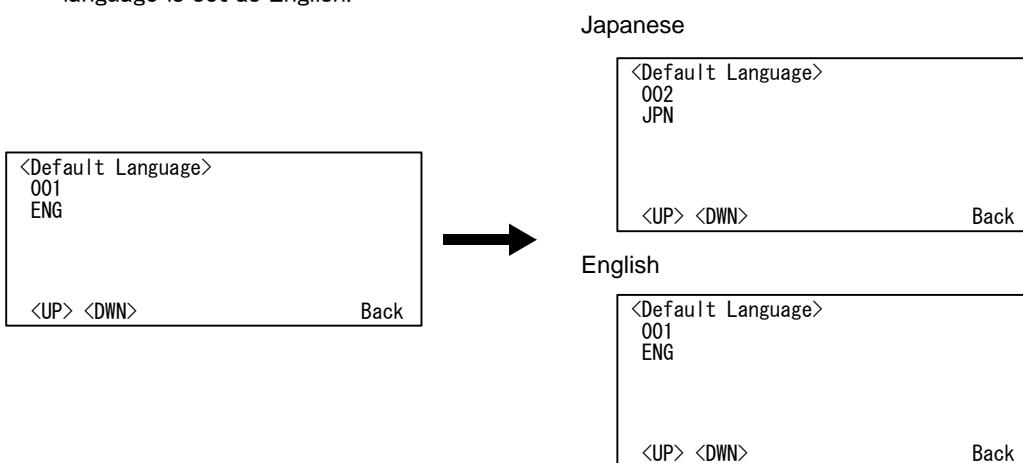
1) Press the [F1] key in the initial-setting screen, and select "1.Configuration"



2) Press the [F1] key , and select "1.Default Language"

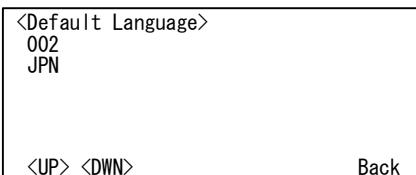


3) Display the "JPN" by [F1] or [F2] key, then language is set as Japanese. And, display the "ENG" , then language is set as English.

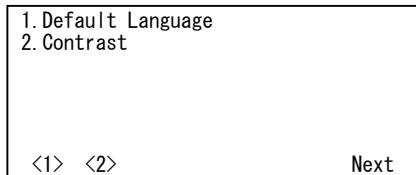
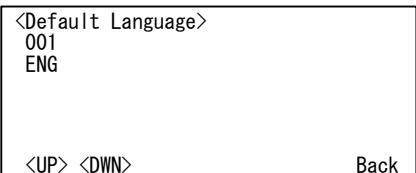


4) Press the [EXE] key, and fix it.

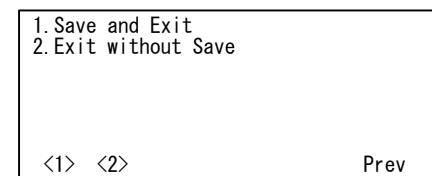
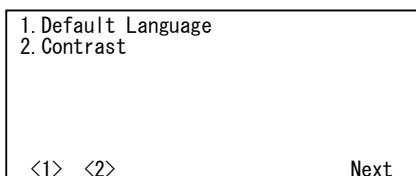
Japanese



English

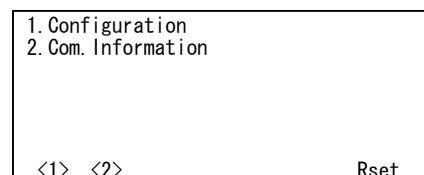
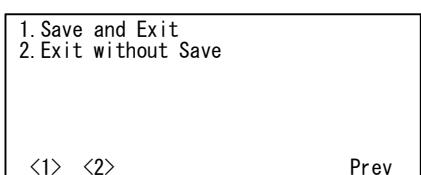


5) Press the [EXE] key, and display finish screen

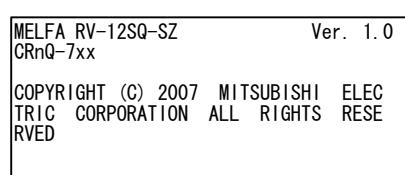
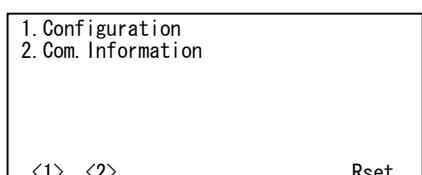


6) Press the [F1] key, and save the setup.

If not saved, press the [F2] key. All return to the initial-setting screen.
And, the setup can be done over again if the [EXE] key is pressed.



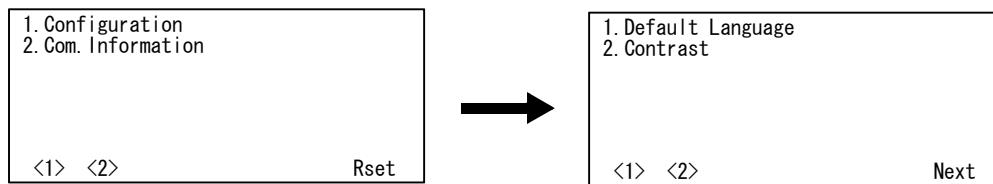
7) T/B starts in the language set up when the [EXE] key was pressed.



(2) Adjustment of contrast

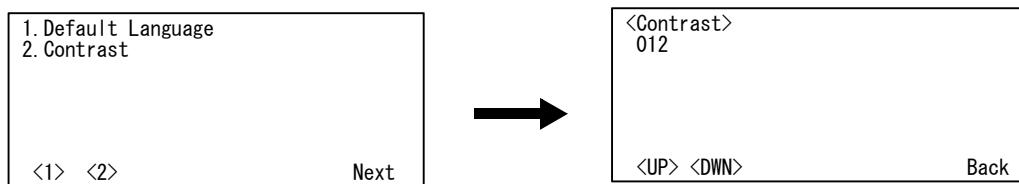
The brightness of the screen of T/B can be adjusted in the 16 steps.

1) Press the [F1] key in the initial-setting screen, and select "1.Configuration"

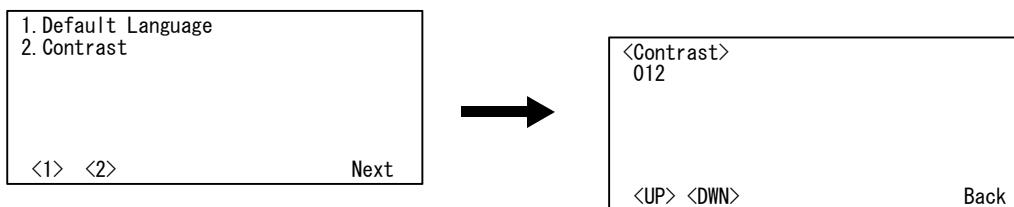


2) Press the [F2] key and select "1.Contrast."

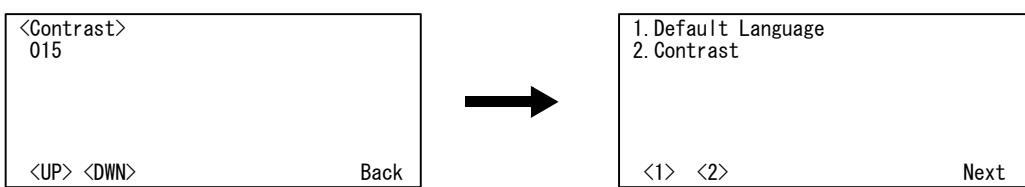
3) The brightness set up now is displayed as the numerical value of 0 to 15.



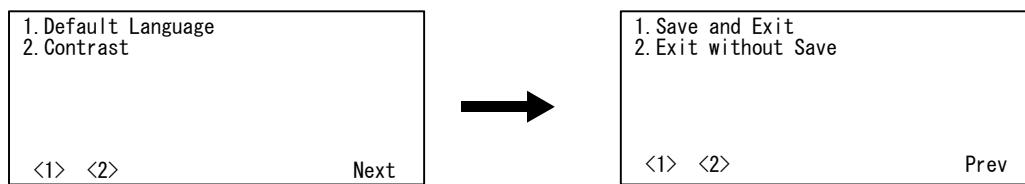
4) If it makes the screen bright, the [F1] key is pressed, if it makes it dark, press the [F2] key, and set it as the good brightness. It becomes so bright that the numerical value is large. .



5) Press the [EXE] key and fix it



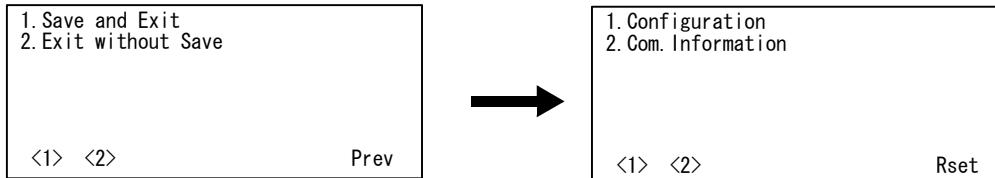
6) Press the [EXE] key, and display finish screen



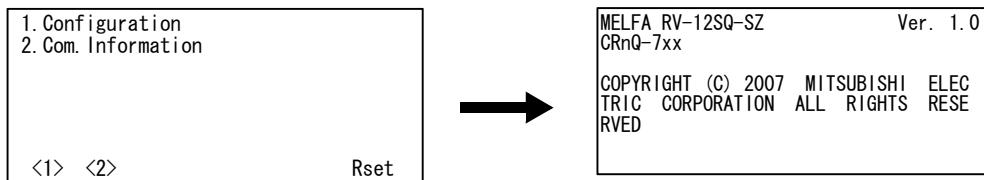
7) Press the [F1] key, and save the setup.

If not saved, press the [F2] key. All return to the initial-setting screen.

And, the setup can be done over again if the [EXE] key is pressed.



8) T/B starts in the contrast set up when the [EXE] key was pressed.



4 MELFA-BASIC V

In this chapter, the functions and the detailed language specification of the programming language "MELFA-BASIC V" are explained.

4.1 MELFA-BASIC V functions

The outline of the programming language "MELFA-BASIC V" is explained in this section. The basic movement of the robot, signal input/output, and conditional branching methods are described.

Table 4-1:List of items described

	Item	Details	Related instructions, etc.
1	4.1.1Robot operation control	(1)Joint interpolation movement	Mov
2		(2)Linear interpolation movement	Mvs
3		(3)Circular interpolation movement	Mvr, Mvr2, Mvr3, Mvc
4		(4)Continuous movement	Cnt
5		(5)Acceleration/deceleration time and speed control	Accel, Oadl
6		(6)Confirming that the target position is reached	Fine, Mov and Dly
7		(7)High path accuracy control	Prec
8		(8)Hand and tool control	HOpen, HClose, Tool
9	4.1.2Pallet operation	-----	Def Plt, Plt
10	4.1.3Program control	(1)Unconditional branching, conditional branching, waiting	GoTo, If Then Else, Wait, etc
11		(2)Repetition	For Next, While WEnd
12		(3)Interrupt	Def Act, Act
13		(4)Subroutine	GoSub, CallP, On GoSub, etc
14		(5)Timer	Dly
15		(6)Stopping	End(Pause for one cycle), Hlt
16	4.1.4Inputting and outputting external signals	(1)Input signals	M_In, M_Inb, M_Inw, etc
17		(2)Output signals	M_Out, M_Outb, M_Outw, etc
18	4.1.5Communication Note 1)	-----	Open, Close, Print, Input, etc
19	4.1.6Expressions and operations	(1)List of operator	+, -, *, /, <>, <, >, etc
20		(2)Relative calculation of position data (multiplication)	P1 * P2
21		(3)Relative calculation of position data (Addition)	P1 + P2
22	4.1.7Appended statement	-----	Wth, WthIf

Note 1)Cannot use in CRnQ series.

For the detailed description of each instruction, please refer to [Page 153, "4.11 Detailed explanation of command words"](#).

4.1.1 Robot operation control

(1) Joint interpolation movement

The robot moves with joint axis unit interpolation to the designated position. (The robot interpolates with a joint axis unit, so the end path is irrelevant.)

*Command word

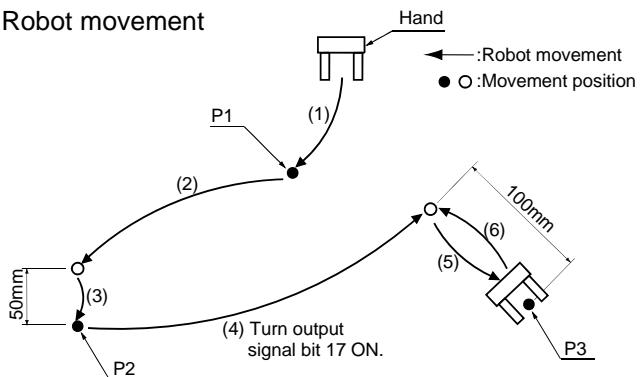
Command word	Explanation
Mov	The robot moves to the designated position with joint interpolation. It is possible to specify the interpolation form using the TYPE instruction. An appended statement Wth or Wthlf can be designated

*Statement example

Statement example	Explanation
Mov P1	' Moves to P1.
Mov P1+P2	' Moves to the position obtained by adding the P1 and P2 coordinate elements. Refer to Page 115 .
Mov P1*P2.....	' Moves to the position relatively converted from P1 to P2. Refer to Page 115 .
Mov P1,-50 *1).....	' Moves from P1 to a position retracted 50mm in the hand direction.
Mov P1 Wth M_Out(17)=1.....	' Starts movement toward P1, and simultaneously turns output signal bit 17 ON.
Mov P1 Wthlf M_In(20)=1, Skip.....	' If the input signal bit 20 turns ON during movement to P1, the movement to P1 is stopped, and the program proceeds to the next stop.
Mov P1 Type 1, 0	' Specify either roundabout (or shortcut) when the operation angle of each axis exceeds 180 deg.. (Default value: Long way around)

*Program example

Robot movement



CAUTION *1) Specification of forward/backward movement of the hand

The statement examples and program examples are for a vertical 6-axis robot (e.g., RV-6SD). The hand advance/retrace direction relies on the Z axis direction (+/- direction) of the tool coordinate set for each model. Refer to the tool coordinate system shown in "Confirmation of movement" in the separate "From Robot unit setup to maintenance", and designate the correct direction.

•Program example

Program	Explanation
1 Mov P1	'(1) Moves to P1.
2 Mov P2, -50 *1)	'(2) Moves from P2 to a position retracted 50mm in the hand direction.
3 Mov P2	'(3) Moves to P2
4 Mov P3, -100 Wth M_Out (17) = 1	'(4) Starts movement from P3 to a position retracted 100mm in the hand direction, and turns ON output signal bit 17.
5 Mov P3	'(5) Moves to P3
6 Mov P3, -100 *1)	'(6) Returns from P3 to a position retracted 100mm in the hand direction.
7 End	'Ends the program.

*Related functions

Function	Explanation page
Designate the movement speed.....	Page 93, "(5) Acceleration/deceleration time and speed control"
Designate the acceleration/deceleration time.	Page 93, "(5) Acceleration/deceleration time and speed control"
Confirm that the target position is reached.	Page 95, "(6) Confirming that the target position is reached"
Continuously move to next position without stopping at target position.....	Page 92, "(4) Continuous movement"
Move linearly.	Page 89, "(2) Linear interpolation movement"
Move while drawing a circle or arc.	Page 90, "(3) Circular interpolation movement"
Add a movement command to the process.....	Page 257, " Wth (With)"

(2) Linear interpolation movement

The end of the hand is moved with linear interpolation to the designated position.

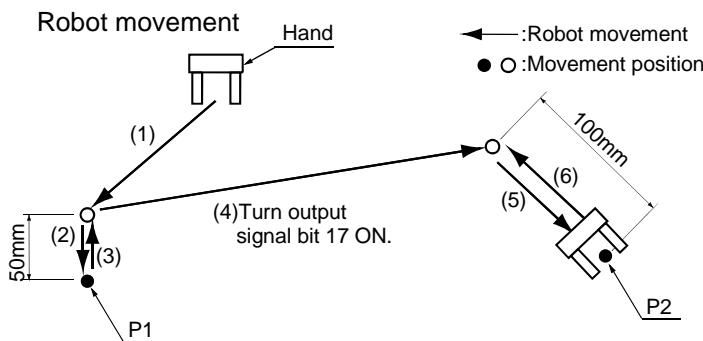
*Command word

Command word	Explanation
Mvs	The robot moves to the designated position with linear interpolation. It is possible to specify the interpolation form using the TYPE instruction. An appended statement Wth or Wthlf can be designated.

*Statement example

Statement example	Explanation
Mvs P1	' Moves to P1
Mvs P1+P2	' Moves to the position obtained by adding the P1 and P2 coordinate elements. Refer to Page 115 .
Mvs P1*P2	' Moves to the position relatively converted from P1 to P2.
Mvs P1, -50 *1)	' Moves from P1 to a position retracted 50mm in the hand direction.
Mvs , -50 *1)	' Moves from the current position to a position retracted 50mm in the hand direction.
Mvs P1 Wth M_Out(17)=1	' Starts movement toward P1, and simultaneously turns output signal bit 17 ON.
Mvs P1 Wthlf M_In(20)=1, Skip	' If the input signal bit 20 turns ON during movement to P1, the movement to P1 is stopped, and the program proceeds to the next stop.
Mvs P1 Type 0, 0	' Moves to P1 with equivalent rotation
Mvs P1 Type 9, 1	' Moves to P1 with 3-axis orthogonal interpolation.

*Program example



CAUTION *1) Specification of forward/backward movement of the hand

The statement examples and program examples are for a vertical 6-axis robot (e.g., RV-6SD). The hand advance/retrace direction relies on the Z axis direction (+/- direction) of the tool coordinate set for each model. Refer to the tool coordinate system shown in "Confirmation of movement" in the separate "From Robot unit setup to maintenance", and designate the correct direction.

•Program example

Program	Explanation
1 Mvs P1, -50 *1)	' (1) Moves with linear interpolation from P1 to a position retracted 50mm in the hand direction.
2 Mvs P1	' (2) Moves to P1 with linear interpolation.
3 Mvs , -50 *1)	' (3) Moves with linear interpolation from the current position (P1) to a position retracted 50mm in the hand direction.
4 Mvs P2, -100 Wth M_Out(17)=1 *1)	(4) Output signal bit 17 is turned on at the same time as the robot starts moving.
5 Mvs P2	(5) Moves with linear interpolation to P2.
6 Mvs , -100 *1)	(6) Moves with linear interpolation from the current position (P2) to a position retracted 50mm in the hand direction.
7 End	' Ends the program.

*Related functions

Function	Explanation page
Designate the movement speed	Page 93, "(5) Acceleration/deceleration time and speed control"
Designate the acceleration/deceleration time	Page 93, "(5) Acceleration/deceleration time and speed control"
Confirm that the target position is reached	Page 95, "(6) Confirming that the target position is reached"
Continuously move to next position without stopping at target position....	Page 92, "(4) Continuous movement"
Move with joint interpolation.....	Page 88, "(1) Joint interpolation movement"
Move while drawing a circle or arc.....	Page 90, "(3) Circular interpolation movement"
Add a movement command to the process	Page 257, "Wth (With)"

(3) Circular interpolation movement

The robot moves along an arc designated with three points using three-dimensional circular interpolation. If the current position is separated from the start point when starting circular movement, the robot will move to the start point with linear operation and then begin circular interpolation.

*Command word

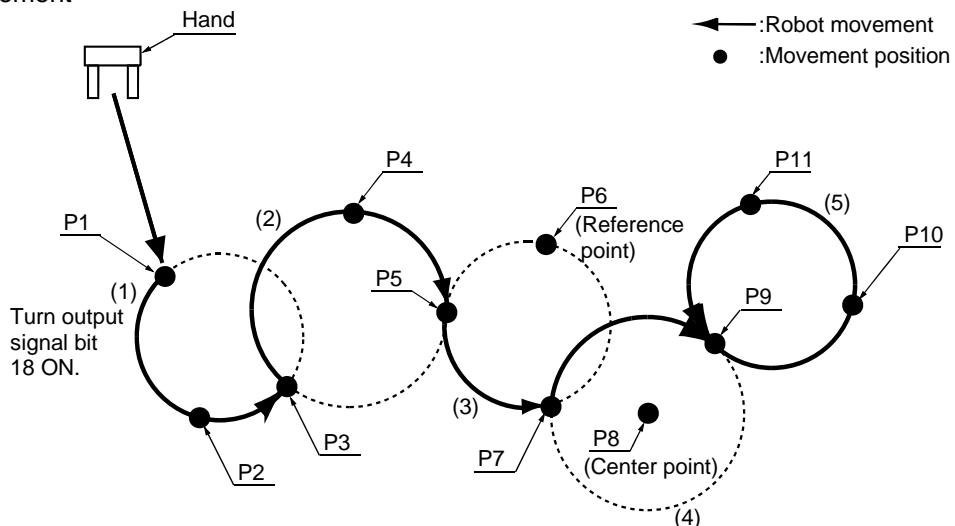
Command word	Explanation
Mvr	Designates the start point, transit point and end point, and moves the robot with circular interpolation in order of the start point - transit point - end point. It is possible to specify the interpolation form using the TYPE instruction. An appended statement Wth or Wthlf can be designated.
Mvr2	Designates the start point, end point and reference point, and moves the robot with circular interpolation from the start point - end point without passing through the reference point. It is possible to specify the interpolation form using the TYPE instruction. An appended statement Wth or Wthlf can be designated.
Mvr3	Designates the start point, end point and center point, and moves the robot with circular interpolation from the start point to the end point. The fan angle from the start point to the end point is 0 deg. < fan angle < 180 deg. It is possible to specify the interpolation form using the TYPE instruction. An appended statement Wth or Wthlf can be designated.
Mvc	Designates the start point (end point), transit point 1 and transit point 2, and moves the robot with circular interpolation in order of the start point - transit point 1 - transit point 2 - end point. An appended statement Wth or Wthlf can be designated.

*Statement example

Statement example	Explanation
Mvr P1, P2, P3	' Moves with circular interpolation between P1 - P2 - P3.
Mvr P1, P2, P3 Wth M_Out (17) = 1	' Circular interpolation between P1 - P2 - P3 starts, and the output signal bit 17 turns ON.
Mvr P1, P2, P3 Wthlf M_In (20) = 1, Skip	' If the input signal bit 20 turns ON during circular interpolation between P1 - P2 - P3, circular interpolation to P1 is stopped, and the program proceeds to the next step.
Mvr P1, P2, P3 TYPE 0, 1	' Moves with circular interpolation between P1 - P2 - P3.
Mvr2 P1, P3, P11	' Circular interpolation is carried out from P1 to P3 in the direction that P11 is not passed. P11 is the reference point.
Mvr3 P1, P3, P10	' Moves with circular interpolation from P1 to P3 in the direction with the smallest fan angle. P10 is the center point.
Mvc P1, P2, P3	' Moves with circular movement from P1 - P2 - P3 - P1.

*Program example

Robot movement



•Program example

Program	Explanation
1 Mvr P1, P2, P3 Wth M_Out(18) = 1	' (1) Moves between P1 - P2 - P3 as an arc. The robot current position before movement is separated from the start point, so first the robot will move with linear operation to the start point. (P1) output signal bit 18 turns ON simultaneously with the start of circular movement.
2 Mvr P3, P4, P5	' (2) Moves between P3 - P4 - P5 as an arc.
3 Mvr2 P5, P7, P6	' (3) Moves as an arc over the circumference on which the start point (P5), reference point (P6) and end point (P7) in the direction that the reference point is not passed between the start point and end point.
4 Mvr3 P7, P9, P8	' (4) Moves as an arc from the start point to the end point along the circumference on which the center point (P8), start point (P7) and end point (P9) are designated.
5 Mvc P9, P10, P11	' (5) Moves between P9 - P10 - P11 - P9 as an arc. The robot current position before movement is separated from the start point, so first the robot will move with linear operation to the start point.(1 cycle operation)
6 End	' Ends the program.

*Related functions

Function	Explanation page
Designate the movement speed.	Page 93, "(5) Acceleration/deceleration time and speed control"
Designate the acceleration/deceleration time.	Page 93, "(5) Acceleration/deceleration time and speed control"
Confirm that the target position is reached.	Page 95, "(6) Confirming that the target position is reached"
Continuously move to next position without stopping at target position.....	Page 92, "(4) Continuous movement"
Move with joint interpolation.....	Page 88, "(1) Joint interpolation movement"
Move linearly.....	Page 89, "(2) Linear interpolation movement"
Add a movement command to the process.	Page 257, " Wth (With)"

(4) Continuous movement

The robot continuously moves to multiple movement positions without stopping at each movement position. The start and end of the continuous movement are designated with the command statement. The speed can be changed even during continuous movement.

*Command word

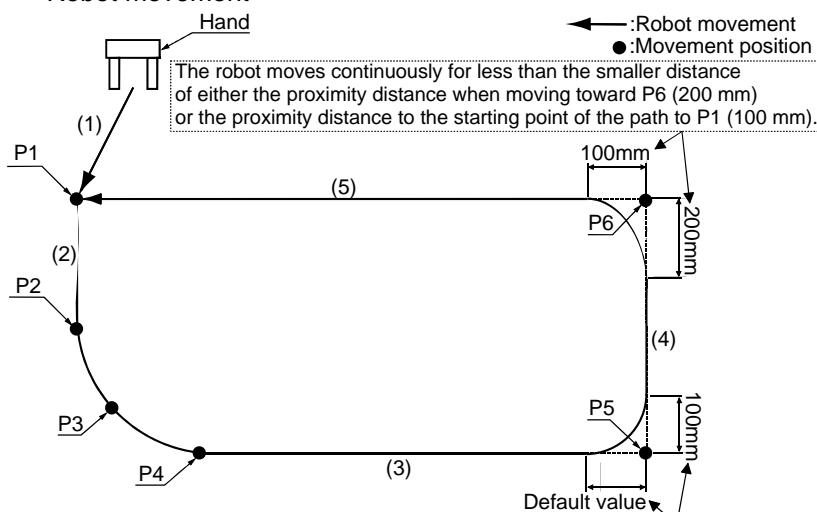
Command word	Explanation
Cnt	Designates the start and end of the continuous movement.

*Statement example

Statement example	Explanation
Cnt 1	Designates the start of the continuous movement.
CNT 1, 100, 200	Designates the start of the continuous movement, and designates that the start point neighborhood distance is 100mm, and the end point neighborhood distance is 200mm.
CNT 0	Designates the end of the continuous movement.

*Program example

Robot movement



CAUTION

*1) Specification of forward/backward movement of the hand

The statement examples and program examples are for a vertical 6-axis robot (e.g., RV-6SD). The hand advance/retrace direction relies on the Z axis direction (+/- direction) of the tool coordinate set for each model. Refer to the tool coordinate system shown in "Confirmation of movement" in the separate "From Robot unit setup to maintenance", and designate the correct direction.

•Program example

Program	Explanation
1 Mov P1	' (1) Moves with joint interpolation to P1.
2 Cnt 1	' Validates continuous movement. (Following movement is continuous movement.)
3 Mvr P2, P3, P4	' (2) Moves linearly to P2, and continuously moves to P4 with arc movement.
4 Mvs P5	' After arc movement, moves linearly to P5.
5 Cnt 1, 200, 100	' (3) Sets the continuous movement's start point neighborhood distance to 200mm, and the end point neighborhood distance to 100mm.
6 Mvs P6	' (4) After moving to previous P5, moves in succession linearly to P6.
7 Mvs P1	' (5) Continuously moves to P1 with linear movement.
8 Cnt 0	' Invalidates the continuous movement.
9 End	' Ends the program.

*Related functions

Function	Explanation page
Designate the movement speed.	Page 93, "(5) Acceleration/deceleration time and speed control"
Designate the acceleration/deceleration time.	Page 93, "(5) Acceleration/deceleration time and speed control"
Confirm that the target position is reached.	Page 95, "(6) Confirming that the target position is reached"
Move with joint interpolation.	Page 88, "(1) Joint interpolation movement"
Move linearly.	Page 89, "(2) Linear interpolation movement"
Move while drawing a circle or arc.	Page 90, "(3) Circular interpolation movement"

(5) Acceleration/deceleration time and speed control

The percentage of the acceleration/deceleration in respect to the maximum acceleration/deceleration, and the movement speed can be designated.

*Command word

Command word	Explanation
Accel	Designates the acceleration during movement and the deceleration as a percentage (%) in respect to the maximum acceleration/deceleration speed.
Ovrd	Designates the movement speed applied on the entire program as a percentage (%) in respect to the maximum speed.
JOvrd	Designates the joint interpolation speed as a percentage (%) in respect to the maximum speed.
Spd	Designate the linear and circular interpolation speed with the hand end speed (mm/s).
Oadl	This instruction specifies whether the optimum acceleration/deceleration function should be enabled or disabled.

*Statement example

Statement example	Explanation
Accel.....	Sets both the acceleration and deceleration to 100%.
Accel 60, 80.....	Sets the acceleration to 60% and the deceleration to 80%. (For maximum acceleration/deceleration is 0.2 sec. acceleration 0.2/0.6=0.33 sec. deceleration 0.2/0.8=0.25 sec.)
Ovrd 50.....	Sets the joint interpolation, linear interpolation and circular interpolation to 50% of the maximum speed.
JOvrd 70.....	Set the joint interpolation operation to 70% of the maximum speed.
Spd 30	Sets the linear interpolation and circular interpolation speed to 30mm/s.
Oadl ON	This instruction enables the optimum acceleration/deceleration function.

*Movement speed during joint interpolation

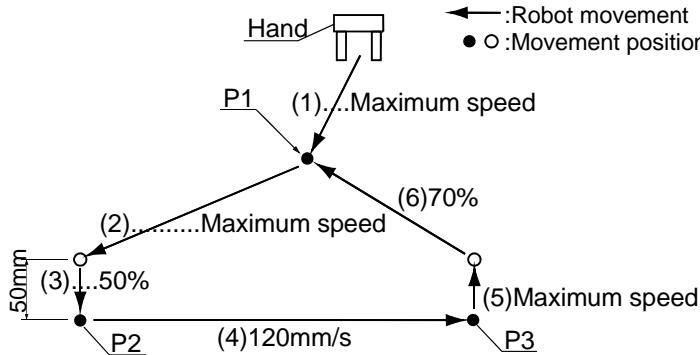
Controller (T/B) setting value x Ovrd command setting value x JOvrd command setting value.

*Movement speed during linear and circular interpolation

Controller (T/B) setting value x Ovrd command setting value x Spd command setting value.

*Program example

Robot movement



CAUTION

¹⁾ Specification of forward/backward movement of the hand

The statement examples and program examples are for a vertical 6-axis robot (e.g., RV-6SD). The hand advance/retrace direction relies on the Z axis direction (+/- direction) of the tool coordinate set for each model. Refer to the tool coordinate system shown in "Confirmation of movement" in the separate "From Robot unit setup to maintenance", and designate the correct direction.

•Program example

Program	Explanation
1 Ovrd 100	' Sets the movement speed applied on the entire program to the maximum speed.
2 Mvs P1	' (1) Moves at maximum speed to P1.
3 Mvs P2, -50 *1)	' (2) Moves at maximum speed from P2 to position retracted 50mm in hand direction.
4 Ovrd 50	' Sets the movement speed applied on the entire program to half of the maximum speed.
5 Mvs P2	' (3) Moves linearly to P2 with a speed half of the default speed.
6 Spd 120	' Sets the end speed to 120mm/s. (Since the override is 50%, it actually moves at 60 mm/s.)
7 Ovrd 100	' Sets the movement speed percentage to 100% to obtain the actual end speed of 120mm/s.
8 Accel 70, 70	' Sets the acceleration and deceleration to 70% of the maximum speed.
9 Mvs P3	' (4) Moves linearly to P3 with the end speed 120mm/s.
10 Spd M_NSpd	' Returns the end speed to the default value.
11 JOvrd 70	' Sets the speed for joint interpolation to 70%.
12 Accel	' Returns both the acceleration and deceleration to the maximum speed.
13 Mvs , -50 *1)	' (5) Moves linearly with the default speed for linear movement from the current position (P3) to a position retracted 50mm in the hand direction.
14 Mvs P1	' (6) Moves to P1 at 70% of the maximum speed.
15 End	' Ends the program.

*Related functions

Function	Explanation page
Move with joint interpolation.....	Page 88, "(1) Joint interpolation movement"
Move linearly.....	Page 89, "(2) Linear interpolation movement"
Move while drawing a circle or arc.....	Page 90, "(3) Circular interpolation movement"
Continuously move to next position without stopping at target position.....	Page 92, "(4) Continuous movement"

(6) Confirming that the target position is reached

The positioning finish conditions can be designated with as No. of pulses. (Fine instruction) This designation is invalid when using continuous movement.

*Command word

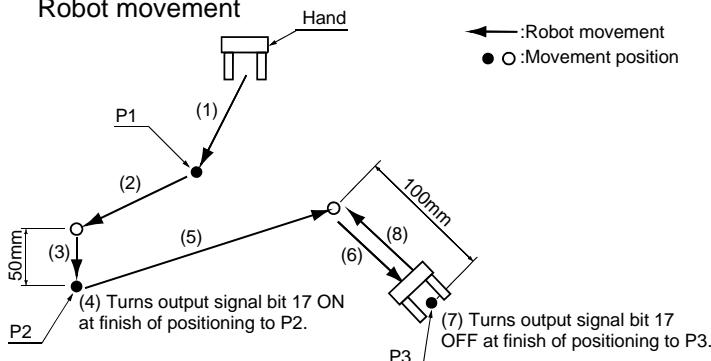
Command word	Explanation
Fine	Designates the positioning finish conditions with a No. of pulses. Specify a small number of pulses to allow more accurate positioning.
Mov and Dly	After the Mov movement command, command the Dly instruction (timer) to complete positioning .

*Statement example

Statement example	Explanation
Fine100	Sets the positioning finish conditions to 100 pulses.
Mov P1	Moves with joint interpolation to P1. (The movement completes at the command value level.)
Dly 0.1	Positioning after the movement instruction is performed by the timer.

*Program example

Robot movement



CAUTION *1) Specification of forward/backward movement of the hand
 The statement examples and program examples are for a vertical 6-axis robot (e.g., RV-6SD). The hand advance/retrace direction relies on the Z axis direction (+/- direction) of the tool coordinate set for each model. Refer to the tool coordinate system shown in "Confirmation of movement" in the separate "From Robot unit setup to maintenance", and designate the correct direction.

•Program example

Program	Explanation
1 Cnt 0	' The Fine instruction is valid only when the Cnt instruction is OFF.
2 Mvs P1	' (1) Moves with joint interpolation to P1.
3 Mvs P2, -50 *1)	' (2) Moves with joint interpolation from P2 to position retracted 50mm in hand direction.
4 Fine 50	' Sets positioning finish pulse to 50.
5 Mvs P2	' (3) Moves with linear interpolation to P2 (Mvs completes if the positioning complete pulse count is 50 or less.)
6 M_Out(17)=1	' (4) Turns output signal 17 ON when positioning finish pulse reaches 50 pulses.
7 Fine 1000	' Sets positioning finish pulse to 1000.
8 Mvs P3, -100 *1)	' (5) Moves linearly from P3 to position retracted 100mm in hand direction.
9 Mvs P3	' (6) Moves with linear interpolation to P3.
10 Dly 0.1	' Performs the positioning by the timer.
11 M_Out(17)=0	' (7) Turns output signal 17 off.
12 Mvs , -100 *1)	' (8) Moves linearly from current position (P3) to position retracted 100mm in hand direction.
13 End	' Ends the program.

*Related functions

Function	Explanation page
Move with joint interpolation.....	Page 90, "(3) Circular interpolation movement"
Move linearly.....	Page 89, "(2) Linear interpolation movement"
Continuously move to next position without stopping at target position.....	Page 92, "(4) Continuous movement"

(7) High path accuracy control

It is possible to improve the motion path tracking when moving the robot. This function is limited to certain types of robot. Currently, the Prec instruction is available for vertical multi-joint type 5-axis and 6-axis robots.

*Command word

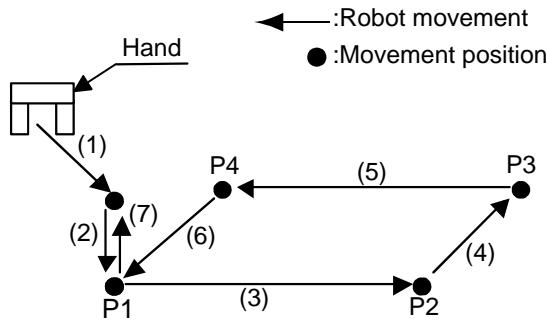
Command word	Explanation
Prec	This instruction specifies whether the high path accuracy mode should be enabled or disabled.

*Statement example

Statement example	Explanation
Prec On	Enables the high path accuracy mode.
Prec Off.....	Disables the high path accuracy mode.

*Program example

Robot movement



CAUTION

^{*1) Specification of forward/backward movement of the hand}

*1) The statement examples and program examples are for a vertical 6-axis robot (e.g., RV-6SD). The hand advance/retrace direction relies on the Z axis direction (+/- direction) of the tool coordinate set for each model. Refer to the tool coordinate system shown in "Confirmation of movement" in the separate "From Robot unit setup to maintenance", and designate the correct direction.

•Program example

Program	Explanation
1 Mov P1, -50 *1)	' (1) Moves with joint interpolation from P1 to position retracted 50mm in hand direction.
2 Ovrd 50	' Sets the movement speed to half of the maximum speed.
3 Mvs P1	' (2) Moves with linear interpolation to P1.
4 Prec On	' The high path accuracy mode is enabled.
5 Mvs P2	' (3) Moves the robot from P1 to P2 with high path accuracy.
6 Mvs P3	' (4) Moves the robot from P2 to P3 with high path accuracy.
7 Mvs P4	' (5) Moves the robot from P3 to P4 with high path accuracy.
8 Mvs P1	' (6) Moves the robot from P4 to P1 with high path accuracy.
9 Prec Off	' The high path accuracy mode is disabled.
10 Mvs P1, -50 *1)	' (7) Returns the robot to the position 50 mm behind P1 in the hand direction using linear interpolation.
11 End	' Ends the program.

CAUTION

The Prec instruction improves the tracking accuracy of the robot's hand tip, but lowers the acceleration/deceleration of the robot movement, which means that the cycle time may become longer. The tracking accuracy will be further improved if the Cnt instruction is not included. However, the hand tip speed cannot be guaranteed in this case.

(8) Hand and tool control

The hand open/close state and tool shape can be designated.

*Command word

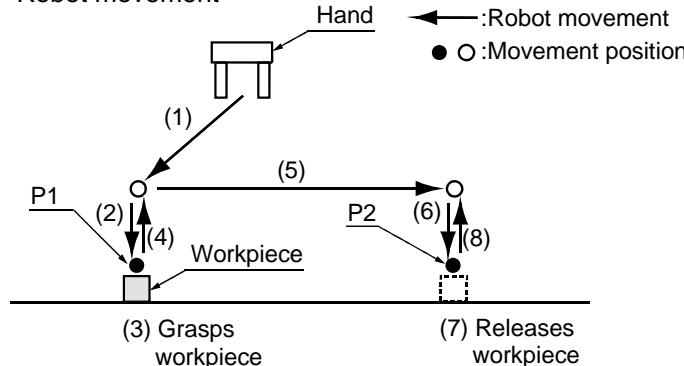
Command word	Explanation
HOpen	Opens the designated hand.
HClose	Closes the designated hand.
Tool	Sets the shape of the tool being used, and sets the control point.

*Statement example

Statement example	Explanation
HOpen 1.....	Opens hand 1.
HOpen 2.....	Opens hand 2.
HClose 1.....	Closes hand 1.
HClose 2.....	Closes hand 2.
Tool (0, 0, 95, 0, 0, 0)	Sets the robot control point to the position 95 mm from the flange plane in the extension direction.

*Program example

Robot movement



CAUTION *1) Specification of forward/backward movement of the hand

The statement examples and program examples are for a vertical 6-axis robot (e.g., RV-6SD). The hand advance/retrace direction relies on the Z axis direction (+/- direction) of the tool coordinate set for each model. Refer to the tool coordinate system shown in "Confirmation of movement" in the separate "From Robot unit setup to maintenance", and designate the correct direction.

•Program example

Program	Explanation
1 Tool(0, 0, 95, 0, 0, 0)	'Sets the hand length to 95 mm.
2 Mvs P1, -50 *1)	'(1) Moves with joint interpolation from P1 to position retracted 50mm in hand direction.
3 Ovrd 50	'Sets the movement speed to half of the maximum speed.
4 Mvs P1	'(2) Moves with linear interpolation to P1. (Goes to grasp workpiece.)
5 Dly 0.5	'Wait for the 0.5 seconds for the completion of arrival to the target position.
6 HClose 1	'(3) Closes hand 1. (Grasps workpiece.)
7 Dly 0.5	'Waits 0.5 seconds.
8 Ovrd 100	'Sets movement speed to maximum speed.
9 Mvs , -50 *1)	'(4) Moves linearly from current position (P1) to position retracted 50mm in hand direction. (Lifts up workpiece.)
10 Mvs P2, -50 *1)	'(5) Moves with joint interpolation from P2 to position retracted 50mm in hand direction.
11 Ovrd 50	'Sets movement speed to half of the maximum speed.
12 Mvs P2	'(6) Moves with linear interpolation to P2. (Goes to place workpiece.)
13 Dly 0.5	'Wait for the 0.5 seconds for the completion of arrival to the target position.
14 HOpen 1	'(7) Opens hand 1. (Releases workpiece.)
15 Dly 0.5	'Waits 0.5 seconds.
16 Ovrd 100	'Sets movement speed to maximum speed.
17 MVS , -50 *1)	'(8) Moves linearly from current position (P2) to position retracted 50mm in hand direction. (Separates from workpiece.)
18 End	'Ends the program.

*Related functions

Function	Explanation page
Appended statement.....	Page 257, "Wth (With)"

4.1.2 Pallet operation

When carrying out operations with the workpieces neatly arranged (palletizing), or when removing workpieces that are neatly arranged (depalletizing), the pallet function can be used to teach only the position of the reference workpiece, and obtain the other positions with operations.

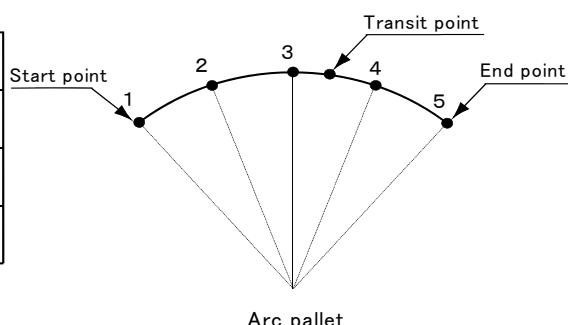
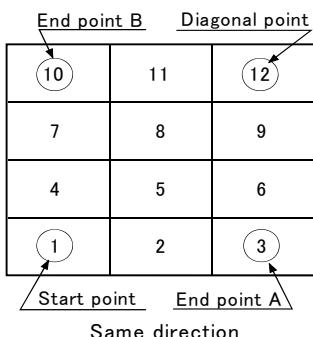
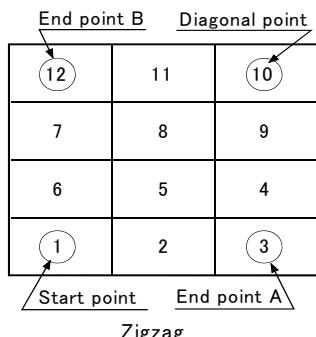
*Command word

Command word	Explanation
Def Plt	Defines the pallet to be used.
Plt	Obtains the designated position on the pallet with operations.

*Statement example

Statement example	Explanation
Def Plt 1, P1, P2, P3, P4, 4, 3, 1	Defines to operate pallet No. 1 with a start point = P1, end point A = P2, end point B = P3 and diagonal point = P4, a total of 12 work positions (quantity A = 4, quantity B = 3), and a pallet pattern = 1(Zigzag).
Def Plt 2, P1, P2, P3, , 8, 5, 2.....	Defines to operate pallet No. 2 with a start point = P1, end point A = P2, and end point B = P3, a total of 40 work positions (quantity A = 8, quantity B = 5), and a pallet pattern = 2 (Same direction).
Def Plt 3, P1, P2, P3, , 5, 1, 3.....	Define that pallet No. 3 is an arc pallet having give five work positions on an arc designated with start point = P1, transit point = P2, end point = P3 (total three points).
(Plt1, 5)	Operate the 5th position on pallet No. 1.
(Plt1, M1)	Operate position in pallet No. 1 indicated with the numeric variable M1.

The relation of the position designation and a pallet pattern is shown below.



<Precautions on the posture of position data in a pallet definition>

⚠ CAUTION

Please read "[*Explanation](#)" below if you use position data whose posture components (A, B and C) are approximately $+\/-180$ degrees as the start point, end points A and B, or the diagonal point.

*Explanation

At a position where a posture component (A, B and C) reaches 180 degrees, the component value can become either +180 degrees or -180 degrees even if the posture is the same. This is due to internal operation errors, and there is no consistency in which sign is employed.

If this position is used for the start point, end points A and B or diagonal point of the pallet definition and the same posture component values include both +180 degrees and -180 degrees, the hand will rotate and move in unexpected ways because the pallet grid positions are calculated by dividing the distance between -180 degrees and +180 degrees.

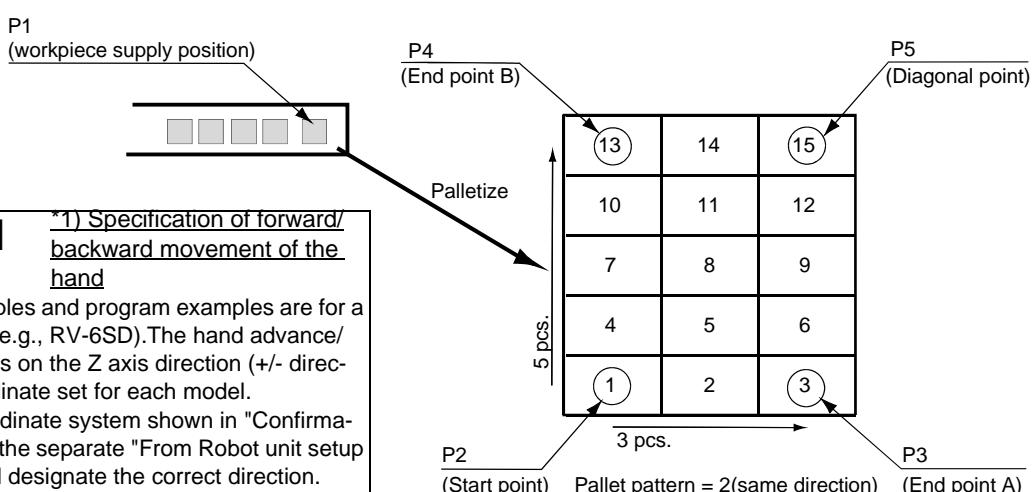
Whether a posture component is +180 degrees or -180 degrees, the posture will be the same. Use the same sign, either + or -, consistently for position data used to define a pallet.

Note also that similar phenomena can occur if posture components are close to $+\/-180$ degrees (e.g., +179 degrees and -179 degrees) as well, if different signs are used. In this case, add or subtract 360 degrees to/ from the posture components and correct the values such that the sign becomes the same. (For example, to change the sign of -179 degrees to +, add 360 degrees and correct the value to +181 degrees.)

"[Program example 1](#)" shows an example where the posture components of the end points (P3 and P4) and diagonal point (P5) are adjusted according to the start point (P2) when the hand direction is the same in all grid points of a pallet (values of the A, B and C axes are identical) (line numbers 10 to 90). "[Program example 2](#)" shows an example where values are corrected to have the same sign as the start point (P2) when the posture components of a pallet definition position are close to $+\/-180$ degrees and the C-axis values of the end points (P3 and P4) and diagonal point (P5) are either less than -178 degrees or greater than +178 degrees (line numbers 10 to 100). ($+\/-178$ degrees are set as the threshold values of correction.) Use these program examples as reference for cases where the pallet precision is not very high and the hand direction thus must be corrected slightly.

*Program example

Robot movement



⚠ CAUTION

The value of the start point of the pallet definition is employed for the structure flag of grid points (FL1 of position data) calculated by pallet operation (PIt instruction).

For this reason, if position data with different structure flags are used for each point of the pallet definition, the desired pallet operation cannot be obtained.

Use position data whose structure flag values are all the same for the start point, end points A and B and the diagonal point of the pallet definition. The value of the start position of the pallet definition is employed for the multi-rotation flag of grid points (FL2 of position data) as well. If position data with different multi-rotation flags are used for each point of the pallet definition, the hand will rotate and move in unexpected ways depending on the robot positions the pallet operation goes through and the type of interpolation instruction (joint interpolation, line interpolation, etc.). In such cases, use the TYPE argument of the interpolation instruction to set the detour/short cut operation of the posture properly and ensure that the hand moves as desired.

•Program example 1

The hand direction is the same in all grid points of a pallet (values of the A, B and C axes are identical)

Program	Explanation
1 P3.A=P2.A	'Assigns the posture component (A) of P2 to the posture component (A) of P3.
2 P3.B=P2.B	'Assigns the posture component (B) of P2 to the posture component (B) of P3.
3 P3.C=P2.C	'Assigns the posture component (C) of P2 to the posture component (C) of P3.
4 P4.A=P2.A	'Assigns the posture component (A) of P2 to the posture component (A) of P4.
5 P4.B=P2.B	'Assigns the posture component (B) of P2 to the posture component (B) of P4.
6 P4.C=P2.C	'Assigns the posture component (C) of P2 to the posture component (C) of P4.
7 P5.A=P2.A	'Assigns the posture component (A) of P2 to the posture component (A) of P5.
8 P5.B=P2.B	'Assigns the posture component (B) of P2 to the posture component (B) of P5.
9 P5.C=P2.C	'Assigns the posture component (C) of P2 to the posture component (C) of P5.
10 Def Plt 1, P2, P3, P4, P5, 3, 5, 2	'Defines the pallet. Pallet No. = 1, start point = P2, end point A = P3, end point B = P4, diagonal point = P5, quantity A = 3, quantity B = 5, pallet pattern = 2 (Same direction).
11 M1=1	'Substitutes value 1 in numeric variable M1. (M1 is used as a counter.)
12 *LOOP	'Designates label LOOP at the jump destination.
13 Mov P1, -50 *1)	'Moves with joint interpolation from P1 to a position retracted 50mm in hand direction.
14 Ovrd 50	'Sets movement speed to half of the maximum speed.
15 Mvs P1	'Moves linearly to P1. (Goes to grasp workpiece.)
16 HClose 1	'Closes hand 1. (Grasps workpiece.)
17 Dly 0.5	'Waits 0.5 seconds.
18 Ovrd 100	'Sets movement speed to maximum speed.
19 Mvs , -50 *1)	'Moves linearly from current position (P1) to a position retracted 50mm in hand direction. (Lifts up workpiece.)
20 P10=(Plt1,M1)	'Operates the position in pallet No. 1 indicated by the numeric variable M1, and substitutes the results in P10.
21 Mov P10, -50 *1)	'Moves with joint interpolation from P10 to a position retracted 50mm in hand direction.
22 Ovrd 50	'Sets movement speed to half of the maximum speed.
23 Mvs P10	'Moves linearly to P10. (Goes to place workpiece.)
24 HOpen 1	'Opens hand 1. (Places workpiece.)
25 Dly 0.5	'Waits 0.5 seconds.
26 Ovrd 100	'Sets movement speed to maximum speed.
27 Mvs , -50	'Moves linearly from current position (P10) to a position retracted 50mm in hand direction. (Separates from workpiece.)
28 M1=M1+1	'Increments numeric variable M1 by 1. (Advances the pallet counter.)
29 If M1<=15 Then *LOOP	'If numeric variable M1 value is less than 15, jumps to label LOOP and repeat process. If more than 15, goes to next step.
30 End	'Ends the program.

•Program example 2

Correction when posture components are close to +/-180 degrees

Program	Explanation
1 If Deg(P2.C)<0 Then GoTo *MINUS	'Checks the sign of the posture component (C) of P2 and, if it is - (negative), jump to the label MINUS line.
2 If Deg(P3.C)<-178 Then P3.C=P3.C+Rad(+360)	'If the posture component (C) of P3 is close to -180 degrees, adds 360 degrees to correct it to a positive value.
3 If Deg(P4.C)<-178 Then P4.C=P4.C+Rad(+360)	'If the posture component (C) of P4 is close to -180 degrees, adds 360 degrees to correct it to a positive value.
4 If Deg(P5.C)<-178 Then P5.C=P5.C+Rad(+360)	'If the posture component (C) of P5 is close to -180 degrees, adds 360 degrees to correct it to a positive value.
5 GoTo *DEFINE	'Jumps unconditionally to the label DEFINE line.
6 *MINUS	'Specifies the label MINUS line as the jump destination.
7 If Deg(P3.C)<+178 Then P3.C=P3.C-Rad(+360)	'If the posture component (C) of P3 is close to +180 degrees, adds 360 degrees to correct it to a negative value.
8 If Deg(P4.C)<+178 Then P4.C=P4.C-Rad(+360)	'If the posture component (C) of P4 is close to +180 degrees, adds 360 degrees to correct it to a negative value.
9 If Deg(P5.C)<+178 Then P5.C=P5.C-Rad(+360)	'If the posture component (C) of P5 is close to +180 degrees, adds 360 degrees to correct it to a negative value.
10 *DEFINE	'Specifies the label DEFINE line as the jump destination.
11 Def Plt 1, P2, P3, P4, P5, 3, 5, 2	'Defines the pallet. Pallet No. = 1, start point = P2, end point A = P3, end point B = P4, diagonal point = P5, quantity A = 3, quantity B = 5, pallet pattern = 2 (Same direction).
12 M1=1	'Substitutes value 1 in numeric variable M1. (M1 is used as a counter.)
13 *LOOP	'Designates label LOOP at the jump destination.
14 Mov P1, -50 *1)	'Moves with joint interpolation from P1 to a position retracted 50mm in hand direction.
15 Ovrd 50	'Sets movement speed to half of the maximum speed.
16 Mvs P1	'Moves linearly to P1. (Goes to grasp workpiece.)
17 HClose 1	'Closes hand 1. (Grasps workpiece.)
18 Dly 0.5	'Waits 0.5 seconds.
19 Ovrd 100	'Sets movement speed to maximum speed.
20 Mvs , -50 *1)	'Moves linearly from current position (P1) to a position retracted 50mm in hand direction. (Lifts up workpiece.)
21 P10=(Plt1,M1)	'Operates the position in pallet No. 1 indicated by the numeric variable M1, and substitutes the results in P10.
22 Mov P10, -50 *1)	'Moves with joint interpolation from P10 to a position retracted 50mm in hand direction.
23 Ovrd 50	'Sets movement speed to half of the maximum speed.
24 Mvs P10	'Moves linearly to P10. (Goes to place workpiece.)
25 HOpen 1	'Opens hand 1. (Places workpiece.)
26 Dly 0.5	'Waits 0.5 seconds.
27 Ovrd 100	'Sets movement speed to maximum speed.
28 Mvs , -50	'Moves linearly from current position (P10) to a position retracted 50mm in hand direction. (Separates from workpiece.)
29 M1=M1+1	'Increments numeric variable M1 by 1. (Advances the pallet counter.)
30 If M1<=15 Then *LOOP	'If numeric variable M1 value is less than 15, jumps to label LOOP and repeat process. If more than 15, goes to next step.
31 End	'Ends the program.

***Related functions**

Function	Explanation page
Substitute, operation	Page 113, "4.1.6 Expressions and operations"
Condition branching	Page 104, "(1) Unconditional branching, conditional branching, waiting"

4.1.3 Program control

The program flow can be controlled with branching, interrupting, subroutine call, and stopping, etc.

(1) Unconditional branching, conditional branching, waiting

The flow of the program to a specified step can be set as unconditional or conditional branching.

*Command word

Command word	Explanation
GoTo	Jumps unconditionally to the designated step.
On GoTo	Jumps according to the value of the designated variable. The value conditions follow the integer value order.
If Then Else (Instructions written in one step)	Executes the command corresponding to the designated conditions.. The value conditions can be designated randomly. There is only one type of condition per command statement. If the conditions are met, the instruction after Then is executed. If the conditions are not met, the instruction after Else is executed. They are written in one step.
If Then Else End If (Instructions written in several steps)	Several steps can be processed according to the specified variables and specified conditions of the values. It is possible to specify any conditions for values. Only one type of condition is allowed for one instruction. If the conditions are met, the steps following Then until the Else step are executed. If the conditions are not met, the steps after Else until End IF are executed.
Select Case End Select	Jumps according to the designated variable and the designated conditions of that value. The value conditions can be designated randomly. Multiple types of conditions can be designated per command statement.
Wait	Waits for the variable to reach the designated value.

*Statement example

Statement example	Explanation
GoTo 200.....	Jumps unconditionally to step 200.
GoTo *FN.....	Jumps unconditionally to the label FIN step.
ON M1 GoTo *L1, *L2, *L3	If the numeric variable M1 value is 1, jumps to step *L1, if 2 jumps to step *L2, and if 3 jumps to step *L3. If the value does not correspond, proceeds to next step.
If M1=1 Then *L1	If the numeric variable M1 value is 1, branches to step *L1. If not, proceeds to the next step.
If M1=1 Then *L2 Else *L2.....	If the numeric variable M1 value is 1, branches to step *L1. If not, branches to step *L2.
If M1=1 Then .. M2=1 M3=2 Else M2=-1 M3=-2 EndIf	If the numerical variable of M1 is 1, the instructions M2 = 1 and M3 = 2 are executed. If the value of M1 is different from 1, the instructions M2 = -1 and M3 = -2 are executed.
Select M1.....	Branches to the Case statement corresponding to the value of numeric variable M1.
Case 10	If the value is 10, executes only between Case 10 and the next Case 11.
: Break	
Case IS 11	If the value is 11, executes only between Case 11 and the next Case IS <5.
: Break	
Case IS <5.....	If the value is smaller than 5, executes only between Case IS <5 and next Case 6 TO 9.
: Break	
Case 6 TO 9	If value is between 6 and 9, executes only between Case 6 TO 9 and next Default.
: Break	
Default	If value does not correspond to any of the above, executes only between Default and next End Select.
: Break	
End Select	Ends the Select Case statement.
Wait M_In(1)=1	Waits for the input signal bit 1 to turn ON.

***Related functions**

Function	Explanation page
Repetition.....	Page 106, "(2) Repetition"
Interrupt.....	Page 107, "(3) Interrupt"
Subroutine.....	Page 108, "(4) Subroutine"
External signal input.....	Page 111, "(1) Input signals"

(2) Repetition

Multiple command statements can be repeatedly executed according to the designated conditions.

*Command word

Command word	Explanation
For Next	Repeat between For statement and Next statement until designated conditions are satisfied.
While WEnd	Repeat between While statement and WEnd statement while designated conditions are satisfied.

*Statement example

Statement example	Explanation
For M1=1 To 10 : Next	Repeat between For statement and Next statement 10 times. The initial numeric variable M1 value is 1, and is incremented by one with each repetition.
For M1=0 To 10 Step 2 : Next	Repeat between For statement and Next statement 6 times. The initial numeric variable M1 value is 0, and is incremented by two with each repetition.
While (M1 >= 1) And (M1 <= 10) : WEnd	Repeat between While statement and WEnd statement while the value of the numeric variable M1 is 1 or more and less than 10.

*Related functions

Function	Explanation page
Unconditional branching, branching.....	Page 104, "(1) Unconditional branching, conditional branching, waiting"
Interrupt.....	Page 107, "(3) Interrupt"
Input signal wait	Page 111, "(1) Input signals"

(3) Interrupt

Once the designated conditions are established, the command statement being executed can be interrupted and a designated step branched to.

*Command word

Command word	Explanation
Def Act	Defines the interrupt conditions and process for generating interrupt.
Act	Designates the validity of the interrupt.
Return	If a subroutine is called for the interrupt process, returns to the interrupt source line.

*Statement example

Statement example	Explanation
Def Act 1, M_In(10)=1 GoSub *SUB1	If input signal bit 10 is turned on for interrupt number 1, the subroutine on step *SUB1 is defined to be called after the robot decelerates and stops. The deceleration time depends on the Accel and Ovrd instructions.
Def Act 2, M_In(11)=1 GoSub *SUB2, L	If input signal bit 11 is turned on for interrupt number 2, the subroutine on step *SUB2 is defined to be called after the statement currently being executed is completed.
Def Act 3, M_In(12)=1 GoSub *SUB3, S.....	If input signal bit 12 is turned on for interrupt number 3, the subroutine on step *SUB3 is defined to be called after the robot decelerates and stops in the shortest time and distance possible.
Act 1=1	Enables the priority No. 1 interrupt.
Act 2=0	Disables the priority No. 1 interrupt.
Return 0.....	Returns to the step where the interrupt occurred.
Return 1.....	Returns to the step following the step where the interrupt occurred.

*Related functions

Function	Explanation page
Unconditional branching, branching.....	Page 104, "(1) Unconditional branching, conditional branching, waiting"
Subroutine.....	Page 108, "(4) Subroutine"
Communication.....	Page 112, "4.1.5 Communication"

(4) Subroutine

Subroutine and subprograms can be used.

By using this function, the program can be shared to reduce the No. of steps, and the program can be created in a hierarchical structure to make it easy to understand.

*Command word

Command word	Explanation
GoSub	Calls the subroutine at the designated step or designated label.
On GoSub	Calls the subroutine according to the designated variable number. The value conditions follow the integer value order. (1,2,3,4,.....)
Return	Returns to the step following the step called with the GoSub command.
CallP	Calls the designated program. The next step in the source program is returned to at the End statement in the called program. Data can be transferred to the called program as an argument.
FPrm	An argument is transferred with the program called with the CallP command.

*Statement example

Statement example	Explanation
GoSub	Calls the subroutine from step.
On GoSub.....	Calls the subroutine from label GET.
ON M1 GoSub *L1, *L2, *L3.....	If the numeric variable M1 value is 1, calls the subroutine at step *L1, if 2 calls the subroutine at step *L2, and if 3 calls the subroutine at step *L3. If the value does not correspond, proceeds to next step.
Return.....	Returns to the step following the step called with the GoSub command.
CallP "10"	Calls the No. 10 program.
CallP "20", M1, P1	Transfers the numeric variable M1 and position variable P1 to the No. 20 program, and calls the program.
FPrm M10, P10	Receives the numeric variable transferred with the CallP in M10 of the subprogram, and the position variable in P10.

*Related functions

Function	Explanation page
Interrupt.....	Page 107, "(3) Interrupt"
Communication	Page 112, "4.1.5 Communication"
Unconditional branching.....	Page 104, "(1) Unconditional branching, conditional branching, waiting"

(5) Timer

The program can be delayed by the designated time, and the output signal can be output with pulses at a designated time width.

***Command word**

Command word	Explanation
Dly	Functions as a designated-time timer.

***Statement example**

Statement example	Explanation
Dly 0.05.....	Waits for only 0.05 seconds.
M_Out(10)=1 Dly 0.5.....	Turns on output signal bit 10 for only 0.5 seconds.

***Related functions**

Function	Explanation page
Pulse signal output.....	Page 111, "(1) Input signals"

(6) Stopping

The program execution can be stopped. The moving robot will decelerate to a stop.

*Command word

Command word	Explanation
Hlt	This instruction stops the robot and pauses the execution of the program. When the program is started, it is executed from the next step.
End	This instruction defines the end of one cycle of a program. In continuous operation, the program is executed again from the start step upon the execution of the End instruction. In cycle operation, the program ends upon the execution of the End instruction when the cycle is stopped.

*Statement example

Statement example	Explanation
Hlt.....	Interrupt execution of the program.
If M_In(20)=1 Then Hlt.....	Pauses the program if input signal bit 20 is turned on.
Mov P1 Wthlf M_In(18)=1, Hlt.....	Pauses the program if input signal bit 18 is turned on while moving toward P1.
End	Terminates the program even in the middle of the execution.

*Related functions

Function	Explanation page
Appended statement	Page 257, " Wth (With)"

4.1.4 Inputting and outputting external signals

This section explains the general methods for signal control when controlling the robot via an external device (e.g., PLC).

(1) Input signals

Signals can be retrieved from an external device, such as a programmable logic controller.

The input signal is confirmed with a robot status variable (M_In(), etc.) Refer to [Page 141, "4.3.26 Robot status variables"](#) for details on the robot status variables.

*Command word

Command word	Explanation
Wait	Waits for the input signal to reach the designated state.

*System variables

M_In, M_Inb, M_Inw, M_DIn

*Statement example

Statement example	Explanation
Wait M_In(1)=1.....	Waits for the input signal bit 1 to turn ON.
M1=M_Inb(20).....	Substitutes the input signal bit 20 to 27, as an 8-bit state, in numeric variable M1.
M1=M_Inw(5).....	Substitutes the input signal bit 5 to 20, as an 16-bit state, in numeric variable M1.

*Related functions

Function	Explanation page
Signal output.....	Page 111, "(2) Output signals"
Branching with input signal	Page 104, "(1) Unconditional branching, conditional branching, waiting"
Interrupting with input signal	Page 107, "(3) Interrupt"

(2) Output signals

Signals can be output to an external device, such as a programmable logic controller.

The signal is output with the robot status variable (M_Out(), etc.). Refer to [Page 141, "4.3.26 Robot status variables"](#) for details on the robot status variables.

*Command word

Command word	Explanation
Clr	Clears the general-purpose output signal according to the output signal reset pattern in the parameter.

*System variables

M_Out, M_Outb, M_Outw, M_DOut

*Statement example

Statement example	Explanation
Clr 1.....	Clears based on the output reset pattern.
M_Out(1)=1	Turns the output signal bit 1 ON.
M_Outb (8)=0	Turns the 8 bits, from output signal bit 8 to 15, OFF.
M_Outw (20)=0	Turns the 16 bits, from output signal bit 20 to 35, OFF.
M_Out(1)=1 Dly 0.5.....	Turns the output signal bit 1 ON for 0.5 seconds. (Pulse output)
M_Outb (10)=&H0F.....	Turns the 4 bits, from output signal bit 10 to 13 ON, and turns the four bits from 14 to 17 OFF.

*Related functions

Function	Explanation page
Signal input.....	Page 111, "(1) Input signals"
Timer.....	Page 109, "(5) Timer"

4.1.5 Communication

Data can be exchanged with an external device, such as a personal computer.
Cannot use in CRnQ series.

*Command word

Command word	Explanation
Open	Opens the communication line.
Close	Closes the communication line.
Print#	Outputs the data in the Ascll format. CR is output as the end code.
Input#	Inputs the data in the Ascll format. The end code is CR.
On Com GoSub	Defines the subroutine to be called when an interrupt is generated from the communication line. The interrupt is generated when data is input from an external device.
Com On	Enables the interrupt process from the communication line.
Com Off	Disables the interrupt process from the communication line. The interrupt will be invalid even if it occurs.
Com Stop	Stops the interrupt process from the communication line. If there is an interrupt, it is saved, and is executed after enabled.

*Statement example

Statement example	Explanation
Open "COM1:" AS #1.....	Opens the communication line COM1 as file No. 1.
Close #1	Closes file No. 1.
Close	Closes all files that are open.
Print#1,"TEST"	Outputs the character string "TEST" to file No. 1.
Print#2,"M1=";M1	Output the character string "M1=" and then the M1 value to file No. 2. Output data example: "M1 = 1" + CR (When M1 value is 1)
Print#3,P1.....	Outputs the position variable P1 coordinate value to file No. 3. Output data example: "(123.7, 238.9, 33.1, 19.3, 0, 0)(1, 0)" +CR (When X = 123.7, Y=238.9, Z=33.1, A=19.3, B=0, C=0, FL1=1, FL2=0)
Print#1,M5,P5.....	Outputs the numeric variable M5 value and position variable coordinate value to file No. 1. M5 and P5 are separated with a comma (hexadecimal, 2C). Output data example: "8, (123.7, 238.9, 33.1, 19.3, 0, 0)(1, 0)" +CR (When M5=8, P5 X=123.7, Y=238.9, Z=33.1, A=19.3, B=0, C=0, FL1=1, FL2=0)
Input#1,M3	Converts the input data into a value, and substitutes it in numeric variable M3. Input data example: "8" + CR (when value 8 is to be substituted)
Input#1,P10	Converts the input data into a value, and substitutes it in position variable P10. Input data example: "8, (123.7, 238.9, 33.1, 19.3, 0, 0)(1, 0)" +CR (P5 will be X= 123.7, Y=238.9, Z=33.1, A=19.3, B=0, C=0, FL1=1, FL2=0)
Input#1,M8,P6.....	Converts the first data input into a value, and substitutes it in numeric variable M8. Converts the data following the command into a coordinate value, and substitutes it in position variable P6. M8 and P6 are separated with a comma (hexadecimal, 2C) Input data example: "7,(123.7, 238.9, 33.1, 19.3, 0, 0)(1, 0)" +CR (The data will be M8 = 7, P6 X=123.7, Y=238.9, Z=33.1, A=19.3, B=0, C=0, FL1=1, FL2=0)
On Com(1) GoSub *SUB3.....	Defines to call step *SUB3 subroutine when data is input in communication line COM1.
On Com(2) GoSub *RECV.....	Defines to call subroutine at label RECV step when data is input in communication line COM2.
Com(1) On.....	Enables the interrupt from communication line COM1.
Com(2) Off.....	Disables (prohibits) the interrupt from communication line COM2.
Com(1) Stop	Stops (holds) the interrupt from communication line COM1.

*Related functions

Function	Explanation page
Subroutine.....	Page 108, "(4) Subroutine"
Interrupt.....	Page 107, "(3) Interrupt"

4.1.6 Expressions and operations

The following table shows the operators that can be used, their meanings, and statement examples.

(1) List of operator

Class	Operator	Meaning	Statement example
Substitution	=	The right side is substituted in the left side.	P1=P2 P5=P_Curr P10.Z=100.0 M1=1 STS\$="OK"
Numeric value operation	+	Add	P10=P1+P2 Mov P8+P9 M1=M1+1 STS\$="ERR"+"001"
	-	Subtract	P10=P1-P2 Mov P8-P9 M1=M1-1
	*	Multiply	P1=P10*P3 M1=M1*5
	/	Divide	P1=P10/P3 M1=M1/2
	^	Exponential operation	M1=M1^2
	\	Integer division	M1=M1\3
	MOD	Remainder operation	M1=M1 Mod 3
	-	Sign reversal	P1=-P1 M1=-M1
Comparison operation	=	Compare whether equal	If M1=1 Then *L1 If STS\$="OK" Then *L2
	<>	Compare whether not equal	If M1<>2 Then *L3 If STS\$<>"OK" Then *L4
	<	Compare whether smaller	If M1< 10 Then *L3 If Len(STS\$)<3 Then *L4
	>	Compare whether larger	If M1>9 Then *L3 If Len(STS\$)>2 Then *L4
	=<	Compare whether equal to or less than	If M1<=10 Then *L3 If Len(STS\$)<=5 Then *L4
	=>	Compare whether equal to or more than	If M1=>11 Then *L3 If Len(STS\$)>=6 Then *L4
	<=		
	>=		

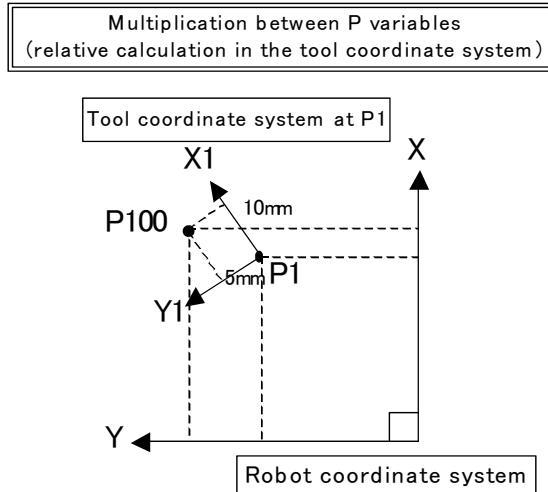
Class	Operator	Meaning	Statement example
Logical operation	And	Logical AND operation	M1=M_Inb(1) And &H0F 'Convert the input signal bit 1 to 4 status and substitute in numeric variable M1. (Input signal bits 5 to 8 remain OFF.)
	Or	Logical OR operation	M_Outb(20)=M1 Or &H80 'Output the numeric variable M1 value to output signal bit 20 to 27. Output bit signal 27 is always ON at this time.
	Not	NOT operation	M1=Not M_Inw(1) 'Reverse the status of input signal bit 1 to 16 to create a value, and substitute in numeric variable M1.
	Xor	Exclusive OR operation	N2=M1 Xor M_Inw(1) 'Obtain the exclusive OR of the states of M1 and the input signal bits 1 to 16, convert into a value and substitute in numeric variable M2.
	<<	Logical left shift operation	M1=M1<<2 'Shift numeric variable M1 two bits to the left.
	>>	Logical right shift operation.	M1=M1>>1 'Shift numeric variable M1 bit to the right.

Note1) Please refer to Page 115, "Relative calculation of position data (multiplication)" .

Note 2) Please refer to Page 115, "Relative calculation of position data (Addition)" .

(2) Relative calculation of position data (multiplication)

Numerical variables are calculated by the usual four arithmetic operations. The calculation of position variables involves coordinate conversions, however, not just the four basic arithmetic operations. This is explained using simple examples.



An example of relative calculation (multiplication)

1 P2=(10,5,0,0,0,0)(0,0)

2 P100=P1*P2

3 Mov P1

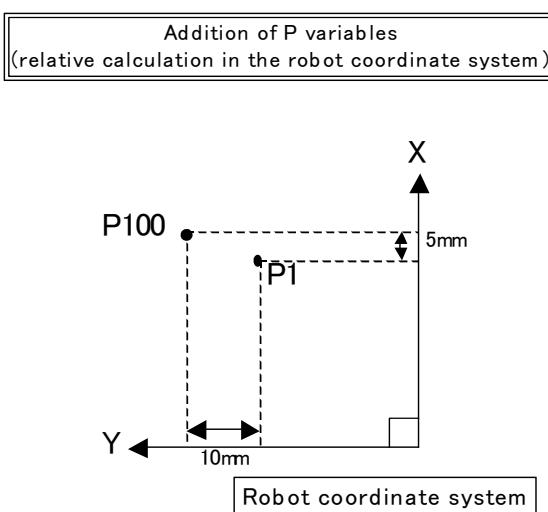
4 Mvs P100

P1=(200,150,100,0,0,45)(4,0)

In this example, the hand tip is moved relatively within the P1 tool coordinate system at teaching position P1. The values of the X and Y coordinates of P2 become the amount of movement within the tool coordinate system. The relative calculation is given by multiplication of the P variables. Be aware that the result becomes different if the order of multiplication is different. The variable that specifies the amount of relative movement (P2) should be entered lastly.

If the posture axis parts of P2 (A, B, and C) are 0, the posture of P1 is used as is. If there are non-zero values available, the new posture is determined by rotating the hand around the Z, Y, and X axes (in the order of C, B, and A) relative to the posture of P1. Multiplication corresponds to addition within the tool coordinate system, while division corresponds to subtraction within the tool coordinate system.

(3) Relative calculation of position data (Addition)



An example of relative calculation (Addition)

1 P2=(5,10,0,0,0,0)(0,0)

2 P100=P1+P2

3 Mov P1

4 Mvs P100

P1=(200,150,100,0,0,45)(4,0)

In this example, the hand is moved relatively within the robot coordinate system at teaching position P1. The values of the X and Y coordinates of P2 become the amount of movement within the robot coordinate system. The relative calculation is given by addition of the P variables.

If a value is entered for the C-axis coordinate of P2, it is possible to change the C-axis coordinate of P100. The resulting value will be the sum of the C-axis coordinate of P1 and the C-axis coordinate of P2.

CAUTION)

In the example above, the explanation is made in two dimensions for the sake of simplicity. In actuality, the calculation is made in three dimensions. In addition, the tool coordinate system changes depending on the posture.

4.1.7 Appended statement

A process can be added to a movement command.

*Appended statement

Appended statement	Explanation
Wth	Unconditionally adds a process to the movement command.
Wthif	Conditionally adds a process to the movement command.

*Statement example

Statement example	Explanation
Mov P1 Wth M_Out(20)=1.....	Turns output signal bit 20 ON simultaneously with the start of movement to P1.
Mov P1 WITHIF M_In(20)=1, Hlt.....	Stops if the input signal bit 20 turns ON during movement to P1.
Mov P1 Wthif M_In(19)=1, Skip	Stops movement to P1 if the input signal bit 19 turns ON during movement to P1, and then proceeds to the next step.

*Related functions

Function	Explanation page
Joint interpolation movement.....	Page 88, "(1) Joint interpolation movement"
Linear interpolation movement.....	Page 89, "(2) Linear interpolation movement"
Circular interpolation movement	Page 90, "(3) Circular interpolation movement"
Stopping.....	Page 110, "(6) Stopping"

4.2 Multitask function

4.2.1 What is multitasking?

The multitask function is explained in this section.

Multitasking is a function that runs several programs as parallel, to shorten the tact time and enable control of peripheral devices with the robot program.

Multitasking is executed by placing the programs, to be run in parallel, in the items called "slots" (There is a total of 32 task slots. The maximum factory default setting is 8.) .

The execution of multitask operation is started by activating it from the operation panel or by a dedicated input signal, or by executing an instruction related to multitask operation.

The execution environment for multitasking is shown in [Fig. 4-1](#).

Multitask slot environment

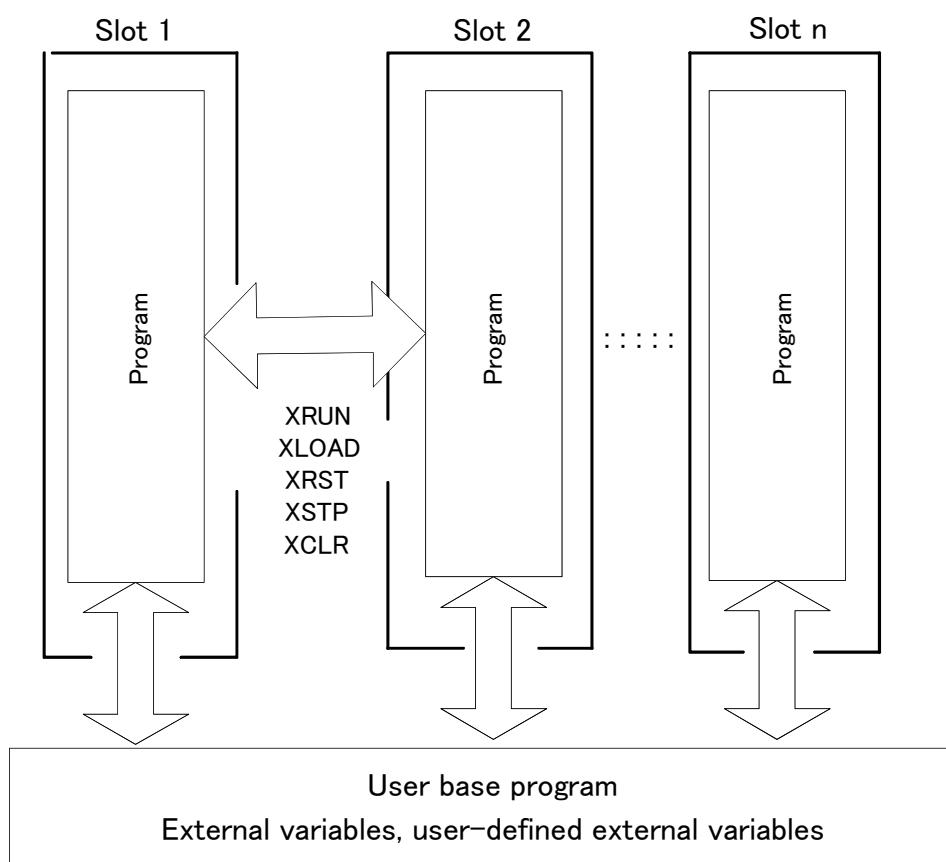


Fig.4-1:Multitask slot environment

Execution of a program

A program is executed by placing it in an item called a "slot" and running it. For example, when running one program (when normally selecting and running the program with the controller's operation panel), the controller system unconditionally places the program selected with the operation panel in slot 1.

4.2.2 Executing a multitask

Table 4-2: The multitask can be executed with the following three methods.

	Types of execution	Explanation
1	Execution from a program	<p>This method starts parallel operation of the programs from a random position in the program using a MELFA-BASIC IV command. The programs to be run in parallel can be designated, and a program running in parallel can be stopped.</p> <p>This method is effective when selecting the programs to be run in parallel according to the program flow.</p> <p>The related commands include the XLoad, XRun, XStp and XRst commands. Refer to Page 153, "4.11 Detailed explanation of command words" in this manual for details.</p>
2	Execution from controller operation panel or external input/output signals	<p>In this execution type, depending on the setting of the information of the "SLT*" parameter, the start operation starts concurrent execution or constant concurrent execution, or starts concurrent execution at error occurrence. It is necessary to set the "SLT*" parameter in advance.</p> <p>This method does not rely on the program flow, and is effective for carrying out simultaneous execution with a preset format, or for sequential execution.</p>
3	Executing automatically when the power is turned on	<p>It is possible to start constant execution immediately after turning the controller's power on. If ALWAYS is specified for the start condition of the SLT* parameter, the program is executed in constant execution mode immediately after the controller's power is turned on.</p> <p>This eliminates the trouble of starting the programs in task slots used for monitoring input/output signals from the PLC side.</p> <p>In addition, it is possible to execute a program from within another program that controls movement continuously. In this case, set the value of the "ALWENA" parameter to 7 in order to execute X** instructions such as XRun and XLoad, the Servo instruction, and the Reset instruction.</p>

4.2.3 Operation state of each slot

The operation state of each slot changes as shown in [Fig. 4-2](#) according to the operations and commands. Each state can be confirmed with the robot status variable or external output signal.

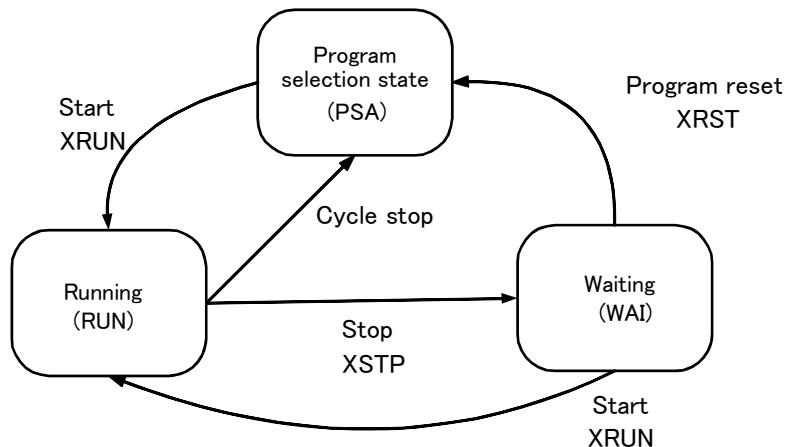


Fig.4-2: Operation state of each slot

<About parameters related to task slots>

The parameters SLT1 to SLT32 contain information about the name of the program to be executed, operation mode, start condition, and priority for each of the 32 task slots (set to 8 slots at the factory default setting). Please refer to [Page 347, "5 Functions set with parameters"](#) for details.

*Designation format

Parameter name = 1. program name, 2. operation format, 3. starting conditions, 4. order of priority

*Various setting values and meanings

Item of parameter	Default value	Setting value	Explanation
1. Program name	SLT1: Program selected on the operation panel. SLT2 to 32: Name of the program to be specified with a parameter.	Possible to set a registered program	Use the parameter to specify the execution of predetermined programs in multitask operation. If the programs to be executed vary depending on conditions, it is possible to specify the program using the XLoad and XRun instructions in another program. The programs selected on the operation panel are set if SLT1 is specified.
2. Operation format	REP	REP : Continuous operation	If REP is specified, the program is executed again from the top after the program ends when the final line of the program is reached, or by execution of the End instruction.
		CYC : One cycle operation	If CYC is specified, the program ends after being executed for one cycle and the selected status is canceled. Change the SLOTON setting of the parameter if it is desired to keep the program in the selected status. Please refer to the section for SLOTON in Page 347, "5 Functions set with parameters" for details.
3. Starting conditions	START	START : Execution of a program using the START button on the operation panel or the I/O START signal	Select START when it is desired to start normally. <small>Note1)</small>
		ALWAYS : Execution of a program when the controller's power is turned on	Use ALWAYS when it is desired to execute the program in constant execution mode. Note, however, that it is not possible to execute movement instructions such as Mov during constant execution of a program. Programs in constant execution mode can be stopped via the XStop instruction. They cannot be stopped via the operation panel and external input signals, or emergency stop.
		Error : Execution of a program when the controller is in error status	Specify Error when it is desired to execute a program in case an error occurs. It is not possible to execute instructions for moving the robot, such as the Mov instruction. The operation mode (REP/CYC) is one-cycle operation (CYC) regardless of the setting value.
4. Order of priority (number of lines executed in priority)	1	1 to 31: Number of lines executed at one time at multitask operation	If this number is increased, the number of lines executed at one time for the task slot is increased. For example, if 10 is specified for SLT1, 5 for SLT2, and 1 for SLT3, then after 10 lines of the program specified in SLT1 have been executed, five lines of the program specified in SLT2 are executed, and then one line of the program specified in SLT3 is executed. Afterward this cycle will be repeated.

Note1) The start operation conducted from the operation panel or by sending the dedicated input signal

START will start the execution of programs of all the task slots whose start conditions are set to "START" at the same time.

To start the program independently, start in slot units with the dedicated input signal (S1 START to S32START). In this case, the line No. is preassigned to the same dedicated input/output parameter. Refer to [Page 428, "6.3 Dedicated input/output"](#) in this manual for details on the assignment of the dedicated input/output.

*Setting example

An example of the parameter settings for designating the following items in slot 2 is shown below.

Designation details) Program name : 5

Operation format : Continuous operation

Starting conditions : Always

Order of priority : 10

SLT2=5, REP, ALWAYS, 10

4.2.4 Precautions for creating multitask program

(1) Relationship between number of tasks and processing time

During multitask operation, it appears as if several robot programs are being processed concurrently. However, in reality, only one line is executed at any one time, and the processing switches from program to program (it is possible to change the number of lines being executed at a time. See the section for the "SLTn" parameter in "Setting Functions by Parameters" on page 247). This means that if the number of tasks increases, the overall program execution time becomes longer. Therefore, when using multitask operation, the number of tasks should be kept to a minimum. However, programs of other tasks executing movement instructions (the Mov and Mvs instructions) are processed at any time.

(2) Specification of the maximum number of programs executed concurrently

The number of programs to be run in parallel is set with parameter TASKMAX. (The default value is 8.) To run more than 8 programs in parallel, change this parameter.

(3) How to pass data between programs via external variables

Data is passed between programs being executed in multitask operation via program external variables such as M_00 and P_00 (refer to [Page 138, "4.3.22 External variables"](#)) and the user-defined external variables (refer to [Page 139, "4.3.24 User-defined external variables"](#)). An example is shown below. In this example, the on/off status of input signal 8 is judged by the program specified in task slot 2. Then this program notifies the program specified in task slot 1 that the signal is turned on by means of the external variable M_00.

<Slot 1>	<pre> 1 M_00=0 ; Substitute 0 in M_00 2 *L 3 If M_00=0 Then *L ; Wait for M_00 value to change from 0. 4 M_00=0 ; Substitute 0 in M_00 5 Mov P1 ; Proceed with the target work. 6 Mov P2 : 10 GoTo *L ; Repeat from step 2. </pre>
-----------------------	---

<Slot 2> (Program of signals and variables)	<pre> 1 If M_In(8) <> 1 Then *A1 ; Branch to line 30 if input signal 8 is not ON. 2 M_00=1 ; Substitute 1 in M_00 3 *A1 4 Mov P1 ; Proceed with the target work. : </pre>
--	---

(4) Confirmation of operating status of programs via robot status variables

The status of the program running with multitask can be referred to from any slot using the robot status variables (M_Run, M_Wai, M-ERR).

Example) M1 = M_Run (2) The operation status of slot 2 is obtained.

Refer to [Page 141, "4.3.26 Robot status variables"](#) for details on the robot status variables.

(5) The program that operates the robot is basically executed in slot 1.

The program that describes the robot arm's movement, such as with the Mov commands, is basically set and executed in slot 1. To run the program in a slot other than slot 1, the robot arm acquisition and release command (GetM, RelM) must be used. Refer to [Page 153, "4.11 Detailed explanation of command words"](#) in this manual for details on the commands.

(6) How to perform the initialization processing via constantly executed programs

Programs specified in task slots whose start condition is set to ALWAYS are executed continuously (repeatedly) if the operation mode is set to REP. Therefore, in order to perform the initialization processing via such programs, they should be programmed in such a way that the initialization processing is not executed more than once, for example by setting an initialization complete flag and perform a conditional branch based on

the flag's status. (This consideration is not necessary for task programs whose operation mode is set to CYC (1 cycle operation) because they are executed only once.)

Mechanism 1 is assigned to slot 1

In the default state, mechanism 1 (robot arm of standard system) is automatically assigned to slot 1.

Because of this, slot 1 can execute the movement command even without acquiring mechanism 1 (without executing GetM command). However, when executing the movement command in a slot other than slot 1, the slot 1 mechanism acquisition state must be released (RelM command executed), and the mechanism must be acquired with the slot that is to execute the movement command (execute the GetM command).

4.2.5 Precautions for using a multitask program

(1) Starting the multitask

When starting from the operation panel or with the dedicated input signal START, the programs in all slots for which the "start request execution" is set in the slot parameter start conditions will start simultaneously. When starting with the dedicated input signals S1START to S32START, the program can be started in each slot. In this case, the line No. is preassigned to the same dedicated input/output parameter. Refer to [Page 428, "6.3 Dedicated input/output"](#) for details on the assignment of the dedicated input/output.

(2) Display of operation status

The LEDs of the [START] and [STOP] switches on the operation panel and the dedicated input/output signals START and STOP display the operation conditions of programs specified in task slots for which the start conditions are set to "START" in the corresponding "SLT*" parameter. If at least one program is operating, the LED of the [START] switch lights up and the dedicated output signal START turns on. If all the programs stop, the LED of the [STOP] switch is lit and the dedicated output signal STOP turns on.

The dedicated output signals S1START to S32START and S1STOP to S32STOP output the operation status for each of the task slots. If it is necessary to know the individual operation status, signal numbers can be assigned to the dedicated input/output parameters and their status checked with the status of the external signals.

For a detailed description of assignment of dedicated input/output, please refer to [Page 428, "6.3 Dedicated input/output"](#) of this manual.

The status of programs whose start condition is set to ALWAYS or Error does not affect the LEDs of the [START] and [STOP] switches. The operation status of programs in constant execution mode can be checked using the monitoring tool of the PC support software (optional).

4.2.6 Example of using multitask

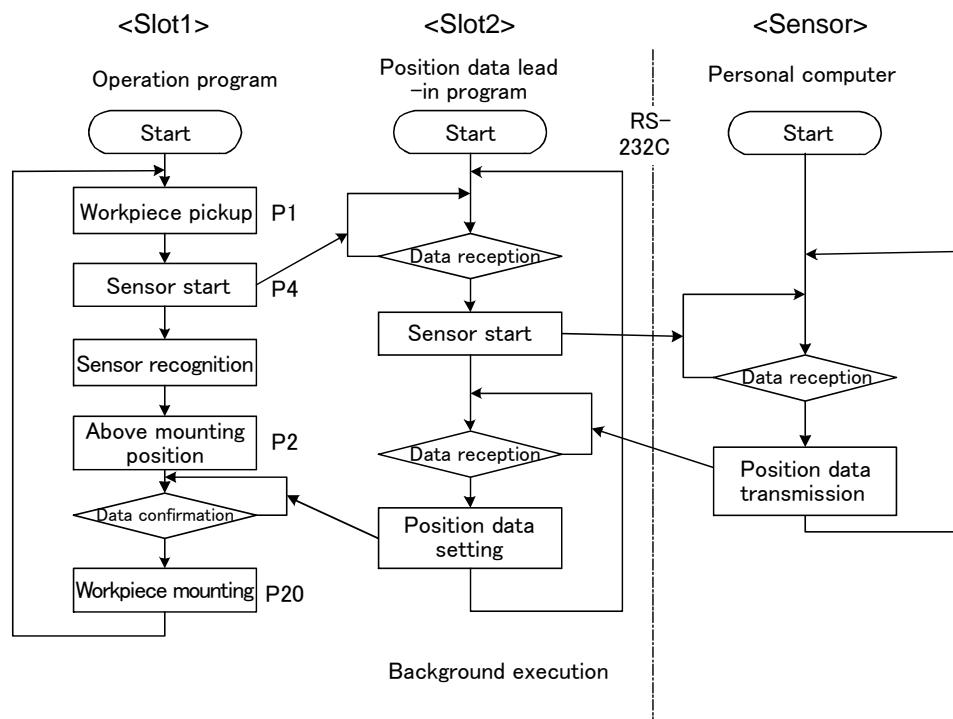
An example of the multitask execution is given in this section.

(1) Robot work details.

The robot programs are the "movement program" and "position data lead-in program".

The "movement program" is executed with slot 1, and the "position data lead-in program" is executed with slot 2. If a start command is output to the sensor while the robot is moving, a request for data will be made to the personal computer via the position data lead-in program. The personal computer sends the position data to the robot based on the data request. The robot side leads in the compensation data via the position data lead-in program.

<Process flow>



P1: Workpiece pickup position (Vacuum timer Dly 0.05)

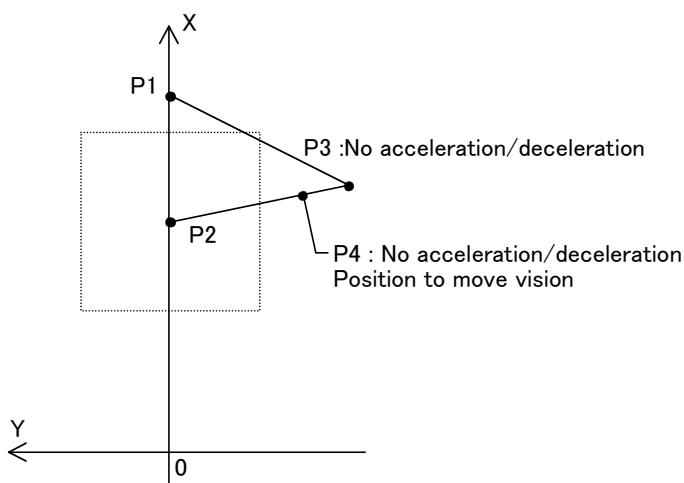
P2: Workpiece placing position (Release timer Dly 0.05)

P3: Vision pre-position (Do not stop at penetration point Cnt)

P4: Vision shutter position (Do not stop at penetration point Cnt)

P_01: Vision compensation data

P20: Position obtained by adding P2 to vision compensation data (relative operation)



(2) Procedures to multitask execution

*Procedure 1: Program creation

```

<1> Movement program (Program name: 1)
1 Cnt 1                                'Validate path connected movement
2 Mov P2,10                             'Move to +10mm above P2
3 Mov P1,10                             'Move to +10mm above P1
4 Mov P1                                'Move to P1 workpiece pickup position
5 M_Out(10)=0                           'Pickup workpiece
6 Dly 0.05                             'Timer 0.05 second
7 Mov P1,10                             'Move to +10mm above P1
8 Mov P3                                'Move to vision pre-position P3
9 Spd 500                               'Set linear speed to 500mm/sec.
10 Mvs P4                               'Start vision lead-in with P4 passage
11 M_02#=0                            'Start data lead-in with background process at interlock variable
                                         (M_01=1/M_02=0)
18 M_01#=1                            'Start data load-in with background process
19 Mvs P2,10                           'Move to +10mm above P2
20 *L                                 'Wait for interlock variable M_02 to reach 1
21 If M_02#=0 Then GoTo *L
22 P20=P2*P_01                         'Add vision compensation P_01 to P20, and move to +10mm above
23 Mov P20,10                           'Move to +10mm above P20
24 Mov P20                             'Go to P20 workpiece placing position
24 M_Out(10)=1                         'Place workpiece
25 Dly 0.05                             'Timer 0.05 second
26 Mov P20,10                           'Move to +10mm above P20
27 Cnt 0                                'Invalidate path connected movement
28 End                                 'End one cycle

```

<2> Position data lead-in program (Program name: 2)

```

1 *R                                 'Wait for interlock variable M_01 to reach 1
2 If M_01#=0 Then GoTo *R
3 Open "COM1:" AS #1                  'Open RS-232-C line
4 Dly M_03#                           'Hypothetical process timer (0.05 second)
5 Print #1,"SENS"                   'Transmit character string "SENS" to RS-232-C (vision side)
6 Input #1,M1,M2,M3                 'Wait to lead-in vision compensation value (relative data)
7 P_01.X=M1                          'Substitute delta X coordinate
8 P_01.Y=M2                          'Substitute delta Y coordinate
9 P_01.Z=0.0                         '
10 P_01.A=0.0                        '
11 P_01.B=0.0                        '
12 P_01.C=Rad(M3)                  'Substitute delta C coordinate
13 Close                             'Close RS-232-C line
14 M_01#=0                           'Interlock variable M_01 = 0
15 M_02#=1                           'Interlock variable M_02 = 0
16 End                               'End process

```

*Procedure 2: Setting the slot parameters

Set the slot parameters as shown below.

Parameters	Program name	Operation mode	Operation format	Number of executed lines
SLT1	1	REP	START	1
SLT2	2	REP	START	1

*Procedure 3: Reflecting the slot parameters

Turn the power OFF and ON to validate the slot parameters.

*Procedure 4: Starting

Start the program 1 and program 2 operation by starting from the operation panel.

4.2.7 Program capacity

There are 3 types of areas that handle robot programs; save, edit and execution. Refer to "[Table 4-3Capacity of each program area](#)" for the capacity of each area.

(1) Program save area

This area is used to save programs. Under normal circumstances, it is possible to save 220 Kbytes of program code in total. The capacity of the program save area can be increased to 2 Mbytes, if it is insufficient, by mounting expansion memory.

(2) Program edit area

This area is used when editing programs and checking the operation in step execution. The program edit area has a capacity of 179 Kbytes, which is the maximum size of one program. The capacity of the program edit area cannot be increased by mounting expansion memory.

(3) Program execution area

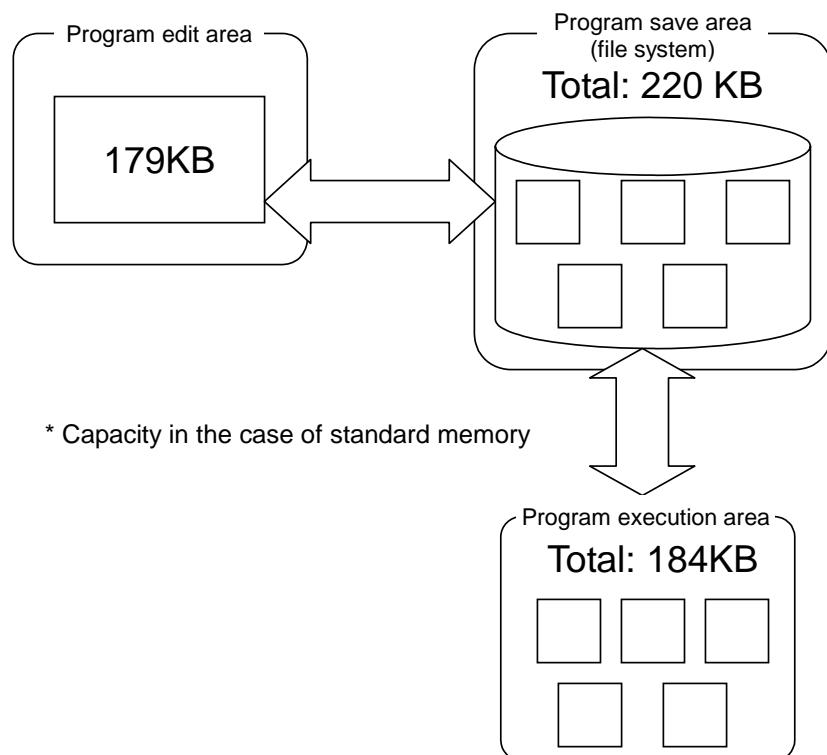
The program execution area is used when operating a program automatically. The capacity of the program execution area is 184 Kbytes. The total capacity of programs loaded into the execution area at the same time via user base programs, for multitasking purposes, or by XRun and CallP instructions, must be 184 Kbytes or less. The capacity of the program execution area cannot be increased by mounting expansion memory.

Table 4-3:Capacity of each program area

Name	Capacity		Remarks
	Standard memory	With expansion memory	
(1) Program save area	220 Kbytes	2 Mbytes	
(2) Program edit area		179 Kbytes	167 Kbytes when the capacity for program external variables is expanded. Note1)
(3) Program execution area		184 Kbytes	

Note1)The capacity for program external variables can be increased by setting the PRGGBL parameter.

The capacity of each program can be checked with the teaching pendant and the Program Manager window of the Personal Computer Support Software.



4.3 Detailed specifications of MELFA-BASIC V

In this section, detailed explanations of the MELFA-BASIC V format and syntax such as configuration are given, as well as details on the functions of each command word. The following explains the components that constitute a statement.

(1) Program name

A program name can be specified using up to 12 characters. However, the operation panel display can display only up to four characters; it is therefore recommended to specify the program name using up to four characters. Moreover, the characters that may be used are as follows.

Class	Usable characters
Alphabetic characters	A B C D E F G H I J K L M N O P Q R S T U V W X Y Z (Use uppercase characters only. If a program name is registered using lowercase characters, the program may not be executed normally.)
Numerals	0 1 2 3 4 5 6 7 8 9

If a program name is specified using more than four characters, the program cannot be selected from the operation panel. In addition, if it is desired to use an external output signal to select a program to be executed, the program name should be specified using the numbers. If a program is executed as a sub-program via the CallP instruction, more than four alphabetic characters may be used. However, such programs may not be selected from the operation panel.

(2) Command statement

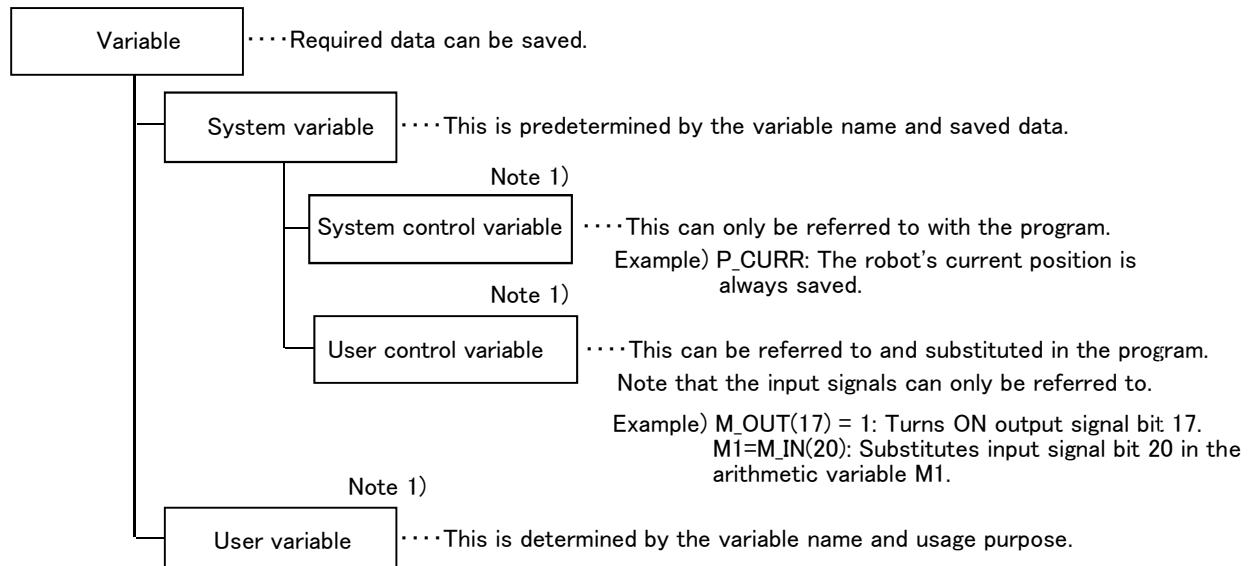
Example of constructing a statement

1 Mov 2) P1 3) Wth M Out(17)=1
1) 2) 3) 4)

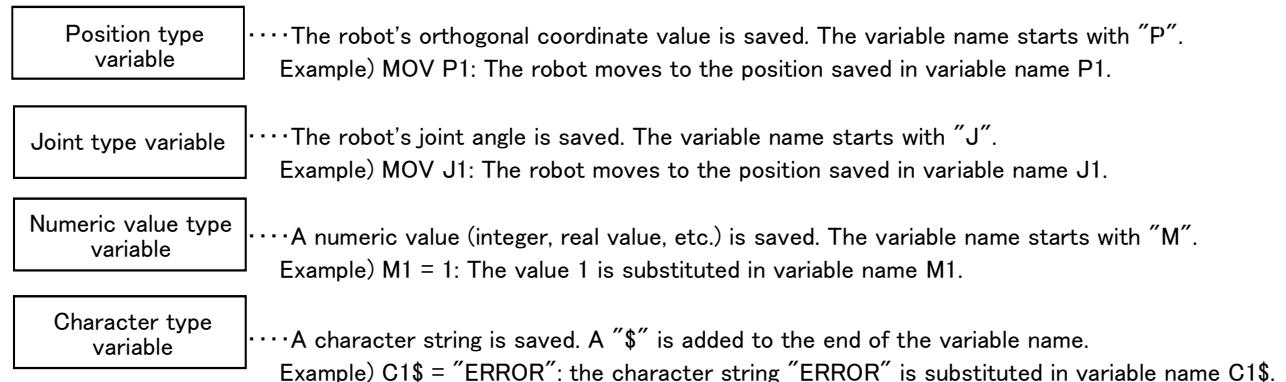
- 1) STEP No. : Numbers for determining the order of execution within the program. steps are executed in ascending order.
- 2) Command word : Instructions for specifying the robot's movement and tasks
- 3) Data : Variables and numerical data necessary for each instruction
- 4) Appended statement: Specify these as necessary when adding robot tasks.

(3) Variable

The following types of variables can be used in a program.



Note 1) Each variable is categorized into the following classes.



4.3.1 Statement

A statement is the minimum unit that configures a program, and is configured of a command word and data issued to the word.

Example) Mov P1
Command word Data
Command statement

4.3.2 Appended statement

Command words can be connected with an appended statement, but this is limited to movement commands.

This allows some commands to be executed in parallel with a movement command.

Example) Mov P1 Wth M_Out (17) = 1
Command statement Appended statement Command statement

Please refer to [Page 257, "Wth \(With\)"](#) or [Page 258, "WthIf \(With If\)"](#), as well as each of the movement instructions (Mov, Mvs, Mvr, Mvr2, Mvr3, Mvc, Mva) for detailed descriptions.

4.3.3 Step

A step is consisted of a step No. and one command statement. Note that if an appended statement is used, there will be two command statements.

One step can have up to 127 characters. (This does not include the last character of the step.)

Only one command statement per step

Multiple command statements cannot be separated with a semicolon and described on one step as done with the general BASIC.

4.3.4 Step No.

Step Nos. should be in ascending order, starting from the first step, in order for the program to run properly. When a program is stored in the memory, it is stored in the order of the step Nos.

Step Nos. can be any integer from 1 to 32767.

Direct execution if step No. is not assigned

If an instruction statement is described without a step number on the instruction screen of the T/B, the statement is executed as soon as it is input. This is called direct execution. In this case, the command statement will not be saved in the memory, but the value substitution to the variable will be saved.

4.3.5 Label

A label is a user-defined name used as a marker for branching.

A label can be created by inserting an asterisk (*) followed by uppercase or lowercase alphanumeric characters after the step No. The head of the label must be an alphabetic character, and the entire label must be within eight characters long. If a label starting with the alphabetic character L is described after the asterisk, an underscore (_) can be used immediately after the character.

* Characters that cannot be used in labels:

- Reserved words (Dly, HOpen, etc.)
- Any name that begins with a symbol or numeral
- Any name that is already used for a variable name or function name

Example) 1 GoTo *LBL

:
10 *LBL

4.3.6 Types of characters that can be used in program

The character which can be used within the program is shown in [Table 4-4](#). However, there are restrictions on the characters that can be used in the program name, variable name and label name. The characters that can be used are indicated by O, those that cannot be used are indicated by X, and those that can be used with restrictions are indicated by @.

Table 4-4:List of characters that can be used

Class	Available characters	Program name	Variable name	Label name
Alphabetic characters	A B C D E F G H I J K L M N O P Q R S T U V W X Y Z a b c d e f g h i j k l m n o p q r s t u v w x y z	O X	O @Note1)	O @Note 1)
Numerals	0 1 2 3 4 5 6 7 8 9	O	@Note2)	O
Symbols	'' & () * + - . , / : ; = < > ? @ ` [\] ^ { } ~ ! # \$ %	X	X	X
	_ (Underscore)	X	Available for type specification @Note3)	@Note4)
Spaces	Space character	X	X	X

Note1) Alphanumerics in variable names and label names in the program are automatically converted into uppercase characters.

Note2) Only alphabetical characters can be used as the first character of the variable name. Numerals can be used as the second and succeeding characters.

Note3) They can be used as the second and succeeding characters. Any variable having an underscore (_) as the second character becomes an external variable.

Note4) If an underscore (_) is used in a label name, start with an asterisk (*) followed by alphabetic character "L."

Refer to [Page 126, "\(1\) Program name"](#) for detail of program names, refer to [Page 134, "4.3.15 Variables"](#) for detail of variable names, and refer to [Page 128, "4.3.5 Label"](#) for detail of label names.

4.3.7 Characters having special meanings

(1) Uppercase and lowercase identification

Lowercase characters will be resigned as lowercase characters when they are used in comments or in character string data. In all other cases, they will be converted to uppercase letters when the program is read.

(2) Underscore (_)

The underscore is used for the second character of an identifier (variable name) to identify the variable as an external variable between programs. Refer to [Page 138, "4.3.22 External variables"](#) for details.

Example) P_Curr, M_01, M_ABC

(3) Apostrophe (')

The apostrophe (') is used at the head of all comments steps. When assigned at the head of a character it is a substitute for the Rem statement.

Example) 1 Mov P1 'GET ;GET will be set as the comment.
2 'GET PARTS ;This is the same as 150 Rem GET PARTS.

(4) Asterisk (*)

The asterisk is placed in front of label names used as the branch destination.

Example) 2 *CHECK

(5) Comma (,)

The comma is used as a delimiter when there are several parameters or suffixes.

Example) P1=(200, 150,)

(6) Period (.)

The period is used for obtaining certain components out of multiple data such as decimal points, position variables and joint variables.

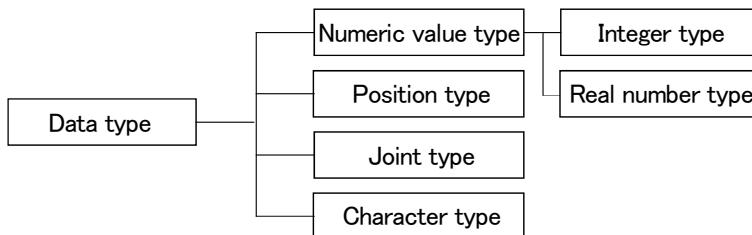
Example) M1 = P2.X ; Substitute the position variable P2.X coordinate element in numeric variable M1.

(7) Space

The space character, when used as part of a character string constant or within a comments step, is interpreted as a character. The space character is required as a delimiter immediately after a step No. or a command word, and between data items. In the [Format] given in section [Page 153, "4.11 Detailed explanation of command words"](#), the space is indicated with a "[] " where required.

4.3.8 Data type

In MELFA BASIC V it is possible to use four data types: numerical values, positions, joints, and character strings. Each of these is called a "data type." The numerical value data type is further classified into real numbers and integers. There can be variables and constants of each data type.



Example)

Numeric value type M1 [Numeric value variables], 1 [Numeric value constants] (Integer),
1.5 [Numeric value constants] (Real number)

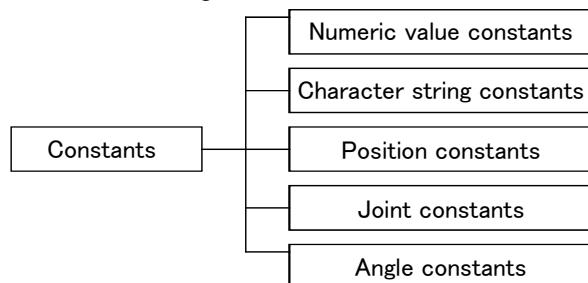
Position type P1 [Position variables], (0,0,0,0,0) (0,0) [Position constants]

Joint type J1 [Joint variables], (0,0,0,0,0) [Joint constants]

Character type C1\$ [Character string variables], "ABC" [Character string constants]

4.3.9 Constants

The constant types include the numeric value constant, character string constant, position constant, joint constant and angle constant.



4.3.10 Numeric value constants

The syntax for numeric value constants is as follows. Numerical constants have the following characteristics.

(1) Decimal number

Example) 1, 1.7, -10.5, +1.2E+5 (Exponential notation)

Valid range -1.7976931348623157e+308 to 1.7976931348623157e+308

(2) Hexadecimal number

Example) &H0001, &HFFFF

Valid range &H0000 to &HFFFF

(3) Binary number

Example) &B0010, &B1111

Valid range &B0000000000000000 to &B1111111111111111

(4) Types of constant

The types of constants are specified by putting symbols after constant characters.

Example) 10% (Integer), 1.0005! (Single-precision real number), 10.00000003# (Double-precision real number)

4.3.11 Character string constants

String constants are strings of characters enclosed by double quotation marks ("").

Example) "ABCDEFGHIJKLMN" "123"

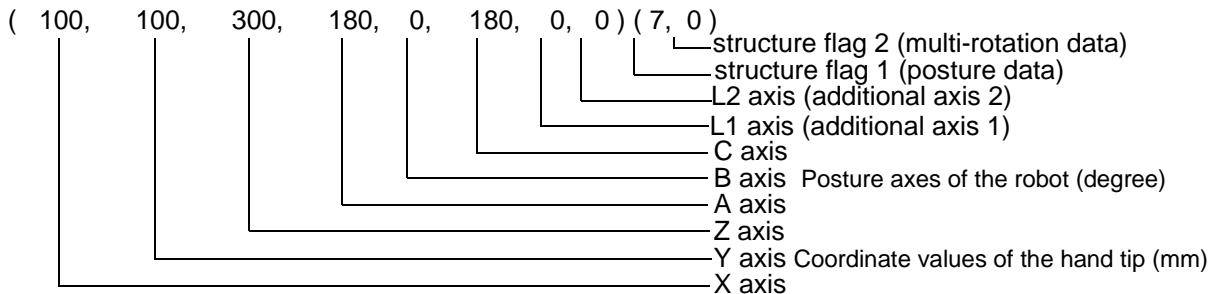
Up to 127 characters for character string

The character string can have up to 127 characters, including the step No. and double quotations.

Enter two double quotation marks successively in order to include the double quotation mark itself in a character string. For the character string AB"CD, input "AB""CD".

4.3.12 Position constants

The syntax for position constants is as shown below. Variables cannot be described within position constants.



Example)

P1=(300, 100, 400, 180, 0, 180, 0, 0) (7, 0)

P2=(0, 0, -5, 0, 0, 0) (0, 0) [A case where there is no traveling axis data]

P3=(100, 200, 300, 0, 0, 90) (4, 0) [A case of a 4-axis horizontal multi-joint robot]

(1) Coordinate, posture and additional axis data types and meanings

[Format] X, Y, Z, A, B, C, L1, L2

[Meaning] X, Y, Z: coordinate data. The position of the tip of the robot's hand in the XYZ coordinates.
(The unit is mm.)

A, B, C: posture data. This is the angle of the posture. (The unit is deg.) Note1)

L1, L2: additional axis data. These are the coordinates for additional axis 1 and additional axis 2, respectively. (The unit is mm or deg.)

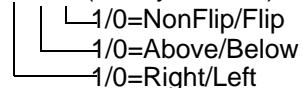
Note1) The T/B and Personal computer support software display the unit in deg; however, the unit of radian is used for substitution and calculation in the program.

(2) Meaning of structure flag data type and meanings

[Format] FL1, FL2

[Meaning] FL1: Posture data. It indicates the robot arm posture in the XYZ coordinates.

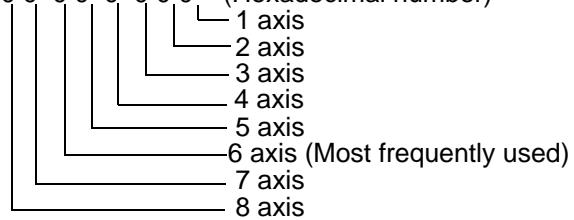
7 = & B 0 0 0 0 0 1 1 1 (Binary number)



FL2: Multiple rotation data. It includes information of the rotational angle of each joint axis at the position (XYZ) and posture (ABC) expressed as XYZ coordinates.

Default value = 0 (The range is 0 to +4294967295 ... Information for eight axes is held with a 1-axis 4-bit configuration.) Two types of screens are available for the PC: screens that display the number of rotations for each axis (-8 to 7) in decimal and those that display the number of rotations for each axis in hexadecimal.

0 = & H 0 0 0 0 0 0 0 0 (Hexadecimal number)



Value of multiple rotation data							
Angle of each axis	-900	-540	-180	0	180	540	900
Value of multiple rotation data	-2 (E)	-1 (F)	0	1	2

The wrist tip axis value in the XYZ coordinates (J6 axis in a vertical multi-joint type robot) is the same after one rotation (360 degrees). For this reason, FL2 is used to count the number of rotations.

Designation of axis No.

1. There is no need to describe the coordinate and posture data for all eight axes. However, if omitted, the following axis data will be processed as undefined.
For a 4-axis robot (X,Y,Z,C axis configuration), describe as (X, Y, Z, , , C) or (X,Y,Z,0,0,C).
2. To omit all axes, insert at least one ","(comma), such as (,).

Use of variables in position element data

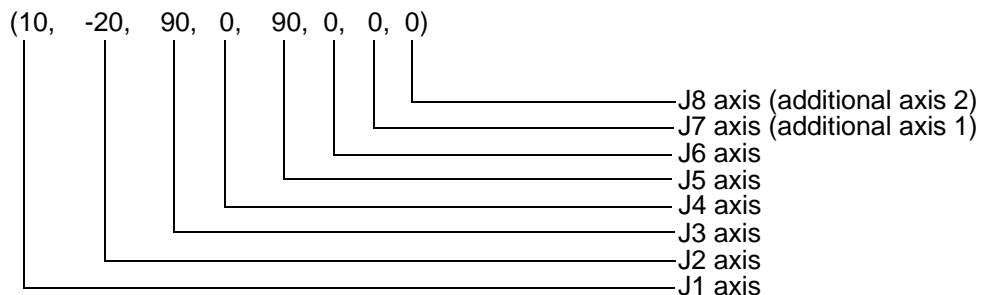
The coordinate, position, additional axis data and structure flag data are called the position element data. A variable cannot be contained in the position element data that configures the position constant.

Omitting the structure flag data

If the structure flag data is omitted, the default value will be applied.((7,0) Varies depending on the machine model.)

4.3.13 Joint constants

The syntax for the joint constants is as shown below



Example)

6 axis robot	J1 = (0, 10, 80, 10, 90, 0)
6 axis + Additional axis	J1 = (0, 10, 80, 10, 90, 0, 10, 10)
5 axis robot	J1 = (0, 10, 80, 0, 90, 0)
5 axis + Additional axis	J1 = (0, 10, 80, 0, 90, 0, 10, 10)
4 axis robot	J1 = (10, 20, 90, 0)
4 axis + Additional axis	J1 = (10, 20, 90, 0, , 10, 10)

(1) Axis data format and meanings

[Format] J1,J2,J3,J4,J5,J6,J7,J8

[Meaning] J1 to J6: Robot axis data (Unit is mm or deg.)

J7, J8: Additional axis data, and may be omitted (optional).
(Unit is mm or deg. Depending on the parameter setting.)

The unit is mm, not degrees, if the J3 axis of a horizontal multi-joint type robot is a direct-driven axis.

Use of variables in joint element data

The axis data is called the joint element data.

A variable cannot be contained in the joint constant data that configures the joint constant.

4.3.14 Angle value

The angle value is used to express the angle in "degrees" and not in "radian".

If written as 100Deg, this value becomes an angle and can be used as an argument of trigonometric functions.

Example) Sin(90Deg)----A 90 degree sine is indicated.

4.3.15 Variables

A variable name should be specified using up to eight characters.

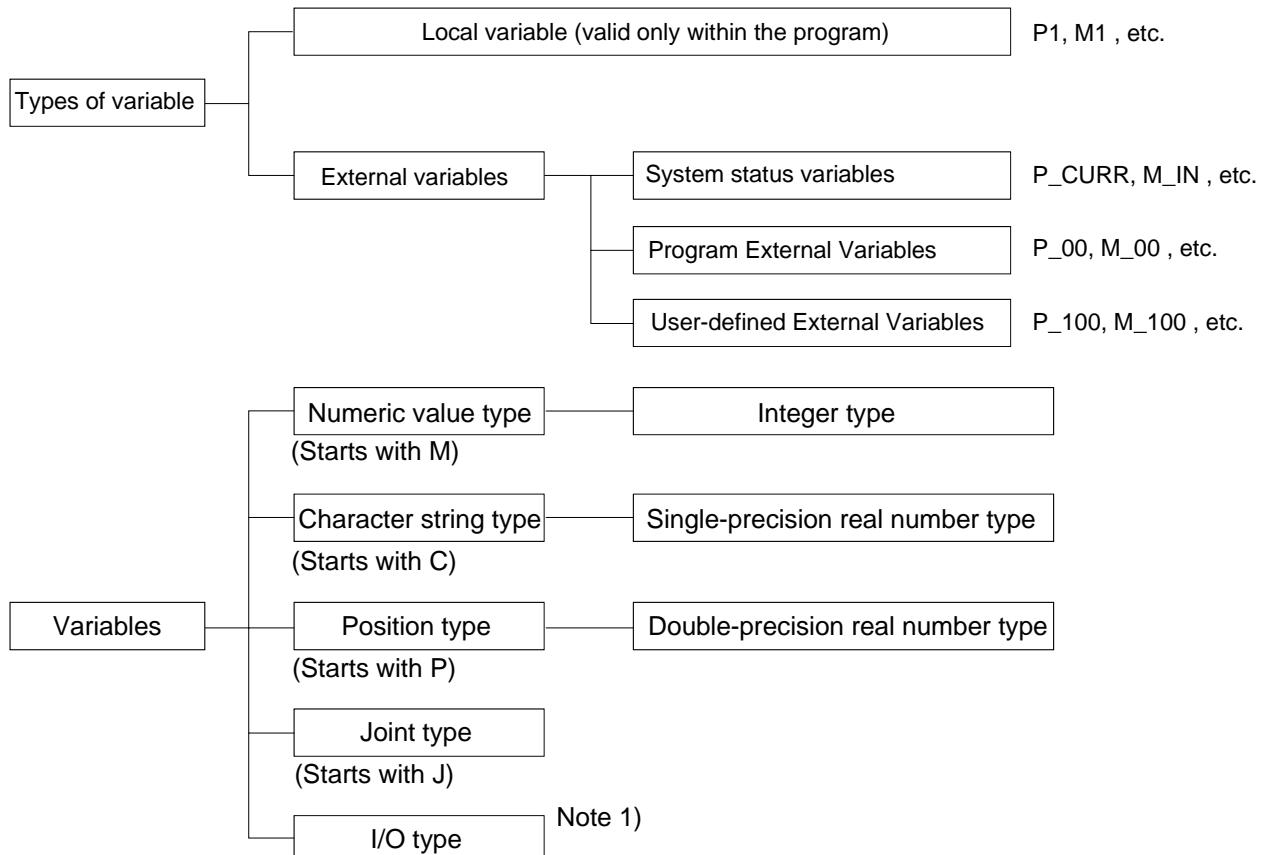
The variable types include the numeric value type, character string type, position type, joint type and I/O type. Each is called a "variable type". The variable type is determined by the head character of the identifier (variable name).

The numeric value type can be further classified as integer type, single-precision real number type, or double-precision real number type.

The following two types of data valid ranges are used.

1. Local variable valid only in one program

2. Robot status variable, program external variable and user-defined external variable valid over programs.
(The user-defined external variable has a _ for the second character of the variable name. Refer to [Page 138, "4.3.22 External variables"](#) for details.)



Note 1) The identifiers include those determined by the robot status variable (M_IN, M_OUT, etc.), and those declared in the program with the DEFIO command.

Variables are not initialized

The variables will not be cleared to zero when generated, when the program is loaded, or when reset.

4.3.16 Numeric value variables

Variables whose names begin with a character other than P, J, or C are considered numeric value variables. In MELFA-BASIC V, it is often specified that a variable is an numeric value variable by placing an M at the head. M is the initial letter of mathematics.

Example) M1 = 100

M2! = -1.73E+10

M3# = 0.123

ABC = 1

- 1) It is possible to define the type of variable by attaching an numeric value type indicator at the end of the variable name. If it is omitted, the variable type is assumed to be of the single-precision real number type.

Numeric value type suffix	Meaning
%	Integer
!	Single-precision real number type
#	Double-precision real number type

- 2) Once the type of a variable is registered, it can only be converted from integer to single-precision real number. For example, it is not possible to convert the type of a variable from integer to double-precision real number, or from single-precision real number to double-precision real number.
- 3) It is not possible to add an numeric value type indicator to an already registered variable. Include the type indicator at the end of the variable name at the declaration when creating a new program.
- 4) If the value is exceeded during a single precision = double precision execution, an error will occur.

Table 4-5:Range of numeric value variable data

Type	Range	
Integer type	-32768 to 32767	
Single-precision real number type	-3.40282347e+38 to 3.40282347e+38	Note) E expresses a power of 10.
Double-precision real number type	-1.7976931348623157e+308 to 1.7976931348623157e+308	

4.3.17 Character string variables

A character string variable should start with C and end with "\$." If it is defined by the Def Char instruction, it is possible to specify a name beginning with a character other than C.

Example) C1\$ = "ABC"
CS\$ = C1\$
Def Char MOJI
MOJI = "MOJIMOJI"

4.3.18 Position variables

Variables whose names begin with character P are considered position variables. If it is defined by the Def Pos instruction, it is possible to specify a name beginning with a character other than P. It is possible to reference individual coordinate data of position variables. In this case, add "." and the name of a coordinate axis, e.g. "X," after the variable name.

P1.X, P1.Y, P1.Z, P1.A, P1.B P1.C, P1.L1, P1.L2

The unit of the angular coordinate axes A, B, and C is radians. Use the Deg function to convert it to degrees.

Example) P1 = PORG
Dim P3(10)
M1 = P1. X (Unit : mm)
M2 = Deg(P1. A) (Unit : degree)
Deg POS L10
Mov L10

4.3.19 Joint variables

A character string variable should start with J. If it is defined by the Def Jnt instruction, it is possible to specify a name beginning with a character other than J.

It is possible to reference individual coordinate data of joint variables.

In this case, add "." and the name of a coordinate axis, e.g. "J1," after the variable name.

JDATA.J1, JDATA.J2, JDATA.J3, JDATA.J4, JDATA.J5, JDATA.J6, JDATA.J7, JDATA.J8

The unit of the angular coordinate axes A, B, and C is radians. Use the DEG function to convert it to degrees.

Example) JSTARAT = (0, 0, 90, 0, 90, 0, 0, 0)

```

JDATA = JSTART
Dim J3 (10)
M1 = J1.J1      (Unit : radian)
M2 = Deg (J1.J2)
Def Jnt K10
Mov K 10

```

4.3.20 Input/output variables

The following types of input/output variables are available. They are provided beforehand by the robot status variables.

Input/output variables name	Explanation
M_In	For referencing input signal bits
M_Inb	For referencing input signal bytes (8-bit signals)
M_Inw	For referencing input signal words (16-bit signals)
M_Out	For referencing/assigning output signal bits
M_OutB	For referencing/assigning output signal bytes (8-bit signals)
M_OutW	For referencing/assigning output signal words (16-bit signals)
M_DIn	For referencing input registers for CC-Link Cannot use in CRnQ series.
M_DOut	For referencing output registers for CC-Link Cannot use in CRnQ series.

Please refer to the robot status variables [Page 285, " M_In/M_Inb/M_Inw"](#), [Page 292, " M_Out/M_Outb/ M_Outw"](#), and [Page 281, " M_DIn/M_DOut"](#).

4.3.21 Array variables

Numeric value variables, character string variables, position variables, and joint variables can all be used in arrays. Designate the array elements at the subscript section of the variables. Array variables should be declared with the Dim instruction. It is possible to use arrays of up to three dimensions.

Example) Example of definition of an array variable

```

Dim M1 (10)  Single-precision real number type
Dim M2% (10) Integer type
Dim M3 ! (10) Single-precision real number type
Dim M4# (10) Double-precision real number type
Dim P1 (20)
Dim J1 (5)
Dim ABC (10, 10, 10)

```

The subscript of an array starts from 1.

However, among the robot status variables, the subscript starts from 0 for individual input/output signal vari-

ables (M_In, M_Out, etc.) only.

Whether it is possible to secure sufficient memory for the variable is determined by the free memory size.

4.3.22 External variables

External variables have a "_" (underscore) for the second character of the identifier (variable name). (It is necessary to register user-defined external variables in the user base program.) The value is valid over multiple programs. Thus, these can be used effectively to transfer data between programs.

There are four types of external variables, numeric value, position, joint and character, in the same manner as the [Page 131, "4.3.8 Data type"](#). The following three types of external variables are available.

Table 4-6:Types of external variables

External variables	Explanation	Example
Program external variables	Types of external variables	P_01,M_01,P_100(1), etc.
User-defined external variables	The user can determine the name freely. Declare the variables using the Def Pos, Def Jnt, Def Char, or DEF INTE/FLOAT/DDOUBLE instructions in the user base program.	P_GENTEN,M_MACHI
Robot status variables (System status variables)	The robot status variables are controlled by the system, and their usage is determined in advance.	M_In,M_Out,P_Curr,M_Pl, etc.

4.3.23 Program external variables

[Table 4-7](#) lists the program external variables that have been prepared for the controller in advance. As shown in the table, the variable name is determined, but the application can be determined by the user.

Table 4-7:Program external variables

Data type	Variable name	Qty.	Remarks
Position	P_00 to P_19 P_20 to P_39 <small>Note)</small>	20 20	
Position array (No. of elements 10)	P_100() to P_104() P_105() to P_109() <small>Note)</small>	5 5	Use the array element in the first dimensions.
Joint	J_00 to J_19 J_20 to J_39 <small>Note)</small>	20 20	
Joint array (No. of elements 10)	J_100() to J_104() J_105() to J_109() <small>Note)</small>	5 5	Use the array element in the first dimensions.
Numeric value	M_00 to M_19 M_20 to M_39 <small>Note)</small>	20 20	The data type of the variables is double-precision real numbers.
Numeric value array (No. of elements 10)	M_100() to M_104() M_105() to M_109() <small>Note)</small>	5 5	Use the array element in the first dimensions. The data type of the variables is double-precision real numbers.
Character string	C_00 to C_19 C_20 to C_39 <small>Note)</small>	20 20	
Character string array (No. of elements 10)	C_100() to C_104() C_105() to C_109() <small>Note)</small>	5 5	Use the array element in the first dimensions.

Note) When you use the extension, change the following parameter.

Parameter	Value	Means
PRGGBL	0:Standard (default) 1:Extension	Sets "1" to this parameter, and turns on the controller power again, then the capacity of each program external variable will double. However, if a variable with the same name is being used as a user-defined external variable, an error will occur when the power is turned ON, and it is not possible to expand. It is necessary to correct the user definition external variable.

4.3.24 User-defined external variables

If the number of program external variables listed above is insufficient or it is desired to define variables with unique names, the user can define program external variables using a user base program.

Procedure before using user-defined external variables

- 1) First, write a user base program. Use "_" for the second character of the variables.
- 2) Register the program name in the "PRGUSR" parameter and turn the power off and on again.
- 3) Write a normal program using the user-defined external variables.
 - (1) By defining a variable having an underscore (_) for the second character of the identifier with the DEF statement in the user base program Note, that variable will be handled as an external variable.
 - (2) It is not necessary to execute the user base program.
 - (3) Write only the lines necessary for declaring variables in the user base program.
 - (4) If it is desired to define array variables in a user base program and use them as external variables, it is necessary to declare them using the Dim instruction again in the program in which they will be used. It is not necessary to declare local variables (variables valid only within programs) again.

Example) Example of using user-defined external variables

On the main program (program name 1) side

10 Dim P_200(10)	' Re-declaration of external variables
20 Dim M_200(10)	' Re-declaration of external variables
30 Mov P_100(1)	
40 If M_200(1) =1 Then Hlt	
50 M1=1	' Local variable

On the user base program (program name UBP) side

10 Def Pos P_900, P_901, P_902, P_903	
20 Dim P_200(10)	' It is necessary to declare this variable again in the program in which they will be used.
30 Def Inte M_100	
40 Dim M_200(10)	' It is necessary to declare this variable again in the program in which they will be used.

Parameter name	Value
PRGUSR	UBP

4.3.25 Creating User Base Programs

Note)

What is a user base program?

A user base program is used when user-defined external variables are used to define such variables, but it is not necessary to actually execute the program. Simply create a program containing the necessary declaration lines and register it in the "PRGUSR" parameter. After changing the parameter, turn the power off and on again.

How to register a new user base program using the Personal Computer Support Software

Using the Personal Computer Support Software, write only instructions to the robot controller first, and write only position data next.

User base programs can be created by using either the teaching box or Personal Computer Support Software, in the same way as the normal programs. To create user base programs using the Personal Computer Support Software, please follow the procedure below:

- 1) Store a program created as a user base program on your personal computer.
- 2) Start Program Manager from Program Editor of the Personal Computer Support Software.
- 3) Specify the program created in step 1) above as the transfer source and the robot as the transfer destination in Program Manager, and perform a "copy" operation. At this point, uncheck the "Position Variables" check box so that only the "Instructions" check box is checked.
- 4) When the copy operation is complete, perform the operation in step 3) above again. Uncheck the "Instructions" check box and check the "Position Variables" check box this time, and then execute.
- 5) Write a user base program in the robot controller first when deleting a program and then register it again in the program management window as well.

4.3.26 Robot status variables

The available robot status variables are shown in [Table 4-8](#). As shown in the table, the variable name and application are predetermined.

The robot status can be checked and changed by using these variables.

Table 4-8:Robot status variables

No	Variable name	Array designation Note1)	Details	Attribute Note2)	Data type, Unit	Page
1	P_Curr	Mechanism No.(1 to 3)	Current position (XYZ)	R	Position type	308
2	J_Curr	Mechanism No.(1 to 3)	Current position (joint)	R	Joint type	271
3	J_ECurr	Mechanism No.(1 to 3)	Current encoder pulse position	R	Joint type	273
4	J_Fbc	Mechanism No.(1 to 3)	Joint position generated based on the feedback value from the servo	R	Joint type	274
5	J_AmpFbc	Mechanism No.(1 to 3)	Current feedback value	R	Joint type	274
6	P_Fbc	Mechanism No.(1 to 3)	XYZ position generated based on the feedback value from the servo	R	Position type	309
7	M_Fbd	Mechanism No.(1 to 3)	Distance between commanded position and feedback position	R	Position type	283
8	M_CmpDst	Mechanism No.(1 to 3)	Amount of difference between a command value and the actual position when the compliance function is being performed	R	Single-precision real number type, mm	277
9	M_CmpLmt	Mechanism No.(1 to 3)	This is used to recover from the error status by using interrupt processing when an error has occurred while the command value in the compliance mode attempted to exceed the limit.	R	Integer type	278
10	P_Tool	Mechanism No.(1 to 3)	Currently designated tool conversion data	R	Position type	310
11	P_Base	Mechanism No.(1 to 3)	Currently designated base conversion data	R	Position type	306
12	P_NTool	Mechanism No.(1 to 3)	System default value (tool conversion data)	R	Position type	310
13	P_NBase	Mechanism No.(1 to 3)	System default value (base conversion data)	R	Position type	306
14	M_Tool	Mechanism No.(1 to 3)	Tool No. (1 to 4)	RW	Integer type	299
15	J_ColMxl	Mechanism No.(1 to 3)	Difference between estimated torque and actual torque	R	Joint type, %	272
16	M_ColSts	Mechanism No.(1 to 3)	Impact detection status (1: Colliding, 0: Others)	R	Integer type	279
17	P_ColDir	Mechanism No.(1 to 3)	Movement direction at collision	R	Position type	307
18	M_OPovrd	None	Speed override on the operation panel (0 to 100%)	R	Integer type, %	286
19	M_Ovrd	Slot No.(1to 32)	Override in currently designated program (0 to 100%)	R	Integer type, %	286
20	M_JOvrd	Slot No.(1to 32)	Currently designated joint override (0 to 100%)	R	Integer type, %	286
21	M_NOvrd	Slot No.(1to 32)	System default value (default value of M_Ovrd) (%)	R	Single-precision real number type, %	286
22	M_NJovrd	Slot No.(1to 32)	System default value (default value of M_JOvrd) (%)	R	Single-precision real number type, %	286
23	M_Wupov	Mechanism No.(1 to 3)	Warm-up operation override (50 to 100%)	R	Single-precision real number type, %	303
24	M_Wuprt	Mechanism No.(1 to 3)	Time until the warm-up operation status is canceled (sec.)	R	Single-precision real number type, sec	304
25	M_Wupst	Mechanism No.(1 to 3)	Time until the warm-up operation status is set again (sec.)	R	Single-precision real number type, sec	305
26	M_Ratio	Slot No.(1to 32)	Fraction of the current movement left before reaching the target position (%)	R	Integer type, %	293
27	M_RDst	Slot No.(1to 32)	Remaining distance left of the current movement (only the three dimensions of X, Y, and Z are taken into consideration: mm)	R	Single-precision real number type, mm	294

No	Variable name	Array designation Note1)	Details	Attribute Note2)	Data type, Unit	Page
28	M_Spd	Slot No.(1to 32)	Current specified speed (valid only for linear/circular interpolation)	R	Single-precision real number type, mm/s	297
29	M_NSpd	Slot No.(1to 32)	System default value (default value of M_Spd) (mm/s)	R	Single-precision real number type, mm/s	297
30	M_RSpd	Slot No.(1to 32)	Current directive speed (mm/s)	R	Single-precision real number type, mm/s	297
31	M_Acl	Slot No.(1to 32)	Current specified acceleration rate (%)	R	Single-precision real number type, %	275
32	M_DAcl	Slot No.(1to 32)	Current specified deceleration rate (%)	R	Single-precision real number type, %	275
33	M_NAcl	Slot No.(1to 32)	System default value (default value of M_Acl) (%)	R	Single-precision real number type, %	275
34	M_NDAcl	Slot No.(1to 32)	System default value (default value of M_DAcl) (%)	R	Single-precision real number type, %	275
35	M_Aclsts	Slot No.(1to 32)	Current acceleration/deceleration status 0 = Stopped, 1 = Accelerating, 2 = Constant speed, 3=Decelerating	R	Integer type	275
36	M_SetAdl	Axis No.(1 to 8)	Specify the acceleration/deceleration time ratio (%) of each axis.	RW	Single-precision real number type, %	295
37	M_LdFact	Axis No.(1 to 8)	The load factor of the servo motor of each axis. (%)	R	Single-precision real number type, %	287
38	M_Run	Slot No.(1to 32)	Operation status (1: Operating, 0: Not operating)	R	Integer type	294
39	M_Wai	Slot No.(1to 32)	Pause status (1: Pausing, 0: Not pausing)	R	Integer type	302
40	M_Psa	Slot No.(1to 32)	Specifies whether or not the program selection is possible in the specified task slot. (1: Selection possible, 0: Selection not possible, in pause status)	R	Integer type	293
41	M_Cys	Slot No.(1to 32)	Cycle operation status (1: Cycle operation, 0: Non-cycle operation)	R	Integer type	280
42	M_Cstp	None	Cycle stop operation status (1: Cycle stop, 0: Not cycle stop)	R	Integer type	280
43	C_Prg	Slot No.(1to 32)	Execution program name	R	Character string type	269
44	M_Line	Slot No.(1to 32)	Currently executed line No.	R	Integer type	288
45	M_SkipCq	Slot No.(1to 32)	A value of 1 is input if execution of an instruction is skipped as a result of executing the line that includes the last executed Skip command, otherwise a value of 0 is input.	R	Integer type	296
46	M_BrkCq	None	Result of the BREAK instruction (1: BREAK, 0: None)	R	Integer type	276
47	M_Err	None	Error occurring (1: An error has occurred, 0: No errors have occurred)	R	Integer type	282
48	M_ErrLvl	None	Reads an error level. caution / low / high1 / high2 = 1/2/3/4	R	Integer type	282
49	M_Erno	None	Reads an error number.	R	Integer type	282
50	M_Svo	Mechanism No.(1 to 3)	Servo motor power on (1: Servo power on, 0: Servo power off)	R	Integer type	297
51	M_Uar	Mechanism No.(1 to 3)	Bit data. (1: Within user specified area, 0: Outside user specified area) (Bit 0:area 1 to Bit 7:area 8)	R	Integer type	300

No	Variable name	Array designation Note1)	Details	Attribute Note2)	Data type, Unit	Page
52	M_Uar32	Mechanism No.(1 to 3)	Bit data. (1: Within user specified area, 0: Outside user specified area) (Bit 0:area 1 to Bit 31:area 32)	R	Integer type	301
53	M_In	Input No.(0 to 32767)	Use this variable when inputting external input signals (bit units). General-purpose bit device: bit signal input 0=off 1=on The signal numbers will be 6000s for CC-Link	R	Integer type	285
54	M_Inb	Input No.(0 to 32767)	Use this variable when inputting external input signals (8-bit units) General-purpose bit device: byte signal input The signal numbers will be 6000s for CC-Link	R	Integer type	285
55	M_Inw	Input No.(0 to 32767)	Use this variable when inputting external input signals (16-bit units) General-purpose bit device: word signal input The signal numbers will be 6000s for CC-Link	R	Integer type	285
56	M_Out	Output No.(0 to 32767)	Use this variable when outputting external output signals (bit units). General-purpose bit device: bit signal input 0=off 1=on The signal numbers will be 6000s for CC-Link	RW	Integer type	292
57	M_OutB	Output No.(0 to 32767)	Use this variable when outputting external output signals (8-bit units) General-purpose bit device: byte signal input The signal numbers will be 6000s for CC-Link	RW	Integer type	292
58	M_OutW	Output No.(0 to 32767)	Use this variable when outputting external output signals (16-bit units) General-purpose bit device: word signal input The signal numbers will be 6000s for CC-Link	RW	Integer type	292
59	M_DIn	Input No.(from 6000)	CC-Link's remote register: Input register Cannot use in CRnQ series.	R	Integer type	281
60	M_DOut	Output No.(from 6000)	CC-Link's remote register: output register Cannot use in CRnQ series.	RW	Integer type	281
61	M_HndCq	Input No.(1 to 8)	Returns a hand check input signal.	R	Integer type	284
62	P_Safe	Mechanism No.(1 to 3)	Returns an safe point position.	R	Position type	309
63	J_Origin	Mechanism No.(1 to 3)	Returns the joint coordinate data when setting the origin.	R	Joint type	274
64	M_Open	File No.(1 to 8)	Returns the open status of the specified file or the communication line.	R	Integer type	290
65	C_Mecha	Slot No.(1 to 32)	Returns the type name of the robot.	R	Character string type	269
66	C_Maker	None	Shows manufacturer information (a string of up to 64 characters).	R	Character string type	268
67	C_User	None	Returns the content of the parameter "USERMSG."(a string of up to 64 characters).	R	Character string type	270
68	C_Date	None	Current date expressed as "year/month/date".	R	Character string type	267
69	C_Time	None	Current time expressed as "time/minute/second".	R	Character string type	270
70	M_BTime	None	Returns the remaining battery capacity time (hours).	R	Integer type, Time	276
71	M_Timer	Timer No. (1 to 8)	Constantly counting. Value can be set. [ms] It is possible to measure the precise execution time by using this variable in a program.	RW	Single-precision real number type	298
72	P_Zero	None	A variable whose position coordinate values (X, Y, Z, A, B, C, FL1, FL2) are all 0	R	Position type	310
73	M_PI	None	Circumference rate (3.1415...)	R	Double-precision real number type	292

No	Variable name	Array designation Note1)	Details	Attribute Note2)	Data type, Unit	Page
74	M_Exp	None	Base of natural logarithm (2.71828...)	R	Double-precision real number type	282
75	M_G	None	Specific gravity constant (9.80665)	R	Double-precision real number type	284
76	M_On	None	1 is always set	R	Integer type	289
77	M_Off	None	0 is always set	R	Integer type	289
78	M_Mode	None	Contains the status of the key switch of the operation panel TEACH/AUTO(OP)/AUTO(Ext.)(1/ 2/3)	R	Integer type	288

Note1) Mechanism No.1 to 3, Specifies a mechanism number corresponding to the multitask processing function.

Slot No.1 to 32, Specifies a slot number corresponding to the multitask function.

Input No.0 to 32767: (theoretical values). Specifies a bit number of an input signal.

Output No.0 to 32767: (theoretical values). Specifies a bit number of an output signal.

Note2) R.....Only reading is possible.

RW.....Both reading and writing are possible.

4.4 Logic numbers

Logic numbers indicate the results of such things as comparison and input/output.

If not 0 when evaluated with an Integer, then it is true, and if 0, it is false. When substituted, if true, 1 is assigned. The processes that can use logic numbers are shown in [Table 4-9](#).

Table 4-9:Values corresponding to true or false logic number

Items expressed with logic number "1"	Items expressed with logic number "0"
*Result of comparison operation (if true)	*Result of comparison operation (if false)
*Result of logic operation (if true)	*Result of logic operation (if false)
*Switch ON	*Switch OFF
*Input/output signal ON	*Input/output signal OFF
*Hand open (supply current to the hand)	*Hand close (do not supply current to the hand)
*Settings for enable/valid such as for interrupts	*Settings for disable/invalid such as for interrupts

4.5 Functions

A function carries out a specific operation for an assigned argument, and returns the result as a numeric value type or character string type. There are built-in functions, that are preassembled, and user-defined functions, defined by the user.

(1) User-defined functions

The function is defined with the Def FN statement.

Example) Def FN MADD(MA, MB)=MA+MB

.....The function to obtain the total of two values is defined with FNMADD.

The function name starts with FN, and the data type identification character (C: character string, M: numeric value, P: position, J: joint) is described at the third character. The function is designated with up to eight characters.

(2) Built-in functions

A list of assembled functions is given in [Table 4-10](#).

Table 4-10:List of built-in functions

Class	Function name (format)	Functions	Page	Result
Numeric functions	Abs (<Numeric expression>)	Produces the absolute value	312	Numeric value
	Cint (<Numeric expression>)	Rounds off the decimal value and converts into an integer.	317	
	Deg (<Numeric expression:radian>)	Converts the angle unit from radian (rad) to degree (deg).	320	
	Exp (<Numeric expression>)	Calculates the value of the expression's exponential function	321	
	Fix (<Numeric expression>)	Produces an integer section	322	
	Int (<Numeric expression>)	Produces the largest integer that does not exceed the value in the expression.	324	
	Len(<Character string expression>)	Produces the length of the character string.	326	
	Ln (<Numeric expression>)	Produces the logarithm.	327	
	Log (<Numeric expression>)	Produces the common logarithm.	327	
	Max (<Numeric expression>...)	Obtains the max. value from a random number of arguments.	328	
	Min (<Numeric expression>...)	Obtains the min. value from a random number of arguments.	329	
	Rad (<Numeric expression: deg.>)	Converts the angle unit from radian (rad) to degree (deg).	333	
	Sgn (<Numeric expression>)	Checks the sign of the number in the expression	340	
	Sqr (<Numeric expression>)	Calculates the square root	341	
	Strpos(<Character string expression>, <Character string expression>)	Obtains the 2nd argument character string position in the 1st argument character string.	341	
	Rnd (<Numeric expression>)	Produces the random numbers.	335	
	Asc(<Character string expression>)	Provides a character code for the first character of the character string in the expression.	314	
	Cvi(<Character string expression>)	Converts a 2-byte character string into integers.	319	
	Cvs(<Character string expression>)	Converts a 4-byte character string into a single-precision real number.	319	
	Cvd(<Character string expression>)	Converts an 8-byte character string into a double-precision real number.	320	
	Val(<Character string expression>)	Converts a character string into a numeric value.	343	
Trigonometric functions	Atn(<Numeric expression>)	Calculates the arc tangent. Unit: radian Definition range: Numeric value, Value range: -PI/2 to +PI/2	314	Numeric value
	Atn2(<Numeric expression>,<Numeric expression>)	Calculates the arc tangent. Unit: radian THETA=Atn2(delta y, deltax) Definition range: Numeric value of delta y or delta x that is not 0 Value range: -PI to +PI	314	

Class	Function name (format)	Functions	Page	Result
Trigonometric functions	Cos(<Numeric expression>)	Calculates the cosine Unit: radian Definition range: Numeric value range, Value range: -1 to +1	318	Numeric value
	Sin(<Numeric expression>)	Calculates the sine Unit: radian Definition range: Numeric value range, Value range: -1 to +1	340	
	Tan(<Numeric expression>)	Calculates the tangent. Unit: radian Definition range: Numeric value range, Value range: Range of numeric value	342	
Character string functions	Bin\$(<Numeric expression>)	Converts numeric expression value into binary character string.	315	Character string
	Chr\$(<Numeric expression>)	Provides character having numeric expression value character code.	317	
	Hex\$(<Numeric expression>)	Converts numeric expression value into hexadecimal character string.	324	
	Left\$(<Character string expression>,<Numeric expression>)	Obtains character string having length designated with 2nd argument from left side of 1st argument character string.	326	
	Mid\$(<Character string expression>,<Numeric expression>,<Numeric expression>)	Obtains character string having length designated with 3rd argument from the position designated with the 2nd argument in the 1st argument character string.	328	
	Mirror\$(<Character string expression>)	Mirror reversal of the character string binary bit is carried out.	329	
	Mki\$(<Numeric expression>)	Converts numeric expression value into 2-byte character string.	330	
	Mks\$(<Numeric expression>)	Converts numeric expression value into 4-byte character string.	330	
	Mkd\$(<Numeric expression>)	Converts numeric expression value into 8-byte character string.	331	
	Right\$(<Character string expression>,<Numeric expression>)	Obtains character string having length designated with 2nd argument from right side of 1st argument character string.	335	
Position variables	Str\$(<Numeric expression>)	Converts the numeric expression value into a decimal character string.	342	Position
	CkSum(<Character string expression>,<Numeric expression>,<Numeric expression>)	Creates the checksum of a character string. Returns the value of the lower byte obtained by adding the character value of the second argument position to that of the third argument position, in the first argument character string.	318	
	Dist(<Position>,<Position>)	Obtains the distance between two points.	321	
	Fram(<Position 1>,<Position 2>,<Position 3>)	Calculates the coordinate system designated with three points. Position 1 is the plane origin, position 2 is the point on the +X axis, and position 3 is the point on the +Y axis direction plane. The plane origin point and posture are obtained from the XYZ coordinates of the three position, and is returned with a return value (position). This is operated with 6-axis three dimensions regardless of the mechanism structure. This function cannot be used in 5-axis robots, because the A, B, and C posture data has different meaning.	323	
	Rdf11(<Position>,<Numeric value>)	Returns the structure flag of the designated position as character data. Argument <numeric value>) 0 = R/L, 1 = A/B , 2 = F/N is returned.	333	Character
	Setfl1(<Position>,<Character>)	Changes the structure flag of the designated position. The data to be changed is designated with characters.(R/L/A/B/F/N)	336	
	Rdf12(<Position>,<Numeric value>)	Returns the multi-rotation data of the designated position as a numeric value (-2 to 1). The argument <numeric expression> returns the axis No. (1 to 8).	334	
	Setfl2(<Position>,<Numeric value>,<Numeric value>)	Changes the multi-rotation data of the designated position as a numeric value (-2 to 1). The left side of the expression is the axis No. to be changed; the right side is the value to be set.	337	
	Align(<Position>)	Returns the value of the XYZ position (0, +/-90, +/-180) closest to the position 1 posture axis (A, B, C). This function cannot be used in 5-axis robots, because the A, B, and C posture data has different meaning.	313	Position
	Inv(<Position>)	Obtains the reverse matrix.	325	
	PtoJ(<Position>)	Converts the position data into joint data.	332	
	JtoP(<Position>)	Converts the joint data into position data.	325	Position
Zone	Zone(<Position 1>,<Position 2>,<Position 3>)	Checks whether position 1 is within the space (Cube) created by the position 2 and position 3 points. Outside the range=0, Within the range=1 For position coordinates that are not checked or non-existent, the following values should be assigned to the corresponding position coordinates: If the unit is degrees, assign -360 to position 2 and 360 to position 3 If the unit is mm, assign -10000 to position 2 and 10000 to position 3	344	Numeric value

Class	Function name (format)	Functions	Page	Result
Position variables	Zone2 (<Position 1>,<Position 2>,<Position 3> <Numeric value1>,<Numeric value2>,<Numeric value3>,<Position 4>)	Checks whether position 1 is within the space (cylinder) created by the position 2 and position 3 points. Outside the range=0, Within the range=1 Only the X, Y, and Z coordinate values are considered; the A, B, and C posture data is ignored.	345	Numeric value
	PosCq(<Position>)	Checks whether <position> is within the movement range.	331	Numeric value
	PosMid (<Position1>,<Position2>,<Numeric value1>,<Numeric value2>)	Calculates the middle position between <position 1> and <position 2>.	332	Position
	CalArc (<Position 1>,<Position 2>,<Position 3> <Numeric value1>,<Numeric value2>,<Numeric value3>,<Position 4>)	Returns information of an arc created from <position 1>, <position 2>, and <position 3>.	316	Numeric value
	SetJnt (<J1 axis>,<J2 axis>,<J3 axis>,<J4 axis> <J5 axis>,<J6 axis>,<J7 axis>,<J8 axis>)	Sets values in joint variables.	338	Joint
	SetPos (<X axis>,<Y axis>,<Z axis>,<A axis> <B axis>,<C axis>,<L1 axis>,<L2 axis>)	Sets values in position variables.	339	Position

4.6 List of Instructions

A list of pages with description of each instruction is shown below. They are listed in the order of presumed usage frequency.

(1) Instructions related to movement control

Command	Explanation	Page
Mov (Move)	Joint interpolation	215
Mvs (Move S)	Linear interpolation	225
Mvr (Move R)	Circular interpolation	219
Mvr2 (Move R2)	Circular interpolation 2	221
Mvr3 (Move R 3)	Circular interpolation 3	223
Mvc (Move C)	Circular interpolation	218
Mva (Move Arch)	Arch motion interpolation	216
Mv Tune (Move Tune)	Specification of the moving characteristics mode	228
Ovrd (Override)	Overall speed specification	237
Spd (Speed)	Speed specification during linear or circular interpolation movement	251
JOvrd (J Override)	Speed specification during joint interpolation movement	211
Cnt (Continuous)	Continuous path mode specification	172
Accel (Accelerate)	Acceleration/deceleration rate specification	154
Cmp Jnt (Comp Joint)	Specification of compliance in the JOINT coordinate system	164
Cmp Pos (Composition Posture)	Specification of compliance in the XYZ coordinate system	166
Cmp Tool (Composition Tool)	Specification of compliance in the Tool coordinate system	168
Cmp Off (Composition OFF)	Compliance setting invalid	170
CmpG (Composition Gain)	Compliance gain specification	171
Mxt (Move External)	Optimum acceleration/deceleration rate specification	230
Loadset (Load Set)	Hand's optional condition specification	214
Prec (Precision)	High accuracy mode specification	239
Torq (Torque)	Torque specification of each axis	254
JRC (Joint Roll Change)	Enables multiple rotation of the tip axis	212
Fine (Fine)	Robot's positioning range specification	197
Fine J (Fine Joint)	Robot's positioning range specification by joint interpolation	199
Fine P (Fine Pause)	Robot's positioning range specification by distance in a straight line	200
Servo (Servo)	Servo motor power ON/OFF	249
Wth (With)	Addition instruction of movement instruction	257
WthIf (With If)	Additional conditional instruction of movement instruction	258

(2) Instructions related to program control

Command	Explanation	Page
Rem (Remarks)	Comment(')	243
If...Then...Else...EndIf (If Then Else)	Conditional branching	209
Select Case (Select Case)	Enables multiple branching	247
GoTo (Go To)	Jump	205
GoSub (Return)(Go Subroutine)	Subroutine jump	204
Reset Err (Reset Error)	Resets an error (use of default is not allowed)	244
CallP (Call P)	Program call	159
FPrm (FPRM)	Program call argument definition	202
Dly (Delay)	Timer	194
Hlt (Halt)	Suspends a program	206
End (End)	End a program	195

Command	Explanation	Page
On ... GoSub (ON Go Subroutine)	Subroutine jump according to the value	234
On ... GoTo (On Go To)	Jump according to the value	235
For - Next (For-next)	Repeat	201
While-WEnd (While End)	Conditional repeat	256
Open (Open)	Opens a file or communication line	236
Print (Print)	Outputs data	240
Input (Input)	Inputs data	210
Close (Close)	Closes a file or communication line	162
ColChk (Col Check)	Enables or disables the impact detection function	175
On Com GoSub (ON Communication Go Subroutine)	Communication interrupt subroutine jump	233
Com On/Com Off/Com Stop (Communication ON/OFF/STOP)	Allows/prohibits/stops communication interrupts	179
HOpen / HClose (Hand Open/Hand Close)	Hand's open/close	207
Error (error)	User error	196
Skip (Skip)	Skip while moving	250
Wait (Wait)	Waiting for conditions	255
Clr (Clear)	Signal clear	163

(3) Definition instructions

Command	Explanation	Page
Dim (Dim)	Array variable declaration	193
Def Plt (Define pallet)	Pallet declaration	190
Plt (Pallet)	Pallet position calculation	238
Def Act (Define act)	Interrupt definition	180
Act (Act)	Starts or ends interrupt monitoring	156
Def Arch (Define arch)	Definition of arch shape for arch motion	183
Def Jnt (Define Joint)	Joint type position variable definition	189
Def Pos (Define Position)	XYZ type position variable definition	192
Def Inte/Def Long/Def Float/Def Double (Define Integer/Long/Float/Double)	Integer or real number variable definition	186
Def Char (Define Character)	Character variable definition	184
Def IO (Define IO)	Signal variable definition	187
Def FN (Define function)	User function definition	185
Tool(Tool)	Hand length setting	253
Base (Base)	Robot base position setting	158
Tool(Tool)	Tool length setting	253

(4) Multi-task related

Command	Explanation	Page
XLoad (X Load)	Loads a program to another task slot	260
XRun (X Run)	Execute the program in another task slot	262
XStop (X Stop)	Stop the program in another task slot	263
XRst (X Reset)	Resets the program in another task slot being suspended	261
XClr (X Clear)	Cancels the loading of the program from the specified task slot	259
GetM (Get Mechanism)	Obtains mechanical control right	203
RelM (Release Mechanism)	Releases mechanical control right	242
Priority (Priority)	Changes the task slot priority	241
Reset Err (Reset Error)	Resets an error (use of default is not allowed)	244

(5) Others

Command	Explanation	Page
ChrSrch (Character search)	Searches the character string out of the character array.	161
Get Pos (Get Position)	Reserved.	-

4.7 Operators

The value's real number or integer type do not need to be declared. Instead, the type may be forcibly converted according to the operation type. (Refer to [Table 4-11](#).) The operation result data type is as follows according to the combination of the left argument and right argument data types.

Example)	Left argument	Operation	Right argument	Operation results
	<u>15</u> (Numeric value type)	<u>AND</u>	<u>256</u> (Numeric value type)	<u>15</u> (Numeric value type)
	<u>P1</u> (Position type)	<u>*</u>	<u>M1</u> (Numeric value type)	<u>P2</u> (Position type)
	<u>M1</u> (Numeric value type)	<u>*</u>	<u>P1</u> (Position type)	Description error

Table 4-11:Table of data conversions according to operations

Left argument type	Operation	Left argument type				
		Character string	Numeric value		Position	Joint
			Integer	Real number		
Character string	Substitution=	Character string	-	-	-	-
Integer	Addition +	Character string	-	-	-	-
	Comparison (Comparison operators)	Integer	-	-	-	-
	Addition +	-	Integer	Real number	-	-
	Subtract -	-	Integer	Real number	-	-
	Multiplication *	-	Integer	Real number	-	-
	Division /	-	Integer	Real number	-	-
	Integer division \	-	Integer	Integer	-	-
	Remainder MOD	-	Integer	Integer	-	-
	Exponent ^	-	Integer	Integer	-	-
	Substitution =	-	Integer	Integer	-	-
Real number	Comparison (Comparison operators)	-	Integer	Integer	-	-
	Logic (Logic operators)	-	Integer	Integer	-	-
	Addition +	-	Real number	Real number	-	-
	Subtract -	-	Real number	Real number	-	-
	Multiplication *	-	Real number	Real number	-	-
	Division /	-	Real number	Real number	-	-
	Integer division \	-	Integer	Integer	-	-
	Remainder MOD	-	Integer	Integer	-	-
	Exponent ^	-	Integer	Real number	-	-
	Substitution =	-	Integer	Real number	-	-
Position	Comparison (Comparison operators)	-	Integer	Integer	-	-
	Logic (Logic operators)	-	Integer	Integer	-	-
	Addition +	-	-	-	Position	-
	Subtract -	-	-	-	Position	-
	Multiplication *	-	Position	Position	Position	-
	Division /	-	Position	Position	Position	-
	Integer division \	-	-	-	-	-
	Remainder MOD	-	-	-	-	-
	Exponent ^	-	-	-	-	-
	Substitution =	-	-	-	Position	-
Joint	Comparison (Comparison operators)	-	-	-	-	-
	Logic (Logic operators)	-	-	-	-	-
	Addition +	-	-	-	-	Joint
	Subtract -	-	-	-	-	Joint
	Multiplication *	-	Joint	Joint	-	-
	Division /	-	Joint	Joint	-	Joint
	Integer division \	-	-	-	-	-
	Remainder MOD	-	-	-	-	-
	Exponent ^	-	-	-	-	-
	Substitution =	-	-	-	-	Joint
Right argument only (Single argument)	Reversal -	-	Integer	Integer	Position	Joint
	Negate NOT	-	Integer	Integer	-	-

Reversal: Sign reversal, Negate: Logical negate, Substitute: Substitute operation, Remainder: Remainder operation, Comparison: Comparison operation, Logic: Logical Operation (excluding logical negate).

[Caution]

- The operation of the section described with a "-" is not defined.
- The results of the integer and the integer multiplication/division is an integer type for multiplication, and a real number type for division.
- If the right argument is a 0 divisor (divide by 0), an operation will not be possible.
- During exponential operation, remainder operation or logical operation (including negate), all real numbers will be forcibly converted into integers (rounded off), and operated.

4.8 Priority level of operations

In the event there are many operators within an expression being calculated, the order of operations is as shown in [Table 4-12](#).

Table 4-12:Priority level of operations

Operations, (operators)	Type of operation	Priority level
1) Operations inside parentheses ()		High
2) Functions	Functions	:
3) Exponents	Numeric value operation	:
4) Single argument operator (+, -)	Numeric value operation	:
5) * /	Numeric value operation	:
6) \	Numeric value operation	:
7) MOD	Numeric value operation	:
8) + -	Numeric value operation	:
9)<< >>	Logic operation	:
10) Comparison operator (=,<>,><,<,<=,>=,=)	Comparison operation	:
11)NOT	Logic operation	:
12)AND	Logic operation	:
13)OR	Logic operation	:
14)XOR	Logic operation	Low

4.9 Depth of program's control structure

When creating a program, the depth of the control structure must be considered.

When using the commands in the table below, the program's level of control structure becomes one level deeper. Each command has a limit to the depth of the control structure. Exceeding these limits will cause an error.

Table 4-13:Limit to control structure depth

	No. of levels	Applicable commands
User stack in program	16 levels	Repeated controls (For-Next,While-WEnd)
	8 levels	Function calling (CallP)
	800 levels max.	Subroutine calling (GoSub) The number decreases by usage frequency of For-Next, While-WEnd, and CallP instructions.

4.10 Reserved words

Reserved words are those that are already used for the system.

A name that is the same as one of the reserved words cannot be used in the program.

Instructions, functions, and system status variables, etc. are considered reserved words.

4.11 Detailed explanation of command words

4.11.1 How to read the described items

[Function]	: Indicates the command word functions.
[Format]	: Indicates how to input the command word argument. The argument is shown in <>. [] indicates that the argument can be omitted. [] indicates that a space is required.
[Terminology]	: Indicates the meaning and range, etc. of the argument.
[Reference Program]	: Indicates a program example.
[Explanation]	: Indicates detailed functions and cautions, etc.
[The available robot type]	: Indicates the available robot type.
[Related parameter]	: Indicates the related parameter.
[Related system variables]	: Indicates the related system variables.
[Related instructions]	: Indicates the related instructions.

4.11.2 Explanation of each command word

Each instruction is explained below in alphabetical order.

Accel (Accelerate)

[Function]

Designate the robot's acceleration and deceleration speeds as a percentage (%). It is valid during optimum acceleration/deceleration.

* The acceleration/deceleration time during optimum acceleration/deceleration refers to the optimum time calculated when using an Oadl instruction, which takes account of the value of the M_SetAdl variable.

[Format]

```
Accel[] [<Acceleration rate>] [, <Deceleration rate>]
          ,[<Acceleration rate when moving upward>], [<Deceleration rate when moving upward>]
          ,[<Acceleration rate when moving downward>], [<Deceleration rate when moving down-
          ward>]
```

[Terminology]

<Acceleration/Deceleration>

1 to 100(%). Designate the acceleration/deceleration to reach the maximum speed from speed 0 as a percentage. This can be described as a constant or variable. A default value of 100 is set if the argument is omitted. A value of 100 corresponds to the maximum rate of acceleration/deceleration. Unit: %

<Acceleration/Deceleration rate when moving upward>

Specify the acceleration/deceleration rate when moving upward in an arch motion due to the Mva instruction.

A default value of 100 is set if the argument is omitted. It is possible to specify the argument either by a constant or variable.

<Acceleration/Deceleration rate when moving downward>

Specify the acceleration/deceleration rate when moving downward in an arch motion due to the Mva instruction.

A default value of 100 is set if the argument is omitted. It is possible to specify the argument either by a constant or variable.

[Reference Program]

1 Accel 50,100	' Heavy load designation (when acceleration/deceleration is 0.2 seconds, the acceleration will be 0.4, and the deceleration will be 0.2 seconds).
2 Mov P1	
3 Accel 100,100	' Standard load designation.
4 Mov P2	
5 Def Arch 1,10,10,25,25,1,0,0	
6 Accel 100,100,20,20,20,20	' Specify the override value to 20 when moving upward or downward due to the Mva instruction.
7 Mva P3,1	

[Explanation]

- (1) The maximum acceleration/deceleration is determined according to the robot being used. Set the corresponding percentage(%). The system default value is 100,100.
- (2) The acceleration percentage changed with this command is reset to the system default value when the program is reset or the End statement executed.
- (3) Although it is possible to describe the acceleration/deceleration time to more than 100%, some models internally set its upper limit to 100%. If the acceleration/deceleration time is set to more than 100%, it may affect the lifespan of the machine. In addition, speed-over errors and overload errors may tend to occur. Therefore, be extra careful when you are setting it to more than 100%.
- (4) The smooth operation when Cnt is valid will have a different locus according to the acceleration speed or operation speed. To move smoothly at a constant speed, set the acceleration and deceleration to the same value. Cnt is invalid in the default state.
- (5) It is also valid during optimum acceleration/deceleration control (Oadl On).

[Related instructions]

[Mxt \(Move External\)](#), [Loadset \(Load Set\)](#)

[Related system variables]

[M_Acl/M_DAcl/M_NAcl/M_NDAcl/M_AclSts](#), [M_SetAdl](#)

[Related parameter]

JADL

Act (Act)

[Function]

This instruction specifies whether to allow or prohibit interrupt processing caused by signals, etc. during operation.

[Format]

```
Act[]<Priority No.> = <1/0>
```

[Terminology]

<Priority No.>	0: Either enables or disables the entire interrupt. 1 to 8: Designate the priority No. for the interrupt defined in the Def Act statement. When entering the priority No., always leave a space (character) after the Act command. If described as Act1, it will be a variable name declaration statement.
<1/0>	1: Allows interrupts, 0:Prohibits interrupts.

[Reference Program]

- (1) When the input signal 1 turns on (set to 1) while moving from P1 to P2, it loops until that signal is set to 0.

```
1 Def Act 1,M_In(1)=1 GoSub *INTR      ' Assign input signal 1 to the interrupt 1 condition
2 Mov P1
3 Act 1=1
4 Mov P2
5 Act 1=0
:
10 *INTR
11 IF M_In(1)=1 GoTo 110
12 Return 0
```

' Enable interrupt 1.
' Disable interrupt 1.
' Loops until the M_In(1) signal becomes 0.
'
- (2) When the input signal 1 turns on (set to 1)while moving from P1 to P2, Operation is interrupted and the output signal 10 turns on.

```
1 Def Act 1,M_In(1)=1 GoSub *INTR      'Assign input signal 1 to the interrupt 1 condition
2 Mov P1
3 Act 1=1
4 Mov P2
:
10 *INTR
11 Act 1=0
12 M_Out(10)=1
13 Return 1
```

' Enable interrupt 1.
' Disable interrupt 1.
' Turn on the output signal 10
' Returns to the next step which interrupted

[Explanation]

- (1) When the program starts, the status of <Priority No.> 0 is "enabled." When <Priority No.> 0 is "disabled," even if <Priority No.> 1 to 8 are set to "enabled," no interrupt will be enabled.
- (2) The statuses of <Priority No.> 1 to 8 are all "disabled" when the program starts.
- (3) An interrupt will occur only when all of the following conditions have been satisfied:
 - *<Priority No.> 0 is set to "enabled."
 - *The status of the Def Act statement has been defined.
 - *When the <Priority No.> designated by Def Act is made valid by an Act statement.
- (4) The return from an interrupt process should be done by describing either RETURN 0 or RETURN 1. However when returning from interruption processing to the next step by RETURN1, execute the statement to disable the interrupt. When that is not so, if interruption conditions have been satisfied, because interruption processing will be executed again and it will return to the next step, the step may be skipped.
- (5) Even if the robot is in the middle of interpolation, an interrupt defined by a Def Act statement will be executed.
- (6) During an interrupt process, that <Priority No.> will be executed with the status as "disable."
- (7) A communications interrupt (COM) has a higher priority than an interrupt defined by a Def Act statement.
- (8) The relationship of priority rankings is as shown below:
COM>Act>WthIf(Wth)>Pulse substitution

[Related instructions] [Def Act \(Define act\)](#), [Return \(Return\)](#)

Base (Base)

[Function]

With this instruction, it is possible to move or rotate the robot coordinate system. Specify the base conversion data for this instruction. Pay extra attention when making changes in a program, as it may be mistaken in jog operations, etc.

[Format]

Base[]<Base conversion data>

[Terminology]

<Base conversion data> Specify with position constants or position variables.

[Reference Program]

1 Base (50,100,0,0,0,90) ' Input the conversion data as a constant.

2 Mvs P1

3 Base P2

' Input the conversion data as a variable.

4 Mvs P1

5 Base P_NBase

' Reset the conversion data to the default values.

[Explanation]

(1) The X, Y, and Z components of the position data represent the amount of parallel movement from the origin of the world coordinate system to the origin of the base coordinate system. The base conversion data can be changed only with the Base command. The components A, B, and C of the position data represent the amount that the base coordinate system is tilted in relation to the world coordinate system.

X.....Distance to move parallel to X axis

Y.....Distance to move parallel to Y axis

Z.....Distance to move parallel to Z axis

A.....Angle to turn toward the X axis

B.....Angle to turn toward the Y axis

C.....Angle to turn toward the Z axis

L1.....Movement amount of additional axis 1

L2.....Movement amount of additional axis 2

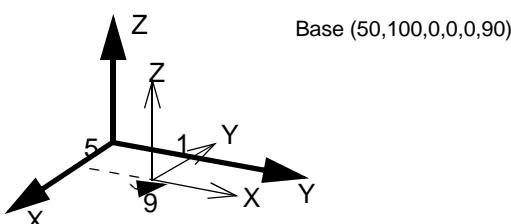
(1) For A, B and C, the clockwise direction looking from the front of the origin of the coordinate, used as the center of rotation, is the forward rotation direction.

(2) The contents of the structural flag have no meaning.

(3) The system's default value for this data is P_NBase=(0,0,0,0,0,0) (0,0). This is calculated with the 6-axis three dimensional regardless of the mechanism structure.

(4) The valid axis element of base conversion data is different depending on the type of robot. Set up the appropriate data referring to the [Page 371, "Table 5-9: Valid axis elements of the base conversion data depending on the robot model"](#).

Fig.4-3:Conceptual diagram of the base coordinate system



[Related parameter]

After it has been changed by the MEXBS Base instruction, the base coordinate system is stored in the MEXBS parameter and maintained even after the controller's power is turned off. Refer to [Page 371, "About Standard Base Coordinates"](#).

[Related system variables]

[P_Base/P_NBase](#)

CallP (Call P)

[Function]

This instruction executes the specified program (by calling the program in a manner similar to using GoSub to call a subroutine). The execution returns to the main program when the End instruction or the final step in the sub program is reached.

[Format]

```
CallP[] "<Program name> " [, <Argument> [, <Argument>]
```

[Terminology]

- | | |
|----------------|--|
| <Program name> | Designate the program name with a character string constant or character string variable.
For the standards for program names, please refer to Page 126, "(1) Program name" . |
| <Argument> | Designate the variable to be transferred to the program when the program is called. Up to 16 variables can be transferred. |

[Reference Program]

(1) When passing the argument to the program to call.

```
Main program
1 M1=0
2 CallP "10" ,M1,P1,P2
3 M1=1
4 CallP "10" ,M1,P1,P2
:
10 CallP "10", M2,P3,P4
:
15 End
"100" sub program side
100 FPrm M01, P01,P02
101 If M01<>0 Then GoTo *LBL1
102 Mov P01
103 *LBL1
104 Mvs P02
105 End           'Return to the main program at this point.
```

* When step 2 and 4 of the main program are executed, M1, P1 and P2 are set in M01, P01 and P02 of the sub program, respectively. When step 100 of the main program is executed, M2, P3 and P4 are set in M01, P01 and P02 of the sub program, respectively.

(2) When not passing the argument to the program to call.

```
Main program
1 Mov P1
2 CallP "20"
3 Mov P2
4 CallP "20"
5 End

"200" sub program side
201 Mov P1           'P1 of the sub program differs from P1 of the main program.
202 Mvs P002
203 M_Out(17)=1
204 End             'Return to the main program at this point.
```

[Explanation]

- (1) A program (sub program) called by the CallP instruction will return to the parent program (main program) when the End instruction (equivalent to the Return instruction of GoSub) is reached. If there is no End instruction, the execution is returned to the main program when the final step of the sub program is reached.
- (2) If arguments need to be passed to the sub program, they should be defined using the FPrm instruction at the beginning of the sub program.
- (3) If the type or the number of arguments passed to the sub program is different from those defined (by the FPrm instruction) in the sub program, an error occurs at execution.
- (4) If a program is reset, the control returns to the beginning of the top main program.
- (5) Definition statements (Def Act, Def FN, Def Plt, and Dim instructions) executed in the main program are invalid in a program called by the CallP instruction. They become valid when the control is returned to the main program from the program called by the CallP instruction again.
- (6) Speed and tool data are all valid in a sub program. Values of Accel and Spd are invalid. The mode of Oadl is valid.
- (7) Another sub program can be executed by calling CallP in a sub program. However, a main program or a program that is currently being executed in another task slot cannot be called. In addition, own program cannot be called, either.
- (8) Eight levels (in a hierarchy) of sub programs can be executed by calling CallP in the first main program.
- (9) Variable values may be passed from a main program to a sub program using arguments, however, it is not possible to pass the processing result of a sub program to a main program by assigning it in an argument. To use the processing result of a sub program in a main program, pass the values using external variables.

[Related instructions]

[FPrm \(FPRM\)](#)

ChrSrch (Character search)

[Function]

Searches the character string out of the character array.

[Format]

ChrSrch[]<Character string array variable>,<Character string>,<Search result storage destination>

[Terminology]

<Character string array variable> Specify the character string array to be searched.
 <Character string> Specify the character string to be searched.
 <Search result storage destination> The number of the element for which the character string to be searched is found is set.

[Reference Program]

```

1 Dim C1$(10)
2 C1$(1)="ABCDEFG"
3 C1$(2)="MELFA"
4 C1$(3)="BCDF"
5 C1$(4)="ABD"
6 C1$(5)="XYZ"
7 C1$(6)="MELFA"
8 C1$(7)="CDF"
9 C1$(8)="ROBOT"
10 C1$(9)="FFF"
11 C1$(10)="BCD"
12 ChrSrch C1$(1), "ROBOT", M1      ' 8 is set in M1.
13 ChrSrch C1$(1), "MELFA", M2      ' 2 is set in M2.
  
```

[Explanation]

- (1) The specified character string is searched from the character string array variables, and the element number of the completely matched character string array is set in <search result storage destination>. Partially matched character strings are not searched.
 Even if ChrSrch C1\$(1), "ROBO", M1 are described in the above statement example, the matched character string is not searched.
- (2) If the character string to be searched is not found, 0 is set in <search result storage destination>.
- (3) Character string search is performed sequentially beginning with element number 1, and the element number found first is set.
 Even if ChrSrch C1\$(3), "MELFA", M2 are described in the above statement example, 2 is set in M2.
 (The same character string is set in C1\$(2) and C1\$(6).)
- (4) The <character string array variable> that can be searched is the one-dimensional array only. If a two-dimensional or higher array is specified as a variable, an error will occur at the time of execution.

Close (Close)

[Function]

Closes the designated file.(including communication lines)

[Format]

```
Close[] [[#]<File No.>[, [[#]<File No.> ...]
```

[Terminology]

<File No. > Specify the number of the file to be closed (1 to 8). Only a numerical constant is allowed.
If this argument is omitted, all open files are closed.

[Reference Program]

```
1 Open "COM1:" AS #1                ' "Open "COM1:" as file No. 1.  
2 Print #1,M1  
:  
10 Input #1,M2  
11 Close #1                        ' Close file No. 1, "COM1:".  
:  
20 Close                            ' Close all open files.
```

[Explanation]

- (1) This instruction closes files (including communication lines) opened by the Open instruction. Data remaining in the buffer is flushed.
The data left in the buffer will be processed as follows when the file is closed:

Table 4-14:Processing of each buffer when the file is closed

Buffer types	Processing when the file is closed
Communication line reception buffer	The contents of the buffer are destroyed
Communication line transmission buffer	(No data remains in the transmission buffer since the data in the transmission buffer is sent immediately by executing the Print instruction.)
File load buffer	The contents of the buffer are destroyed.
File unload buffer	The contents of the buffer are written into the file, and then the file is closed.

- (2) Executing an End statement will also close a file.
(3) If the file number is omitted, all files will be closed.

[Related instructions]

[Open \(Open\)](#), [Print \(Print\)](#), [Input \(Input\)](#)

Clr (Clear)

[Function]

This instruction clears general-purpose output signals, local numerical variables in a program, and numerical external variables.

[Format]

Clr[]<Type>

[Terminology]

<Type>

It is possible to specify either a constant or a variable.

0 : All steps 1 to 3 below are executed.

1 : The general-purpose output signal is cleared based on the output reset pattern.

The output reset pattern is designated with parameters ORST0 to ORST224.

Refer to [Page 379, "5.14 About the output signal reset pattern"](#).

(0: OFF, 1: ON, *: Hold)

2 : All local numeric variables and numeric array variables used in the program are cleared to zero

3 : Clears all external numerical variables (External system variables and user-defined external variables) and external numerical array variables, setting them to 0. External position variables are not cleared.

[Reference Program]

(1) The general-purpose output signal is output based on the output reset pattern.

1 Clr 1

(2) The local numeric variables and numeric array variables in the program are cleared to 0.

1 Dim MA(10)

2 Def Inte IVAL

3 Clr 2 ' Clears MA(1) through MA(10), IVAL and local numeric variables in the program to 0.

(3) All external numeric array variables and external numeric array variables are cleared to 0

1 Clr 3

(4) (1) through (3) above are performed simultaneously.

1 Clr 0

[Related parameter]

ORST0 to ORST224

[Related system variables]

[M_In/M_Inb/M_Inw, M_Out/M_Outb/M_Outw](#)

Cmp Jnt (Comp Joint)

[Function]

Start the soft control mode (compliance mode) of the specified axis in the JOINT coordinates system.
Note) The available robot type is limited. Refer to "[Available robot type]".

[Format]

Cmp[]JNT, <Axis designation>

[Terminology]

<Axis designation> Specify the axis to be controlled in a pliable manner with the bit pattern.
1 : Enable, 0 : Disable &B00000000
This corresponds to axis 87654321.

[Reference Program]

```

1 Mov P1
2 CmpG 0.0,0.0,1.0,1.0, , , , ' Set softness.
3 Cmp Jnt,&B11           ' The J1 and J2 axes are put in the state where they are controlled in a
pliable manner.
4 Mov P2
5 HOpen 1
6 Mov P1
7 Cmp Off               ' Return to normal state.

```

[Explanation]

- (1) It is possible to control each of the robot's axes in the joint coordinate system in a pliable manner. For example, if using a horizontal multi-joint robot to insert pins in a workpiece by moving the robot's hand up and down, it is possible to insert the pins more smoothly by employing pliable control of the J1 and J2 axes (see the statement example above).
- (2) The degree of compliance can be specified by the CmpG instruction, which sets the spring constant. If the robot is of the RH-SDH type, specify 0.0 for the horizontal axes J1 and J2 to make the robot behave equivalently to a servo free system (the spring constant is zero). (Note that the vertical axes cannot be made to behave equivalently to a servo free system even if 0.0 is set for them. Also, be careful not to let these axes reach a position beyond the movement limit or where the amount of diversion becomes too large.) Note that 4) and 5) below do not function if this servo-free equivalent behavior is in use.
- (3) The soft state is maintained even after the robot program execution is stopped. To cancel the soft status, execute the "Cmp Off" command or turn Off the power.
- (4) When pressing in the soft state, the robot cannot move to positions that exceed the operation limit of each joint axis.
- (5) If the amount of difference between the original target position and the actual robot position becomes greater than 200 mm by pushing the hand, etc., the robot will not move any further and the operation shifts to the next step of the program.
- (6) It is not possible to use Cmp Jnt, POS, and Tool at the same time. In other words, an error occurs if the Cmp Pos or Cmp Tool instruction is executed while the Cmp Jnt instruction is being performed. Cancel the Cmp Jnt instruction once using the Cmp Off instruction to execute these instructions.
- (7) Be aware that the position of the robot may change if the servo status is switched on while this instruction is active.
- (8) It is possible to perform jog operations while the robot is in compliance mode. However, the setting of the compliance mode cannot be canceled by the T/B; in order to do so, execute this instruction in a program or execute it directly via the program edit screen of the T/B.
- (9) To change the axis specification, cancel the compliance mode with the Cmp Off instruction first, and then execute the Cmp Jnt instruction again.
- (10) The compliance mode is valid only for the robot arm axes. It is not valid for additional axes, even if specified.
- (11) If a positioning completion condition is specified using the Fine instruction while the compliance mode is activated, depending on the operation the robot may be unable to reach the positioning completion

pulse of the target position, and will wait indefinitely for the completion of the operation instruction. As a result, the program execution comes to a halt. Do not use the compliance mode and the Fine instruction at the same time.

⚠ CAUTION

The compliance mode is in effect continuously until the Cmp Off instruction is executed, or the power is turned off.

⚠ CAUTION

To execute a jog operation after setting the compliance mode with the Cmp Jnt instruction, use the JOINT jog mode.

If any other jog mode is used, the robot may operate in a direction different from the expected moving direction because the directions of the coordinate systems controlled by the jog operation and the compliance mode differ.

⚠ CAUTION

When performing the teaching of a position while in the compliance mode, perform servo OFF first.

Be careful that if teaching operation is performed with Servo ON, the original command position is taught, instead of the actual robot position. As a result, the robot may move to a location different from what has been taught.

[Available robot type]

RH-SDH series

[Related system variables]

M_BTime

[Related instructions]

[Cmp Off \(Composition OFF\)](#), [CmpG \(Composition Gain\)](#), [Cmp Pos \(Composition Posture\)](#), [Cmp Tool \(Composition Tool\)](#)

Cmp Pos (Composition Posture)

[Function]

Start the soft control mode (compliance mode) of the specified axis in the XYZ coordinates system.
Note) The available robot types for this instruction are limited. Refer to "[Available robot type]".

[Format]

Cmp[]Pos, <Axis designation>

[Terminology]

<Axis designation> Designate axis to move softly with a bit pattern.
1 : Enable, 0 : Disable &B00000000
This corresponds to axis L2L1CBAZYX

[Reference Program]

1 Mov P1	' Move in front of the part insertion position.
2 CmpG 0.5, 0.5, 1.0, 0.5, 0.5, , ,	' Set softness
3 Cmp Pos, &B011011	' The X, Y, A, and B axes are put in the state where they are controlled in a pliable manner.
4 Mvs P2	' Moves to the part insertion position.
5 M_Out(10)=1	' Instructs to close the chuck for positioning.
6 Dly 1.0	' Waits for the completion of chuck closing.(1 sec.)
7 HOpen 1	' Open the hand.
8 Mvs, -100	' Retreats 100 mm in the Z direction of the Tool coordinate system.
9 Cmp Off	' Return to normal state.

[Explanation]

- (1) The robot can be moved softly with the XYZ coordinate system.
For example, when inserting a pin in the vertical direction, if the X, Y, A and B axes are set to soft operation, the pin can be inserted smoothly.
- (2) The degree of softness can be designated with the CmpG command.
- (3) The soft state is maintained even after the robot program execution is stopped. To cancel the soft status, execute the "Cmp Off" command or turn Off the power.
- (4) When pressing in the soft state, the robot cannot move to positions that exceed the operation limit of each joint axis.
- (5) The deviation of the command position and actual position can be read with M_CmpDst. The success/failure of pin insertion can be checked using this variable.
- (6) If the amount of difference between the original target position and the actual robot position becomes greater than 200 mm by pushing the hand, etc., the robot will not move any further and the operation shifts to the next step of the program.
- (7) It is not possible to use Cmp Jnt, POS, and Tool at the same time. In other words, an error occurs if the Cmp Pos or Cmp Tool instruction is executed while the Cmp Jnt instruction is being performed. Cancel the Cmp Jnt instruction once using the Cmp Off instruction to execute these instructions.
- (8) If the servo turns from Off to On while this command is functioning, the robot position could change.
- (9) It is possible to perform jog operations while the robot is in compliance mode. However, the setting of the compliance mode cannot be canceled by the T/B; in order to do so, execute this instruction in a program or execute it directly via the program edit screen of the T/B.
- (10) To change the axis specification, cancel the compliance mode with the Cmp Off instruction first, and then execute the Cmp Pos instruction again.
- (11) If the robot is operated near a singular point, an alarm may be generated or control may be disabled.
Do not operate the robot near a singular point. If this situation occurs, cancel the compliance mode by executing a Cmp Off instruction once with servo Off (or turning Off and then On the power again), keep the robot away from a singular point, and then make the compliance mode effective again.
- (12) The compliance mode is valid only for the robot arm axes. It is not valid for additional axes, even if specified.

(13) If a positioning completion condition is specified using the Fine instruction while the compliance mode is activated, depending on the operation the robot may be unable to reach the positioning completion pulse of the target position, and will wait indefinitely for the completion of the operation instruction. As a result, the program execution comes to a halt. Do not use the compliance mode and the Fine instruction at the same time.

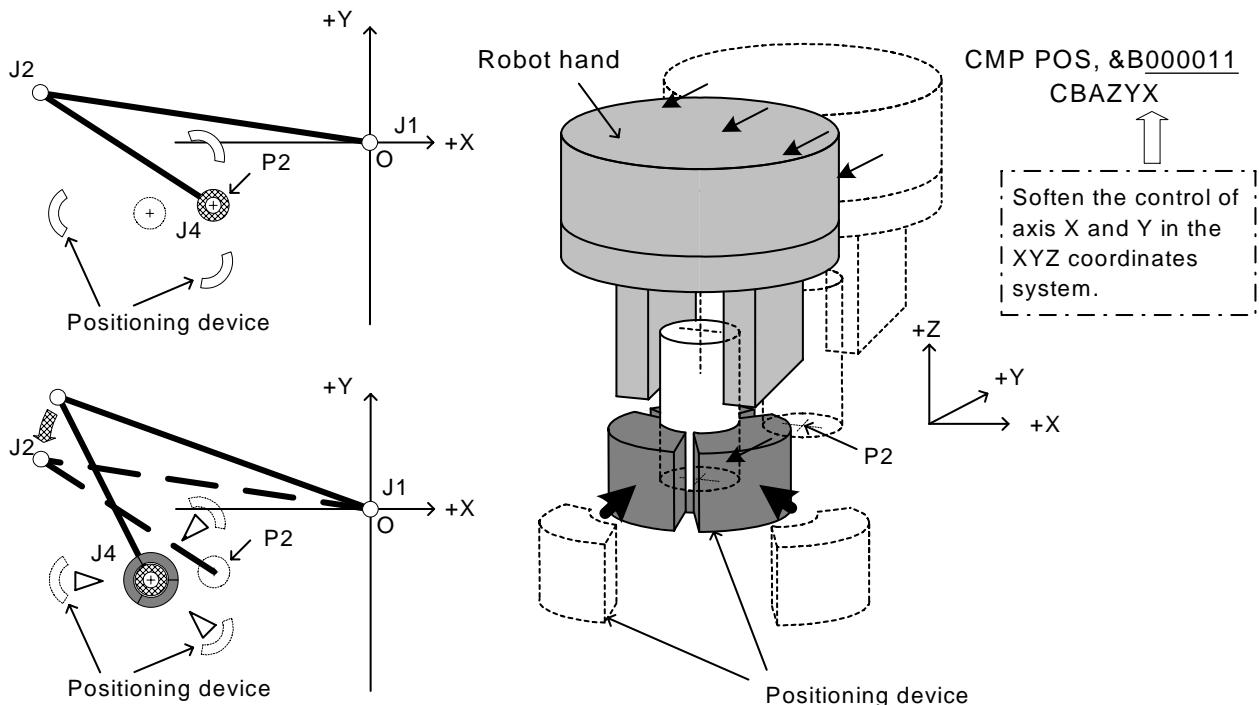


Fig.4-4:The example of compliance mode use



The compliance mode is in effect continuously until the Cmp Off instruction is executed, or the power is turned off. Exercise caution when changing the executable program number or operating the jog.



To execute a jog operation after setting the compliance mode with the Cmp Pos instruction, use the XYZ jog mode.
If any other jog mode is used, the robot may operate in a direction different from the expected moving direction because the directions of the coordinate systems controlled by the jog operation and the compliance mode differ.



When performing the teaching of a position while in the compliance mode, perform servo OFF first.
Be careful that if teaching operation is performed with Servo ON, the original command position is taught, instead of the actual robot position. As a result, the robot may move to a location different from what has been taught.

[Available robot type]

RV-6SD/6SDL/12SD/12SDL series

[Related system variables]

M_BTime

[Related instructions]

[Cmp Off \(Composition OFF\)](#), [CmpG \(Composition Gain\)](#), [Cmp Tool \(Composition Tool\)](#), [Cmp Jnt \(Comp Joint\)](#)

Cmp Tool (Composition Tool)

[Function]

Start the soft control mode (compliance mode) of the specified axis in the Tool coordinates system.
Note) The available robot types for this instruction are limited. Refer to "[Available robot type]".

[Format]

Cmp[]Tool, <Axis designation>

[Terminology]

<Axis designation> Designate axis to move softly with a bit pattern.
1 : Enable, 0 : Disable &B00000000
This corresponds to axis L2L1CBAZYX

[Reference Program]

1 Mov P1	' Moves to in front of the part insertion position.
2 CmpG 0.5, 0.5, 1.0, 0.5, 0.5, , ,	' Set softness.
3 Cmp Tool, &B011011	' The X, Y, A, and B axes are put in the state where they are controlled in a pliable manner.
4 Mvs P2	' Moves to the part insertion position.
5 M_Out(10)=1	' Instructs to close the chuck for positioning.
6 Dly 1.0	' Waits for the completion of chuck closing.(1 sec.)
7 HOpen 1	' Open the hand.
8 Mvs, -100	' Retreats 100 mm in the Z direction of the Tool coordinate system.
9 Cmp Off	' Return to normal state.

[Explanation]

- (1) The robot can be moved softly with the tool coordinate system. For the tool coordinate system, please refer to [Page 369, "5.6 Standard Tool Coordinates"](#).
- (2) For example, when inserting a pin in the tool coordinate Z axis direction, if the X, Y, A and B axes are set to soft operation, the pin can be inserted smoothly.
- (3) The degree of softness can be designated with the CmpG command.
- (4) The soft state is maintained even after the robot program execution is stopped. To cancel the soft status, execute the "Cmp Off" command or turn Off the power.
- (5) When pressing in the soft state, the robot cannot move to positions that exceed the operation limit of each joint axis.
- (6) The deviation of the command position and actual position can be read with M_CmpDst. The success/failure of pin insertion can be checked using this variable.
- (7) If the amount of difference between the original target position and the actual robot position becomes greater than 200 mm by pushing the hand, etc., the robot will not move any further and the operation shifts to the next step of the program.
- (8) It is not possible to use Cmp Jnt, POS, and Tool at the same time. In other words, an error occurs if the Cmp Pos or Cmp Tool instruction is executed while the Cmp Jnt instruction is being performed. Cancel the Cmp Jnt instruction once using the Cmp Off instruction to execute these instructions.
- (9) If the servo turns from Off to On while this command is functioning, the robot position could change.
- (10) It is possible to perform jog operations while the robot is in compliance mode. However, the setting of the compliance mode cannot be canceled by the T/B; in order to do so, execute this instruction in a program or execute it directly via the program edit screen of the T/B.
- (11) To change the axis specification, cancel the compliance mode with the Cmp Off instruction first, and then execute the Cmp Tool instruction again.
- (12) For vertical 5-axis robots (such as the RV-3SDJ), only the X and Z axes can be used for axis specification.
- (13) If the robot is operated near a singular point, an alarm may be generated or control may be disabled. Do not operate the robot near a singular point. If this situation occurs, cancel the compliance mode by exe-

- cuting a Cmp Off instruction once with servo Off (or turning Off and then On the power again), keep the robot away from a singular point, and then make the compliance mode effective again.
- (14) The compliance mode is valid only for the robot arm axes. It is not valid for additional axes, even if specified.
- (15) If a positioning completion condition is specified using the Fine instruction while the compliance mode is activated, depending on the operation the robot may be unable to reach the positioning completion pulse of the target position, and will wait indefinitely for the completion of the operation instruction. As a result, the program execution comes to a halt. Do not use the compliance mode and the Fine instruction at the same time.

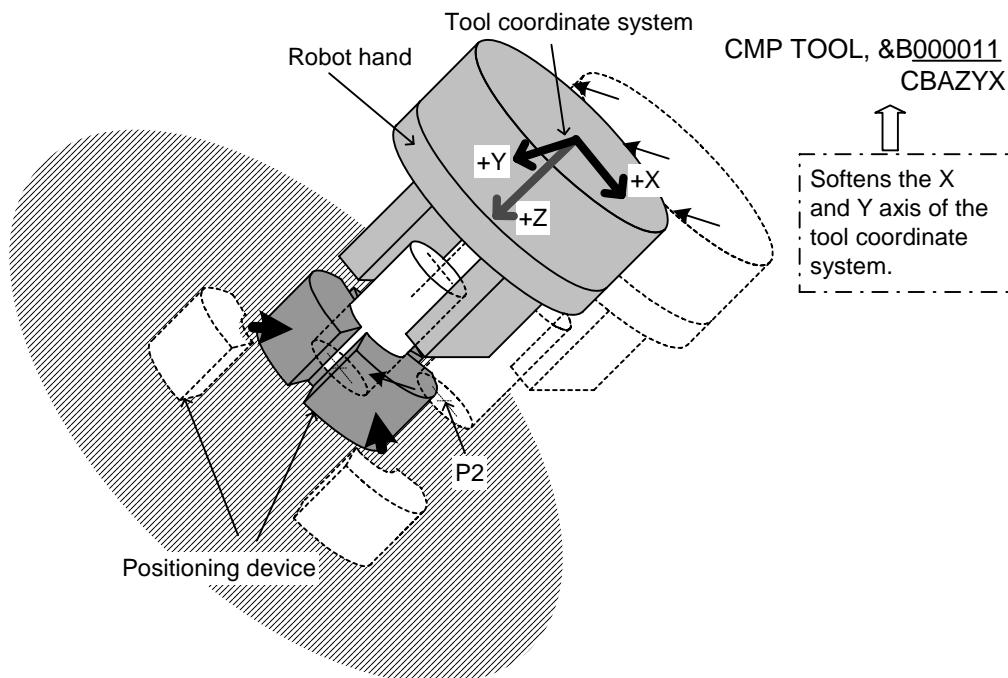


Fig.4-5:The example of using the compliance mode



CAUTION The compliance mode is in effect continuously until the Cmp Off instruction is executed, or the power is turned off. Exercise caution when changing the executable program number or operating the jog.



To execute a jog operation after setting the compliance mode with the Cmp Tool instruction, use the Tool jog mode.
If any other jog mode is used, the robot may operate in a direction different from the expected moving direction because the directions of the coordinate systems controlled by the jog operation and the compliance mode differ.



When performing the teaching of a position while in the compliance mode, perform servo Off first.
Be careful that if teaching operation is performed with Servo On, the original command position is taught, instead of the actual robot position. As a result, the robot may move to a location different from what has been taught.

[Available robot type]

RV-6SD/6SDL/12SD/12SDL series

[Related system variables]

[M_BTIme](#)

[Related instructions]

[Cmp Off \(Composition OFF\)](#), [CmpG \(Composition Gain\)](#), [Cmp Pos \(Composition Posture\)](#), [Cmp Jnt \(Comp Joint\)](#)

Cmp Off (Composition OFF)

[Function]

Release the soft control mode (compliance mode).

Note) The available robot types for this instruction are limited. Refer to "[Available robot type]".

[Format]

```
Cmp[]Off
```

[Reference Program]

1 Mov P1	' Moves to in front of the part insertion position.
2 CmpG 0.5, 0.5, 1.0, 0.5, 0.5, , ,	' Set softness.
3 Cmp Tool, &B011011	' The X, Y, A, and B axes are put in the state where they are controlled in a pliable manner.
4 Mvs P2	' Moves to the part insertion position.
5 M_Out(10)=1	' Instructs to close the chuck for positioning.
6 Dly 1.0	' Waits for the completion of chuck closing.(1 sec.)
7 HOpen 1	' Open the hand.
8 Mvs, -100	' Retreats 100 mm in the Z direction of the Tool coordinate system.
9 Cmp Off	' Return to normal state.

[Explanation]

- (1) This instruction cancels the compliance mode started by the Cmp Tool, Cmp Pos, or Cmp Jnt instruction.
- (2) In order to cancel jog operations in the compliance mode, either execute this instruction in a program or execute it directly via the program edit screen of the T/B.

[Available robot type]

```
RV-6SD/6SDL/12SD12SDL series
```

[Related instructions]

[CmpG \(Composition Gain\)](#), [Cmp Tool \(Composition Tool\)](#), [Cmp Pos \(Composition Posture\)](#), [Cmp Jnt \(Composition Joint\)](#)

CmpG (Composition Gain)

[Function]

Specify the softness of robot control.

Note) The available robot types for this instruction are limited. Refer to "[Available robot type]".

[Format]

Cmp Pos, Cmp Tool

CmpG[] [<X axis gain>], [<Y axis gain>], [<Z axis gain>], [<A axis gain>],
[<B axis gain>], [<C axis gain>], ,

Cmp Jnt

CmpG[] [<J1 axis gain>], [<J2 axis gain>], [<J3 axis gain>], [<J4 axis gain>],
[<J5 axis gain>], [<J6 axis gain>], ,

[Terminology]

<X to C axis gain>

<J1 to J6 axis gain>

Specify this argument using a constant.

The softness can be set for each axis.

Value 1.0 indicates the normal status, and the 0.2 is the softest.

If the value is omitted, the current setting value will be applied.

[Reference Program]

1 CmpG , ,0.5, , , , ' This statement selects only the Z-axis. For axes that are omitted, keep the corresponding entries blank and just enter commas.

[Explanation]

- (1) The softness can be designated in each axis units.
- (2) The soft state will not be entered unless validated with the Cmp Pos or Cmp Tool commands.
- (3) A spring-like force will be generated in proportion to the deviation of the command position and actual position. CmpG designates that spring constant.
- (4) The deviation of the command position and actual position can be read with M_CmpDst. The success/failure of pin insertion can be checked using this variable.
- (5) If a small gain is set, and the soft state is entered with the Cmp Pos, Cmp Tool, and Cmp Jnt commands, the robot position could drop. Set the softness state gradually while checking it.
- (6) The softness can be changed halfway when this command executed under the soft control status.
- (7) The gain value of less than 0.2 is invalid. The robot is controlled using the value 0.2.
Also, two or more decimal positions can be set for gain values.
- (8) The compliance mode is valid only for the robot arm axes. It is not valid for additional axes (J7, J8 or L1, L2), even if specified.

[Available robot type]

RV-6SD/6SDL/12SD/12SDL series

Cnt (Continuous)

[Function]

Designates continuous movement control for interpolation. Shortening of the operating time can be performed by carrying out continuous movement.

[Format]

Cnt[] <Continuous movement mode/acceleration/deceleration movement mode>
 [, <Numeric value 1>] [, <Numeric value 2>]

[Terminology]

<1/0> Designate the continuous operation or acceleration/deceleration operation mode.
 1 : Continuous movement.

0 : Acceleration/deceleration movement.(default value.)

<Numeric value 1> Specify the maximum proximity distance in mm for starting the next interpolation when changing to a new path segment.

The default value is the position where the acceleration/deceleration is started.

<Numeric value 2> Specify the maximum proximity distance in mm for ending the previous interpolation when changing to a new path segment.

The default value is the position where the acceleration/deceleration is started.

[Reference Program]

When the maximum neighborhood distance is specified when changing a locus.

1 Cnt 0	' Invalidate Cnt (Continuous movement).
2 Mvs P1	' Operate with acceleration/deceleration
3 Cnt 1	' Validate Cnt (Continuous movement). (Operate with continuous movement after this step.)
4 Mvs P2	' The connection with the next interpolation is continuous movement.
5 Cnt 1,100,200	' Continuous operation specification at 100 mm on the starting side and at 200 mm on the end side.
6 Mvs P3	' Continuous operation at a specified distance before and after an interpolation.
7 Cnt 1,300	' Continuous operation specification at 300 mm on the starting side and at 300 mm on the end side.
8 Mov P4	' Continuous operation specification at 300 mm on the starting side.
9 Cnt 0	' Invalidate Cnt (Continuous movement).
10 Mov P5	' Operate with acceleration/deceleration

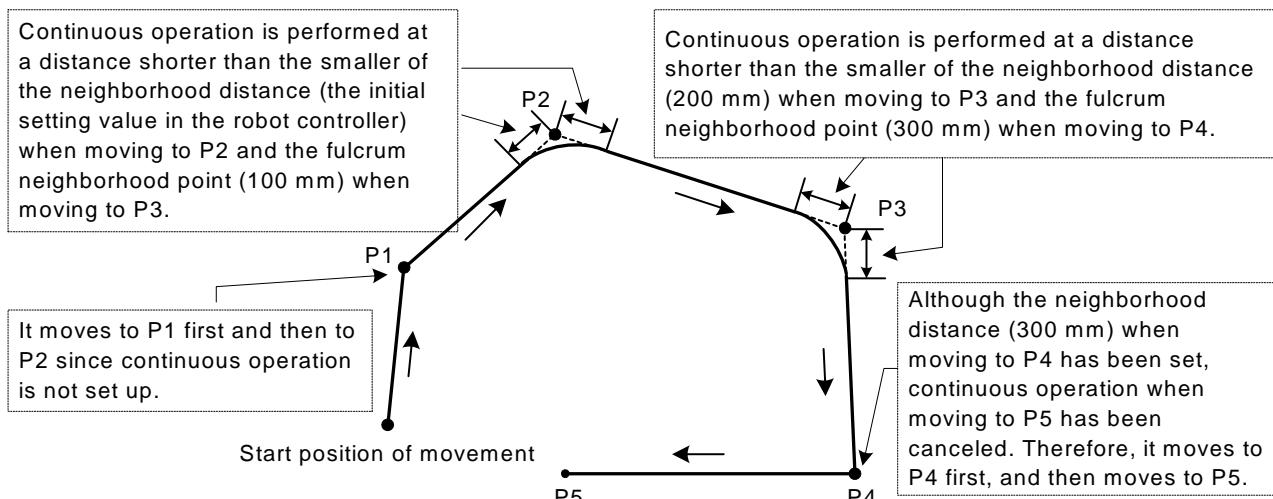


Fig.4-6:Example of continuous path operation

[Explanation]

- (1) The interpolation (40 step to 80 step of the example) surrounded by Cnt 1 - Cnt 0 is set as the target of continuous action.
- (2) The system default value is Cnt 0 (Acceleration/deceleration movement).
- (3) If values 1 and 2 are omitted, the connection with the next path segment is started from the time the deceleration is started.
- (4) As shown in Fig. 4-7, in the acceleration and deceleration operating mode, the speed is reduced in front of the target position. After moving to the target position, the speed for moving to the next target position starts to be accelerated. On the other hand, in the continuous operating mode, the speed is reduced in front of the target position, but it does not stop completely. The speed for moving to the next target position starts to be accelerated at that point. Therefore, it does not pass through each target position, but it passes through the neighborhood position.

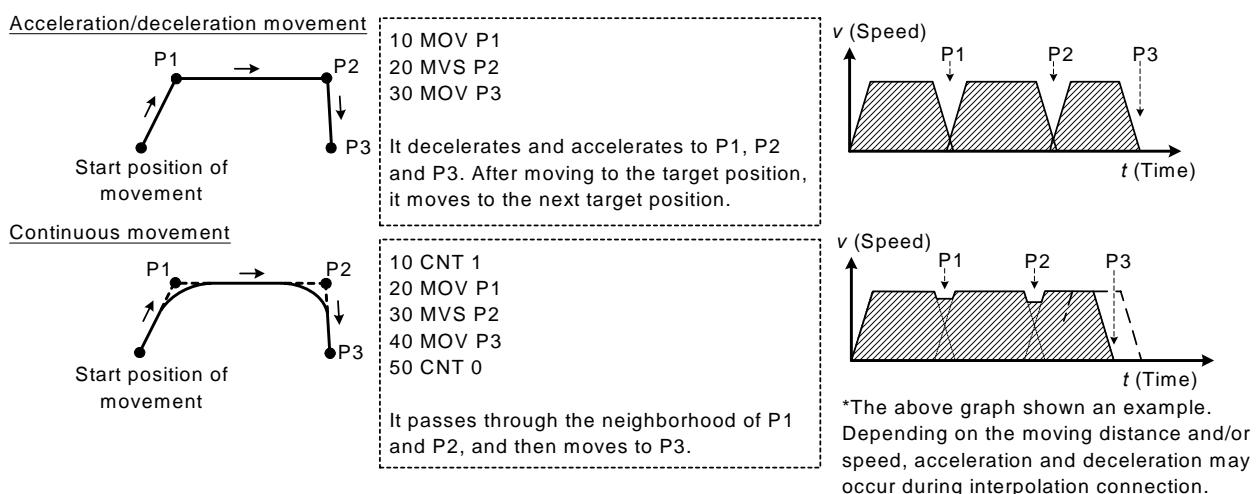


Fig.4-7:Acceleration/deceleration movement and continuous movement

- (5) The neighborhood distance denotes the changing distance to the interpolation operation at the next target position. If this neighborhood distance (numerical value 1, numerical value 2) is omitted, the accelerate and deceleration starting position will be the changing position to the next interpolation. In this case, it passes through a location away from the target position, but the operating time will be the shortest. To pass through a location closer to the target position, set this neighborhood distance (numerical value 1, numerical value 2).

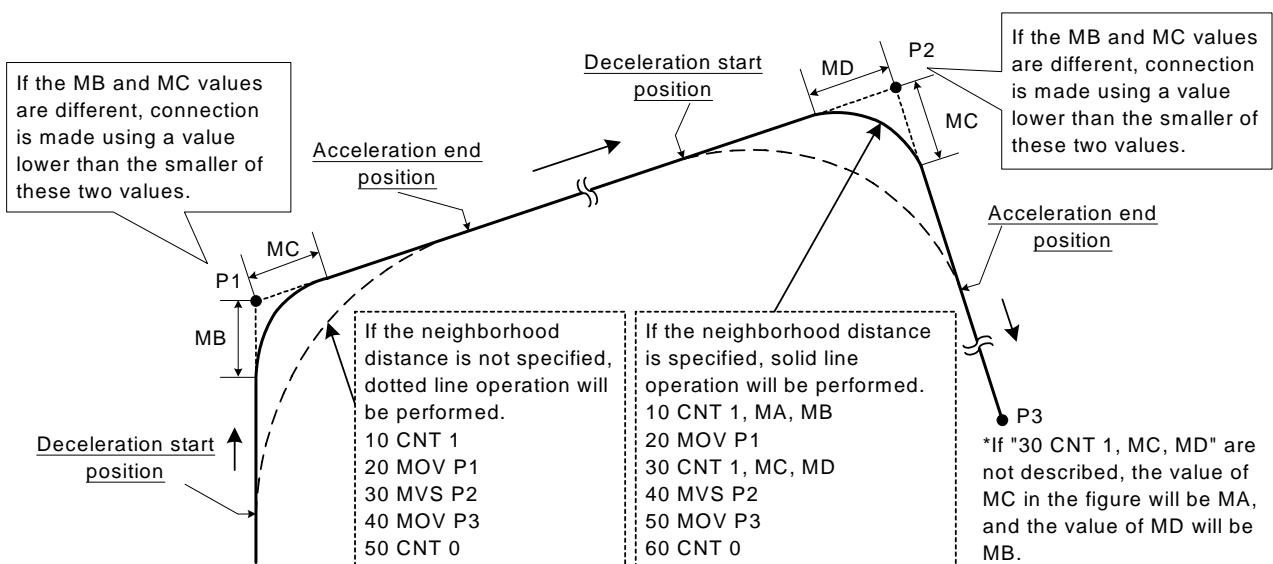


Fig.4-8:Setting Up the Neighborhood Distance

- (6) If the specifications of numerical value 1 and numerical value 2 are different, continuous operation will be performed at the position (distance) that is the smaller of these two.
- (7) If numeric value 2 is omitted, the same value as numeric value 1 will be applied.
- (8) When continuous operation is specified, the positioning completion specification by the Fine instruction will be invalid.
- (9) If the proximity distance (value 1, value 2) is set small, the movement time may become longer than in the status where Cnt 0.
- (10) Even when continuous operation is specified, acceleration/deceleration is performed for the interpolation instruction that specifies singular point passage as the interpolation method.

ColChk (Col Check)

[Function]

Set to enable/disable the impact detection function in automatic operation.

The impact detection function quickly stops the robot when the robot's hand and/or arm interferes with peripheral devices so as to minimize damage to and deformation of the robot's tool part or peripheral devices. However, it cannot completely prevent such damage and deformation.

The impact detection function can only be used in certain models (Refer to "[Available robot type]".).

[Format]

ColChk[]On [, NOErr] / Off

[Terminology]

On	Enable the impact detection function. Once an impact is detected, it immediately stops the robot, issues an error numbered in 1010's, and turns OFF the servo.
Off	Disable the impact detection function
NOErr	Even if an impact is detected, no error is issued. (If omitted, an error will occur.)

[Reference Program 1]

If an error is set in the case of impact	
1 COILvl 80,80,80,80,80,,	'Specify the allowable level for impact detection.
2 ColChk On	'Enable the impact detection function.
3 Mov P1	
4 Mov P2	
5 Dly 0.2	'Wait until the completion of operation (Fine instruction can also be used).
6 ColChk Off	'Disable the impact detection function.
7 Mov P3	

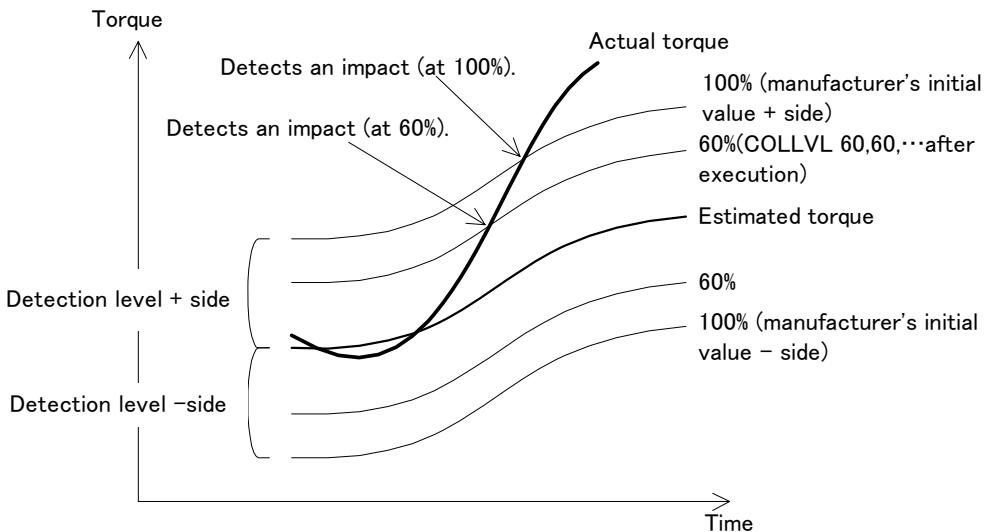
[Reference Program 2]

If interrupt processing is used in the case of impact

1 Def Act 1,M_ColSts(1)=1 GoTo *HOME,S	'Define the processing to be executed when an impact is detected using an interrupt.
2 Act 1=1	
3 ColChk On,NOErr	'Enable the impact detection function in the error non-occurrence mode.
4 Mov P1	
5 Mov P2	'If an impact is detected while executing steps 40 through 70, it jumps to interrupt processing.
6 Mov P3	
7 Mov P4	
8 Act 1=0	
:	
10 *HOME	'Interrupt processing during impact detection
11 ColChk Off	'Disable the impact detection function.
12 Servo On	'Turn the servo on.
13 PESC=P_ColDir(1)*(-2)	'Create the amount of movement for escape operation.
14 PDst=P_Fbc(1)+PESC	'Create the escape position.
15 Mvs PDst	'Move to the escape position.
16 Error 9100	'Stop operation by generating a user-defined L level error.

[Explanation]

- (1) The impact detection function estimates the amount of torque that will be applied to the axes during movement executed by a Move instruction. It determines that there has been an impact if the difference between the estimated torque and the actual torque exceeds the tolerance, and immediately stops the robot.



- (2) Immediately after power ON, the impact detection function is disabled. Enable the Col parameter before using. This instruction specifies whether to enable or disable the impact detection function during program operation (including step feed and step jump). The enable/disable status when no program is executed, such as pause status and during jog operation, depends on the setting of element 3 of the Col parameter.
- (3) The detection level can be adjusted by a ColLvl instruction. The initial value of the detection level is the setting value of the ColLvl parameter.
- (4) After the impact detection function is enabled by this instruction, that state is maintained continuously until it is disabled by the ColChk Off instruction, the program is reset, an End instruction is executed or the power is turned OFF.
- (5) Even if the impact detection function is disabled by this instruction, the impact detection level set by a ColLvl instruction is retained.
- (6) When the continuity function is enabled, the previous impact detection setting state is restored at next power ON even if the power is turned OFF.
- (7) Error 3950 occurs if an interrupt by the M_Colsts status variable (an interrupt with the interrupt condition of M_Colsts(*)=1 and * denotes a machine number) is not enabled when specifying NOErr (error non-occurrence mode). See [Syntax Example 2]. Error 3960 also occurs if this interrupt processing is disabled while in the error non-occurrence mode.
- (8) If an impact is detected while in the error non-occurrence mode, the robot turns OFF the servo and stops. Therefore, no error occurs and operation also continues. However, it is recorded in the error log that an impact was detected. (The recording into the log is done only if no other errors occur simultaneously.)
- (9) If an attempt is made to execute ColChk On and ColChk On,NOErr on a robot that cannot use the impact detection function, low level error 3970 occurs. In the case of ColChk Off, neither error occurs nor processing is performed.
- (10) The impact detection function cannot be enabled while compliance is being enabled by a Cmp instruction or the torque limit is being enabled by a Torq instruction. In this case, error 3940 will occur if an attempt is made to enable the impact detection function. Conversely, error 3930 will occur if an attempt is made to enable a Cmp or Torq instruction while impact detection is being enabled.
- (11) If ColChk Off is described immediately after an operation instruction, impact detection may not work near the last stop position of a given operation. As shown in syntax example 1, execute ColChk Off upon completion of positioning by a Dly or Fine instruction between an operation instruction and a ColChk Off instruction.

- (12) The impact detection function may not work properly if the hand weight (HNDDATn parameter) and workpiece weight (WRKDATn parameter) are not set correctly. Be sure to set these parameters correctly before using.
- (13) If the impact detection function is enabled by this instruction, the execution time (tact time) may become long for some programs. Use the impact detection function only for operations that may interfere with peripheral devices, rather than enabling it for the entire program.
- (14) This function cannot be used together with the multi-mechanism control function.

[Related instructions and variables]

[ColLvl](#) ([Col Level](#)), [M_ColSts](#), [J_ColMxl](#), [P_ColDir](#)

[Related parameter]

[COL](#), [COLLVL](#), [COLLVLJG](#), [HNDDATn](#), [WRKDATn](#)

[Available robot type]

[RV-6SD/6SDL/12SD/12SDL series](#)

ColLvl (Col Level)

[Function]

Set the detection level of the impact detection function in automatic operation.

The impact detection function can only be used in certain models (Refer to "[Available robot type]").

[Format]

```
ColLvl[] [<J1 axis>],[<J2 axis>],[<J3 axis>],[<J4 axis>],[<J5 axis>],[<J6 axis>],
```

[Terminology]

<J1 to J6 axis> Specify the detection level in a range between 1 and 500%.
 If omitted, the previously set value is retained.
 This instruction is invalid for the J7 and J8 axes.
 The initial value is the setting value of the ColLvl parameter.

[Reference Program]

1 ColLvl 80,80,80,80,80,,	'Specify the allowable level for impact detection.
2 ColChk On	'Enable the impact detection function.
3 Mov P1	
4 ColLvl ,50,50,,,,	'Change the allowable level of the J2 and J3 axes for impact detection.
5 Mov P2	
6 Dly 0.2	'After arriving at P2, disable impact detection.
7 ColChk Off	'Disable the impact detection function.
8 Mov P3	

[Explanation]

- (1) Set the allowable level of each axis for the impact detection function during program operation.
- (2) This instruction affects the impact detection function in automatic operations (including step feed and step jump operations). If a program is not running (pause status or during jog operation), the setting level of the ColLvlJG parameter is used.
- (3) Normally, the setting value of the allowable level immediately after power ON is the setting value of the ColLvl parameter.
- (4) "All axes 100%" is set as the initial value of the ColLvl parameter.
- (5) If this value is increased, the detection level (sensitivity) lowers; if this value is lowered, the detection level increases.
- (6) If the detection level is increased, the probability of erroneous detection becomes high. Adjust the level such that it does not become too high. Depending on the posture and operation speed, erroneous detection may also occur with the initial value. In this case, the detection level should be lowered.
- (7) The impact detection function may not work properly if the hand weight (HNDDATn parameter) and workpiece weight (WRKDATn parameter) are not set correctly. Be sure to set these parameters correctly before using.
- (8) When the continuity function is enabled, the previously set value is restored at next power ON even if the power is turned OFF.
- (9) The allowable level is reset to the setting value of the ColLvl parameter when a program reset or an End instruction is executed.
- (10) Even if an attempt is made to execute this instruction on robots that cannot use the impact detection function, the instruction is ignored and thus no error occurs.
- (11) The impact detection function is not valid for the J7 and J8 axes.
- (12) The correct setting value may vary even among robots of the same type due to individual differences of units. Check the operation with each robot.

[Related instructions and variables]

[ColChk \(Col Check\)](#), [M_ColSts](#), [J_ColMxl](#), [P_ColDir](#)

[Related parameter]

[COL](#), [COLLVL](#), [HNDDATn](#), [WRKDATn](#)

[Available robot type]

RV-6SD/6SDL/12SD/12SDL series

Com On/Com Off/Com Stop (Communication ON/OFF/STOP)

[Function]

- | | |
|----------|---|
| Com On | :Allows interrupts from a communication line. |
| Com Off | :Prohibits interrupts from a communication line. |
| Com Stop | :Prevents interrupts from a communication line temporarily (data is received).
Jump immediately to the interrupt routine the next time the Com On instruction is executed. |

[Format]

Com[(<Communication Line No.>)][On]

Com[(<Communication Line No.>)][Off]

Com[(<Communication Line No.>)][Stop]

[Terminology]

- | | |
|--------------------------|--|
| <Communication Line No.> | Describes numbers 1 to 3 assigned to the communication line.
(If the argument is omitted, 1 is set as the default value.) |
|--------------------------|--|

[Reference Program]

Refer to [Page 233, "On Com GoSub \(ON Communication Go Subroutine\)"](#).

[Explanation]

- (1) When Com On Off is executed, even if communications are attempted, the interrupt will not be generated.
- (2) For information on communication line Nos., refer to the [Page 236, "Open \(Open\)"](#).
- (3) After Com Stop is executed, even if communication is attempted, the interrupt will not be generated.
Note that the receiving data and the fact of the interrupt will be recorded, and be executed the next time the line is reopened.

Def Act (Define act)

[Function]

This instruction defines the interrupt conditions for monitoring signals concurrently and performing interrupt processing during program execution, as well as the processing that will take place when an interrupt occurs.

[Format]

Def[]Act[]<Priority No.>, <Expression>[]<Process> [, <Type>]

[Terminology]

<Priority No. >	This is the priority No. of the interrupt. It can be set with constant Nos. 1 to 8.
<Expression>	For the interrupt status, use the formats described below: (Refer to the syntax diagram) <Numeric type data> <Comparison operator> <Numeric type data> or <Numeric type data> <Logical operator> <Numeric type data> <Numeric type data> refers to the following: <Numeric type constant> <Numeric variable> <Numeric array variable> <Component data>
<Process>	Refer to a GoTo statement or a GoSub statement used to process an interrupt when it occurs.
<Type>	When omitted: Stop type 1 The robot stops at the stop position, assuming 100% execution of the external override. If the external override is small, the time required for the robot to stop becomes longer, but it will always stop at the same position. S : Stop type 2 The robot decelerates and stops in the shortest time and distance possible, independently of the external override. L : Execution complete stop The interrupt processing is performed after the robot has moved to the target position (the step being executed is completed).

[Reference Program]

```

1 Def Act 1,M_In(17)=1 GoSub *SUB1          ' Defines the subroutine at step 100 to be the one to be
                                              called up when the status for the general purpose input
                                              signal No. 17 is ON.
2 Def Act 2,MFG1 AND MFG2 GoTo *SUB2        ' Defines the step 200 as the one to jump to when the
                                              logic operation of AND applied to MFG1 or MFG2
                                              results in "true."
3 Def Act 3,M_Timer(1)>10.5 GoSub *LBL       ' When 10.5 seconds pass, the program jumps to the
                                              step 300 subroutine.

:
9 *SUB1
10 M_Timer(1)=0                               ' Sets the timer to zero.
11 Act 3=1                                     ' Enables Act 3.
12 Return 0

:
19 *SUB2
20 Mov P_Safe
21 End
30 *LBL
31 M_Timer(1)=0.0                            ' Resets the timer to zero.
32 Act 3=0                                     ' Disables Act 3.
33 Return 0

```

[Explanation]

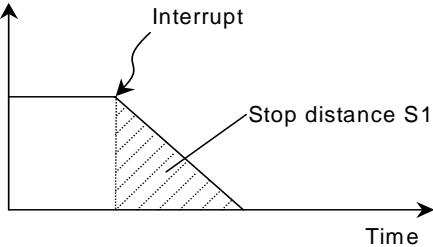
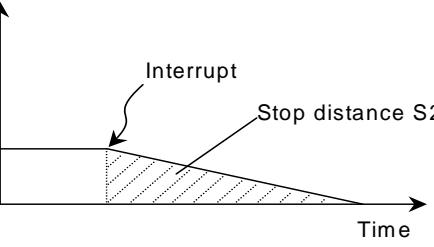
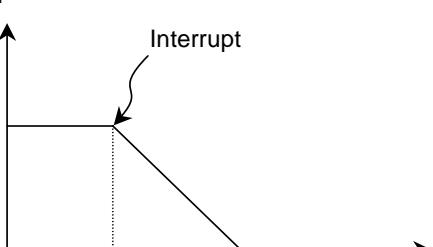
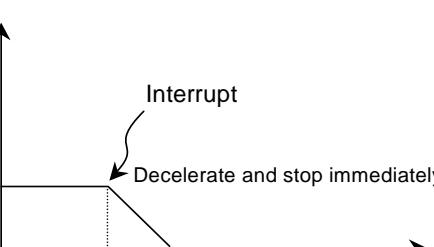
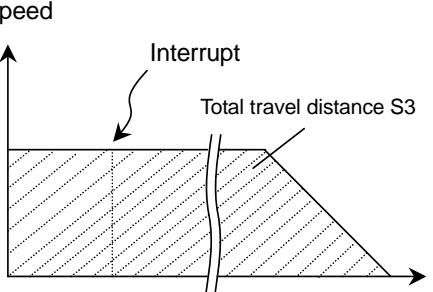
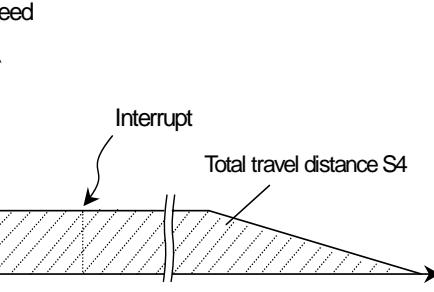
- (1) The priority level for the interrupts is decided by the <Priority No.>, and the priority level, from the highest ranges from 1 to 8.
- (2) There can be up to 8 settings for the interrupts. Use the <Priority No.> to differentiate them.
- (3) An <expression> should be either a simple logical operation or a comparison operation (one operator). Parentheses cannot be used either.
- (4) If two Def Act instructions with the same priority number are included in a program, the latter one defined becomes valid.
- (5) Since Def Act defines only the interrupt, always use the Act command to designate the enable/disable status of the interrupt.
- (6) The communications interrupt (COM) has a higher priority level than any of the interrupts defined by Def Act.
- (7) Def Act definitions are valid only in the programs where they are defined. These are invalid when called up in a program by CallP. If necessary, the data in a sub program may need to be redefined.
- (8) If an interrupt is generated when a GoTo command is designated by <Process> for a Def Act command, during execution of the remaining program, the interrupt in progress will remain, and only interrupts of a higher level will be accepted. The interrupt in progress for a GoTo statement can be canceled with the execution of an End statement.
- (9) Expressions containing conditional expressions combined with logical operations, such as (M1 AND &H001) = 1, are not allowed.

⚠ CAUTION

Specify the proper interrupt stop type according to the purpose. Specify "S" for the stop type if it is desired to stop the robot in the shortest time and distance possible by an interrupt while the robot is executing a movement instruction.

Table 4-15 shows conceptual diagrams that illustrate the effects of the 3 types of program execution stop commands when the interrupt conditions are met while the robot is moving according to a movement instruction.

Table 4-15:Conceptual diagram showing the effects of different stop commands

	External override 100% (maximum speed)	External override 50%
Stop type 1 (If the argument is omitted) S1=S2		
Stop type 2(S)		
Execution complete stop(L) S3=S4		

[Related instructions]

[Act \(Act\)](#)

Def Arch (Define arch)

[Function]

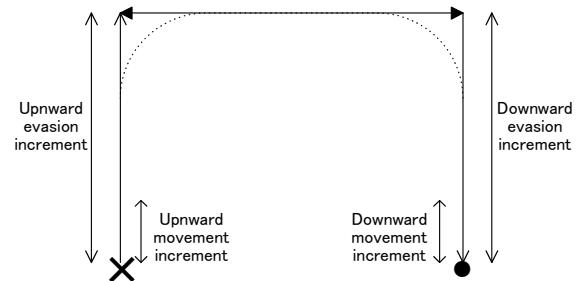
This instruction defines an arch shape for the arch motion movement corresponding to the Mva instruction.

[Format].

```
Def[]Arch[]<Arch number>, [<upward movement increment>][<downward movement increment >],  
[<Upward evasion increment>], [<downward evasion increment>],  
[<interpolation type>], [<interpolation type 1>, <interpolation type 2> ]
```

[Terminology]

<Arch number>	Arch motion movement pattern number. Specify a number from 1 to 4 using a constant or a variable.
<Upward movement increment>	
<Downward movement increment >	Refer to figure at right. It is possible to specify either a constant or a variable.
<Upward evasion increment>	
<Downward evasion increment>	
<Interpolation type>	Interpolation type for upward and downward movements. Linear/joint = 1/0
<Interpolation type 1>	Detour/short cut = 1/0,
<Interpolation type 2>	3-axis XYZ/Equivalent rotation = 1/0



If any of the arguments besides the arch number is omitted, the default value is employed.

The default values are set by the following parameters. Check the corresponding parameters to see the values; it is also possible to modify the values.

Parameter name	Arch number	Upward movement increment (mm)	Downward movement increment (mm)	Upward evasion increment (mm)	Downward evasion increment (mm)
ARCH1S	1	0.0	0.0	30.0	30.0
ARCH2S	2	10.0	10.0	30.0	30.0
ARCH3S	3	20.0	20.0	30.0	30.0
ARCH4S	4	30.0	30.0	30.0	30.0

Vertical multi-joint robot(RV-6SDL, etc.)

Parameter name	Arch number	Interpolation type	Interpolation type 1	Interpolation type 2
ARCH1T	1	1	0	0
ARCH2T	2	1	0	0
ARCH3T	3	1	0	0
ARCH4T	4	1	0	0

Horizontal multi-joint robot(RH-12SDH **, etc.)

Parameter name	Arch number	Interpolation type	Interpolation type 1	Interpolation type 2
ARCH1T	1	0	0	0
ARCH2T	2	0	0	0
ARCH3T	3	0	0	0
ARCH4T	4	0	0	0

[Reference Program]

1 Def Arch 1,5,5,20,20

2 Mva P1,1

'Performs the arch motion movement defined in step 10.

3 Mva P2,2

'The robot moves according to the default values specified by the parameters.

[Explanation]

(1) If the Mva instruction is executed without the Def Arch instruction, the robot moves according to the arch shape specified by the parameters.

(2) Used to change the increments in a program, etc.

[Related instructions]

[Mva \(Move Arch\)](#), [Accel \(Accelerate\)](#), [Ovrd \(Override\)](#), [Mvs \(Move S\)](#)(Used as a reference for interpolation types 1 and 2)

Def Char (Define Character)

[Function]

Declares a character string variable. It is used when using a variable with a name that begins with a character other than "C." It is not necessary to declare variables whose names begin with the character "C" using the Def Char instruction.

[Format]

```
Def[]Char[]<Character string variable name>
[, <Character string variable name>...]
```

[Terminology]

<Character string variable name> Designate a variable name.

[Reference Program]

1 Def Char MESSAGE	' Declare "MESSAGE" as a character string variable.
2 MESSAGE = "WORKSET"	' Substitute "WORKSET" in the MESSAGE variable.
3 CMSG = "ABC"	' Substitute "ABC" for variable CMSG. For variables starting with C, the definition of "Def Char" is not required.

[Explanation]

- (1) The variable name can have up to eight characters. Refer to the [Page 129, "4.3.6 Types of characters that can be used in program"](#) for the characters that can be used.
- (2) When designating multiple variable names, the maximum value (127 characters including command) can be set on one step.
- (3) A variable becomes a global variable that is shared among programs by placing "_" after C in the variable name and writing it in a base program.
Refer to [Page 139, "4.3.24 User-defined external variables"](#) for details.

Def FN (Define function)

[Function]

Defines a function and gives it name.

[Format]

```
Def[]FN <Identification character><Name> [<Dummy Argument> [, <Dummy Argument>]...]
      = <Function Definition Expression>
```

[Terminology]

<Identification character>	The identification character has the following four type. Numeric value type:M Character string type:C Position type:P Joint type:J
<Name>	Describe a user-selected character string. (5 is the maximum)
<Dummy argument>	When a function has been called up, it is transferred to the function. It is possible to describe all the variables, and up to 16 variables can be used.
<Function Definition Expression>	Describe the expression for what operation to use as a function.

[Reference Program]

```
1 Def FNMAve(MA,MB)=(MA+MB)/2      ' Define FNMAVE to obtain the average of two numeric values.
2 MDATA1=20
3 MDATA2=30
4 MAVE=FNMAve(MDATA1,MDATA2)      ' Substitute average value 25 of 20 and 30 in numeric variable MAVE.
5 Def FNPADD(PA,PB)=PA+PB          ' Position type addition.
6 P10=FNPADD(P1,P2)
```

[Explanation]

- (1) FN + <Name> becomes the name of the function. The function name can be up to 8 characters long.
Example) Numeric value type FNMMAX Identification character: M
Character string type ... FNCAME\$ Identification character: C (Describe \$ at the end of the name)
- (2) A function defined with Def FN is called a user-defined function. A function as long as one step can be described.
- (3) Built-in functions and user-defined functions that have already been defined can be used in the function definition expression. In this case, up to 16 levels of user-defined functions can be written.
- (4) If the variables used in <Function Definition Expression> are not located in <Dummy Argument>, then the value that the variable has at that time will be used. Also, an error will occur if during execution, the number or argument type (numeric value or character string) of arguments differs from the number or type declared.
- (5) A user-defined function is valid only in the program where it is defined. It cannot be used by a CallP designation program.

Def Inte/Def Long/Def Float/Def Double (Define Integer/Long/Float/Double)

[Function]

Use this instruction to declare numerical values. INTE stands for integer, FLOAT stands for single-precision real number, and DOUBLE stands for double-precision real number.

[Format]

```
Def[]Inte[] <Numeric value variable name> [, <Numeric value variable name>]...
Def[] Long[] <Numeric value variable name> [, <Numeric value variable name>]...
Def[] Float[] <Numeric value variable name> [, <Numeric value variable name>]...
Def[]Double[] <Numeric value variable name> [, <Numeric value variable name>]...
```

[Terminology]

<Numeric value variable name> Designate the variable name.

[Reference Program]

(1) Definition of the integer type variable.

```
1 Def Inte WORK1, WORK2      ' Declare WORK 1 and WORK 2 as an numeric value variable name.
2 WORK1 = 100                ' Substitute the value 100 in WORK 1.
3 WORK2 = 10.562             ' Numerical "11" is set to WORK2.
4 WORK2 = 10.12              ' Numerical "10" is set to WORK2.
```

(2) Definition of long precision integer type variable

```
1 Def Long WORK3
2 WORK3 = 12345
```

(3) Definition of the single precision type real number variable.

```
1 Def Float WORK3
2 WORK3 = 123.468          ' Numerical "123.468000" is set to WORK3.
```

(4) Definition of the double precision type real number variable.

```
1 Def Double WORK4
2 WORK4 = 100/3            ' Numerical "33.333332061767599" is set to WORK4.
```

[Explanation]

- (1) The variable name can have up to eight characters. Refer to the [Page 129, "4.3.6 Types of characters that can be used in program"](#) for the characters that can be used.
- (2) When designating multiple variable names, the maximum value (123 characters including command) can be set on one step.
- (3) The variable declared with Inte will be an integer type.(-32768 to +32767)
- (4) The variable declared with Long will be a long precision integer type.(-2147483648 to 2147483647)
- (5) The variable declared with Float will be a single-precision type.(+/-1.70141E+38)
- (6) The variable declared with Double will be a double-precision type.(+/-1.701411834604692E+308)

Def IO (Define IO)

[Function]

Declares an input/output variable. Use this instruction to specify bit widths. M_In and M_Out variables are used for normal single-bit signals, M_Inb and M_Outb are used in the case of 8-bit bytes, and M_Inw and M_Outw are used in the case of 16-bit words.

Be aware that it is not allowed to reference output signals with variables declared using this instruction.

[Format]

```
Def[]IO[]<Input/output variable name> = <Type designation>, <Input/output bit No.>
                                                [, <Mask information>]
```

[Terminology]

<Input/output variable name>	Designate the variable name.
<Type designation>	Designate BIT(1bit), BYTE(8bit), WORD(16bit) or INTEGER.
<Input/output bit No.>	Designate the input(When referencing) or output(When assigning) bit No.
<Mask information>	Designate when only a specific signal is to be validated.

[Reference Program]

(1) Assign the input variable named PORT1 to input/output signal number 6 in bit type.

```
1 Def IO PORT1 = BIT,6
:
10 PORT1 = 1      ' Output signal number 6 turns on.
:
20 PORT1 = 2      ' Output signal number 6 turns off.(Because the lowest bit of the numerical value 2 is 0.)
21 M1 = PORT1    ' Substitute the state of the input signal number 6 for M11.
```

(2) Assign the input variable named PORT2 to input/output signal number 5 in byte type, and specify the mask information as 0F in hexadecimal.

```
1 Def IO PORT2 = BYTE, 5, &H0F
:
10 PORT2 = &HFF    ' Output signal number 5 to 8 turns on.
:
20 M2 = PORT2    ' Substitute the value of the input signals 5 to 8 for the variable M2.
```

(3) Assign the input variable named PORT3 to input/output signal number 8 in word type, and specify the mask information as 0FFF in hexadecimal.

```
1 Def IO PORT3 = WORD, 8, &H0FFF
:
10 PORT3 = 9      ' Output signal number 8 and 11 turns on.
:
20 M3 = PORT3    ' Substitute the value of the input signals 8 to 19 for the variable M3.
```

[Explanation]

- (1) An input signal is read when referencing this variable.
- (2) An output signal is written when assigning a value to this variable.
- (3) It is not allowed to reference an output signal by this variable. Use the M_Out variable in order to reference an output signal.
- (4) The variable name can have up to eight characters. Refer to the [Page 129, "4.3.6 Types of characters that can be used in program"](#) for the characters that can be used.
- (5) When mask information is designated, only the specified signal will be validated.

Example) In the above example on the 20th step, the input/output data with a bit width of eight is masked by 0F in hexadecimal. Thus, if PORT 2 is used thereafter,

- When used as an input signal (M1 = PORT 2):

Numbers 5 to 8 are used for input, and numbers 9 to 12 are always treated as 0.

No. 12 No.5 (Input/output bit No.)

0000 1111

 Invalid Valid

- When used as an output signal (PORT 2 = M1):

Data to be output this time is output to numbers 5 to 8, and the status currently being output is retained at numbers 9 to 12.

No. 12 No.5 (Input/output bit No.)

**** 1111
| |

Retains the current output status Output data of this time

Def Jnt (Define Joint)

[Function]

This instruction declares joint type position variables. It is used when using a variable with a name that begins with a character other than "J." It is not necessary to declare variables whose names begin with the character "J" using the Def Jnt instruction.

[Format]

Def[]Jnt[] <Joint variable name> [, <Joint variable name>]...

[Terminology]

<Joint variable name> Designate a variable name.

[Reference Program]

1 Def Jnt SAFE	' Declare "SAFE" as a joint variable.
2 Mov J1	' For joint type position variables starting with J, the definition of "Def Jnt" is not required.
3 SAFE = (-50,120,30,300,0,0,0,0)	
4 Mov SAFE	' Move to SAFE.

[Explanation]

- (1) Use this instruction to define a joint position variable by a name beginning with a character other than J.
- (2) The variable name can have up to eight characters. Refer to the [Page 129, "4.3.6 Types of characters that can be used in program"](#) for the characters that can be used. When designating multiple variable names, the maximum value (127 characters including command) can be set on one step.
- (3) A variable becomes a global variable that is shared among programs by placing "_" after J in the variable name and writing it in a base program.
Refer to [Page 139, "4.3.24 User-defined external variables"](#) for details.

Def Plt (Define pallet)

[Function]

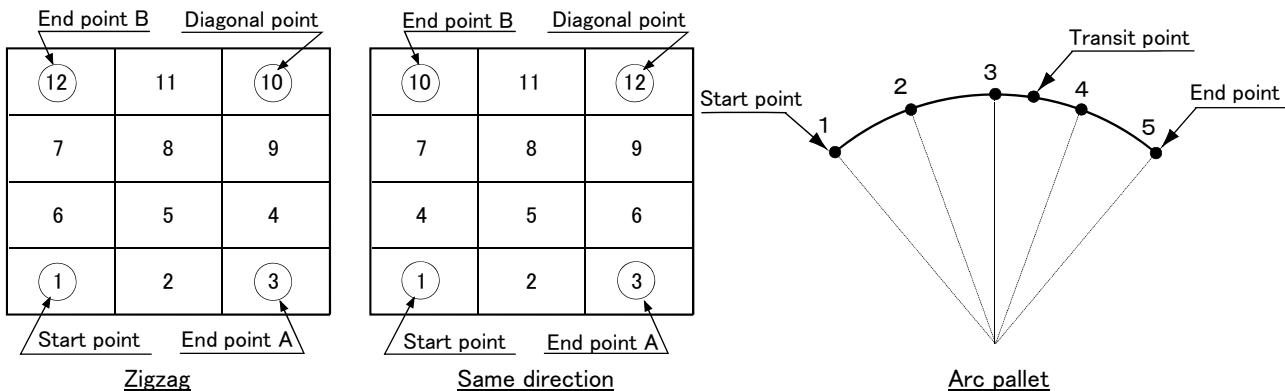
Defines the pallet. (3-point pallet, 4-point pallet)

[Format]

```
Def[]Plt[] <Pallet No.>, <Start Point>, <End Point A>, <End Point B>, [<Diagonal Point>],  
          <Quantity A>, <Quantity B>, <Pallet Pattern>
```

[Terminology]

<Pallet No. >	This is the selection No. of the set pallet. (Constants from 1 to 8 only).
<Start Point>	Refers to the pallet's start point.
<End Point A>	One of the ending points for the pallet. Transit point of arc for arc pallet.
<End Point B>	Another ending point for the pallet. Ending point of arc for arc pallet.
<Diagonal Point>	The diagonal point from the pallet's start point. Insignificant for arc pallet.
<Quantity A>	The No. of workpieces from the pallet's start point to the end point A.
	The No. of workpieces between the pallet start point and arc end point when using an arc pallet.
<Quantity B>	The No. of workpieces from the pallet's start point to the end point B. Insignificant for an arc pallet. (1, etc., must be designated.)
<Pallet Pattern>	Specify the pallet pattern and fixation/equal division of the posture when numbering divided grid points. Constant only. 1: Zigzag (posture equal division) 2: Same direction (posture equal division) 3: Arc pallet (posture equal division) 11: Zigzag (posture fixation) 12: Same direction (posture fixation) 13: Arc pallet (posture fixation)



[Reference Program]

```
1 Def Plt 1,P1,P2,P3, ,4,3,1      ' Define a 3-point pallet.  
2 Def Plt 1,P1,P2,P3,P4,4,3,1      ' Define a 4-point pallet.
```

[Explanation]

- (1) The accuracy of the position calculation will be higher for a 4-point pallet than for a 3-point pallet.
- (2) The command is valid only within the program being executed. The command is invalid in the program that calls up the command from another program. If necessary, redefine.
- (3) Quantity A and B should be a non-zero positive number, while if 0 or a negative number is assigned, an error will occur.
- (4) If Quantity A x Quantity B exceeds 32,767, an error will occur when operation starts.
- (5) The value of Quantity B is insignificant for the arc pallet, but it must not be omitted. Set 0 or a dummy value. The diagonal point will be insignificant even if specified.

- (6) If the hand is facing downward, the sign of the A, B and C axis coordinates at the start point, end point A, end point B and diagonal point must match. If the hand is facing downward, A = 180 (or -180), B = 0, and C = 180 (or -180). If the signs of the A and C axis coordinates at the three positions do not match, the hand may rotate in the middle position. In this case, modify the signs so that they match in the position edit screen of the T/B. +180 and -180 result in the same posture; modifying signs poses no problem.
- (7) If a value from 11 to 13 is specified for the pallet pattern, the posture at <Start Point> is assigned to the posture data of the position variable obtained by the pallet operation. If a value from 1 to 3 is specified, the distance between <Start Point> and <End Point> is divided equally and assigned to the posture data.

⚠ CAUTION

If position data whose posture components (A, B and C) are close to +/-180 degrees is set to <Start Point>, <End Point A>, <End Point B> and <Diagonal Point> of the pallet definition, the hand will rotate and move in unexpected ways if different signs are used for the same posture component of the position data. To use position data whose posture components are close to +/-180 degrees, please read [<Precautions on the posture of position data in a pallet definition>](#) in [Page 98, "4.1.2 Pallet operation"](#).

⚠ CAUTION

The value of the start point of the pallet definition is employed for the structure flag of grid points (FL1 of position data) calculated during pallet operation (Plt instruction). For this reason, if position data with different structure flags are used for each point of the pallet definition, the desired pallet operation cannot be obtained. Use position data whose structure flag values are all the same for the start point, end points A and B and the diagonal point of the pallet definition. The value of the start position of the pallet definition is employed for the multi-rotation flag of grid points (FL2 of position data) as well. If position data with different multi-rotation flags are used for each point of the pallet definition, the hand will rotate and move in unexpected ways depending on the robot positions the pallet operation goes through and the type of interpolation instruction (joint interpolation, line interpolation, etc.). In such cases, use the TYPE argument of the interpolation instruction to set the detour/short cut operation of the posture properly and ensure that the hand moves as desired.

Please refer to the illustrations in [Page 98, "4.1.2 Pallet operation"](#), which explain this concept.

[Related instructions]

[Plt \(Pallet\)](#)

Def Pos (Define Position)

[Function]

This instruction declares XYZ type position variables. It is used when using a variable with a name that begins with a character other than "P." It is not necessary to declare variables whose names begin with the character "P" using the Def Pos instruction.

[Format]

```
Def[]Pos[] <Position variable name> [, <Position variable name>]...
```

[Terminology]

<Position variable name> Designate a variable name.

[Reference Program]

1 Def Pos WORKSET	' Declare "WORKSET" as the XYZ type position variable.
2 Mov P1	' For XYZ type position variables starting with P, the definition of "Def Pos" is not required.
3 WORKSET=(250,460,100,0,0,-90,0,0)(0,0)	
4 Mov WORKSET	' Move to WORKSET.

[Explanation]

- (1) Use this instruction to define a XYZ type position variable by a name beginning with a character other than P.
- (2) The variable name can have up to eight characters. Refer to the [Page 129, "4.3.6 Types of characters that can be used in program"](#) for the characters that can be used.
- (3) When designating multiple variable names, the maximum value (127 characters including command) can be set on one step.
- (4) A variable becomes a global variable that is shared among programs by placing "_" after P in the variable name and writing it in a base program.
Refer to [Page 139, "4.3.24 User-defined external variables"](#) for details.

Dim (Dim)

[Function]

Declares the quantity of elements in the array variable. (Arrays up to the third dimension are possible.)

[Format]

```
Dim[]<Variable name> (<Element Value> [, <Element Value> [, <Element Value>]])  
[, <Variable name> (<Element Value> [, <Element Value> [, <Element Value>]])]...
```

[Terminology]

<Variable name>	Describe the name of the array variable.
<Element Value>	Describe in terms of constants, the number of elements in an array variable.

[Reference Program]

1 Dim PDATA(10)	' Define the position array variable PDATA having ten elements.
2 Dim MDATA#(5)	' Define double-precision type array variable MDATA# having the five elements.
3 Dim M1%(6)	' Define integer-type array variable M1% having the six elements.
4 Dim M2!(4)	' Define single-precision real number type array variable M2! having the four elements.
5 Dim CMOJI(7)	' Define the character-string type variable CMOJI having the seven elements.
6 Dim MD6(2,3), PD1(5,5)	' Define the 2-dimensional single precision real number type array variable MDATA having the element of 2x3. ' Define the 2-dimensional position array variable PD 1 having the element of 5x5.

[Explanation]

- (1) A one-dimensional, two-dimensional or three-dimensional array can be used.
- (2) In the case of numeric variables, it is possible to use integer, single-precision real and double-precision real variables differently by adding a symbol that indicates the type of each variable to the variable name. If the variable type is omitted, a single-precision real variable will be assumed.
Dim MABC(10) ' Define the single-precision real number type array variable MABC having ten elements.
- (3) Element number start from 1 when actually referencing array variables. For PDATA on step 10 of the statement example, the element number will be 1 to 10.
- (4) <Element Value> can be described with numeric constants from 1 to 999. It is not allowed to use a numerical value operation expression.
If the number of elements is specified using a real number, an integer with rounded decimal part will be assumed. Depending on the system memory's free space, arrays may not be allocated for the number of specified elements. In this case, an error will occur when lines are registered.
- (5) If an element number larger than the number of defined elements is specified, an error will occur at the time of execution.
- (6) At the point when array variables are defined, variable values are indeterminate.
- (7) To use array variables, it is necessary to define them using the Dim instruction.
- (8) The arrays defined by the Dim instruction are valid only in the program where they are defined. To use these arrays by a sub program called by the CallP instruction, it is necessary to define them again.
- (9) Array variables can be used similar to normal variables. However, note that variables of which variable names and/or the number of characters for specifying element numbers exceed eight characters cannot be used on the monitor variable screen and position edit screen of the teaching pendant.
- (10) If a variable name whose second character is underlined "_" is registered in a user program, a user defined external variable (a variable common among programs) will be assumed..
Refer to Page 139, "4.3.24 User-defined external variables" for details.

Dly (Delay)

[Function]

- 1) When used as a single command:

At a designated time, it causes a wait. It is used for positioning the robot and timing input/output signals.

- 2) When used as an additional pulse output:

Designates an output time for a pulse.

[Format]

- 1) When used as a single command

Dly[]<Time>

- 2) When used as an additional pulse output

Example) $M_{Out}(1) = 1$ Dly[]<Time>

[Terminology]

<Time> Describes the waiting time or the output time for the pulse output, in terms of a numeric operation expression. Unit: [Seconds]

The minimum value that can be set is 0.01 seconds. It is allowed to specify 0.00 as well.

The maximum value is the maximum single-precision real number.

[Reference Program]

- (1) Waiting for time

1 Dly 30 ' Wait for 30 seconds

- (2) Pulse output of the signal

2 $M_{Out}(17)=1$ Dly 0.5 ' Send the signal output to the general-purpose output signal 17 for 0.5 seconds.

3 $M_{Outb}(18)=1$ Dly 0.5

' Among general-purpose output signals 18 to 25, only signal 18 is output (on) for the first 0.5 seconds, and signals 19 to 25 are output (on) after 0.5 seconds have passed.

- (3) Wait for the completion of positioning.

1 Mov P1 ' Moves to P1.

2 Dly 0.1 ' Positions to 1.

- (4) Wait for completion of hand opening. (closing)

1 HOpen 1 ' Open the hand 1.

2 Dly 0.5 ' Wait for hand 1 to open securely.

[Explanation]

- (1) This instruction sets the wait time in a program. It is used for timing input/output signals, positioning movement instructions, and for specifying pulse output times when used in a signal output statement (such as step 20 in [statement example] above).

- (2) The pulse output will be executed simultaneously as the next command in the steps that follow.

- (3) Up to 50 pulse outputs can be issued of all programs simultaneously. Exceeding this, an error will occur when the program tries to execute it.

- (4) A pulse output reverses each of its bits after the specified time. This means that if M_{Outb} (8-bit signal) or M_{Outw} (16-bit signal) is used, the corresponding number of bits are reversed.

- (5) As for pulse output, the execution of a program ends without waiting the elapse of the specified duration if the End instruction or the last step of the program is executed during the specified duration. However, output turns off after the specified duration.

- (6) The relation of the priority levels for other interrupts is as shown below:

COM>Act>WthIf (Wth) >Pulse output (Time setting ON)

- (7) Even if stop is input during the execution of a pulse output, the pulse output operation will not stop.

Note1) If stop is input at step 20 in the following program, the output signal state will be held, and the execution is stopped.

1 $M_{Out}(17)=1$

2 Dly 10

3 $M_{Out}(17)=0$

Note2) If a pulse output by the M_{Outb} (8-bit signal) or the M_{Outw} (16-bit signal) is used, each bits in the corresponding bit width are reversed after the designated time.

$M_{Outb}(1)=1$ Dly 1.0

In this case the bit pattern 00000001 is output for one second, and the bit pattern 11111110 is output thereafter.

End (End)

[Function]

This instruction defines the final step of a program.

It is also used to indicate the end of a program explicitly, by entering the End instruction at the end of the main processing, in case a sub program is attached after the main program. In the case of a sub program called up by the CallP instruction, the control is returned to the main program when the End instruction is executed.

[Format]

End

[Reference Program]

```

1 Mov P1
2 GoSub *ABC
3 End           ' End the program.
:
10 *ABC
11 M1=1
12 Return

```

[Explanation]

- (1) This instruction defines the final step of a program. Use the Hlt instruction to stop a program in the middle and put it in the pause status.
- (2) If executed from the operation panel, a program is executed in the continuos operation mode; it will be executed again from the top even if it contains an End instruction. If it is desired to end a program at the End instruction, press the End key on the operation panel to stop the cycle.
- (3) It is allowed to have several End statements within one program.
- (4) The End statement does not need to be described at the end of the program.
- (5) If the End command is executed by the sub program called by CallP, control will return to the main program. The operation will be similar to the RETURN command of GoSub.
- (6) The file and communication line which are opened are all closed by execution of the End command.
- (7) At program End, the Spd, Accel, Oadl, JOvrd, Ovrd, Fine and Cnt settings will be initialized.

[Related instructions]

[Hlt \(Halt\)](#), [CallP \(Call P\)](#)

Error (error)

[Function]

This instruction makes a program generate an error (9000s number).

[Format]

Error[]<Error No.>

[Terminology]

<Error No. > Either a constant or numeric operation expression can be set. Designate the No. within the range of 9000 to 9299.

[Reference Program]

(1) Generate the error 9000.

:

10 Error 9000

(2) Change the error number to generate corresponding to the value of M1.

4 If M1 <> 0 Then *LERR ' When M1 is not 0, branches to "*LERR".

:

14 *LERR

15 MERR=9000+M1*10 ' Calculate the error number according to the value of M1.

16 Error MERR ' The calculated error number is generated.

17 End

[Explanation]

(1) It is possible to generate any error in the 9000's number range by executing this instruction.

(2) If a LOW level or HIGH level error is generated, the program is paused.

Steps after the Error instruction are not executed. A CAUTION error does not pause a program; the next step and onward are executed. The action of system by error number is shown in the [Table 4-16](#).

(3) It is possible to create up to 20 error messages using parameters UER1 to UER20.

(4) A system error occurs if a value outside the error number range shown in [Table 4-16](#) is specified.

Table 4-16:Action of system by error number

No.	System behavior
9000 to 9099 (H level error)	The program execution is stopped, and the servo power is shut off. The error state is reset when error reset is input.
9100 to 9199 (L level error)	The program execution is stopped. The error state is reset when error reset is input.
9200 to 9299 (CAUTION)	The program execution is continued. The error state is reset when error reset is input.

[Related parameter]

UER1 to 20

Fine (Fine)

[Function]

This instruction specifies completion conditions of the robot's positioning. It is invalid during the smooth movement control (Cnt 1).

Depending on the type of robot (RP series), positioning using the Dly instruction may be more effective than using the Fine instruction.

[Format]

Fine[]<No. of pulses> [, <Axis No.>]

[Terminology]

<No. of pulses> Specify the positioning pulses number.

This will be invalid to when set to 0. The default value is 0.

<Axis No.> Designate the axis No. to which the positioning pulses are to be designated. The positioning pulses will be applied on all axes when omitted.

[Reference Program]

1 Fine 300	' Designate 300 for the positioning pulses.
2 Mov P1	
3 Fine 100,2	' Change the 2nd axis positioning pulses to 100.
4 Mov P2	
5 Fine 0	' Invalidate the positioning pulse designation.
6 Mov P3	
7 Fine 100	' Designate 100 for the positioning pulses.
8 Mov P4	

[Explanation]

- (1) The Fine instruction does not complete movement instructions such as Mov by giving commands to the servo; rather, it completes positioning by determining whether or not the feedback pulse value from the servo is within the specified range. It is thus possible to confirm positioning more accurately.
- (2) There are cases when the Dly instruction (timer) is used for positioning instead of the Fine instruction. This instruction is easier to specify.
10 Mov P1
20 Dly 0.1
- (3) Fine is invalid in the program until the Fine command is executed. Once Fine is validated, it remains valid until invalidated.
- (4) Fine is invalidated at the end of the program (Execution of the End instruction, program reset after pausing).
- (5) When the continuous movement control valid state (Cnt 1) is entered, the Fine command will be ignored even if it is valid (i.e., it will be treated as invalid, but the status will be kept).
- (6) To the addition axis (general-purpose servo axis), although the valid/invalid change of Fine is possible, specification of the pulse number cannot be performed. The value registered in the "INP" parameter on the servo amplifier side is used. Thus, when the integers other than zero are specified, the Fine becomes effective by the parameter set value of servo amplifier, and the Fine becomes invalid when 0 is specified.
- (7) If a positioning completion condition is specified using the Fine instruction while the compliance mode is activated, depending on the operation the robot may be unable to reach the positioning completion pulse of the target position, and will wait indefinitely for the completion of the operation instruction. As a result, the program execution comes to a halt. Do not use the compliance mode and the Fine instruction at the same time.

⚠ CAUTION

The RH-A/RH-S series and the RV-SD/RH-SDH series robots use different encoder resolutions (number of pulses) for joint axes. If the value of <No. of pulses> of the Fine instruction is the same, the RV-S/RH-S series, which normally has higher encoder resolution (number of pulses), takes longer to complete positioning. For this reason, if robots are replaced and the robot model is changed from the RV-A/RH-A series to the RV-S/RH-S series, the time it takes for the Fine instruction to complete positioning may change.

In such cases, adjust the value of <No. of pulses> of the Fine instruction.

Fine J (Fine Joint)

[Function]

Specifies the robot positioning complete conditions with a joint axis value.

The Fine J command will be disabled during continuous operation control (Cnt 1).

The Fine command or Fine P command will be disabled for all axes when the Fine J command is executed.

[Format]

Fine[]<Positioning Width>, J [, <Axis No.>]

[Terminology]

< Positioning Width > The positioning width is specified with either a variable or constant and will be disabled if 0 is specified. The default value is set to 0.

Units will be in either "mm" or "deg.", depending on the joint axis unit system.

The minimum value that can be specified is 0.001.

< Axis No. > Specifies the number of the axis that specifies the positioning pulse, and will apply to all axes if omitted. Specify with either a constant or numeric value variable.

[Reference Program]

1 Fine 1, J	' Specifies the positioning width for all axes to 1 [mm] (or [deg.]).
2 Mov P1	
3 Fine 0.5, J, 2	' Changes the no. 2 axis positioning width to 0.5 [mm] (or [deg.]).
4 Mov P2	
5 Fine 0, J, 5	' Disables the no. 5 axis positioning width specification.
6 Mov P3	
7 Fine 0, J	' Disables the positioning width specification for all axes.
8 Mov P4	

[Explanation]

- (1) The Fine J command specifies the operation command complete condition (positioning accuracy) with a feedback joint value. Operation completion is determined with a joint value, resulting in more accurate positioning.
- (2) The Fine J command deems the operation to be complete when the difference between the command joint position and feedback joint position for all enabled axes is within the <Positioning Width>.
- (3) Furthermore, there are also times when positioning is performed with a Dly command (timer) instead of the Fine J command. This is easier to specify.
 - 1 Mov P1
 - 2 Dly 0.1
- (4) Fine J is disabled for all axes by default. Once Fine J is enabled, the enabled status is applied continuously until disabled.
- (5) Fine J is disabled when a program is terminated (End command execution, program reset following an interruption).
- (6) The Fine J enabled status is temporarily ignored (disabled, status is maintained) when in the continuous operation control enabled status (Cnt 1).
- (7) The Fine command or Fine P command will be disabled for all axes when the Fine J command is executed. (The status is not maintained.)
- (8) Fine J can be enabled and disabled and the <positioning width> can be specified for additional axes (multi-purpose servo axes) also.
- (9) If the positioning complete condition is specified with the Fine J command when the compliance mode is functioning, depending on the operation, there may be times when the robot is unable to reach the positioning completion pulse for its target position, the system waits for completion of the operation command, and program execution does not proceed any further. Do not use compliance mode and the Fine J command simultaneously.

Fine P (Fine Pause)

[Function]

Specifies the robot positioning complete conditions with a linear distance.

The Fine P command will be disabled during continuous operation control (Cnt 1).

The Fine command or Fine J command will be disabled for all axes when the Fine P command is executed.

[Format]

```
Fine[]<Linear Distance>, P
```

[Terminology]

<Linear Distance> The positioning linear distance [mm] is specified with either a variable or constant and will be disabled if 0 is specified. The default value is set to 0. The minimum value that can be specified is 0.001.

[Reference Program]

```
1 Fine 1, P           ' Specifies the positioning linear distance to 1 mm.  
2 Mov P1  
3 Fine 0, P           ' Disables the positioning linear distance specification.  
4 Mov P2
```

[Explanation]

- (1) The Fine P command specifies the operation command complete condition (positioning accuracy) with a feedback linear distance. Operation completion is determined with a linear distance, resulting in more accurate positioning.
- (2) The operation is deemed to be complete when the linear distance between the respective robot current positions obtained from the command pulse and feedback pulse is within the <Linear Distance>.
- (3) Furthermore, there are also times when positioning is performed with a Dly command (timer) instead of the Fine P command. This is easier to specify.
1 Mov P1
2 Dly 0.1
- (4) Fine P is disabled for all axes by default. Once Fine P is enabled, the enabled status is applied continuously until disabled.
- (5) Fine P is disabled when a program is terminated (End command execution, program reset following an interruption).
- (6) The Fine P enabled status is temporarily ignored (disabled, status is maintained) when in the continuous operation control enabled status (Cnt 1).
- (7) The Fine command or Fine J command will be disabled for all axes when the Fine P command is executed. (The status is not maintained.)
- (8) Fine P cannot be enabled and disabled for additional axes (multi-purpose servo axes). Fine P is always disabled.
- (9) If the positioning complete condition is specified with the Fine P command when the compliance mode is functioning, depending on the operation, there may be times when the robot is unable to reach the positioning completion pulse for its target position, the system waits for completion of the operation command, and program execution does not proceed any further. Do not use compliance mode and the Fine P command simultaneously.

For - Next (For-next)

[Function]

Repeatedly executes the program between the For statement and Next statement until the end conditions are satisfied.

[Format]

```
For[]<Counter> = <Default value> To <End Value> [Step <Increment>]
:
Next[] [<Counter 1>]
```

[Terminology]

<Counter>	Describe the numerical variable that represents the counter for the number of repetitions. Same for <Counter 1> and <Counter 2>.
<Default Value>	Set default value of the counter for the number of repetitions as a numeric operation expression.
<End Value>	Set the end value of the counter for the number of repeats as a numeric operation expression.
<Increment>	Set the value of the increments for the counter for the number of repetitions as a numeric operation expression. It is allowed to omit this argument, including STEP.

[Reference Program]

(1) A program that adds the numbers 1 to 10

```
1 MSUM=0          ' Initialize the total MSUM.
2 For M1=1 TO 10  ' Increase the counter by 1 from 1 to 10 for the numeric variable M1.
3 MSUM=MSUM+M1    ' Add M1 value to numeric variable MSUM.
4 Next M1          ' Return to step 2.
```

(2) A program that puts the result of a product of two numbers into a 2-dimensional array variable

```
1 Dim MBOX(10,10)  ' Reserve space for a 10×10 array.
2 For M1=1 To 10 Steo 1 ' Increase the counter by 1 from 1 to 10 for the numeric variable M1.
3 For M2=1 To 10 Step 1 ' Increase the counter by 1 from 1 to 10 for the numeric variable M2.
4 MBOX(M1,M2)=M1*M2    ' Substitute the value of M1*M2 for the array variable MBOX (M1, M2).
5 Next M2              ' Return to step 3.
6 Next M1              ' Return to step 2.
```

[Explanation]

- (1) It is possible to describe For-Next statements between other For-Next statements. Jumps in the program caused by the For-Next instruction will add one more level to the control structure in a program. It is possible to make the control structure of a program up to 16 levels deep. An error occurs at execution if 16 levels are exceeded.
- (2) If a GoTo instruction forces the program to jump out from between a For statement and a Next statement, the free memory available for control structure (stack memory) decreases. Thus, if a program is executed continuously, an error will eventually occur. Write a program in such a way that the loop exits when the condition of the For statement is met.
- (3) A run-time error occurs under the following conditions.
 - *The counter's <Default Value> is greater than <End Value> and <Increment> is a positive number.
 - *The counter's <Default Value> is smaller than <End Value>, and <Increment> is a negative number.
- (4) A run-time error occurs if a For statement and a Next statement are not paired.
- (5) When the Next statement corresponds to the closest For statement, the variable name in the Next statement can be omitted. In the example, "M2" in step 5 and "M1" in step 6 can be omitted. The processing speed will be slightly faster to omit the counter variable.

FPrm (FPRM)

[Function]

Defines the order of the arguments, the type, and number for the main program that uses arguments in a sub program (i.e., when the host program uses another program with CALL P).

[Format]

```
FPrm[]<Dummy Argument> [,<Dummy Argument>] ...
```

[Terminology]

<Dummy Argument> The variable in the sub program that is transferred to the main statement when executed. All variables can be used. Up to 16 variables may be used.

[Reference Program]

```
<Main program>
1 M1=1
2 P2=P_Curr
3 P3=P100
4 CallP "100",M1,P2,P3      ' It can be described like "CallP "100", 1, P_Curr, P100" also.

<Sub program "100">
1 FPrm M1,P1,P2
2 If M1=1 Then GoTo 40
3 Mov P1
4 Mvs P2
5 End                      ' Return to the main program.
```

[Explanation]

- (1) FPrm is unnecessary if there are no arguments in the sub program that is called up.
- (2) An error occur when the type or number is different between the argument of CallP and the dummy argument that defined by FPrm.
- (3) It is not possible to pass the processing result of a sub program to a main program by assigning it in an argument.
To use the processing result of a sub program in a main program, pass the values using external variables.

[Related instructions]

[CallP \(Call P\)](#)

GetM (Get Mechanism)

[Function]

This instruction is used to control the robot by a program other than the slot 1 program when a multi-task is used, or to control a multi-mechanism by setting an additional axis as a user-defined mechanism.

Control right is acquired by specifying the mechanism number of the robot to be controlled. To release control right, use the RelM instruction.

[Format]

GetM[]<Mechanism No.>

[Terminology]

<Mechanism No.> 1 to 3, Specify this argument using a numerical or a variable.
The standard system's robot arm is assigned to mechanism 1.

[Reference Program]

(1) Start the task slot 2 from the task slot 1, and control the mechanism 1 in the task slot 2.

Task slot 1.

1 RelM	' Releases the mechanism in order to control mechanism 1 using slot 2.
2 XRun 2,"10"	' Start the program 10 in slot 2.
3 Wait M_Run(2)=1	' Wait for the starting confirmation of the slot 2.
:	

Task slot 2. (Program "10")

1 GetM 1	' Get the control of mechanism 1.
2 Servo On	' Turn on the servo of mechanism 1.
3 Mov P1	
4 Mvs P2	
5 P3=P_Curr	' Substitute P3 in mechanism 1 current position.
6 Servo Off	' Turn mechanism 1 servo OFF.
7 RelM	' Releases the control right of mechanism 1.
8 End	

[Explanation]

- (1) Normally (in single task operation), mechanism 1 is obtained in the initial status; it is not necessary to use the GetM instruction.
- (2) Because the control right of the same mechanism cannot be acquired simultaneously by multiple tasks, the following procedure is required in order to operate the robot by other than slot 1:
First, release control right using the RelM instruction by the slot 1 program. Next, acquire control right using the GetM instruction by the slot program that operates the robot. An error will be generated if the GetM instruction is executed again using a slot that has already acquired control right.
- (3) The instructions requiring control right include the motor power ON/OFF instruction, the interpolation instruction, the speed acceleration deceleration specification instruction, and the Tool/Base instruction.
- (4) If the argument is omitted from the system status variable requiring the mechanism designation, the currently acquired mechanism will be designated.
- (5) If the program is stopped, RelM will be executed automatically by the system. When the program is restarted, GetM will be executed automatically.
- (6) This instruction cannot be used in a constantly executed program.

[Related instructions]

[RelM \(Release Mechanism\)](#)

GoSub (Return)(Go Subroutine)

[Function]

Calls up the subroutine at the designated step label. Be sure to return from the jump destination using the Return instruction.

[Format]

```
GoSub[]<Call Destination>
```

[Terminology]

<Call Destination> Describe the step label name.

[Reference Program]

```
1 GoSub *LBL
2 End
:
20 *LBL
21 Mov P1
22 Return      ' Be sure to use the Return instruction to return.
```

[Explanation]

- (1) Make sure to return from the subroutine by using the Return command. If return by GoTo command, the memory for control structure (stack memory) will decrease, and it will cause the error at continuous executing.
- (2) The call of other subroutines is possible again by the GoSub command out of the subroutine. This approach can be employed approximately up to 800 times.
- (3) When the step or label of the call place does not exist, it becomes the execution-time error.

[Related instructions]

[Return \(Return\)](#)

GoTo (Go To)

[Function]

This instruction makes a program branch to the specified label step unconditionally.

[Format]

```
GoTo[]<Branch Destination>
```

[Terminology]

<Branch Destination> Describe the label name.

[Reference Program]

```
:
10 GoTo *LBL                ' Branches to the label *LBL.
:
100 *LBL
101 Mov P1
```

[Explanation]

(1) If a branch destination or label does not exist, an error will occur during execution.

Hlt (Halt)

[Function]

Interrupts the execution of the program and movement of the robot, and stops. The program which was being executed at this time becomes standby status.

[Format]

```
Hlt
```

[Reference Program]

(1) Stop the robot without condition during program execution.

```
15 Hlt
```

' Stop the program without condition.

(2) Stop the robot on some conditions.

```
10 If M_In(18)=1 Then Hlt
```

' Stop the program execution when the input signal 18 turns on.

```
11 Mov P1 WthIf M_In(17)=1, Hlt
```

' When the input signal 17 turns on during moving to P1, the program execution is stopped.

[Explanation]

- (1) Interrupts the execution of a program and decelerates the robot to a stop. The system will enter the waiting state.
- (2) If the Hlt instruction is used in multitask operation, only the task slot that executed the Hlt instruction is paused.
- (3) To restart, start the O/P or issue the start signal from an external source. The program will be restarted at the next step after the Hlt statement. Note that if the Hlt statement is an appended statement, the operation will restart from the same step of the program where it was interrupted.

[Related instructions]

[End \(End\)](#)

HOpen / HClose (Hand Open/Hand Close)

[Function]

Commands the hand to open or close.

[Format]

```
HOpen[]<Hand No.> [, <Starting grasp force>, <Holding grasp force>,
<Starting grasp force holding time>]
HClose[]<Hand No.>
```

[Terminology]

<Hand No.>

Select a numeric value between 1 and 8. Specify this argument using a constant or a variable.

<Starting grasp force>

This parameter is valid for the motorized hand, and invalid for any other type of hand.

Set the required grasping force for starting the hand open/close.
Set the grasping force as a step between 0 and 63 (63 = 3.5kg).

The default value is 63. When omitted, the previous setting value will be applied.

<Holding grasp force>

This parameter is valid for the motorized hand, and invalid for any other type of hand.

Set the required grasping force for holding the hand open/close.

Set the grasping force as a step between 0 and 63 (63 = 3.5kg).

The default value is 63. When omitted, the previous setting value will be applied.

<Starting grasp force holding timer>This parameter is valid for the motorized hand.

Set the duration to hold the starting grasp force as a constant or variable.
It can be set in the range of 0.00 (sec) to the maximum single-precision real number.

The default value is 0.3 sec.

[Reference Program]

```
1 HOpen 1      ' Open hand 1.
2 Dly 0.2      ' Set the timer to 0.2 sec. (Wait for the hand to open securely.)
3 HClose 1      ' Close hand 1.
4 Dly 0.2      ' Set the timer to 0.2 sec. (Wait for the hand to close securely.)
5 Mov PUP      '
```

[Explanation]

- (1) The operation (single/double) of each hand is set with parameter HANDTYPE.
- (2) If the hand type is set to double solenoid, hands 1 to 4 can be supported. If the hand type is set to single solenoid, hands 1 to 8 can be supported.
- (3) The status of the hand output signal when the power is turned ON is set with parameter HANDINIT.
- (4) The hand input signal can be confirmed with the robot status variable M_HndCq ("Hand input number").

The signal can also be confirmed with the input signals No. 900 to 907 (when there is one mechanism).

```
10 HClose 1
20 If M_HndCq(1)<>1 Then GoTo 20
30 Mov P1
```

- (5) There are related parameters. Refer to [Page 374, "5.10 Automatic return setting after jog feed at pause"](#) and, [Page 378, "5.13 About default hand status"](#) of this manual.

[Related system variables]

[M_In/M_Inb/M_Inw](#) (900s number), [M_Out/M_Outb/M_Outw](#) (900s number), [M_HndCq](#)

[Related instructions]

[Loadset \(Load Set\)](#), [Mxt \(Move External\)](#)

[Related parameter]

HANDTYPE, HANDINIT

Refer to [Page 374, "5.10 Automatic return setting after jog feed at pause"](#) and, [Page 378, "5.13 About default hand status"](#).

If...Then...Else...EndIf (If Then Else)

[Function]

A process is selected and executed according to the results of an expression.

[Format]

```
If[]<Expression>[]Then[]<Process>[]][Else <Process>]
```

```
If[]<Expression>[]Then
  <Process>
  <Process>
  Break
  :
[Else]
  <Process>
  <Process>
  Break
  :
EndIf
```

[Terminology]

- | | |
|--------------|---|
| <Expression> | Describe the expression targeted for comparison as a comparison operation expression or logic operation expression. |
| <Process> | Describe the process following Then for when the comparison results are true, and the process following Else for when the comparison results are false. |

[Reference Program]

```

1 If M1>10 Then *L1
  ' When M1 is larger than 10, jump to the step
  ' *L1.
11 If M1>10 Then GoTo *L2 Else GoTo *L3
  ' If M1 is larger than 10, it jumps to step *L2; if
  ' smaller than 10, it jumps to label *L3.
  ' The "GoTo" after "Then" or "Else" can be
  ' omitted.

  :
19 *L1
20 M1=10
21 Mov P1
22 GoTo *LC
23 *L2
24 M1=-10
25 Mov P2
26 GoTo *LC
```

[Explanation]

- (1) The If .. Then .. Else .. statements should be contained in one step.
- (2) It is allowed to split an If .. Then .. Else .. EndIf block over several steps.
- (3) Else can be omitted.
- (4) Make sure to include the EndIf statement in the If .. Then .. Else .. EndIf block.
- (5) If the GoTo instruction is used to jump out from inside an If .. Then .. Else .. EndIf block, an error will occur when the memory for control structure (stack memory) becomes insufficient.
- (6) For If .. Then .. Else .. EndIf, it is possible to describe If .. Then .. Else .. EndIf inside Then or Else. (UP to eight levels of nesting is allowed.)
- (7) GoTo following Then or Else may be omitted.

Example) If M1 > 10 Then 200 Else 300

Also, only when Then is followed by GoTo, either one of Then or GoTo may be omitted. Else cannot be omitted.

Example) If M1 > 10 Then GoTo 200 (The program at left can be rewritten as shown below.)

--- If M1 > 10 Then 200
--- If M1 > 10 GoTo 200

- (8) In the Then or the Else, it can escape to the next step of EndIf by Break. That is, process of If Then EndIf can be skipped.

Input (Input)

[Function]

Inputs data into a file (including communication lines). Only Ascii character data can be received.

Please refer to [Page 381, "5.15 About the communication setting\(RS-232\)"](#), which lists related parameters.

[Format]

```
Input[]#<File No.>, <Input data name> [, <Input data name>] ...
```

[Terminology]

<File No. > Describe a number between 1 and 8.

This corresponds to the file No. assigned with the Open command.

<Input data name> Describe the variable name for saving the input data. All variables can be described.

[Reference Program]

```
1 Open "COM1:" AS #1      ' Assign RS-232-C to file No. 1.
2 Input #1, M1            ' The value will be set to the numerical variable M1 if data are inputted from the
                           keyboard.
3 Input #1, CABC$          '
                           :
10 Close #1
```

[Explanation]

- (1) Data is input from file having the file No. opened with the Open statement, and is substituted in the variable. If the Open statement has not been executed, an error will occur.
- (2) The type of data input and the type of variable that is substituting it must be the same.
- (3) When describing multiple variable names, use a comma (,) between variable names as delimiters.
- (4) When the Input statement is executed, the status will be "standby for input. "The input data will be substituted for the variables at the same time as the carriage return (CR and LF) are input.
- (5) If the protocol (in the case of the standard port: the "CRPC232" parameter is 0) of the specified port is for PC support (non procedure), it is necessary to attach "PRN" at the head of any data sent from a PC. Normally, the standard port is connected to a PC and used for transferring and debugging robot programs. Therefore, it is recommended to use the optional expansion serial interface if a data link is used.
- (6) If the number of elements input is greater than the number of arguments in the Input statement, they will be read and discarded.
When the End or Close statement is executed, the data saved in the buffer will be erased.
Example) To input both a character string, numeric value and position.
10 Input #1,C1\$,M1,P1

Data sent from the PC side

(when received by the standard port of the robot: the "CRPS232" parameter is 0)

```
PRNMELFA,125.75,(130.5,-117.2,55.1,16.2,0,0)(1,0) CR
```

MELFA is substituted in C1\$, 125.75 in M1, and (130.5, -117.2,55.1,16.2,0,0)(1,0) in P1.

[Related instructions]

[Open \(Open\)](#), [Close \(Close\)](#), [Print \(Print\)](#)

JOvrd (J Override)

[Function]

Designates the override that is valid only during the robot's joint movements.

[Format]

JOvrd[]<Designated override>

[Terminology]

<Designated override> Describe the override as a real number.
 A numeric operation expression can also be described.
 Unit: [%] (Recommended range: 1 to 100.0)

[Reference Program]

```
1 JOvrd 50
2 Mov P1
3 JOvrd M_NJovrd      ' Set the default value.
```

[Explanation]

- (1) The JOvrd command is valid only during joint interpolation.
- (2) The actual override is = (Operation panel (T/B) override setting value) x (Program override (Ovrd command)) x (Joint override (JOvrd command)). The JOvrd command changes only the override for the joint interpolation movement.
- (3) The 100% <Designate override> is the maximum capacity of the robot. Normally, the system default value (M_NOvrd) is set to 100%. The value is reset to the default value when the End statement is executed or the program is reset.

[Related instructions]

[Ovrd \(Override\)](#), [Spd \(Speed\)](#)

[Related system variables]

[M_JOvrd/M_NJovrd/M_OPovrd/M_Ovrd/M_NOvrd](#)
 (M_NJovrd:System default value, M_JOvrd:Currently specified joint override)

JRC (Joint Roll Change)

[Function]

- This instruction rewrites the current coordinate values by adding +/-360 degrees to the current joint coordinate values of the applicable axis (refer to <Axis No> in [Terminology]) of the robot arm.
- User-defined axis (additional axis, user defined mechanism)
This instruction rewrites the current coordinate values by adding/subtracting the value specified by a parameter to/from the current joint coordinate values of the specified axis. This instruction can be used for both rotating and linear axes. The origin can also be reset at the current position.

[Format]

```
JRC <[+] 1 / -1 / 0 > [, <Axis No>]
```

```
JRC <[+] <Numeric Value> / -<Numeric Value> / 0 > [, <Axis No>]
```

[Terminology]

- <+1> The current joint angle of the designated axis is incremented by the amount designated in parameter JRCQTT (The sign can be omitted.). For the priority axes of the robot arm, it is fixed at 360 degrees.
- <-1> The current joint angle of the designated axis is decremented by the amount designated in parameter JRCQTT. For the priority axes of the robot arm, it is fixed at 360 degrees.
- <0> The origin for the designated axis is reset at the value designated in parameter JRCORG. This can be used only for the user-defined axis.
- <Axis No> The target axis is specified with the number. The priority axes are used if omitted. Note that this argument cannot be omitted if additional axes and/or user-defined mechanical axes are the targets.

[Applicable Models and Applicable Axes]

- (1) Applicable models and priority axes

RV-SD series: J6 axis
RH-SDH series: J4 axis

- (2) Additional axes of all models

- (3) All axes of user defined mechanisms

- <Numeric Value> Specify an incremental/decremental number (a multiple of 360 degrees). Description by the constant or the variable is possible (J1 edition or later is possible).
Example) +3: Increases the applicable axis angle by 1080 degrees.
-2: Decreases the angle by 720 degrees..

[Reference Program]

- | | |
|----------|--|
| 1 Mov P1 | ' Moves to P1. (The movement to which the J6 axis moves in the minus direction) |
| 2 JRC +1 | ' Add 360 degrees to the current coordinate values of the applicable axis. |
| 3 Mov P1 | ' Moves to P1. |
| 4 JRC +1 | ' Add 360 degrees to the current coordinate values of the applicable axis. |
| 5 Mov P1 | ' Moves to P1. |
| 6 JRC -2 | ' Subtract 720 degrees from the current coordinate values of the applicable axis.
(Reverts) |

[Explanation]

- (1) With the JRC 1/1 instruction (JRC n/-n), the current joint coordinate values of the specified axis are incremented/decremented.
The origin for the designated axis is reset with the JRC 0 command.
Although the values of the joint coordinates change, the robot does not move.
- (2) When using this command, change the movement range of the target axis beforehand so that it does not leave the movement range when the command is executed. The range can be changed by changing the - side and + side value of the corresponding axis in the joint movement range parameter "MEJAR".
Set the movement range for the rotating axis in the range of -2340 deg. to 2340 deg.
- (3) If the designated axis is omitted, the priority axis will be the target. The priority axis is the rotating axis (J6 axis) at the end of the robot.
- (4) If the designated axis is omitted when a priority axis does not exist (robot incapable of JRC), or if the designated axis is not a target for JRC, an error will occur when the command is executed.
- (5) If the origin is not set, an error will occur when the command is executed.
- (6) The robot is stopped while the JRC command is executed. Even if Cnt is validated, the interpolation connection will not be continuous when this command is executed.
- (7) The following parameter must be set before using the JRC command.
 - Set JRCEXE to 1. (JRC execution enabled)
 - Change the movement range of the target axis with MEJAR.
 - Set the position change amount during the JRC 1/1(JRC n/-n) execution with JRCQTT.
(Only for the additional axis or user-defined mechanism.)
 - Set the origin position for executing JRC 0 with JRCORG.
(Only for the additional axis or user-defined mechanism.)
- (8) When parameter JRCEXE is set to 0, no process will take place even if JRC command is executed.
- (9) If the movement amount designated with parameter JRCQTT is not within the pulse data 0 to Max., an error will occur during the initialization. Here, Max. is $2^{\wedge}(\text{Number of encoder bits} + 15) - 1$. For example, with a 13-bit encoder (8192 pulses), this will be Max. = $2^{\wedge}(13+15)-1 = 0x0fffff$, and for a 14-bit encoder (16384 pulses), this will be Max. $2^{\wedge}(14+15)-1 = 0x1fffff$.

The movement amount to pulse data conversion is as follows:

For rotating axis

Pulse data = movement amount (deg.)/360 * gear ratio denominator/gear ratio numerator * Number of encoder pulses

For linear axis

Pulse data = movement amount (mm) * gear ratio denominator/gear ratio numerator * Number of encoder pulses

- (10) The origin data will change when JRC is executed, so the default origin data will be unusable.
If the controller needs to be initialized due to a version upgrade, etc., the parameters must be backed up beforehand in the original state.
- (11) Step return operation is not possible with the JRC command.
- (12) This instruction cannot be used in a constantly executed program.

[Related parameter]**JRCEXE**

Set whether to enable/disable the JRC execution.

Execution disabled = 0 (default value)/execution enabled = 1

JRCQTT

Designate the amount to move (1 deg./1mm unit) when incrementing or decrementing with the JRC command in additional axis or user-defined mechanism.

For the JRC's applicable axis on the robot arm side, it is fixed at 360 degrees regardless of this setting.

JRCORG

Designate the origin for executing JRC 0. in additional axis or user-defined mechanism.

Refer to [Page 347, "5 Functions set with parameters"](#) for detail.

Loadset (Load Set)

[Function]

This instruction specifies the condition of the hand/workpiece at execution of the Oadl instruction.

[Format]

```
LoadSet[]<Hand condition No.>, <Workpiece condition No.>
```

[Terminology]

<Hand condition No. > 1 to 8. Designate the hand condition (HNDDAT 1 to 8) No. for which the weight and size are designated.

<Workpiece condition No. > 1 to 8. Designate the hand condition (WRKDAT 1 to 8) No. for which the weight and size are designated. .

[Reference Program]

```
1 Oadl On
2 LoadSet 1,1      ' Hand 1(HNDDAT1) and workpiece 1(WRKDAT1) conditions.
3 Mov P1
4 Mov P2
5 LoadSet 1,2      ' Hand 1(HNDDAT1) and workpiece 2(WRKDAT1) conditions.
6 Mov P1
7 Mov P2
8 Oadl Off
```

[Explanation]

- (1) Set the hand conditions and workpiece conditions used for optimum acceleration/deceleration. This is used when setting the optimum acceleration/deceleration for workpiece types having different weights.
- (2) The maximum load is set for the hand when the program execution starts.
- (3) Set the weight, size (X, Y, Z) and center of gravity position (X, Y, Z) as the hand conditions in parameter (HNDDAT 1 to 8).
- (4) Set the weight, size (X, Y, Z) and center of gravity position (X, Y, Z) as the workpiece conditions in parameter (WRKDAT 1 to 8).
- (5) The hand conditions and workpiece conditions changed when this command is executed are reset to the system default value when the program is reset and when the End statement is executed.
As the system default values, the hand conditions are set to the rated load, and the workpiece conditions are set to none (0kg).
- (6) Refer to [Page 389, "5.18 Hand and Workpiece Conditions \(optimum acceleration/deceleration settings\)"](#) for details on the optimum acceleration/deceleration.

[Related instructions]

[Mxt \(Move External\)](#), [HOpen / HClose \(Hand Open/Hand Close\)](#)

[Related parameter]

HNDDAT1 to 8, WRKDAT1 to 8, HNDHOLD1 to 8

Refer to [Page 389, "5.18 Hand and Workpiece Conditions \(optimum acceleration/deceleration settings\)"](#).

Refer to [Page 355, "Table 5-2: List Signal parameter"](#) for the ACCMODE.

Mov (Move)

[Function]

Using joint interpolation operation, moves from the current position to the destination position.

[Format]

```
Mov[]<Target Position> [, <Close Distance>] [|]Type[]<Constants 1>, <Constants 2>|[]  
[<Appended conditions>]
```

[Terminology]

<Movement Target Position> This is the final position for interpolation operation. This position may be specified using a position type variable and constant, or a joint variable.

<Close Distance> If this value is designated, the actual movement target position will be a position separated by the designated distance in the tool coordinate system Z axis direction (+/- direction).

<Constants 1> 1/0 : Detour/short cut. The default value is 1(detour).

<Constants 2> Invalid (Specify 0).

<Appended conditions> The Wth and WHTIF statements can be used.

[Reference Program]

```
1 Mov P1 Type 1,0  
2 Mov J1  
3 Mov (Plt 1,10),100.0 Wth M_Out(17)=1  
4 Mov P4+P5,50.0 Type 0,0 WthIf M_In(18)=1,M_Out(20)=1
```

[Explanation]

- (1) The joint angle differences of each axis are evenly interpolated at the starting point and endpoint positions. This means that the path of the tip cannot be guaranteed.
- (2) By using the Wth and WthIf statement, the signal output timing and motion can be synchronized.
- (3) The numeric constant 1 for the TYPE designates the posture interpolation amount.
- (4) Detour refers to the operating exactly according to the teaching posture. Short cut operation may take place depending on the teaching posture.
- (5) Short cut operation refers to posture interpolation between the start point and end point in the direction with less motion.
- (6) The detour/short cut designation is significant when the posture axis has a motion range of (180 deg. or more).
- (7) Even if short cut is designated, if the target position is outside the motion range, the axis may move with the detour in the reverse direction.
- (8) The TYPE numeric constant 2 setting is insignificant for joint interpolation.
- (9) This instruction cannot be used in a constantly executed program.
- (10) If paused during execution of a Mov instruction and restarted after jog feed, the robot returns to the interrupted position and restarts the Mov instruction. The interpolation method (JOINT interpolation / XYZ interpolation) which returns to the interrupted position can be changed by the "RETPATH" parameter. Moreover, it is also possible by changing the value of this RETPATH parameter to move to the direct target position, without returning to the interrupted position. (Refer to [Page 374, "5.10 Automatic return setting after jog feed at pause"](#))

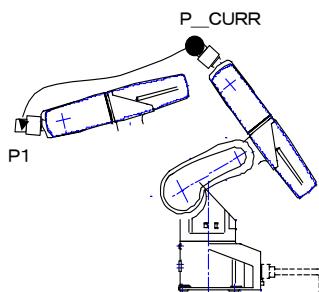


Fig.4-9:Example of joint interpolation motion path

Mva (Move Arch)

[Function]

This instruction moves the robot from the current position to the target position with an arch movement (arch interpolation).

[Format].

```
Mva[]<Target Position> [, <Arch number>]
```

[Terminology]

<Target Position>

Final position of interpolation movement. This position may be specified using a position type variable and constant, or a joint variable.

<Arch number>

A number defined by the Def Arch instruction (1 to 4).

If the argument is omitted, 1 is set as the default value.

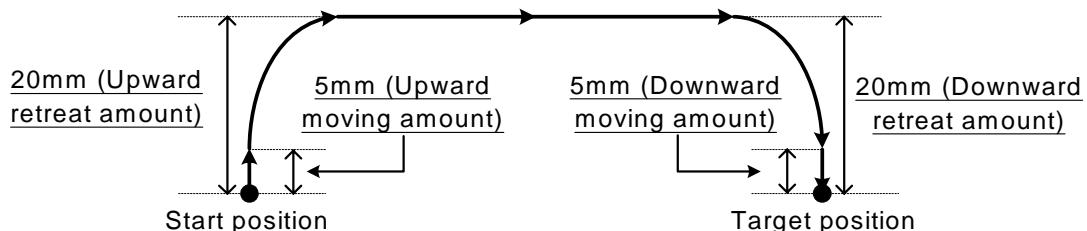
[Reference Program]

1 Def Arch 1,5,5,20,20	' Defines the arch shape configuration.
2 Ovrd 100,20,20	' Specifies override.
3 Accel 100,100,50,50,50,50	' Specifies acceleration/deceleration rate.
2 Mva P1,1	' Performs the arch motion movement according to the shape configuration defined in step 1.
3 Mva P2,2	' Moves the robot according to the default values registered in the parameters.

[Explanation]

- (1) The robot moves upward along the Z-axis direction from the current position, then moves to a position above the target position, and finally moves downward, reaching the target position. This so-called arch motion movement is performed with one instruction.
- (2) If the Mva instruction is executed without the Def Arch instruction, the robot moves with the arch shape configuration set in the parameters. Refer to [Page 183, "Def Arch \(Define arch\)"](#) for a detailed description about the parameters.
- (3) The interpolation form, type and other items are also defined by the Def Arch instruction; refer to [Page 183, "Def Arch \(Define arch\)"](#).
- (4) This instruction cannot be used in a constantly executed program.
- (5) If paused during execution of a Mva instruction and restarted after jog feed, the robot returns to the interrupted position and restarts the Mva instruction. (this can be changed by the "RETPATH" parameter). The interpolation method (JOINT interpolation / XYZ interpolation) which returns to the interrupted position can be changed by the "RETPATH" parameter. (Refer to [Page 374, "5.10 Automatic return setting after jog feed at pause"](#))

DEF ARCH 1,5,5,20,20



*If Z is different between the movement starting position and the target position, it will operate as follows:

DEF ARCH 1,5,5,20,20

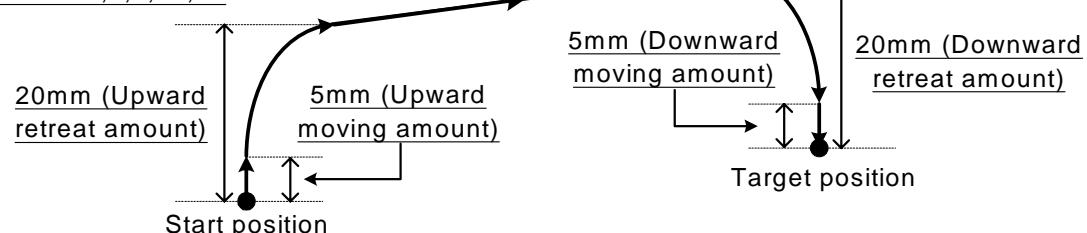


Fig.4-10:Example of arch interpolation motion path (seen from the side)

[Related instructions]

[Def Arch \(Define arch\)](#), [Accel \(Accelerate\)](#), [Ovrd \(Override\)](#)

Mvc (Move C)

[Function]

Carries out 3D circular interpolation in the order of start point, transit point 1, transit point 2 and start point.

[Format]

Mvc[]<Start point>,<Transit point 1>,<Transit point 2>[]<Additional condition>

[Terminology]

- <Start point> The start point and end point for a circle. Describe a position operation expression or joint operation expression.
- <Transit point 1> Transit point 1 for a circular arc. Describe a position operation expression or joint operation expression.
- <Transit point 2> Transit point 2 for a circular arc. Describe a position operation expression or joint operation expression.
- <Additional condition> Describe a Wth conjunction or a Wthlf conjunction

[Reference Program]

- 1 Mvc P1,P2,P3
- 2 Mvc P1,J2,P3
- 3 Mvc P1,P2,P3 Wth M_Out(17)=1
- 4 Mvc P3,(Plt 1,5),P4 Wthlf M_In(20)=1,M_Out(21)=1

[Explanation]

- (1) In circular interpolation motion, a circle is formed with the 3 given points, and the circumference is moved. (360 degrees)
- (2) The posture at the starting point is maintained during circle interpolation. The postures while passing points 1 and 2 are not considered.
- (3) If the current position and the starting position do not match, the robot automatically moves to the starting point based on the linear interpolation (3-axis XYZ interpolation), and then performs the circle interpolation.
- (4) If paused during execution of a Mvc instruction and restarted after jog feed, the robot returns to the interrupted position by JOINT interpolation and restarts the remaining circle interpolation.
The interpolation method (JOINT interpolation / XYZ interpolation) which returns to the interrupted position can be changed by the "RETPATH" parameter. (Refer to [Page 374, "5.10 Automatic return setting after jog feed at pause"](#))
- (5) This instruction cannot be used in a constantly executed program.

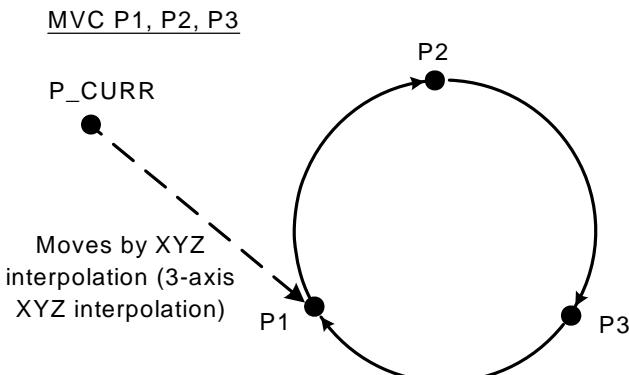


Fig.4-11:Example of circle interpolation motion path

Mvr (Move R)

[Function]

Carries out 3-dimensional circular interpolation movement from the start point to the end point via transit points.

[Format]

```
Mvr[]<Start Point>, <Transit Point>, <End Point>
  [[TYPE[]<Constants 1>, <Constants 2>]] [<Appended Condition>]
```

[Terminology]

<Start Point>	Start point for the arc. Describe a position operation expression or joint operation expression.
<Transit Point>	Transit point for the arc. Describe a position operation expression or joint operation expression.
<End Point>	End point for the arc. Describe a position operation expression or joint operation expression.
<Constants 1>	Short cut/detour = 1/0, The default value is 0.
<Constants 2>	Equivalent rotation/3-axis XYZ/singular point passage = 0/1/2. The default value is 0.
<Appended conditions>	The Wth and Wthlf statements can be used.

[Reference Program]

- 1 Mvr P1,P2,P3
- 2 Mvr P1,J2,P3
- 3 Mvr P1,P2,P3 Wth M_Out(17)=1
- 4 Mvr P3,(Plt 1,5),P4 Wthlf M_In(20)=1,M_Out(21)=1

[Explanation]

- (1) In circular interpolation motion, a circle is formed with three given points, and robot moves along the circumference.
- (2) The posture is interpolation from the start point to the end point; the transit point posture has no effect.
- (3) If the current position and start point do not match, the robot will automatically move with linear interpolation (3-axis XYZ interpolation) to the start point.
- (4) If paused during execution of a Mvr instruction and restarted after jog feed, the robot returns to the interrupted position by JOINT interpolation and restarts the remaining circle interpolation. The interpolation method (JOINT interpolation / XYZ interpolation) which returns to the interrupted position can be changed by the "RETPATH" parameter. (Refer to [Page 374, "5.10 Automatic return setting after jog feed at pause"](#))
- (5) If the start point and end point structure flags differ when equivalent rotation (constant 2 = 0) is specified, an error will occur at the execution.
- (6) Of the three designated points, if any points coincide with the other, or if three points are on a straight line, linear interpolation will take place from the start point to the end point. An error will not occur.
- (7) If 3-axis XYZ is designated for the constant 2, the constant 1 will be invalidated, and the robot will move with the taught posture.
- (8) Constant 2 designates the posture interpolation type. 3-axis XYZ is used when carrying out interpolation on the (X, Y, Z, J4, J5, J6) coordinate system, and the robot is to move near a particular point.
- (9) This instruction cannot be used in a constantly executed program.

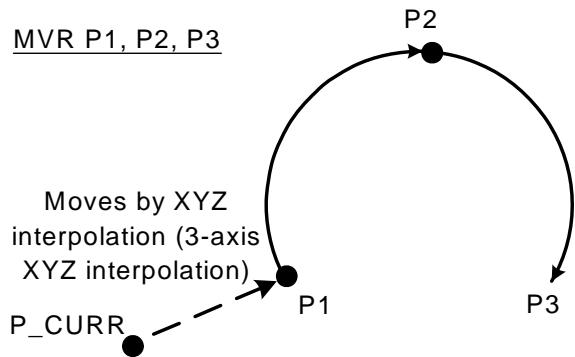


Fig.4-12:Example of circular interpolation motion path 1

Mvr2 (Move R2)

[Function]

Carries out 3-dimensional circular interpolation motion from the start point to the end point on the arc composed of the start point, end point, and reference points.

The direction of movement is in a direction that does not pass through the reference points.

[Format]

```
Mvr2[]<Start Point>, <End Point>, <Reference point>
[[]]Type[]<Constants 1>, <Constants 2>][[]]<Appended Condition>]
```

[Terminology]

<Start Point>	Start point for the arc. This position may be specified using a position type variable and constant, or a joint variable.
<End Point>	End point for the arc. This position may be specified using a position type variable and constant, or a joint variable.
<Reference point>	Reference point for a circular arc. This position may be specified using a position type variable and constant, or a joint variable.
<Constants 1>	Short cut/detour = 1/0, The default value is 0.
<Constants 2>	Equivalent rotation/3-axis XYZ/singular point passage = 0/1/2. The default value is 0.
<Appended conditions>	The Wth and Wthlf statements can be used.

[Reference Program]

```
1 Mvr2 P1,P2,P3
2 Mvr2 P1,J2,P3
3 Mvr2 P1,P2,P3 Wth M_Out(17)=1
4 Mvr2 P3,(Plt 1,5),P4 Wthlf M_In(20)=1,M_Out(21)=1
```

[Explanation]

- (1) In circular interpolation motion, a circle is formed with three given points, and robot moves along the circumference.
- (2) The posture is interpolation from the start point to the end point; the reference point posture has no effect.
- (3) If the current position and start point do not match, the robot will automatically move with linear interpolation (3-axis XYZ interpolation) to the start point.
- (4) If paused during execution of a Mvr instruction and restarted after jog feed, the robot returns to the interrupted position by JOINT interpolation and restarts the remaining circle interpolation.
The interpolation method (JOINT interpolation / XYZ interpolation) which returns to the interrupted position can be changed by the "RETPATH" parameter. (Refer to [Page 374, "5.10 Automatic return setting after jog feed at pause"](#))
- (5) The direction of movement is in a direction that does not pass through the reference points.
- (6) If the start point and end point structure flags differ when equivalent rotation (constant 2 = 0) is specified, an error will occur at the execution.
- (7) Of the three designated points, if any points coincide with the other, or if three points are on a straight line, linear interpolation will take place from the start point to the end point. An error will not occur.
- (8) If 3-axis XYZ is designated for the constant 2, the constant 1 will be invalidated, and the robot will move with the taught posture.
- (9) Constant 2 designates the posture interpolation type. 3-axis XYZ is used when carrying out interpolation on the (X, Y, Z, J4, J5, J6) coordinate system, and the robot is to move near a particular point.
- (10) This instruction cannot be used in a constantly executed program.

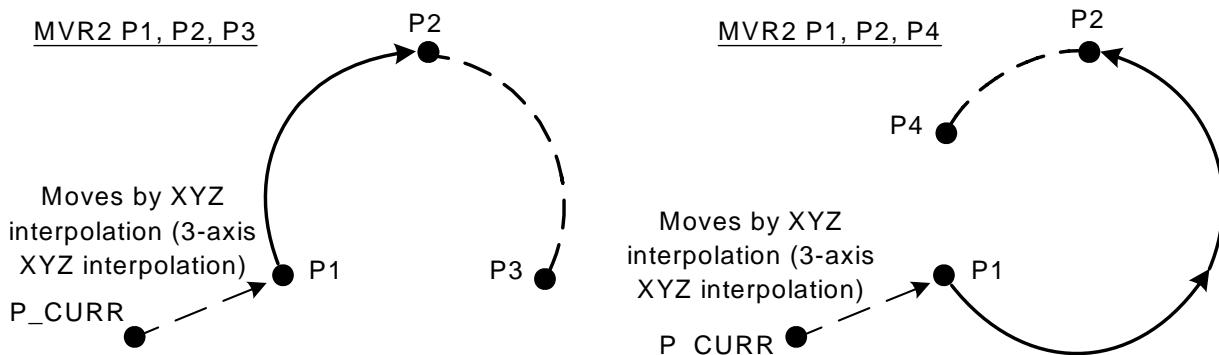


Fig.4-13:Example of circular interpolation motion path 2

Mvr3 (Move R 3)

[Function]

Carries out 3-dimensional circular interpolation movement from the start point to the end point on the arc composed of the center point, start point and end point.

[Format]

```
Mvr3[]<Start Point>, <End Point>, <Center Point>
[[]]Type[]<Constants 1>ÅC<Constants 2>[]] [<Appended Condition>]
```

[Terminology]

<Start Point>	Start point for the arc. This position may be specified using a position type variable and constant, or a joint variable.
<End Point>	End point for the arc. This position may be specified using a position type variable and constant, or a joint variable.
<Center Point>	Center point for the arc. This position may be specified using a position type variable and constant, or a joint variable.
<Constants 1>	Short cut/detour = 1/0, The default value is 0.
<Constants 2>	Equivalent rotation/3-axis XYZ/singular point passage = 0/1/2. The default value is 0.
<Appended conditions>	The Wth and Wthlf statements can be used.

[Reference Program]

- 1 Mvr3 P1,P2,P3
- 2 Mvr3 P1,J2,P3
- 3 Mvr3 P1,P2,P3 Wth M_Out(17)=1
- 4 Mvr3 P3,(Plt 1,5),P4 Wthlf M_In(20)=1,M_Out(21)=1

[Explanation]

- (1) In circular interpolation motion, a circle is formed with three given points, and robot moves along the circumference.
- (2) The posture is interpolation from the start point to the end point; the center point posture has no effect.
- (3) If the current position and start point do not match, the robot will automatically move with linear interpolation (3-axis XYZ interpolation) to the start point.
- (4) If paused during execution of a Mvr3 instruction and restarted after jog feed, the robot returns to the interrupted position by JOINT interpolation and restarts the remaining circle interpolation. The interpolation method (JOINT interpolation / XYZ interpolation) which returns to the interrupted position can be changed by the "RETPATH" parameter. (Refer to [Page 374, "5.10 Automatic return setting after jog feed at pause"](#))
- (5) If the start point and end point structure flags differ when equivalent rotation (constant 2 = 0) is specified, an error will occur at the execution.
- (6) If 3-axis XYZ is designated for the constant 2, the constant 1 will be invalidated, and the robot will move with the taught posture.
- (7) Constant 2 designates the posture interpolation type. 3-axis XYZ is used when carrying out interpolation on the (X, Y, Z, J4, J5, J6) coordinate system, and the robot is to move near a particular point.
- (8) The central angle from the start point to the end point always satisfies $0 < \text{central angle} < 180$ degrees.
- (9) Designate the positions so that the difference from the center point to the end point and the center point to the distance is within 0.01mm.
- (10) If the three points are on the same line, or if the start point and center point, or end point and center point are the same, an error will occur.
- (11) If the start point and end point are the same or if three points are the same, an error will not occur, and the next command will be executed. Note that if the posture changes at this time, only the posture will be interpolated.
- (12) This instruction cannot be used in a constantly executed program.

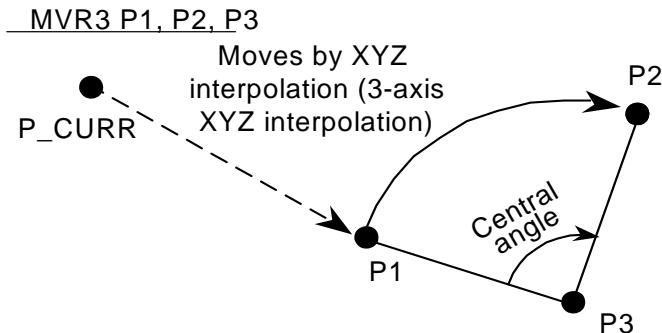


Fig.4-14:Example of circular interpolation motion path 3

Mvs (Move S)

[Function]

Carries out linear interpolation movement from the current position to the movement target position.

[Format 1]

```
Mvs[]<Movement Target Position> [, <Close Distance>]
  [Type[]<Constants 1>,<Constants 2>][][<Appended Condition>]
```

[Format 2]

```
Mvs[], <Separation Distance>
  [Type[]<Constants 1>,<Constants 2>][][<Appended Condition>]
```

[Terminology]

<Movement Target Position>	The final position for the linear interpolation. This position may be specified using a position type variable and constant, or a joint variable.
<Close Distance>	If this value is designated, the actual movement target position will be a position separated by the designated distance in the tool coordinate system Z axis direction (+/- direction).
<Constants 1>	Short cut/detour = 1/0, The default value is 0.
<Constants 2>	Equivalent rotation/3-axis XYZ/singular point passage = 0/1/2.
<Appended conditions>	The default value is 0.
<Separation Distance>	The Wth and Wthlf statements can be used. When this value is designated, the axis will move the designated distance from the current position to the Z axis direction (+/- direction) of the tool coordinate system.

[Reference Program]

(1) Move to the target position P1 by XYZ interpolation.

1 Mvs P1

(2) Turns on the output signal 17 at the same time if it moves to the target position P1 by linear interpolation.

1 Mvs P1,100.0 Wth M_Out(17)=1

(3) Turns on output signal 20 if the input signal 18 is turned on while moving 50 mm in the Z direction of the tool coordinate system of the target position P4+P5 (relative operation position obtained by addition) by linear interpolation.

2 Mvs P4+P5, 50.0 Wthlf M_In(18)=1, M_Out(20)=1

(4) Moves 50 mm in the Z direction of the tool coordinate system from the current position by linear interpolation.

3 Mvs ,50

[Explanation]

- (1) Linear interpolation motion is a type of movement where the robot moves from its current position to the movement target position so that the locus of the control points is in a straight line.
- (2) The posture is interpolation from the start point to the end point.
- (3) In the case of the tool coordinate system specified by using <proximity distance> or <separation distance>, the + and - directions of the Z axis vary depending on the robot model. Refer to [Page 369, "5.6 Standard Tool Coordinates"](#) for detail. The "[Fig.4-15:Example of movement at linear interpolation](#)" is the example of RV-6SD movement.

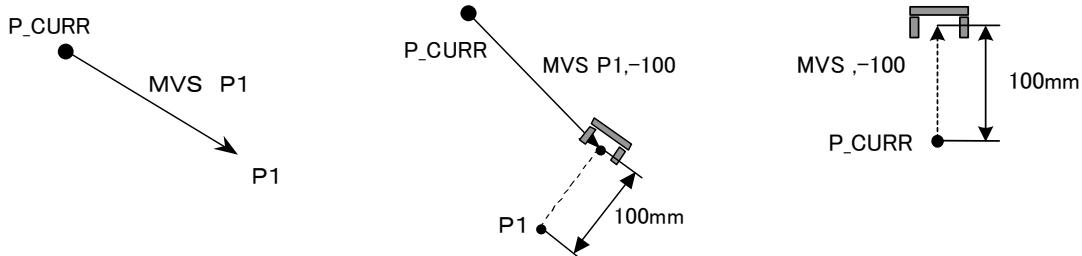
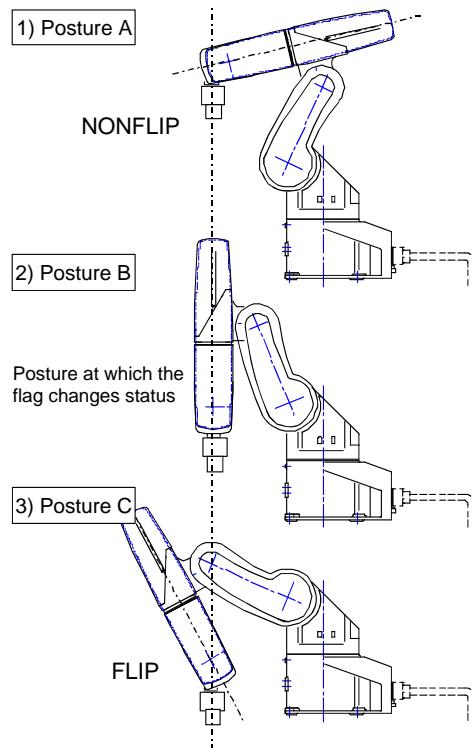


Fig.4-15:Example of movement at linear interpolation

- (4) If paused during execution of a Mvs instruction and restarted after jog feed, the robot returns to the interrupted position and restarts the Mvs instruction. This can be changed by the "RETPATH" parameter, and also the interpolation method (JOINT interpolation / XYZ interpolation) which returns to the interrupted position can be changed by same parameter. Some robots for liquid crystal transportation have different default values of this parameter. Refer to [Page 374, "5.10 Automatic return setting after jog feed at pause"](#).
- (5) This instruction cannot be used in a constantly executed program.
- (6) If the start point and end point structure flags differ when equivalent rotation (constant 2 = 0) is specified, an error will occur at the execution.
- (7) If 3-axis XYZ is designated for the numeric constant 2, the numeric constant 1 will be invalidated, and the robot will move with the taught posture.
- (8) Constant 2 designates the posture interpolation type. 3-axis XYZ is used when carrying out interpolation on the (X, Y, Z, J4, J5, J6) coordinate system, and the robot is to move near a particular point.

(9) Description of singular points.

About singular points of vertical 6-axis robots



<In the case of a vertical 6-axis robot>

Movement from posture A, through posture B, to posture C cannot be performed using the normal linear interpolation (Mvs).

This limitation applies only when J4 axis is at zero degrees at all the postures A, B, and C. This is because the structure flag of axis J5 (wrist axis) is FLIP for posture A and NONFLIP for posture C. Moreover, in posture B, the wrist is fully extended and axes J4 and J6 are located on the same line. In this case, the robot cannot perform a linear interpolation position calculation.

The 3-axis XYZ (TYPE 0, 1) method in the command option of Mvs should be used if it is desired to perform linear interpolation based on such posture coordinates. Note that, strictly speaking, this 3-axis XYZ method does not maintain the postures as it evenly interpolates the joint angle of axes J4, J5, and J6 at posture A and C. Therefore, it is expected that the robot hand's posture may move forward and backward while moving from posture A to posture C.

In this case, add one point in the middle to decrease the amount of change in the hand's posture.

Another singular point is when the center of axis J5 is on the Z axis of the base coordinates and the wrist is facing upward. In this case, J1 and J6 are located on the same axis and it is not possible to calculate the robot position.

Fig.4-16:Singular point 1

Mv Tune (Move Tune)

[Function]

Select the robot operating characteristics from one of the following three modes. The robot operating performance will improve by selecting the optimum operating characteristics based on the application.

Operating characteristics are optimized based on the hands and workpieces specified with the LoadSet command. Set the correct weight, shape and barycentric position of hands and workpieces actually used.

Moving Characteristics Mode		Features
1	Standard mode (default)	This is the maker standard setting. This mode has been tuned to standard characteristics that can be used for all applications.
2	High-speed positioning mode	This mode reduces the time taken to reach the target position. This setting is effective for reducing tact time, however, the load on the motor is greater when compared to "Standard mode". Set to standard mode when there is no load ratio leeway.
3	Trajectory priority mode	This mode improves the trajectory accuracy. This mode is used when route accuracy is required midway through operation, however, the operating time is longer when compared to "Standard mode". Set to "Standard mode" or "High-speed positioning mode" to prioritize tact time. This mode has the same effect as when high-accuracy mode is enabled with the Prec command.

[Format]

MvTune[]<Operating Characteristics Mode>

[Terminology]

<Operating Characteristics Mode> The robot operating characteristics mode (1 to 3) is specified with either a constant or numeric value variable.
 1: Standard mode (default)
 2: High-speed positioning mode
 3: Trajectory priority mode

[Reference Program]

```
LoadSet 1, 1      ' Sets to hand 1/workpiece 1.
MvTune 2          ' Changes the operating characteristics mode to high-speed positioning
Mvs P1
MvTune 3          ' Changes the operating characteristics mode to trajectory priority mode
Mvs P2
```

[Explanation]

- (1) This has been adjusted to ensure the optimum characteristics based on the hand and workpiece conditions specified with the LoadSet command. If the hand and workpiece conditions have not been set correctly, there is a possibility that sufficient performance will not be achieved. (Refer to [Page 389, "5.18 Hand and Workpiece Conditions \(optimum acceleration/deceleration settings\)"](#))
- (2) Standard mode is specified as the default immediately after the power is turned ON.
- (3) The operating characteristics mode returns to standard mode when a program is terminated (End command execution, program reset following an interruption), however, the current operating characteristics mode is retained with the sub-program End command executed with the CallP command.
- (4) The current operating characteristics mode is retained when restarting after a program interruption (including interruption, restart after program editing).
- (5) The differences between the standard mode and the other operating characteristics modes are as follows.

Operating characteristics mode	Target position arrival time	Trajectory accuracy	Load ratio
High-speed positioning	Shorter	Improvement (small)	Increased
Trajectory priority	Longer	Improvement (large)	Decreased

- (6) If optimum acceleration/deceleration control (specified with the Oadl command or ACCMODE parameter) is disabled, it is automatically enabled by executing the MvTune command. Furthermore, if OadleOff is executed after executing the MvTune command, optimum acceleration/deceleration control only will be disabled. (The operating mode will not change.)
- (7) Depending on the robot model or program, there may be times when the effect is not clear even after changing the operating characteristics mode.
- (8) With high-speed positioning mode, the vibration level when decelerating and stopping is sometimes greater when compared to standard mode. Select standard mode if there is any interference.
- (9) Trajectory priority mode has been adjusted so that the optimum effect is obtained with operating speeds in the mid to low speed range. Consequently, when performing an operation such as drawing a small circle at high speed, the robot may vibrate more than when in standard mode. In such a case, use the Spd command and lower the speed until the robot no longer vibrates.
- (10) This command does not function for the jog operation.

[Related instructions]

[Loadset \(Load Set\)](#)、[Mxt \(Move External\)](#)、[Prec \(Precision\)](#)

[Related parameter]

ACCMODE、HNDDAT 0～8、WRKDAT 0～8

Mxt (Move External)

[Function]

The real-time external control function by ethernet I/F

The absolute position data is retrieved from an external source at each controller control time (currently approx 7.1msec), and the robot is directly moved.

[Format]

Mxt <File No.>, <Reply position data type> [, <Filter time constant>]

[Terminology]

<File No.>

Describe a number between 1 and 8 assigned with the Open command.
If the communication destination is not designated with the Open command, an error will occur, and communication will not be possible.
In addition, data received from a source other than the communication destination will be ignored.

<Reply position data type> Designate the type of the position data to be received from the personal computer.
A XYZ/joint/motor pulse can be designated.

- 0: XYZ coordinate data
- 1: Joint coordinate data
- 2: Motor pulse coordinate data

<Filter time constant> If 0 is designated, the filter will not be applied. (0 will be set when omitted.) A filter is applied on the reception position data, an obtuse command value is created and output to the servo.

[Reference Program]

1 Open "ENET:192.168.0.2"	As #1'Set Ethernet communication destination IP address
2 MovP1	'Move to P1
3 Mxt1,1,50	'Move with real-time external control with filter time constant set to 50msec
4 Mov P1	'Move to P1
5 Hlt	'Halt program

[Explanation]

- * When the Mxt command is executed, the position command for movement control can be retrieved from the personal computer connected on the network. (One-on-one communication)
- * One position command can be retrieved and operated at the operation control time (currently 7.1msec).
- * Operation of Mxt command

- (1) When this command is executed with the controller, the controller enters the command value reception enabled state. The workpiece grasp/not grasp for when the hand is opened or closed is set with parameter HNDHOLD 1 to 8.
- (2) When the controller receives the command value from the personal computer, it will output the received command value to the servo within the next control process cycle.
- (3) After the command value is sent to the servo, the controller information, such as the current position is sent from the controller to the personal computer.
- (4) A reply is made from the controller to the personal computer only when the command value from the personal computer is sent to the controller.
- (5) If the data is not received, the current position is maintained.

(6) When the real-time external command end command is received from the personal computer, the Mxt command is ended.

(7) When the operation is stopped from the operating panel or external input, the Mxt command will be halted, and the transmission/reception will also be halted until restart.

* The timeout is designated with the parameter MXTTOUT.

* One randomly designated (head bit, bit width) input/output signal can be transmitted and received simultaneously with the position data.

* A personal computer with sufficient processing speed must be used to command movement in the movement control time.

* Refer to [Page 382, "5.16 About the communication setting\(Ethernet\)"](#) for details.

A Windows NT or 2000/Pentium II 450MHz or higher console application is recommended.

[Related instructions]

[Open \(Open\)](#)

Oadl (Optimal Acceleration)

[Function]

Automatically sets the optimum acceleration/deceleration according to the robot hand's load state (Optimum acceleration/deceleration control).

By employing this function, it becomes possible to shorten the robot's motion time (tact).

The acceleration/deceleration speed during optimum acceleration/deceleration can be calculated using the following equation:

Acceleration/deceleration speed (sec) = Optimum acceleration/deceleration speed (sec) x Accel instruction (%) x M_SetAdl (%)

* The optimum acceleration/deceleration speed is the optimum acceleration/deceleration speed calculated when an Oadl instruction is used.

[Format]

Oadl[]<On / Off>

[Terminology]

<On / Off>

ON : Start the optimum acceleration/deceleration speed.

OFF : End the optimum acceleration/deceleration speed.

[Reference Program]

1 Oadl On	' Move with maximum load.
2 Mov P1	' Set hand 1 and workpiece 1.
3 LoadSet 1ÅC1	' Move with hand 1 + workpiece 1 load.
4 Mov P2	'
5 HOpen 1	'
6 Mov P3	' Move with hand 1 load.
7 HClose 1	'
8 Mov P4	' Move with hand 1 + workpiece 1 load.
9 Oadl Off	

*When parameter HNDHOLD1 is set to 0, 1

[Explanation]

- (1) The robot moves with the optimum acceleration/deceleration according to the hand conditions and workpiece conditions designated with the LoadSet command.
- (2) The workpiece grasp/not grasp for when the hand is opened or closed is set with parameter HNDHOLD 1 to 8.
- (3) Initial setting of Oadl can be changed by the ACCMODE parameter. (Refer to [Page 355, "Table 5-2: List Signal parameter"](#))
- (4) Once Oadl is On, it is valid until Oadl Off is executed or until the program End is executed.
- (5) Depending on the conditions of the hand and/or workpiece, the motion time may become longer than usual.
- (6) It is possible to perform the optimum acceleration/deceleration operation by using the LoadSet and Oadl instructions, and by setting the HNDDAT1(0) through 8 and WRKDAT1(0) through 8 parameters to appropriate values. (Refer to [Page 389, "5.18 Hand and Workpiece Conditions \(optimum acceleration/deceleration settings\)"](#))
- (7) The value of the acceleration/deceleration speed distribution rate in units of axes are predetermined by the JADL parameter. This value varies with models in the S series. Refer to the JADL parameter.

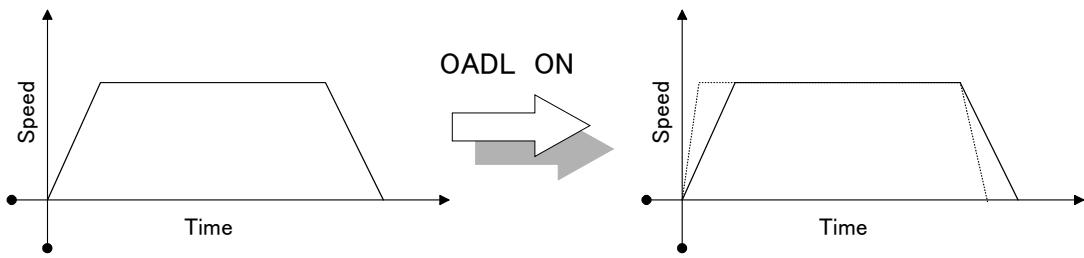


Fig.4-17:Acceleration/deceleration pattern at light load

[Related instructions]

[Accel \(Accelerate\)](#), [Loadset \(Load Set\)](#), [HOpen / HClose \(Hand Open/Hand Close\)](#)

[Related parameter]

HNDDAT 0 to 8, WRKDAT 0 to 8, HNDHOLD 1 to 8, ACCMODE, JADL

On Com GoSub (ON Communication Go Subroutine)

[Function]

Defines the starting line of a branching subroutine when an interrupt is generated from a designated communication line.

[Format]

On[]Com[][(<File No.>)][]GoSub[]<Call Destination>

[Terminology]

<File No.> Describe a number between 1 and 3 assigned to the communication line.

<Call Destination> Describe the line No. and label name.

[Reference Program]

If an interrupt is generated from the file No. 1 communication line (COM1:), carry out the label RCV process.

```

1 Open "COM1:" AS #1      ' Communication line opening.
2 On Com(1) GoSub *RCV   ' The definition of interruption.
3 Com(1) On               ' Enable interrupt from file No. 1 communication line.
4
:
10 ' <<If the communicative interrupt occurs here, it will branch to label *RCV.>>
11 '
12 Mov P1
13 Com(1) Stop           ' Suspend the interrupt during movement only from P1 to P2.
14 Mov P2
15 Com(1) On               ' If there are some communications during movement from P1 to P2, the
                           ' interrupt occurs here.
16
:
22 ' <<If the communicative interrupt occurs here, it will branch to label *RCV.>>
23 '
24 Com(1) Off             ' Disable interrupt from file No. 1 communication line.
25 Close #1
26 End
:
:
30 *RCV                  ' Communication interruption processing.
31 Input #1, M0001         ' Set the received information as M0001 and P0001.
32 Input #1, P0001
:
39 Return 1               ' Returns control to the next step of interrupted step.

```

[Explanation]

- (1) If the file No. is omitted, 1 will be used as the file No.
- (2) The file Nos. with the smallest No. have the order of priority for the interrupt.
- (3) If the communicative interrupt occurs while the robot is moving, robots operating within the same slot will stop. It is possible to use Com Stop to stop the interrupt, and prevent the robot from stopping.
- (4) Interrupts are prohibited in the initial state. To enable interrupts, execute the Com On instruction after this instruction.
- (5) Make sure to return from a subroutine using the Return instruction. An error occurs if the GoTo instruction is used to return, because the free memory available for control structure (stack memory) decreases and eventually becomes insufficient.

[Related instructions]

[Com On/Com Off/Com Stop \(Communication ON/OFF/STOP\)](#), [Return \(Return\)](#), [Open \(Open\)](#), [Input \(Input\)](#), [Print \(Print\)](#), [Close \(Close\)](#)

On ... GoSub (ON Go Subroutine)

[Function]

Calls up the subroutine at the step label corresponding to the value.

[Format]

On[]<Terminology>[]GoSub[] [<Expression>] [, [<Call Destination>]] ...

[Terminology]

<Terminology> Designate the step label on the step to branch to with a numeric operation expression.
 <Call Destination> Describe the step label No. The maximum number is 32.

[Reference Program]

Sets the value equivalent to three bits of input signal 16 in M1, and branches according to the value of M1 (1 through 7).
 (Calls line 1000 if M1 is 1, label LSUB if M1 is 2, line 2000 if M1 is 3, 4 or 5, and label L67 if M1 is 6 or 7.)
 1 M1 = M_Inb(16) AND &H7
 2 On M1 GoSub *L1,*LSUB,*L2,*L2,*L2,*L67,*L67
 :
 10 *L1
 11 ' Describes processing when M1=1.
 12 '
 13 Return ' Be sure to return by using Return.
 20 *LSUB
 21 ' Describes processing when M1=2.
 22 Return ' Be sure to return by using Return.
 30 *L67
 31 ' Describes processing when M1=6 or M1=7.
 32 Return ' Be sure to return by using Return.
 40 *L2
 41 ' Describes processing when M1=3, M1=4, or M1=5.
 42 '
 43 Return ' Be sure to return by using Return.

[Explanation]

- (1) The value of <Expression> determines which step label subroutine to call.
 For example, if the value of <Expression> is 2, the step label described for the second value is called.
- (2) If the value of <expression> is larger than the number of <destinations called up>, the program control jumps to the next step. For example, the program control jumps to the next step if the value of <expression> is 5 and there are only three <destinations called up>.
- (3) When a step No. or label that is called up does not exist, or when there are two definitions, an error will occur.
- (4) Make sure to return from a subroutine using the Return instruction. An error occurs if the GoTo instruction is used to return, because the free memory available for control structure (stack memory) decreases and eventually becomes insufficient.

Value of <Expression>	Process <Control>
Real number	Value is converted to an integer by rounding it off, and then branching is executed.
When 0, or when the value exceeds the number of step labels	Control proceeds to the next step
Negative number or 32767 is exceeded	Execution error

On ... GoTo (On Go To)

[Function]

Branches to the step with the step label that corresponds to the designated value.

[Format]

On[]<Expression>[]GoTo[] [<Branch Destination>] [, [<Branch Destination>]] ...

[Terminology]

<Expression> Designate the step label on the line to branch to with a numeric operation expression.
 <Call Destination> Describe the step label No. The maximum number is 32.

[Reference Program]

Branches based on the value (1-7) of the numerical variable M1.

(Branches to step 20 if M1 is 1, to label LJMP if M1 is 2, to step 50 if M1 is 3, 4 or 5, and to label L67 if M1 is 6 or 7.)

10 On M1 GoTo L1,*LJMP,*L2,*L2,*L2,*L67,*L67

11 ' Control is passed to this line when M1 is other than 1 through 7 (i.e., 0, or 8 or larger).

20 *L1

21 ' Describes processing when M1=1.

22 ' :

30 *LJMP ' When M1=2.

31 ' Describes processing when M1=2.

32 ' :

40 *L67

41 ' Describes processing when M1=6 or M1=7.

42 ' :

50 *L2

51 ' Describes processing when M1=3, M1=4, or M1=5.

52 ' :

[Explanation]

- (1) This is the GoTo version of On GoSub.
- (2) If the value of <expression> is larger than the number of <destinations called up>, the program control jumps to the next step. For example, the program control jumps to the next step if the value of <expression> is 5 and there are only three <destinations called up>.
- (3) When a step No. or label that is called up does not exist, or when there are two definitions, an error will occur.

Value of <Expression>	Process <Control>
Real number	Value is converted to an integer by rounding it off, and then branching is executed.
When 0, or when the value exceeds the number of step labels	Control proceeds to the next step
Negative number or 32767 is exceeded	Execution error

Open (Open)

[Function]

Open the file or communication lines.

[Format]

```
Open[] "<File Descriptor>" [] [For <Mode>] [] AS[] [#] <File No.>
```

[Terminology]

<File Descriptor> Describe a file name (including communication lines).
 *To use a communication line, set "<Communication Line File Name>:
 *When not using a communications line, set "<File Name>"

File type	File name	Access method
File	Describe with 16 characters or less.	Input,OUTPUT,APPEND
Communication line	COM1: Standard RS-232C(default value) COM2:The setting in the "COMDEV" parameter. : COM8:The setting in the "COMDEV" parameter.	Omitted = random mode only
	ENET:192.168.0.2 ^{Note 1)}	Mxt command

Note 1) It is specification in the case of using the real-time external control by the Ethernet interface. Specify the IP address which takes absolute position data by the "Mxt" command following "ENET:.."

<Mode> Designate the method to access a file.

*Omitted = random mode. This can be omitted when using a communication line.

*ÆInput = input mode. Inputs from an existing file.

*OUTPUT = output mode (new file). Creates a new file and outputs it there.

*APPEND = Output mode (existing file). Appends output to the end of an existing file.

<File No.> Specify a constant from 1 to 8.

To interrupt from communication line: 1 to 3.

[Reference Program]

(1) Communication line.

```
1 Open "COM1:" AS #1      ' Open standard RS-232C line as file No. 1.20 Mov P_01
2 Mov P_01
3 Print #1,P_Curr        ' Output current position to external source.
40Input #1,M1,M2,M3      ' Receive from external source with "101.00,202.00,303.00" Ascll for-
                          mat.
5 P_01.X=M1
6 P_01.Y=M2
7 P_01.C=Rad(M3)         ' Copy to global data.
8 Close                   ' Close all opened files.
9 End
```

(2) File operation. (Create the file "temp.txt" to the controller and write "abc")

1 Open "temp.txt" FOR APPEND AS #1

2 Print #1, "abc"

3 Close #1

[Explanation]

(1) Opens the file specified in <File name> using the file number.

Use this file No. when reading from or writing to the file.

A communication line is handled as a file.

(2) [Related instructions]

[Close \(Close\)](#), [Print \(Print\)](#), [Input \(Input\)](#), [Mxt \(Move External\)](#)

[Related parameter]

COMDEV

Ovrd (Override)

[Function]

This instruction specifies the speed of the robot movement as a value in the range from 1 to 100%. This is the override applied to the entire program.

[Format]

Ovrd[]<Override>

Ovrd[]<Override> [, <Override when moving upward> [, <Override when moving downward>]]

[Terminology]

<Override>	Designate the override with a real number. The default value is 100. Unit: [%] (Recommended range: 0.1 to 100.0) A numeric operation expression can also be described. If 0 or a value over 100 is set, an error will occur.
<Override when moving upward/downward>	Sets the override value when moving upward/downward by the arch motion instruction (Mva).

[Reference Program]

```

1 Ovrd 50
2 Mov P1
3 Mvs P2
4 Ovrd M_NOvrd      ' Set default value.
5 Mov P1
6 Ovrd 30,10,10      ' Sets the override when moving upward/downward by the arch motion
                      instruction to 10.
70 Mva P3,3

```

[Explanation]

- (1) The Ovrd command is valid regardless of the interpolation type.
- (2) The actual override is as follows:
 - *During joint interpolation: Operation panel (T/B) override setting value) x (Program override (Ovrd command)) x (Joint override (JOvrd command)).
 - *During linear interpolation: Operation panel (T/B) override setting value) x (Program override (Ovrd command)) x (Linear designated speed (Spd command)).
- (3) The Ovrd command changes only the program override. 100% is the maximum capacity of the robot. Normally, the system default value (M_NOvrd) is set to 100%. The designated override is the system default value until the Ovrd command is executed in the program.
- (4) Once the Ovrd command has been executed, the designated override is applied until the next Ovrd command is executed, the program End is executed or until the program is reset. The value will return to the default value when the End statement is executed or the program is reset.

[Related instructions]

[JOvrd \(J Override\)](#) (For joint interpolation), [Spd \(Speed\)](#) (For linear/circular interpolation)

[Related system variables]

[M_JOvrd/M_NJOvrd/M_OPovrd/M_Ovrd/M_NOvrd](#)
(M_NOvrd (System default value), M_Ovrd (Current designated speed))

Plt (Pallet)

[Function]

Calculates the position of grid in the pallet.

[Format]

```
Plt[]<Pallet No. > , <Grid No. >
```

[Terminology]

<Pallet No. >	Select a pallet No. between 1 and 8 that has already been defined with a Def Plt command.
	Specify this argument using a constant or a variable.
<Grid No. >	The position number to calculate in the palette. Specify this argument using a constant or a variable.

[Reference Program]

```
10 Def Plt 1,P1,P2,P3,P4,4,3,1  ' The definition of the four-point pallet. (P1,P2,P3,P4)
11
12 M1=1                         ' Initialize the counter M1.
13 *LOOP
14 Mov PICK, 50                  ' Moves 50 mm above the work unload position.
15 Ovrd 50
16 Mvs PICK
17 HClose 1                      ' Close the hand.
18 Dly 0.5                        ' Wait for the hand to close securely (0.5 sec.)
19 Ovrd 100
20 Mvs,50                         ' Moves 50 mm above the current position.
21 PLACE = Plt 1, M1              ' Calculates the M1th position
22 Mov PLACE, 50                  ' Moves 50 mm above the pallet top mount position.
23 Ovrd 50
24 Mvs PLACE
25 HOpen 1                        ' Open the hand.
26 Dly 0.5
27 Ovrd 100
28 Mvs,50                         ' Moves 50 mm above the current position.
29 M1=M1+1                        ' Add the counter.
30 If M1 <=12 Then *LOOP        ' If the counter is within the limits, repeats from *LOOP.
31 Mov PICK,50
32 End
```

[Explanation]

- (1) The position of grid of a pallet defined by the Def Plt statement is operated.
- (2) The pallet Nos. are from 1 to 8, and up to 8 can be defined at once.
- (3) Note that the position of the grid may vary because of the designated direction in the pallet definition.
- (4) If a grid No. is designated that exceeds the largest grid No. defined in the pallet definition statement, an error will occur during execution.
- (5) When using the pallet grid point as the target position of the movement command, an error will occur if the point is not enclosed in parentheses as shown above. Refer to [Page 98, "4.1.2 Pallet operation"](#) for detail.

[Related instructions]

[Def Plt \(Define pallet\)](#)

Prec (Precision)

[Function]

This instruction is used to improve the motion path tracking. It switches between enabling and disabling the high accuracy mode.

Note) The available robot types for this instruction are limited. Refer to "[Available robot type]".

[Format].

Prec[]<On / Off>

[Terminology]

<On / Off>

On : When enabling the high accuracy mode.

Off : When disabling the high accuracy mode.

[Reference Program]

1 Prec On	' Enables the high accuracy mode.
2 Mvs P1	
3 Mvs P2	
4 Prec Off	' Disables the high accuracy mode.
5 Mov P1	

[Explanation]

- (1) The high accuracy mode is enabled using the Prec On instruction if it is desired to perform interpolation movement with increased path accuracy.
- (2) When this instruction is used, the path accuracy is improved but the program execution time (tact time) may become longer because the acceleration/deceleration times are changed internally.
- (3) The enabling/disabling of the high accuracy mode is activated from the first interpolation instruction after the execution of this instruction.
- (4) The high accuracy mode is disabled if the Prec Off or End instruction is executed, or a program reset operation is performed.
- (5) The high accuracy mode is disabled immediately after turning the power on.
- (6) The high accuracy mode is always disabled in jog movement.

[Available robot type]

RV-6SD/6SDL/12SD/12SDL series

Print (Print)

[Function]

Outputs data into a file (including communication lines). All data uses the Ascii format.

[Format]

```
Print[]#<File No.>[] [, [<Expression> ; ] ...[<Expression>[ ; ]]]
```

[Terminology]

<File No.> Described with numbers 1 to 8.

Corresponds to the control No. assigned by the Open command.

<Expression> Describes numeric operation expressions, position operation expressions and character string expressions.

[Reference Program]

1 Open "COM1" AS #1	' Open standard RS-232-C line as file No. 1.20 Mov P_01.
2 MDATA=150	' Substitute 150 for the numeric variable MDATA.
3 Print #1,"***Print TEST***"	' Outputs the character string "***Print TEST***".
4 Print #1	' Issue a carriage return
5 Print #1,"MDATA=",MDATA	' Output the character string "MDATA" followed by the value of MDATA, (150).
6 Print #1	' Issue a carriage return.
4 Print #1,"*****"	' Outputs the character string "*****".
5 End	' End the program.

The output result is shown below.

```
***Print TEST***
```

```
MDATA=150
```

```
*****
```

[Explanation]

(1) If <Expression> is not described, then a carriage return will be output.

(2) Output format of data (reference)

The output space for the value for <Expression> and for the character string is in units of 14 characters.

When outputting multiple values, use a comma between each <Expression> as a delimiter.

If a semicolon (;) is used at the head of each space unit, it will output after the item that was last displayed. The carriage return code will always be returned after the output data.

(3) The error occurs when Open command is not executed.

(4) If data contains a double quotation mark ("), only up to the double quotation mark is output.

Example)

```
[1 M1=123.5
```

```
2 P1=(130.5,-117.2,55.1,16.2,0.0,0.0)(1,0) ]
```

1)[3 Print #1,"OUTPUT TEST",M1,P1]is described,

OUTPUT TEST 123.5 (130.5,-117.2,55.1,16.2,0.0,0.0)(1,0) is output.

2)[3 Print #1,"OUTPUT TEST";M1;P1]is described,

OUTPUT TEST 123.5(130.5,-117.2,55.1,16.2,0.0,0.0)(1,0) is output.

If a comma or semicolon is inserted after a <Expression>, the carriage return will not be issued, and instead, printing will continue on the same line.

3) 3 Print #1,"OUTPUT TEST",

4 Print #1,M1;

5 Print #1,P1]is described,

OUTPUT TEST 123.5(130.5,-117.2,55.1,16.2,0.0,0.0)(1,0) is output.

[Related instructions]

[Open \(Open\)](#), [Close \(Close\)](#), [Input \(Input\)](#)

Priority (Priority)

[Function]

In multitask program operation, multiple program lines are executed in sequence (one by one line according to the default setting). This instruction specifies the priority (number of lines executed in priority) when programs are executed in multitask operation.

[Format].

Priority[]<Number of executed lines> [, <Slot number>]

[Terminology]

<Number of executed lines>	Specify the number of lines executed at once . Use a numerical value from 1 to 31.
<Slot number>	1 to 32. If this argument is omitted, the current slot number is set.

[Reference Program]

Slot 1	
1 Priority 3	' Sets the number of executed steps for the current slot to 3.

Slot 2	
1 Priority 4	' Sets the number of executed steps for this slot to 4.

[Explanation]

- (1) Programs of other slots are not executed until the specified number of steps is executed. For example, as in the statement example above, if Priority 3 is set for slot 1's program and Priority 4 is set for slot 2's program, three steps of the slot 1 program are executed first, then four steps of the slot 2 program are executed. Afterward, this cycle is repeated.
- (2) The default value is 1 for all the slots. In other words, the execution moves to the next slot every time one step has been executed.
- (3) An error occurs if there is no program corresponding to the specified task slot.
- (4) It is possible to change the priority even while the program of the specified task slot is being executed.

ReLM (Release Mechanism)

[Function]

This instruction is used in connection with control of a mechanism via task slots during multitask operation. It is used to release the mechanism obtained by the GetM instruction.

[Format]

```
ReLM
```

[Reference Program]

(1) Start the task slot 2 from the task slot 1, and control the mechanism 1 in the task slot 2.

Task slot 1

```
1 ReLM
```

' Releases the mechanism in order to control mechanism 1 using slot 2.

```
2 XRun 2,"10"
```

' Start the program 10 in slot 2.

```
3 Wait M_Run(2)=1
```

' Wait for the starting confirmation of the slot 2.

```
:
```

Task slot 2. (Program "10")

```
1 GetM 1
```

' Get the control of mechanism 1.

```
2 Servo On
```

' Turn on the servo of mechanism 1.

```
3 Mov P1
```

```
4 Mvs P2
```

```
5 Servo Off
```

' Turn off the servo of mechanism 1.

```
6 ReLM
```

' Releases the control right of mechanism 1.

```
7 End
```

[Explanation]

(1) Releases the currently acquired mechanism resource.

(2) If an interrupt is applied while the mechanism is acquired and the program execution is stopped, the acquired mechanism resource will be automatically released.

(3) This instruction cannot be used in a constantly executed program.

[Related instructions]

[GetM \(Get Mechanism\)](#)

Rem (Remarks)

[Function]

Uses the following character strings as comments.

[Format]

```
Rem[] [<Comment>]
```

[Terminology]

<Comment> Describe a user-selected character string.
 Descriptions can be made in the range of position steps.

[Reference Program]

```
1 Rem ***MAIN PROGRAM***  
2 ' ***MAIN PROGRAM***  
3 Mov P1                      ' Move to P1.
```

[Explanation]

- (1) Rem can be abbreviated to be a single quotation mark (').
- (2) It can be described after the instruction like an 30 step in reference program.

Reset Err (Reset Error)

[Function]

This instruction resets an error generated in the robot controller. It is not allowed to use this instruction in the initial status. If an error other than warnings occurs, normal programs other than constantly executed programs cannot be operated. This instruction is effective if used in constantly executed programs.

[Format].

Reset Err

[Reference Program]

Example of execution in a constantly executed program

1 If M_Err=1 Then Reset Err 'Resets an error when an error occurs in the controller.

[Explanation]

- (1) This instruction is used in a program whose start condition is set to constant execution (ALWAYS) by the "SLT*" parameter when it is desired to reset system errors of the robot.
 - (2) It becomes enabled when the controller's power is turned on again after changing the value of the "ALWENA" parameter from 0 to 7.

[Related parameter]

ALWENA

[Related system variables]

M_Err/M_ErrLvl/M_Errno

Return (Return)

[Function]

- (1) When returning from a normal subroutine returns to the next step after the GoSub.
- (2) When returning from an interrupt processing subroutine, returns either to the step where the interrupt was generated, or to the next step.

[Format]

- (1) When returning from a normal subroutine:

```
Return
```

- (2) When returning from an interrupt processing subroutine:

```
Return <Return Designation No.>
```

[Terminology]

<Return Designation No.> Designate the step number where control will return to after an interrupt has been generated and processed.

0 ... Return control to the line where the interrupt was generated.
 1 ... Return control to the next line after the line where the interrupt was issued.

[Reference Program]

- (1) The example of Return from the usual subroutine .

```
1 '***MAIN PROGRAM***  

2 GoSub *SUB_INIT           ' Subroutine jumps to label SUB_INIT.  

3 Mov P1  

:  

100 '***SUB INIT***        ' Subroutine  

11 *SUB_INIT  

12 PSTART=P1  

13 M100=123  

14 Return                  ' Returns to the step immediately following the step where the subroutine was called from.
```

- (2) The example of Return from the subroutine for interruption processing. Calls the subroutine on step 10 when the input signal of general-purpose input signal number 17 is turned on.

```
1 Def Act 1,M_In(17)=1 GoSub *SUB1 ' Definition of interrupt of Act 1.  

2 Act 1=1                      ' Enable the Act 1.  

:  

10 *SUB1                       ' The subroutine for interrupt of Act 1.  

11 Act 1=0                      ' Disable the interrupt.  

12 M_Timer(1)=0                 ' Set the timer to zero.  

13 Mov P2                        ' Move to P2.  

14 Wait M_In(17)=0               ' Wait until the input signal 17 turns off.  

15 Act 1=1                      ' Set up interrupt again.  

16 Return 0                      ' Returns control to the interrupted step.
```

[Explanation]

- (1) Writes the Return instruction at the end of the jump destination processing called up by the GoSub instruction.
- (2) An error occurs if the Return instruction is executed without being called by the GoSub instruction.
- (3) Always use the Return instruction to return from a subroutine when called by the GoSub instruction. An error occurs if the GoTo instruction is used to return, because the free memory available for control structure (stack memory) decreases and eventually becomes insufficient.
- (4) When there is a Return command in a normal subroutine with a return-to designation number, and when there is a Return command in an interrupt-processing subroutine with no return-to destination number, an error will occur.
- (5) when returning from interruption processing to the next step by Return1, execute the statement to disable the interrupt. When that is not so, if interruption conditions have been satisfied, because interruption processing will be executed again and it will return to the next step, the step may be skipped.
Please refer to [Page 180, "Def Act \(Define act\)"](#) for the interrupt processing.

[Related instructions]

[GoSub \(Return\)\(Go Subroutine\)](#), [On ... GoSub \(ON Go Subroutine\)](#), [On Com GoSub \(ON Communication Go Subroutine\)](#), [Def Act \(Define act\)](#)

Select Case (Select Case)

[Function]

Executes one of multiple statement blocks according to the condition expression value.

[Format]

```
Select[] <Condition>
  Case[]<Expression>
    [<Process>]
    Break
  Case[]<Expression>
    [<Process>]
    Break
  .
  Default
    [<Process>]
    Break
End[]Select
```

[Terminology]

<Condition>	Describe a numeric operation expression.
<Expression>	Describe an expression using the following format. The type must be the same as the condition expression.
*IS <Comparison operator> <Constant>	
*<Constant>	
*<Constant> TO <Constant>	
<Process>	Writes any instruction (other than the GoTo instruction) provided by MELFA-BASIC V.

[Reference Program]

```
1 Select MCNT
2 M1=10          ' This line is not executed
3 Case Is <= 10  ' MCNT <= 10
4 Mov P1
5 Break
6 Case 11        'MCNT=11 OR MCNT=12
7 Case 12
8 Mov P2
9 Case 13 TO 18  '13 <= MCNT <= 18
10 Mov P4
11 Break
12 Default        ' Other than the above.
13 M_Out(10)=1
14 Break
15 End Select
```

[Explanation]

- (1) If the condition matches one of the Case items, the process will be executed until the next Case, Default or EndSelect. If the case does not match with any of the Case items but Default is described, that block will be executed.
- (2) If there is no Default, the program will jump to the step after EndSelect without processing.
- (3) The Select Case and End Select statements must always correspond. If a GoTo instruction forces the program to jump out from a Case block of the Select Case statement, the free memory available for control structure (stack memory) decreases. Thus, if a program is executed continuously, an error will eventually occur.
- (4) If an End Select statement that does not correspond to Select Case is executed, an execution error will occur.
- (5) It is possible to write While-WEnd and For-Next within a Case block.
- (6) Use "Case IS", when using the comparison operators (<, =, >, etc.) for the "<Expression>".

Servo (Servo)

[Function]

Controls the ON and OFF of the servo motor power.

[Format]

(1) The usual program

Servo[]<On / Off>

(2) The program of always (ALWAYS) execution.

Servo[]<On / Off> , <Mechanism No.>

[Terminology]

<On / Off>

On : When turning the servo motor power on.

Off : When turning the servo motor power off.

<Mechanism No.>

This is valid only within the program of always execution.

The range of the value is 1 to 3, and describe by constant or variable.

[Reference Program]

1 Servo On	' Servo On.
2 *L2	
3 IF M_Svo<>1 GoTo *L2	' Wait for servo On.
4 Spd M_NSpd	
5 Mov P1	
6 Servo Off	

[Explanation]

- (1) The robot arm controls the servo power for all axes.
- (2) If additional axes are attached, the servo power supply for the additional axes is also affected.
- (3) If used in a program that is executed constantly, this instruction is enabled by changing the value of the "ALWENA" parameter from 0 to 7 and then turning the controller's power on again.

[Related system variables]

[M_Svo](#) (1 : On, 0 : Off)

[Related parameter]

ALWENA

Skip (Skip)

[Function]

Transfers control of the program to the next step.

[Format]

```
Skip
```

[Reference Program]

```
1 Mov P1 Wthlf M_In(17)=1,Skip
```

' If the input signal (M_In(1 7)) turns ON while moving with joint interpolation to the position indicated with position variable P1, stop the robot interpolation motion, and stop execution of this command, and execute the next step.

```
2 If M_SkipCq=1 Then Hlt
```

' Pauses the program if the execution is skipped.

[Explanation]

- (1) This command is described with the WHT or Wthlf statements. In this case, the execution of that step is interrupted, and control is automatically transferred to the next step. Execution of skip can be seen with the M_SkipCq information.

[Related system variables]

[M_SetAdl](#) (1: Skipped, 0: Not skipped)

Spd (Speed)

[Function]

Designates the speed for the robot's linear and circular movements. This instruction also specifies the optimum speed control mode.

[Format]

```
Spd[]<Designated Speed
Spd[]M_NSpd (Optimum speed control mode)
```

[Terminology]

<Designated Speed> Designate the speed as a real number. Unit: [mm/s]

[Reference Program]

```
1 Spd 100
2 Mvs P1
3 Spd M_NSpd      ' Set the default value.(The optimal speed-control mode .)
4 Mov P2
5 Mov P3
6 Ovrd 80          ' Countermeasure against an excessive speed error in the optimal speed mode
7 Mov P4
8 Ovrd 100
```

[Explanation]

- (1) The Spd command is valid only for the robot's linear and circular movements.
- (2) The actual designated override is (Operation panel (T/B) override setting value) x (Program override (Ovrd command)) x (Linear designated speed (Spd command)).
- (3) The Spd command changes only the linear/circular designated speed.
- (4) When M_NSpd (The default value is 10000) is designated for the designated speed, the robot will always move at the maximum possible speed, so the line speed will not be constant(optimum speed control).
- (5) An error may occur depending on the posture of the robot despite of the optimal speed control. If an excessive speed error occurs, insert an Ovrd instruction in front of the error causing operation instruction in order to lower the speed only in that segment.
- (6) The system default value is applied for the designated speed until the Spd command is executed in the program. Once the Spd command is executed, that designated speed is held until the next Spd command.
- (7) The designated speed will return to the system default value when the program End statement is executed.

[Related system variables]

[M_Spd/M_NSpd/M_RSpd](#)

Title (Title)

[Function]

Appends the title to the program. The characters specified in the program list display field of the robot controller can be displayed using the separately sold personal computer support software.

[Format]

```
Title[]"<Character String>"
```

[Terminology]

<Character String> Message for title

[Reference Program]

- 1 Title "ROBOT Loader program"
- 2 Mov P1
- 3 Mvs P2

[Explanation]

- (1) Although characters can be registered up to the maximum allowed for each step in the program, only a maximum of 20 characters can be displayed in the program list display field of the robot controller using the personal computer support software.

Tool(Tool)

[Function]

Designates the tool conversion data. This instruction specifies the length, position of the control point from the mechanical interface, and posture of the tools (hands).

[Format]

Tool[]<Tool Conversion Data>

[Terminology]

<Tool Conversion Data> Specifies the tool conversion data using the position operation expression. (Position constants, position variables, etc.)

[Reference Program]

(1) Set up the direct numerical value.

1 Tool (100,0,100,0,0,0)

' Changes the control position to an X-axis coordinate value of 100 mm and a Z-axis coordinate value of 100 mm in the tool coordinate system.

2 Mvs P1

3 Tool P_NTool

' Returns the control position to the initial value. (mechanical interface position, flange plane.)

(2) Set up the position variable data in the XYZ coordinates system.

(If (100,0,100,0,0,0,0) are set in PTL01, it will have the same meaning as (1).)

1 Tool PTL01

2 Mvs P1

[Explanation]

(1) The Tool instruction is used to specify the control points at the tip of each hand in a system using double hands. If both hands are of the same type, the control point should be set by the "MEXTL" parameter instead of by the Tool instruction.

(2) The tool conversion data changed with the Tool command is saved in parameter MEXTL, and is saved even after the controller power is turned OFF.

(3) The system default value (P_NTool) is applied until the Tool command is executed.

Once the Tool command is executed, the designated tool conversion data is applied until the next Tool command is executed. This is operated with 6-axis three-dimension regardless of the mechanism structure.

(4) If different tool conversion data are used at teaching and automatic operation, the robot may move to an unexpected position. Make sure that the settings at operation and teaching match.

The valid axis element of tool conversion data is different depending on the type of robot.

Set up the appropriate data referring to the [Page 370, "Table 5-8: Valid axis elements of the tool conversion data depending on the robot model"](#).

(5) Using the M_Tool variable, it is possible to set the MEXTL1 to 4 parameters as tool data.

[Related parameter]

MEXTL, MEXTL 1 to 4 Refer to [Page 369, "5.6 Standard Tool Coordinates"](#) for detail.

[Related system variables]

[P_Tool/P_NTool](#), [M_Tool](#)

Torg (Torque)

[Function]

Designates the torque limit for each axis. By specifying the torque limit, an excessive load (overload) on works and so forth can be avoided. An excessive error is generated if the torque limit value ratio is exceeded.

[Format]

```
Torq[]<Axis No.>, <Torque Limitation Rate>
```

[Terminology]

<Axis No. > Designate the axis No. with a numeric constant. (1 to 6)
<Torque Limitation Rate> Designate the limit of the force generated from the axis as a percentage. (1 to 100)

[Reference Program]

```
1 Def Act 1,M_Fbd>10 GoTo *SUB1,S
2 Act 1=1
3 Torq 3,10
4 Mvs P1
5 Mov P2
:
10 *SUB1
11 Mov P_Fbc
12 M_Out(10)=1
13 Hlt
```

' Generate an interrupt when the difference between the command position and the feedback position reaches 10 mm or more.
' Enable the interrupt 1
' Set the torque limit of the three axes to 10% of the normal torque using the torque instruction.
' Moves
' Align the command position with the feedback position.
' Signal No. 10 output
' Stop when a difference occurs.

[Explanation]

- (1) Restrict the torque limit value of the specified axis so that a torque exceeding the specified torque value will not be applied during operation. Specify the ratio relative to the standard torque limit value. The standard torque limit value is predefined by the manufacturer.
- (2) The available rate of torque limitation is changed by robot type. The setting is made for each servo motor axis; thus, it may not be the torque limit ratio at the control point of the tip of the actual robot. Try various ratios accordingly.
- (3) If the robot is stopped while still applying the torque limit, it may stop at the position where the command position and the feedback position deviate (due to friction, etc.). In such a case, an excessive error may occur when resuming the operation. To avoid this, program so as to move to the feedback position before resuming the operation, as shown on the 10th step of the above example.
- (4) This instruction is valid only for standard robot axes. It cannot be used for general-purpose servo axes (additional axes and user-defined mechanisms). Change the parameters on the general-purpose servo side to obtain similar movement.

[Related system variables]

P_Fbc, M_Fbd

Wait (Wait)

[Function]

Waits for the variable to reach the designated value.

[Format]

Wait[]<Numeric variable>=<Numeric constant>

[Terminology]

<Numeric variable> Designate a numeric variable. Use the input/output signal variable (in such cases as M_In, M_Out) well.
<Numeric constant> Designate a numeric constant.

[Reference Program]

(1) Signal state

(2) Task slot state

3 Wait M_Run(2)=1

(3) Variable state

4 Wait M_01=100

[Explanation]

- (1) This command is used as the interlock during signal input wait and during multitask execution.
 - (2) The Wait instruction allows the program control to continue to the next step once the specified condition is met.
 - (3) In case the Wait instruction is executed in several tasks at one time in the multitask execution status, the processing time (tact time) may become longer and affect the system. In such cases, use the If-Then instruction instead of the Wait instruction.

Example) 50 Wait M_ABC=0 4 *L5
5 If M_ABC<>0 Then GoTo *L5

While-WEnd (While End)

[Function]

The program between the While statement and WEnd statement is repeated until the loop conditions are satisfied.

[Format]

```
While[]<Loop Condition>
:
WEnd
```

[Terminology]

<Loop Condition> Describe a numeric operation expression. (Refer to the syntax diagram)

[Reference Program]

Repeat the process while the numeric variable M1 value is between -5 and +5, and transfer control to step after WEnd statement if range is exceeded.

1 While (M1>=-5) AND (M1<=5)	' Repeat the process while the value of numeric variable M1 is between -5 and +5.
2 M1=-(M1+1)	' Add 1 to M1, and reverse the sign.
3 Print# 1, M1	' Output the M1 value.
4 WEnd	' Return to the While statement (step 1)
5 End	' End the program.

[Explanation]

- (1) The program between the While statement and WEnd statement is repeated.
- (2) If the result of <Expression> is true (not 0), the control moves to the step following the While statement and the process is repeated.
- (3) If the result of <Expression> is false (is 0), then the control moves to the step following the WEnd statement.
- (4) If a GoTo instruction forces the program to jump out from between a While statement and a WEnd statement, the free memory available for control structure (stack memory) decreases. Thus, if a program is executed continuously, an error will eventually occur. Write a program in such a way that the loop exits when the condition of the While statement is met.

Wth (With)

[Function]

A process is added to the interpolation motion.

[Format]

Example) MOV P1 Wth[]<Process>

[Terminology]

<Process>

Describe the process to be added. The commands that can be described are as follow.

1. <Numeric type data B> <Substitution operator><Numeric type data A> [Substitute, signal modifier command (refer to syntax diagram)]

[Reference Program]

1 Mov P1 Wth M_Out(17)=1 Dly M1+2

' Simultaneously with the start of movement to P1, the output signal No. 17 will turn ON for the value indicated with the numeric variable M1 + two seconds.

[Explanation]

- (1) This command can only be used to describe the additional condition for the movement command.
- (2) An error will occur if the Wth command is used alone.
- (3) The process will be executed simultaneously with the start of movement.
- (4) The relationship between the interrupts regarding the priority order is shown below.
Com > Act > WthIf(Wth) > Pulse substitution

WthIf (With If)

[Function]

A process is conditionally added to the interpolation motion command.

[Format]

```
WthIf[]<Additional Condition>, <Process>
```

[Terminology]

<Additional Condition>	Describe the condition for adding the process. (Same as Act condition expression)
<Process>	Describe the process to be added when the additional conditions are established. (Same as Wth)
	The commands that can be described as a process are as follow. (Refer to syntax diagram.)
1.	<Numeric type data B> <Substitution operator><Numeric type data A> Example) M_Out(1)=1, P1=P2
2.	Hlt statement
3.	Skip statement

[Reference Program]

(1) If the input signal 17 turns on, the robot will stop.

```
1 Mov P1 WthIf M_In(17)=1, Hlt
```

(2) If the current command speed exceeds 200 mm/s, turn on the output signal 17 for the M1+2 seconds.

```
2 Mvs P2 WthIf M_RSpd>200, M_Out(17)=1 Dly M1+2
```

(3) If the rate of arrival exceeds 15% during movement to P3, turn on the output signal 1.

```
3 Mvs P3 WthIf M_Ratio>15, M_Out(1)=1
```

[Explanation]

- (1) This command can only be used to describe the additional conditions to the movement command.
- (2) Monitoring of the condition will start simultaneously with the start of movement.
- (3) It is not allowed to write the Dly instruction at the processing part.
- (4) If the robot is stopped using the Hlt instruction, it decelerates and stops in the same way as for "Stop type 1" of the Def Act instruction.(Refer to [Page 180, "Def Act \(Define act\)"](#))

XClr (X Clear)

[Function]

This instruction cancels the program selection status of the specified task slot from within a program. It is used during multitask operation.

[Format]

```
XClr[]<Slot No.>
```

[Terminology]

<Slot No. > Specify a slot number in the range from 1 to 32 as a constant or variable.

[Reference Program]

```

1 XRun 2,"1"           ' Executes the first program in task slot 2.
:
10 XStp 2              ' Pauses the program of task slot 2.
11 Wait M_Wai(2)=1     ' Waits until the program of task slot 2 pauses.
12 XRst 2              ' Cancels the pause status of the program of task slot 2.
:
20 XClr 2              ' Cancels the program selection status of task slot 2.
21 End

```

[Explanation]

- (1) An error occurs at execution if the specified slot does not select the program.
- (2) If the designated program is being operating, an error will occur at execution.
- (3) If the designated program is being pausing, an error will occur at execution.
- (4) If this instruction is used within a constantly executed program, it becomes enabled by changing the value of the "ALWENA" parameter from 0 to 7 and turning the controller's power off and on again.

[Related instructions]

[XLoad \(X Load\)](#), [XRst \(X Reset\)](#), [XRun \(X Run\)](#), [XStp \(X Stop\)](#)

[Related parameter]

ALWENA

XLoad (X Load)

[Function]

This instruction commands the specified program to be loaded into the specified task slot from within a program.

It is used during multitask operation.

[Format]

```
XLoad[]<Slot No.> <Program Name>
```

[Terminology]

<Slot No.> Specify a slot number in the range from 1 to 32 as a constant or variable.
<Program Name> Designate the program name.

[Reference Program]

```
1 If M_Psa(2)=0 Then *L1          ' Checks whether slot 2 is in the program selectable state.  
2 XLoad 2,"10"                   ' Select program 10 for slot 2.  
3 *L3  
4 If C_Prg(2)<>"10" Then GoTo *L3  ' Waits for a while until the program is loaded.  
5 XRun 2                         ' Start slot 2.  
6 Wait M_Run(2)=1                 ' Wait to confirm starting of slot 2.  
7 *L1  
8 ' When the slot 2 is already operating, execute from here.
```

[Explanation]

- (1) An error occurs at execution if the specified program does not exist.
- (2) If the designated program is already selected for another slot, an error will occur at execution.
- (3) If the designated program is being edited, an error will occur at execution.
- (4) If the designated program is being executed, an error will occur at execution.
- (5) Designate the program name in double quotations.
- (6) If used in a program that is executed constantly, this instruction is enabled by changing the value of the "ALWENA" parameter from 0 to 7 and then turning the controller's power on again.
- (7) If XRun is executed immediately after executing XLoad, an error may occur while loading a program. If necessary, perform a load completion check as shown on the 3rd step of the statement example.

[Related instructions]

[XClr \(X Clear\)](#), [XRst \(X Reset\)](#), [XRun \(X Run\)](#), [XStop \(X Stop\)](#)

[Related parameter]

ALWENA

XRst (X Reset)

[Function]

This instruction returns the program control to the first step if the program of the specified task slot is paused by a command within the program (program reset). It is used during multitask operation.

[Format]

XRst[]<Slot No.>

[Terminology]

<Slot No.> Specify a slot number in the range from 1 to 32 as a constant or variable.

[Reference Program]

```

1 XRun 2          ' Start.
2 Wait M_Run(2)=1 ' Wait to confirm starting.
:
10 XStp 2         ' Stop.
11 Wait M_Wai(2)=1 ' Wait for stop to complete.
:
15 XRst 2         ' Set program execution start step to head step.
16 Wait M_Psa(2)=1 ' Wait for program reset to complete.
:
20 XRun 2         ' Restart.
21 Wait M_Run(2)=1 ' Wait for restart to complete.

```

[Explanation]

- (1) This is valid only when the slot is in the stopped state.
- (2) If used in a program that is executed constantly, this instruction is enabled by changing the value of the "ALWENA" parameter from 0 to 7 and then turning the controller's power on again.

[Related instructions]

[XClr \(X Clear\)](#), [XLoad \(X Load\)](#), [XRun \(X Run\)](#), [XStp \(X Stop\)](#)

[Related parameter]

ALWENA

[Related system variables]

[M_Psa](#) (Slot number) (1: Program selection is possible, 0: Program selection is impossible)

[M_Run](#) (Slot number) (1: Executing, 0: Not executing)

[M_Wai](#) (Slot number) (1: Stopping, 0: Not stopping)

XRun (X Run)

[Function]

This instruction executes concurrently the specified programs from within a program. It is used during multitask operation.

[Format]

XRun[]<Slot No.> [, "<Program Name>"] [, <Operation Mode>]]

[Terminology]

<Slot No. > Specify a slot number in the range from 1 to 32 as a constant or variable.
<Program Name> Designate the program name.
<Operation Mode> 0 = Continuous operation,
1 = Cycle stop operation. If the operation mode is omitted, the current operation mode will be used. Specify this argument using a constant or a variable.

[Reference Program]

- (1) When the program of execution is specified by XRun command (continuous executing).

1 XRun 2,"1" ' Start the program 1 with slot 2.
2 Wait M_Run(2)=1 ' Wait to have started.

(2) When the program of execution is specified by XRun command (cycle operation)

1 XRun 3,"2",1 ' Start the program 2 with slot 3 in the cycle operation mode
2 Wait M_Run(3)=1 ' Wait to have started.

(3) When the program of execution is specified by XLoad command (continuous executing).

1 XLoad 2, "1" ' Select the program 1 as the slot 2.
2 *L2
3 If C_Prg(2)<>"1" Then GoTo *L3 ' Wait for load complete.
4 XRun 2 ' Start the slot 2.

(4) When the program of execution is specified by XLoad command (cycle operation)

1 XLoad 3, "2" ' Select the program 2 as the slot 3.
2 *L2
3 If C_Prg(2)<>"1" Then GoTo *L2 ' Wait for load complete
4 XRun 3, ,1 ' Start the program 1 with cycle operation.

[Explanation]

- (1) An error occurs at execution if the specified program does not exist.
 - (2) If the designated slot No. is already in use, an error will occur at execution.
 - (3) If a program has not been loaded into a task slot, this instruction will load it. It is thus possible to operate the program without executing the XLoad instruction.
 - (4) If XRun is executed in the "Pausing" state with the program stopped midway, continuous execution will start.
 - (5) Designate the program name in double quotations.
 - (6) If the operation mode is omitted, the current operation mode will be used.
 - (7) If it is used in programs that are constantly executed, change the value from 0 to 7 in the ALWENA parameter, and power ON the controller again.
 - (8) If XRun is executed immediately after executing XLoad, an error may occur while loading a program. If necessary, perform a load completion check as shown on the 3rd step of both statement examples [3] and [4].

[Related instructions]

XClr (X Clear), XLoad (X Load), XRst (X Reset), XStop (X Stop)

Related parameter

ALWENA

[Related system variables]

M_Run (Slot number) (1: Executing, 0: Not executing)

XStp (X Stop)

[Function]

This instruction pauses the execution of the program in the specified task slot from within a program. If the robot is being operated by the program in the specified task slot, the robot stops. It is used in multitask operation.

[Format]

```
XStp[]<Slot No. >
```

[Terminology]

<Slot No. > Specify a slot number in the range from 1 to 32 as a constant or variable.

[Reference Program]

```
1 XRun 2          ' Execute.  
:  
10 XStp 2        ' Stop.  
11 Wait M_WAI(2)=1  ' Wait for stop to complete.  
:  
20 XRun 2        ' Restart.
```

[Explanation]

- (1) If the program is already stopped, an error will not occur.
- (2) XStp can also stop the constant execution attribute program.
- (3) If used in a program that is executed constantly, this instruction is enabled by changing the value of the "ALWENA" parameter from 0 to 7 and then turning the controller's power on again.

[Related instructions]

[XClr \(X Clear\)](#), [XLoad \(X Load\)](#), [XRst \(X Reset\)](#), [XRun \(X Run\)](#)

[Related parameter]

ALWENA

[Related system variables]

[M_Wai](#) (Slot number) (1: Stopping, 0: Not stopping)

Substitute

[Function]

The results of an operation are substituted in a variable or array variable.

[Format]

```
<Variable Name> = <Expression 1>
```

For pulse substitution

```
<Variable Name> = <Expression 1> Dly <Expression 2>
```

[Terminology]

<Variable Name> Designate the variable name of the value is to be substituted.
(Refer to the syntax diagram for the types of variables.)

<Expression 1> Substitution value. Describe an numeric value operation expression.

<Expression 2> Pulse timer. Describe an numeric value operation expression.

[Reference Program]

(1) Substitution of the variable operation result .

1 P100=P1+P2*2

(2) Output of the signal.

2 M_Out(10)=1 ' Turn on the output signal 10.

(3) Pulse output of the signal.

3 M_Out(17)=1 Dly 2.0 ' Turn on the output signal 17 for 2 seconds.

[Explanation]

- (1) When using this additionally for the pulse output, the pulse will be executed in parallel with the execution of the commands on the following steps.
- (2) Be aware that if a pulse is output by M_Outb or M_Outw, the bits are reversed in 8-bit units or 16-bit units, respectively. It is not possible to reverse at any bit widths.
- (3) If the End command or program's last step is executed during the designated time, or if the program execution is stopped due to an emergency stop, etc., the output state will be held. But, the output reversed after the designated time.

(Label)

[Function]

This indicates the jump site.

[Format]

*<Label Name>

*<Label Name> [:<Command line>]

[Terminology]

<Label Name> Describe a character string that starts with an alphabetic character.

Up to 8 characters can be used. (Up to 9 characters including *.)

<Command line> The command line can be described after the colon after the label (:).

[Reference Program]

```
1 *SUB1
2 If M1=1 Then GoTo *SUB1
```

```
3 *LBL1 : IF M_In(19)=0 Then GoTo *LBL1      ' Wait by the 300 lines until the input signal of No. 10 turns
on.
```

[Explanation]

(1) An error will not occur even if this is not referred to during the program.

(2) If the same label is defined several times in the same program, an error will occur at the execution.

(3) The reserved words can't be used for the label.

(4) If the underscore is used for the label name, the 1st character is "L." only. If the characters except "L" are used (ex. *A_LABEL), an error occurs.

Ex.) The correct example of the label with using the underscore. (The 1st character is "L")
 *L_ABC, *L12_345, *LABEL_1

The mistake example of the label with using the underscore.

*H_ABC, *ABC_123, *NG_, *_LABEL

(5) The software J1 or later, the command line can be described after the colon after the label (:). However, after the command line, the colon cannot be described and the command line cannot be described again.

4.12 Detailed explanation of Robot Status Variable

4.12.1 How to Read Described items

[Function]	: This indicates a function of a variable.
[Format]	: This indicates how to enter arguments of an instruction. [] means that arguments may be omitted. System status variables can be used in conditional expressions, as well as in reference and assignment statements. In the format example, only reference and assignment statements are given to make the description simple.
[Reference Program]	: An example program using variables is shown.
[Terminology]	: This indicates the meaning and range of an argument.
[Explanation]	: This indicates detailed functions and precautions.
[Reference]	: This indicates related items.

4.12.2 Explanation of Each Robot Status Variable

Each variable is explained below in alphabetical order.

C_Com

[Function]

Sets the parameters for the line to be opened by the Open instruction. This is used when the communication destination is changed frequently.

* Character string type

* Only for a client with the Ethernet.

[Format]

C_Com (<communication line number>) = "ETH: <server side IP address> [, <port number>]"

[Terminology]

ETH:

An identifier to indicate that the target is an Ethernet

<Communication line number> The number of the COM to be specified by the Open instruction (The line type is assigned by the COMDEV parameter.) Specify 1 through 8.

<Server side IP address>

Server side IP address (May be omitted.)

<Port number>

Port number on the server side (If omitted, the set value of the NETPORT parameter is used.)

[Reference Program]

Example when the Ethernet option is installed in an option slot and OPT12 is set in the second element of the COMDEV parameter

```

1 C_Com(2)="ETH:192.168.0.10,10010" Set the IP address of the communication destination server corresponding to communication line COM2
2 *O1
3 Open "COM2:" AS #1          ' As 192.168.0.10 and the port number as 10010, and then open the line.
4 If M_Open(1)<>1 Then *O1  ' Loops if unable to connect to the server.
5 Print #1, "HELLO"          ' Sends a character string.
6 Input #1, C1$              ' Receives a character string.
7 Close #1                  ' Closes the line.
8 C_Com(2)="ETH:192.168.0.11,10011" Set the IP address of the communication destination server corresponding to communication line COM2
9 *O2
10 Open "COM2:" AS #1         ' As 192.168.0.11 and the port number as 10011, and then open the line.
11 If M_Open(1)<>1 Then *O2  ' Loops if unable to connect to the server.
12 Print #1, C1$              ' Sends a character string.
13 Input #1, C2$              ' Receives a character string.
14 Close #1                  ' Closes the line.
15 Hlt                      ' Halts the program.
16 End                       ' Ends.

```

[Explanation]

- (1) It is not necessary to use this command when the communication counterpart of the robot controller is specified with the NETHSTIP and NETPORT parameters and the specified communication counterpart will not be changed at all.
- (2) Currently, this function is valid only for a client of a data link with the Ethernet option.
- (3) Because the communication parameters of the OPEN instruction are set, it is necessary to execute this command before the OPEN instruction.
- (4) When the power is turned on, the set values specified by the NETHSTIP and NETPORT parameters are used. When this command is executed, the values specified by the parameters of this command are changed temporarily. They are valid until the power is turned off. When the power is turned on again, the values revert to the original values set by the parameters.

- (5) If this command is executed after the OPEN instruction, the current open status will not change. In such a case, it is necessary to close the line with the CLOSE instruction once, and then execute the OPEN instruction again.
- (6) If an incorrect syntax is used, an error occurs when the program is executed, not when the program is edited.

[Related parameter]

NETHSTIP, NETPORT

C_Date

[Function]

This variable returns the current date in the format of year/month/date.

[Format]

Example) <Character String Variable >=C_Date

[Reference Program]

1 C1\$=C_Date ' "2000/12/01" is assigned to C1\$.

[Explanation]

- (1) The current date is assigned.
- (2) This variable only reads the data. Use the T/B to set the date.

[Reference]

C_Time

C_Maker

[Function]

This variable returns information on the manufacturer of the robot controller.

[Format]

Example) <Character String Variable >=C_Maker

[Reference Program]

1 C1\$=C_Maker ' "COPYRIGHT1999....." is assigned to C1\$.

[Explanation]

- (1) This variable returns information on the manufacturer of the robot controller.
- (2) This variable only reads the data.

[Reference]

C_Mecha

C_Mecha

[Function]

This variable returns the name of the mechanism to be used.

[Format]

Example) <Character String Variable >=C_Mecha[(<Mechanism Number>)]

[Terminology]

<Character String Variable > Specify a character string variable to be assigned.

<Mechanism Number> Enter the mechanism number. 1 to 3, If the argument is omitted, 1 is set as the default value.

[Reference Program]

1 C1\$=C_Mecha(1) ' "RV-6SD" is assigned to C1\$. (If the robot type name is RV-6SD)

[Explanation]

(1) This variable returns the name of the mechanism to be used.

(2) This variable only reads the data.

C_Prg

[Function]

This variable returns the selected program number (name).

[Format]

Example) <Character String Variable >=C_Prg [(<Numeric>)]

[Terminology]

<Character String Variable > Specify a character string variable to be assigned.

<Numeric> 1 to 32, Enter the task slot number. If the argument is omitted, 1 is set as the default value.

[Reference Program]

1 C1\$=C_Prg(1) ' "10" is assigned to C1\$. (if the program number is 10.)

[Explanation]

(1) The program number (name) set (loaded) into the specified task slot is assigned.

(2) If this variable is used in single task operation, the task slot number becomes 1.

(3) If it is set in the operation panel, that number is set.

(4) This variable only reads the data.

(5) If a task slot for which a program is not loaded is specified, an error occurs at execution.

C Time

[Function]

This variable returns the current time in the format of time:minute:second (24 hours notation).

[Format]

Example) <Character String Variable >=C_Time

[Reference Program]

1 C1\$=C_Time ' "01/05/20" is assigned to C1\$.

[Explanation]

- (1) The current clock is assigned.
 - (2) This variable only reads the data.
 - (3) Use the T/B to set the time.

[Reference]

c_com

C:\User\

[Function]

This variable returns the data registered in the `USERMSG` parameter.

[Format]

Example) <Character String Variable >=C_User

[Reference Program]

' The characters registered in "USERMSG" are assigned to C1\$.

[Explanation]

- (1) This variable returns the data registered in the "USERMSG" parameter.
 - (2) This variable only reads the data.
 - (3) Use the PC support software or the T/B to change the parameter setting.

J_Curr

[Function]

Returns the joint type data at the current position.

[Format]

Example) <Joint Type Variable>=J_Curr [<Mechanism Number>]

[Terminology]

<Joint Type Variable> Specify a joint type variable to be assigned.

<Mechanism Number> Enter the mechanism number. 1 to 3, If the argument is omitted, 1 is set as the default value.

[Reference Program]

1 J1=J_Curr ' J1 will contain the current joint position.

[Explanation]

- (1) The joint type variable for the current position of the robot specified by the mechanism number will be obtained.
- (2) This variable only reads the data.

[Reference]

P_Curr

J_CoMxI

[Function]

Return the maximum value of the differences between the estimated torque and actual torque while the impact detection function is being enabled.

The impact detection function can only be used in certain models (Refer to "[Available robot type]").

[Format]

Example) <Joint Type Variable>=J_CoMxI [(<Mechanism Number>)]

[Terminology]

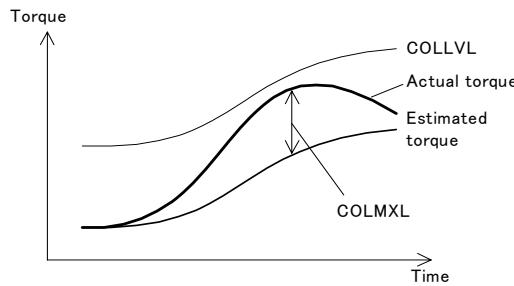
- | | |
|-----------------------|--|
| <Joint Type Variable> | Specify a joint type variable to be assigned.(Joint type variable will be used even if this is a pulse value.) |
| <Mechanism Number> | Enter the mechanism number. 1 to 3, If the argument is omitted, 1 is set as the default value. |

[Reference Program]

```
1 M1=100          'Set the initial value of the allowable impact level of each axis.
2 M2=100
3 M3=100
4 M4=100
5 M5=100
6 M6=100
7 ColLvl M1,M2,M3,M4,M5,M6,,          'Set the allowable impact level of each axis.
8 ColChk On          'Enable the impact detection function.
                      '(Start the calculation of the maximum value of torque error.)
9 Mov P1
:
:
50 ColChk Off          'Disable the impact detection function.
                      '(End the calculation of the maximum value of torque error.)
51 M1=J_CoMxI(1).J1+10          'For each axis, the allowable impact level with a margin of 10% is
                                 calculated.
                                 '(10% is a reference value for the reference program and not an
                                 actual guaranteed value.)
52 M2=J_CoMxI(1).J2+10
53 M3=J_CoMxI(1).J3+10
54 M4=J_CoMxI(1).J4+10
55 M5=J_CoMxI(1).J5+10
56 M6=J_CoMxI(1).J6+10
57 GoTo 70
```

[Explanation]

- (1) Keep the maximum value of the error of the estimated torque and actual torque of each axis while impact detection function is valid.



- (2) When this value is 100%, it indicates that the maximum error value is the same as the manufacturer's initial value of the allowable impact level.
 (3) For robots that prohibit the use of impact detection, 0.0 is always returned for all axes.
 (4) The maximum error value is initialized to 0.0 when the servo is turned ON during the execution of a ColChk ON or COLLVL instruction.
 (5) Because they are joint-type variables, it will be conversion values from rad to deg if they are read as joint variables. Therefore, substitute each axis element by a numeric variable as shown in the syntax example when using these joint-type variables.
 (6) This variable only reads the data.

[Reference]

ColChk (Col Check), ColLvl (Col Level), M_ColSts, P_ColDir

[Available robot type]

RV-6SD/6SDL/12SD/12SDL series

J_ECurr

[Function]

Returns the current encoder pulse value.

[Format]

Example) <Joint Type Variable>=J_ECurr [(<Mechanism Number>)]

[Terminology]

- <Joint Type Variable> Specify a joint type variable to be assigned.
 <Mechanism Number> Enter the mechanism number. 1 to 3, If the argument is omitted, 1 is set as the default value.

[Reference Program]

1 J1=J_ECurr(1)	' JA will contain the encoder pulse value of mechanism 1.
2 MA=JA. 1	' Loads the encoder pulse value of the J1 axis to the MA.

[Explanation]

- (1) Although the value to be returned is a pulse value, use the joint type as the substitution type. Then, specify joint component data, and use by substituting in a numeric variable.
 (2) This variable only reads the data.

J_Fbc/J_AmpFbc

[Function]

J_Fbc:Returns the current position of the joint type that has been generated by encoder feedback.
J_AmpFbc:Returns the current feedback value of each axis

[Format]

Example) <Joint Type Variable>=J_Fbc [(<Mechanism Number>)]

Example) <Joint Type Variable>=J_AmpFbc [(<Mechanism Number>)]

[Terminology]

<Joint Type Variable> Specify a joint type variable to be assigned.
<Mechanism Number> Enter the mechanism number. 1 to 3, If the argument is omitted, 1 is set as the default value.

[Reference Program]

1 J1=J_Fbc ' J1 will contain the current position of the joint that has been generated by servo feedback.
2 J1=J_AmpFbc ' The present current feedback value is entered in J2.

[Explanation]

- (1) J_Fbc returns the present position of the joint type generated by the feedback of the encoder.
- (2) J_Fbc can check the difference between the command value to the servo and the delay in the actual servo.
- (3) J_Fbc can also check if there is a difference as a result of executing a Cmp Jnt instruction.
- (4) This variable only reads the data.

[Reference]

P_Fbc

J_Origin

[Function]

Returns the joint data when the origin has been set.

[Format]

Example) <Joint Type Variable>=J_Origin [(<Mechanism Number>)]

[Terminology]

<Joint Type Variable> Specify a joint type variable to be assigned.
<Mechanism Number> Enter the mechanism number. 1 to 3, If the argument is omitted, 1 is set as the default value.

[Reference Program]

1 J1=J_Origin(1) ' J1 will contain the origin setting position of mechanism 1.

[Explanation]

- (1) Returns the joint data when the origin has been set.
- (2) This can be used to check the origin, for instance, when the position of the robot shifted.
- (3) This variable only reads the data.

M_Acl/M_DAcl/M_NAcl/M_NDAcl/M_AclSts

[Function]

Returns information related to acceleration/deceleration time.

M_Acl : Returns the ratio of current acceleration time. (%)

M_DAcl : Returns the ratio of current deceleration time. (%)

M_NAcl : Returns the initial acceleration time value. (100%)

M_NDAcl : Returns the initial deceleration time value. (100%)

M_AclSts : Returns the current acceleration/deceleration status.

(Current status: 0 = Stopped, 1 = Accelerating, 2 = Constant speed, 3 = Decelerating)

[Format]

Example) <Numeric Variable>=M_Acl [(<Equation>)]

Example) <Numeric Variable>=M_DAcl [(<Equation>)]

Example) <Numeric Variable>=M_NAcl [(<Equation>)]

Example) <Numeric Variable>=M_NDAcl [(<Equation>)]

Example) <Numeric Variable>=M_AclSts [(<Equation>)]

[Terminology]

<Numeric Variable> Specifies the numerical variable to assign.

<Equation> 1 to 32, Enter the task slot number. If this argument is omitted, the current slot will be used as the default.

[Reference Program]

1 M1=M_Acl	' M1 will contain the ratio of acceleration time set for task slot 1.
2 M1=M_DAcl(2)	' M1 will contain the ratio of deceleration time set for task slot 2.
3 M1=M_NAcl	' M1 will contain the ratio of initial acceleration time value set for task slot 1.
4 M1=M_NDAcl(2)	' M1 will contain the ratio of initial deceleration time value set for task slot 2.
5 M1=M_AclSts(3)	' M1 will contain the current acceleration/deceleration status for task slot 3.

[Explanation]

- (1) The ratio of acceleration/deceleration time is the ratio against each robot's maximum acceleration/deceleration time (initial value). If this value is 50%, the amount of time needed to accelerate/decelerate is doubled, resulting in slower acceleration/deceleration.
- (2) M_NAcl and M_NDAcl always return 100 (%).
- (3) This variable only reads the data.

M_BrkCq

[Function]

Returns the result of executing a line containing a BREAK command that was executed last.

1 : BREAK was executed

0 : BREAK was not executed

[Format]

Example) <Numeric Variable>=M_BrkCq [(<Equation>)]

[Terminology]

<Numeric Variable> Specifies the numerical variable to assign.

<Equation> 1 to 32, Enter the task slot number. If this argument is omitted, the current slot will be used as the default.

[Reference Program]

1 While M1<>0

2 If M2=0 Then Break ' The remaining battery capacity time is assigned to M1.

3 WEnd

4 If M_BrkCq=1 Then Hlt ' Hlt, if Break in While is executed.

[Explanation]

(1) Check the state of whether the Break command was executed.

(2) This variable only reads the data.

(3) If the M_BrkCq variable is referenced even once, the Break status is cleared. (The value is set to zero.) Therefore, to preserve the status, save it by substituting it into a numeric variable.

(4) The Break status is also cleared even if it is referenced on T/B monitor screen and so forth.

M_BTime

[Function]

Returns the remaining hour of battery left. (Unit: hour)

[Format]

Example)<Numeric Variable>=M_BTime

[Terminology]

<Numeric Variable> Specifies the numerical variable to assign.

[Reference Program]

1 M1=M_BTime ' The remaining battery capacity time is assigned to M1.

[Explanation]

(1) Returns the remaining hours the battery can last from now.

(2) As for the battery life, 14,600 hours are stored as the initial value.

(3) After summing the total amount of time the power of robot controller has been off, this value will be subtracted from 14,600 and the result is returned.

(4) This variable only reads the data.

M_CmpDst

[Function]

Returns the amount of difference (in mm) between the command value and the actual value from the robot when executing the compliance function.

[Format]

Example)<Numeric Variable>=M_CmpDst [<Mechanism Number>]

[Terminology]

<Numeric Variable> Specifies the numerical variable to assign.

<Mechanism Number> Enter the mechanism number. 1 to 3, If the argument is omitted, 1 is set as the default value.

[Reference Program]

```

1 Mov P1
2 CmpG 0.5,0.5,1.0,0.5,0.5, , , ' Set softness.
3 Cmp Pos, &B00011011      ' Enter soft state.
4 Mvs P2
5 M_Out(10)=1
6 Mvs P1
7 M1=M_CmpDst(1)          ' M1 will contain the difference between the position specified by the
                           operation command and the actual current position.
8 Cmp Off                 ' Return to normal state.

```

[Explanation]

- (1) This is used to check the positional discrepancy while executing the compliance function.
- (2) This variable only reads the data.

M_CmpLmt

[Function]

Returns whether or not the command value when the compliance function is being executed is about to exceed various limits.

1: The command value is about to exceed a limit.

0: The command value is not about to exceed a limit.

[Format]

```
Example) Def Act 1, M_CmpLmt [(<Mechanism Number>)] = 1 GoTo *Lmt
```

[Terminology]

<Mechanism Number> Specify the mechanism number 1 to 3. The default value is 1.

[Reference Program]

```
1 Def Act 1, M_CmpLmt(1)=1 GoTo *Lmt      ' Define the conditions of interrupt 1.
2 '
3 '
;
10 Mov P1
11 CmpG 1,1,0,1,1,1,1,1
12 Cmp Pos, &B100                      ' Enable compliance mode.
13 Act 1=1                                ' Enable interrupt 1.
14 Mvs P2
15 '
16 '
;
100 *Lmt
101 Mvs P1                                ' Movement to P2 is interrupted and returns to P1.
102 Reset Err                            ' Reset the error.
103 Hlt                                  ' Execution is stopped.
```

[Explanation]

- (1) This is used to recover from the error status by using interrupt processing if an error has occurred while the command value in the compliance mode attempted to exceed a limit.
- (2) For various limits, the joint operation range and operation speed of the command value in the compliance mode, and the dislocation between the commanded position and the actual position are checked.
- (3) 0 is set if the servo power is off, or the compliance mode is disabled.
- (4) This is a read only variable.

M_Colsts

[Function]

Return the impact detection status..

1: Detecting an impact

0: No impact has been detected

The impact detection function can only be used in certain models (Refer to "[Available robot type]").

[Format]

Example) Def Act 1, M_Colsts [(<Mechanism Number>)=1 GoTo *LCOL,S

[Terminology]

<Mechanism Number> Specify the mechanism number 1 to 3. The default value is 1.

[Reference Program]

1 Def Act 1,M_Colsts(1)=1 GoTo *HOME,S	'Define the processing to be executed when an impact is detected using an interrupt.
2 Act 1=1	
3 ColChk ON,NOERR	'Enable the impact detection function in the error non-occurrence mode.
4 Mov P1	
5 Mov P2	'If an impact is detected while executing lines 40 through 70, it jumps to interrupt processing.
6 Mov P3	
7 Mov P4	
8 Act 1=0	
:	
:	
100 *HOME	'Interrupt processing during impact detection.
101 ColChk Off	'Disable the impact detection function.
102 Servo On	'Turn the servo on.
103 PESC=P_ColDir(1)*(-2)	'Create the amount of movement for escape operation
104 PDst=P_Fbc(1)+PESC	'Create the escape position.
105 Mvs PDst	'Move to the escape position.
106 Error 9100	'Stop operation by generating a user-defined L level error.

[Explanation]

- (1) When an impact is detected, it is set to 1. When the servo is turned off and the impact state is canceled, it is set to 0.
- (2) It is used as an interrupt condition in the Def Act instruction when used in the NOERR mode.
- (3) This variable only reads the data.

[Available robot type]

RV-6SD/6SDL/12SD/12SDL series

M_Cstp

[Function]

Returns the status of whether or not a program is on cycle stop

1: Cycle stop is entered, and cycle stop operation is in effect.

(The input of the End key on the operation panel, or the input of a cycle stop signal)

0: Other than above

[Format]

Example)<Numeric Variable>=M_Cstp

[Terminology]

<Numeric Variable> Specifies the numerical variable to assign.

[Reference Program]

1 M1=M_Cstp ' 1 is assigned to M1. (When under a cycle stop)

[Explanation]

- (1) When the End key on the operation panel is pressed while the program is under continuous execution, the system enters a cycle operation state. The status at this time is returned as 1.
- (2) This variable only reads the data.

M_Cys

[Function]

Returns the status of whether or not a program is on cycle operation

1: In cycle operation (operating mode set by the slot parameter SLT* to ...)

0: Other than above.

[Format]

Example)<Numerical variable> = M_Cys

[Terminology]

<Numerical variable> Specify the numerical variable to substitute.

[Reference Program]

1 M1=M_Cys ' The numerical value 1 is substituted for M1. (When under a cycle operation)

[Explanation]

- (1) When starting a program, the cycle mode - either continuous operation or cycle operation - can be specified using a parameter, etc. Returns this operation mode.
- (2) Even if CYC has been specified in the slot parameter, the value will be 0 when continuous operation is specified by XRun.
- (3) This is a read only variable.

M_DIn/M_DOut

[Function]

This is used to write or reference the remote register of CC-Link (optional). Cannot use in CRnQ series.

M_DIn : References the input register.

M_DOut : Writes or reference the output register.

[Format]

Example)<Numeric Variable>=M_DIn (<Equation 1>)

Example)<Numeric Variable>=M_DOut (<Equation 2>)

[Terminology]

<Numeric Variable>	Specifies the numerical variable that assigns the CC-Link register value.
<Equation 1>	Specifies the CC-Link register number (6000 or above).
<Equation 2>	Specifies the CC-Link register number (6000 or above).

[Reference Program]

1 M1=M_DIn(6000)	' M1 will contain the CC-Link input register value. ' (If CC-Link station number is 1.)
2 M1=M_DOut(6000)	' M1 will contain the CC-Link output register value.
3 M_DOut(6000)=100	' Writes 100 to the CC-Link output register.

[Explanation]

- (1) For details, refer to the "CC-Link Interface Instruction Manual."
- (2) Signal numbers in 6,000's will be used for CC-Link.
- (3) M_DIn is read-only.

M_Err/M_ErrLvl/M_Errno

[Function]

Returns information regarding the error generated from the robot.

M_Err : Returns whether an error has been generated. (1: Error has been generated, 0: No error)

M_ErrLvl : Returns the level of the generated error. (Caution/Low/High1/High2 = 1/2/3/4)

M_Errno : Returns the error number of the generated error.

[Format]

Example) <Numeric Variable>=M_Err
 Example) <Numeric Variable>=M_ErrLvl
 Example) <Numeric Variable>=M_Errno

[Terminology]

<Numeric Variable> Specifies the numerical variable to assign.

[Reference Program]

1 If M_Err=0 Then 10	' Waits until an error is generated.
2 M2=M_ErrLvl	' M2 will contain the error level
3 M3=M_Errno	' M3 will contain the error number.

[Explanation]

- (1) Normal programs will pause when an error (other than cautions) is generated. The error status of the controller may be monitored using this variable for programs whose startup condition is set to ALWAYS by the SLT* parameter. The program set to ALWAYS will not stop even when an error is generated from other programs.
- (2) Level 1 errors are warnings, level 2 errors pause programs. Level 3 errors pause programs and turn the servo power OFF, but error reset can be performed. Level 4 errors pause programs, turn the servo power OFF, and error reset cannot be performed. Thus, when a level 4 error occurs, it is necessary to turn the controller power OFF.
- (3) This variable only reads the data.

[Related instructions]

Reset Err (Reset Error)

M_Exp

[Function]

Returns the base of natural logarithm (2.718281828459045).

[Format]

Example) <Numeric Variable>=M_Exp

[Terminology]

<Numeric Variable> Specifies the numerical variable to assign.

[Reference Program]

1 M1=M_Exp	' Base of natural logarithm (2.718281828459045) is assigned to M1.
------------	--

[Explanation]

- (1) This is used when processing exponential and logarithmic functions.
- (2) This variable only reads the data.

M_Fbd

[Function]

Returns the difference between the command position and the feedback position.

[Format]

Example) <Numeric Variable>=M_Fbd[(<Mechanism Number>)]

[Terminology]

<Numeric Variable>

Specifies the numerical variable to assign.

<Mechanism Number>

Specify the mechanism number 1 to 3. The default value is 1.

[Reference Program]

```

1 Def Act 1,M_Fbd>10 GoTo *SUB1,S      ' Generate an interrupt when the difference between the
                                           command position and the feedback position reaches 10
                                           mm or more.
2 Act 1=1                                ' An interrupt takes effect.
3 Torq 3,10                               ' Set the torque limit of the three axes to 10% or less using
                                           the torque instruction.
4 Mvs P1                                 ' Moves.
5 End
;
10 *SUB1
11 Mov P_Fbc                            ' Align the command position with the feedback position.
12 M_Out(10)=1                           ' Signal No. 10 output
13 Hlt                                    ' Stop when a difference occurs.

```

[Explanation]

- (1) This function returns the difference between the command position specified by the operation instruction and the feedback position from the motor. When using the torque instruction, use this in combination with a Def Act instruction to prevent the occurrences of excessive errors (960, 970, etc.).
- (2) This variable only reads the data.

[Reference]

Torq (Torque), P_Fbc

M_G

[Function]

Returns gravitational constant (9.80665).

[Format]

Example) <Numeric Variable>=M_G

[Terminology]

<Numeric Variable> Specifies the numerical variable to assign.

[Reference Program]

1 M1=M_G ' Gravitational constant (9.80665) is assigned to M1.

[Explanation]

- (1) This is used to perform calculation related to gravity.
- (2) This variable only reads the data.

M_HndCq

[Function]

Returns the hand check input signal value.

[Format]

Example) <Numeric Variable>=M_HndCq (<Equation>)

[Terminology]

<Numeric Variable> Specifies the numerical variable to assign.

<Equation> Enter the hand input signal number.

1 to 8, (Corresponds to input signals 900 to 907.)

[Reference Program]

1 M1=M_HndCq(1) ' M1 will contain the status of hand 1.

[Explanation]

- (1) Returns one bit of the hand check input signal status (such as a sensor).
- (2) M_HndCq(1) corresponds to input signal number 900. Same result will be obtained using M_In (900).
- (3) This variable only reads the data.

M_In/M_Inb/M_Inw

[Function]

Returns the value of the input signal.

M_In : Returns a bit.

M_Inb : Returns a byte (8 bits).

M_Inw : Returns a word (16 bits).

[Format]

Example) <Numeric Variable>=M_In(<Equation>)

Example) <Numeric Variable>=M_Inb(<Equation>)

Example) <Numeric Variable>=M_Inw(<Equation>)

[Terminology]

<Numeric Variable> Specifies the numerical variable to assign.

<Equation> Enter the input signal number. 0 to 32767 (Theoretical value)

0 to 255 : Standard remote inputs (Normally 32 points. 0 to 31)

900 to 907 : Hand input.

2000 to 5071 : Input signal of PROFIBUS.

6000 to 8047 : Remote input for CC-Link.

[Reference Program]

1 M1=M_In(0) ' M1 will contain the value of the input signal 0 (1 or 0).

2 M2=M_Inb(0) ' M2 will contain the 8-bit information starting from input signal 0.

3 M3=M_Inb(3) AND &H7 ' M3 will contain the 3-bit information starting from input signal 3.

4 M4=M_Inw(5) ' M4 will contain the 16-bit information starting from input signal 5.

[Explanation]

(1) Returns the status of the input signal.

(2) M_Inb and M_Inw will return 8- or 16-bit information starting from the specified number.

(3) Although the signal number can be as large as 32767, only the signal numbers with corresponding hardware will return a valid value. Value for a signal number without corresponding hardware is set as undefined.

(4) This variable only reads the data.

M_JOvrd/M_NJovrd/M_OPovrd/M_Ovrd/M_NOvrd

[Function]

Returns override value.

M_JOvrd : Value specified by the override JOvrd instruction for joint interpolation.

M_NJovrd : Initial override value (100%) for joint interpolation.

M_OPovrd : Override value of the operation panel.

M_Ovrd : Current override value, value specified by the Ovrd instruction.

M_NOvrd : Initial override value (100%).

[Format]

Example)<Numeric Variable>=M_JOvrd [(i<Equation>)]

Example)<Numeric Variable>=M_NJovrd[(i<Equation>)]

Example)<Numeric Variable>=M_OPovrd

Example)<Numeric Variable>=M_Ovrd[(<Equation>)]

Example)<Numeric Variable>=M_NOvrd[(<Equation>)]

[Terminology]

<Numeric Variable> Specifies the numerical variable to assign.

<Equation> 1 to 32, Enter the task slot number. If this parameter is omitted, the current slot will be used as the default.

[Reference Program]

- | | |
|----------------|--|
| 1 M1=M_Ovrd | ' M1 will contain the current override value. |
| 2 M2=M_NOvrd | ' M2 will contain the initial override value (100%). |
| 3 M3=M_JOvrd | ' M3 will contain the current joint override value. |
| 4 M4=M_NJovrd | ' M4 will contain the initial joint override value. |
| 5 M5=M_OPovrd | ' M5 will contain the current OP (operation panel) override value. |
| 6 M6=M_Ovrd(2) | ' M6 will contain the current override value for slot 2. |

[Explanation]

- (1) If the argument is omitted, the current slot status will be returned.
- (2) This variable only reads the data.

M_LdFact

[Function]

The load ratio for each joint axis can be referenced.

[Format]

Example)<Numeric Variable>=M_LdFact(<Axis Number>)

[Terminology]

<Numeric Variable> The load ratio of each axis is substituted. The range is 0 to 100%.

<Axis Number> 1 to 8, Specifies the axis number.

[Reference Program]

```

1 Accel 100,100           ' Lower the overall deceleration time to 50%.
2 Mov P1
3 Mov P2
4 If M_LdFact(2)>90 Then
5   Accel 50,50           ' Lower the acceleration/deceleration ratio to 50%.
6   M_SetAdl(2)=50        ' Furthermore, lower the acceleration/deceleration ratio of the J2 axis to
                           50%. (In actuality, 50% x 50% = 25%)
7 Else
8   AccelL 100.,100       ' Return the acceleration/deceleration time.
9 EndIf
10 GoTo 20

```

[Explanation]

- (1) The load ratio of each axis can be referenced.
- (2) The load ratio is derived from the current that flows to each axis motor and its flow time.
- (3) The load ratio rises when the robot is operated with a heavy load in a severe posture for a long period of time.
- (4) When the load ratio reaches 100%, an overload error occurs. In the above example statement, once the load ratio exceeds 90%, the k acceleration/deceleration time is lowered to 50%.
- (5) To lower the load ratio, measures, such as decreasing the acceleration/deceleration time, having the robot standing by in natural posture, or shutting down the servo power supply, are effective.

[Related instructions]

Accel (Accelerate), Ovrd (Override), M_SetAdl

M_Line

[Function]

Returns the line number that is being executed.

[Format]

Example)<Numeric Variable>=M_Line [(<Equation>)]

[Terminology]

<Numeric Variable>	Specifies the numerical variable to assign.
<Equation>	1 to 32, Specifies the task slot number. If this parameter is omitted, the current slot will be used as the default.

[Reference Program]

1 M1=M_Line(2) ' M1 will contain the line number being executed by slot 2.

[Explanation]

- (1) This can be used to monitor the line being executed by other tasks during multitask operation.
- (2) This variable only reads the data.

M_Mode

[Function]

Returns the key switch mode of the operation panel.

- 1 : TEACH
- 2 : AUTO(OP)
- 3 : AUTO(Ext.)

[Format]

Example)<Numeric Variable>=M_Mode

[Terminology]

<Numeric Variable>	Specifies the numerical variable to assign.
--------------------	---

[Reference Program]

1 M1=M_Mode ' M1 will contain the key switch status.

[Explanation]

- (1) This can be used in programs set to ALWAYS (constantly executed) during multitask operation.
- (2) This variable only reads the data.

M_On/M_Off

[Function]

Always returns 1 (M_On) or 0 (M_Off).

[Format]

Example)<Numeric Variable>=M_On

Example)<Numeric Variable>=M_Off

[Terminology]

<Numeric Variable> Specifies the numerical variable to assign.

[Reference Program]

1 M1=M_On	' 1 is assigned to M1.
2 M2=M_Off	' 0 is assigned to M2.

[Explanation]

- (1) Always returns 1 or 0.
- (2) This variable only reads the data.

M_Open

[Function]

Returns the status indicating whether or not a file is opened.
Returns the status of other end of the RS-232C cable.

[Format]

Example)<Numerical variable>=M_Open [<File number>]

[Terminology]

<Numerical variable> Specify the numerical variable to substitute.
<File number> Specify the file number 1-8 by constant value of communication line opened by Open command. The default value is 1. If 9 or more are specified, the error will occur when executing.

[Reference Program]

```
1 Open "COM2:" AS #1           ' Open the communication line COM2 as the file number 1.
2 If M_Open(1)<>1 Then GoTo 110      ' Wait until the file number 1 opens.

<Using the ethernet I/F>
1 ' Client Program -----
2 M1=0
3 M_Timer(1)=0           'Resets the timer to 0.
4 *O1
5 Open "COM2:" As #1           'Opens the line.
6 If M_Timer(1)>10000.0 Then *E1      'Jumps when 10 seconds elapses.
7 If M_Open(1)<>1 Then Goto *O1      'Loops if no connection is made.
8 Def Act 1,M_Open(1)=0 GoSub 300      'Monitors the down state of the server using an interrupt.
9 Act 1=1           'Starts monitoring.
10 *M1
11 M1=M1+1
12 If M1<10 Then C1$="MELFA" Else C1$="END"  'Sends END after sending the "MELFA" string nine
                                                times.
13 Print #1,C1$           'Sends a character string.
14 Input #1,C2$           'Receives a character string.
15 If C1$="END" Then *C1      'Jumps to CLOSE after sending "END."
16 GoTo *M1           'Loops.
17 *C1
18 Close #1           'Closes the line.
19 Hlt
20 End
21 *E1
22 Error 9100           'Generates error 9100 if no connection can be made to the server.
23 Close #1
24 Hlt
25 End
26 Error 9101           'Generates error 9101 if the server is down during processing.
27 Close #1
28 Hlt
29 End
```

[Explanation]

- (1) This is a read only variable.
- (2) The return value differ corresponding to the file type specified by Open command as follows.

Kind of files	Meaning	Value
File	Returns the status indicating whether or not a file is opened. Returns 1 until the Close instruction, the End instruction or End in a program is executed after executing the Open instruction.	1: Already opened -1: Undefined file number (not opened)
Communication line RS-232C	*Returns the status of other end of the RS-232C cable. Returns the status of the CTS signal input as is. (This can be used only when the RTS signal of other end is enabled using the Mitsubishi genuine cable specification.)	1: Already connected (CTS signal is ON) 0: Unconnected (CTS signal is OFF) -1: Undefined file number (not opened)

[Related instructions]

[Open \(Open\)](#)

[Related parameter]

COMDEV, CPRE**, NETMODE

M_Out/M_Outb/M_Outw

[Function]

Writes or references external output signal.

M_Out:Output signal bit.

M_Outb:Output signal byte (8 bits).

M_Outw:Output signal word (16 bits).

[Format]

Example)M_Out(<Equation>)=<Numeric Variable>

Example)M_Outb(<Equation>)=<Numeric Variable>

Example)M_Outw(<Equation>)=<Numeric Variable>

Example)M_Outw(<Equation>)=<Numeric Variable> Dly <Time>

[Terminology]

<Numeric Variable> Specifies the numerical variable to assign.

<Equation> Specify the output signal number.

0 to 255 : Standard remote outputs.

900 to 907 : Hand output.

2000 to 5071 : Output signal of PROFIBUS.

6000 to 8047 : Remote output for CC-Link.

<Time> Describe the output time for the pulse output as a constant or numeric operation expression. Unit: [Seconds]

[Reference Program]

1 M_Out(2)=1 ' Turn ON output signal 2 (1 bit).

2 M_Outb(2)=&HFF ' Turns ON 8-bits starting from the output signal 2.

3 M_Outw(2)=&HFFFF ' Turns ON 16-bits starting from the output signal 2.

4 M4=M_Outb(2) AND &H0F ' M4 will contain the 4-bit information starting from output signal 2.

[Explanation]

(1) This is used when writing or referencing external output signals.

(2) Numbers in 900's will be used as I/O signals for the hand.

(3) Numbers 6000 and beyond will be referenced/assigned to the CC-Link (optional).

(4) Refer to [Page 194](#), " Dly (Delay)" for the explanation of pulse output.

M_PI

[Function]

Returns pi (3.14159265358979).

[Format]

Example)<Numeric Variable>=M_PI

[Terminology]

<Numeric Variable> Specifies the numerical variable to assign.

[Reference Program]

1 M1=M_PI ' 3.14159265358979 is assigned to M1.

[Explanation]

(1) A variable to be assigned will be a real value.

(2) This variable only reads the data.

M_Psa

[Function]

Returns whether the program is selectable by the specified task slot.

1 : Program is selectable.

0 : Program not selectable (when the program is paused).

[Format]

Example)<Numeric Variable>=M_Psa [(<Equation>)]

[Terminology]

<Numeric Variable> Specifies the numerical variable to assign.

<Equation> 1 to 32, Specifies the task slot number. If this parameter is omitted, the current slot will be used as the default.

[Reference Program]

1 M1=M_Psa(2) ' M1 will contain the program selectable status of task slot 2.

[Explanation]

- (1) Returns whether the program is selectable by the specified task slot.
- (2) This variable only reads the data.

M_Ratio

[Function]

Returns how much the robot has approached the target position (0 to 100%) while the robot is moving.

[Format]

Example)<Numeric Variable>=M_Ratio [(<Equation>)]

[Terminology]

<Numeric Variable> Specifies the numerical variable to assign.

<Equation> 1 to 32, Specifies the task slot number. If this parameter is omitted, the current slot will be used as the default.

[Reference Program]

1 Mov P1 WthIf M_Ratio>80, M_Out(1)=1' The output signal 1 will turn ON when the robot has moved 80% of the distance until the target position is reached while moving toward P1.

[Explanation]

- (1) This is used, for instance, when performing a procedure at a specific position while the robot is moving.
- (2) This variable only reads the data.

M_RDst

[Function]

Returns the remaining distance to the target position (in mm) while the robot is moving.

[Format]

Example)<Numeric Variable>=M_RDst [(<Equation>)]

[Terminology]

<Numeric Variable>	Specifies the numerical variable to assign.
<Equation>	1 to 32, Specifies the task slot number. If this parameter is omitted, the current slot will be used as the default.

[Reference Program]

1 Mov P1 WthIf M_RDst<10 M_Out(10)=1 ' The output signal 1 will turn ON when the remaining distance until the target position is reached becomes 10 mm or less while moving toward P1.

[Explanation]

- (1) This is used, for instance, when performing a procedure at a specific position while the robot is moving.
- (2) This variable only reads the data.

M_Run

[Function]

Returns whether the program for the specified task slot is being executed.

- 1 : Executing.
0 : Not executing (paused or stopped).

[Format]

Example)<Numeric Variable>=M_Run [(<Equation>)]

[Terminology]

<Numeric Variable>	Specifies the numerical variable to assign.
<Equation>	1 to 32, Specifies the task slot number. If this parameter is omitted, the current slot will be used as the default.

[Reference Program]

1 M1=M_Run(2) ' M1 will contain the execution status of slot 2.

[Explanation]

- (1) This will contain 1 if the specified slot is running, or 0 if the slot is stopped (or paused).
- (2) Combine M_Run and M_Wai to determine if the program has stopped (in case the currently executed line is the top line).
- (3) This variable only reads the data.

M_SetAdl

[Function]

Set the acceleration/deceleration time distribution rate of the specified axis when optimum acceleration/deceleration control is enabled (Oadl ON). Since it can be set for each axis, it is possible to reduce the motor load of an axis with a high load. Also, unlike a method that sets all axes uniformly, such as Ovrd, Spd and Accel instructions, the effect on the tact time can be minimized as much as possible. The initial value is the setting value of the JADL parameter.

This status variable can only be used in certain models (Refer to "[Available robot type]").

[Format]

Example) M_SetAdl(<Axis Number>)=<Numeric Variable>

[Terminology]

<Axis Number>	1 to 8, Specifies the axis number.
<Numeric Variable>	Specify the ratio for the standard acceleration/deceleration time, between 1 and 100. The unit is %. The initial value is the value of the optimum acceleration/deceleration adjustment rate parameter (JADL).

[Reference Program]

```

1 Accel 100,50          ' Set the overall acceleration/deceleration distribution rate to 50%.
2 If M_LdFact(2)>90 Then
3   M?SetAdl(2)=70      ' If the load rate of the J2 axis exceeds 90%,
4 EndIf                   ' set the acceleration/deceleration time distribution rate of the J2
                           ' axis to 70%.
5 Mov P1                  ' Acceleration 70% (= 100% x 70%), deceleration 35% (= 50% x
6 Mov P2                  70%)
7 M_SetAdl(2)=100          ' Return the acceleration/deceleration time distribution rate of the
                           ' J2 axis to 100%.
8 Mov P3                  ' Acceleration 100%, deceleration 50%
9 Accel 100,100            ' Return the overall deceleration distribution rate to 100%.
10 Mov P4

```

[Explanation]

- (1) The acceleration/deceleration time distribution rate when optimum acceleration/deceleration is enabled can be set in units of axes. If 100% is specified, the acceleration/deceleration time becomes the shortest.
- (2) Using this status variable, the acceleration/deceleration time can be set so as to reduce the load on axes where overload and overheat errors occur.
- (3) The setting of this status variable is applied to both the acceleration time and deceleration time.
- (4) When this status variable is used together with an Accel instruction, the specification of the acceleration/deceleration distribution rate of the Accel instruction is also applied to the acceleration/deceleration time calculated using the optimum acceleration/deceleration speed.
- (5) With the Accel instruction, the acceleration/deceleration time changes at the specified rate. Because this status variable is set independently for each axis and also the acceleration/deceleration time that takes account of the motor load is calculated, the change in the acceleration/deceleration time may show a slightly different value than the specified rate.

[Reference]

[Accel \(Accelerate\)](#), [Ovrd \(Override\)](#), [Spd \(Speed\)](#), [M_LdFact](#)

[Available robot type]

RV-6SD/6SDL/12SD/12SDL series

M_SkipCq

[Function]

Returns the result of executing the line containing the last executed Skip command.

1 : Skip has been executed.

0 : Skip has not been executed.

[Format]

Example) <Numeric Variable>=M_SkipCq [(<Equation>)]

[Terminology]

<Numeric Variable> Specifies the numerical variable to assign.

<Equation> 1 to 32, Specifies the task slot number. If this parameter is omitted, the current slot will be used as the default.

[Reference Program]

1 Mov P1 WthIf M_In(10)=1,Skip	' If the input signal 10 is 1 when starting to move to P1, skip the Mov instruction.
2 If M_SkipCq=1 Then GoTo 1000	' If Skip instruction has been executed, jump to line 1000.
;	
1000 End	

[Explanation]

(1) Checks if a Skip instruction has been executed.

(2) This variable only reads the data.

(3) If the M_SkipCq variable is referenced even once, the Skip status is cleared. (The value is set to zero.)
Therefore, to preserve the status, save it by substituting it into a numeric variable.

M_Spd/M_NSpd/M_RSpd

[Function]

Returns the speed information during XYZ and JOINT interpolation.

M_Spd : Currently set speed.

M_NSpd : Initial value (optimum speed control).

M_RSpd : Directive speed.

[Format]

Example)<Numeric Variable>=M_Spd [(<Equation>)]

Example)<Numeric Variable>=M_NSpd [(<Equation>)]

Example)<Numeric Variable>=M_RSpd [(<Equation>)]

[Terminology]

<Numeric Variable> Specifies the numerical variable to assign.

<Equation> 1 to 32, Specifies the task slot number. If this parameter is omitted, the current slot will be used as the default.

[Reference Program]

1 M1=M_Spd ' M1 will contain the currently set speed.

2 Spd M_NSpd ' Reverts the speed to the optimum speed control mode.

[Explanation]

(1) M_RSpd returns the directive speed at which the robot is operating.

(2) This can be used in M_RSpd multitask programs or with Wth and Wthlf statements.

(3) This variable only reads the data.

M_Svo

[Function]

Returns the current status of the servo power supply.

1 : Servo power ON

0 : Servo power OFF

[Format]

Example)<Numeric Variable>=M_Svo [(<Mechanism Number>)]

[Terminology]

<Numeric Variable> Specifies the numerical variable to assign.

<Mechanism Number> Enter the mechanism number. 1 to 3, If the argument is omitted, 1 is set as the default value.

[Reference Program]

1 M1=M_Svo(1) ' M1 will contain the current status of the servo power supply.

[Explanation]

(1) The status of the robot's servo can be checked.

(2) This variable only reads the data.

M_Timer

[Function]

Time is measured in milliseconds. This can be used to measure the operation time of the robot or to measure time accurately.

[Format]

Example)<Numeric Variable>=M_Timer (<Equation>)

[Terminology]

<Numeric Variable>	Specifies the numerical variable to assign.
<Equation>	Enter the number to 8 from 1. Parentheses are required.

[Reference Program]

1 M_Timer(1)=0	
2 Mov P1	
3 Mov P2	
4 M1=M_Timer(1)	' M1 will contain the amount of time required to move from P1 to P2 (in ms). Example) If the time is 5.346 sec. the value of M1 is 5346.
5 M_Timer(1)	' Set to 1.5 sec.

[Explanation]

- (1) A value may be assigned. The unit is seconds when set to M_Timer.
- (2) Since measurement can be made in milliseconds (ms), precise execution time measurement is possible.

M_Tool

[Function]

In addition to using the tool data (MEXTL1 to 4) of the specified number as the current tool data, it is also set in the MEXTL parameter.

The current tool number can also be read.

[Format]

Example)<Numeric Variable>=M_Tool [(<Mechanism Number>)]'Referencing the Current Tool Number

Example)M_Tool [(<Mechanism Number>)] = [<Equation>] 'Set a tool number.

[Terminology]

<Numeric Variable> Specifies the numerical variable to assign.

<Mechanism Number> Enter the mechanism number to 3 from 1.

If the argument is omitted, 1 is set as the default value.

<Equation> Enter the tool number to 4 from 1.

[Reference Program]

Setting Tool Data

1 Tool (0,0,100,0,0,0)	' Specify tool data (0,0,100,0,0,0), and write it into MEXTL.
2 Mov P1	
3 M_Tool=2	' Change the tool data to the value of tool number 2 (MEXTL2).
4 Mov P2	

Referencing the Tool Number

1 If M_In(900)=1 Then	' Change the tool data by a hand input signal.
2 M_Tool=1	' Set tool 1 in tool data.
3 Else	
4 M_Tool=2	' Set tool 2 in tool data.
5 EndIf	
6 Mov P1	

[Explanation]

- (1) The values set in the MEXTL1, MEXTL2, MEXTL3 and MEXTL4 tool parameters are reflected in the tool data. It is also written into the MEXTL parameter.
- (2) Tool numbers 1 to 4 correspond to MEXTL1 to 4.
- (3) While referencing, the currently set tool number is read.
- (4) If the reading value is 0, it indicates that tool data other than MEXTL1 to 4 is set as the current tool data.
- (5) The same setting can be performed on the Tool Setup screen of the teaching pendant. For more information, see [Page 23, "3.2.8 Switching Tool Data"](#).

[Reference]

[Tool\(Tool\)](#), MEXTL, MEXTL1, MEXTL2, MEXTL3, MEXTL4

M_Uar

[Function]

Returns whether the robot is in the user-defined area.

Bits 0 through 7 correspond to areas 1 to 8 and each bit displays the following information.

1 : Within user-defined area

0 : Outside user-defined area

[Format]

Example)<Numeric Variable>=M_Uar [<Mechanism Number>]

[Terminology]

<Numeric Variable> Specifies the numerical variable to assign.

<Mechanism Number> Enter the mechanism number. 1 to 3, If the argument is omitted, 1 is set as the default value.

[Reference Program]

1 M1=M_Uar(1) ' M1 indicates whether the robot is within or outside the user-defined area.
The value 4 indicates that the robot is in the user-defined area 3.

[Explanation]

- (1) For details on how to use user-defined areas, refer to [Page 372, "About user-defined area".](#)
- (2) This variable only reads the data.

M_Uar32

[Function]

Returns whether contained in the user-defined area.

Bits 0 to 31 correspond to areas 1 to 32, with the respective bits displaying the information below.

1: Within user-defined area

2: Outside user-defined area

[Format]

Example) <Numeric Variable> = M_Uar32 [(<Mechanism Number>)]

[Terminology]

<Numeric Variable> Specifies the numerical variable to assign.

<Mechanism Number> Enter the mechanism number from 1 to 3. If the argument is omitted, 1 is set as the default value.

[Reference Program]

```
1 Def Long M1
2 M1& = M_Uar32(1) AND &H00080000      ' The result for area 20 only is assigned to M1.
3 If M1&<>0 Then M_Out(10)=1           ' Output signal 10 turns ON if contained in area 20.
```

[Explanation]

- (1) Refer to, [Page 372, "5.8 About user-defined area"](#) for details on how to use a user-defined area.
- (2) An error will occur if a 16-bit integer type is used for the <Numeric Variable> and the value is over. If so, use a 32-bit integer type.
- (3) The area in which 1 (signal output) is specified for parameter AREAnAT (n is the area no. (n = 1 to 32)) is applicable.
- (4) When performing a comparison operation or logic operation, a negative value results in decimal notation if bit 31 is 1, and therefore it is recommended that hexadecimal notation be used.
- (5) This variable only reads the data.

[Related System Variables]

M_Uar

[M_Uar32 and User-defined Area Compatibility]

Bit	Area	Decimal Value	Hexadecimal Value
0	1	1	&H00000001
1	2	2	&H00000002
2	3	4	&H00000004
3	4	8	&H00000008
4	5	16	&H00000010
5	6	32	&H00000020
6	7	64	&H00000040
7	8	128	&H00000080
8	9	256	&H00000100
9	10	512	&H00000200
10	11	1024	&H00000400
11	12	2048	&H00000800
12	13	4096	&H00001000
13	14	8192	&H00002000
14	15	16384	&H00004000
15	16	32768	&H00008000

Bit	Area	Decimal Value	Hexadecimal Value
16	17	65536	&H00010000
17	18	131072	&H00020000
18	19	262144	&H00040000
19	20	524288	&H00080000
20	21	1048576	&H01000000
21	22	2097152	&H02000000
22	23	4194304	&H04000000
23	24	8388608	&H08000000
24	25	16777216	&H01000000
25	26	33554432	&H02000000
26	27	67108864	&H04000000
27	28	134217728	&H08000000
28	29	268435456	&H10000000
29	30	536870912	&H20000000
30	31	1073741824	&H40000000
31	32	-2147483648	&H80000000

Example) If contained in user-defined area 5 and 10, this will be the combined value of &H00000010, the value indicating area 5, and H00000400, the value indicating area 10, however, this will be returned as an M_Uar32 value.

M_Wai

[Function]

Returns the standby status of the program for the specified task slot.

1 : Paused (The program has been paused.)

0 : Not paused (Either the program is running or is being stopped.)

[Format]

Example) <Numeric Variable>=M_Wai [<Equation>]

[Terminology]

<Numeric Variable> Specifies the numerical variable to assign.

<Equation> 1 to 32, Specifies the task slot number. If this parameter is omitted, the current slot will be used as the default.

[Reference Program]

1 M1=M_Wai(1) ' M1 will contain the standby status of slot 1.

[Explanation]

(1) This can be used to check whether the program has been paused.

(2) Combine M_Run and M_Wai to determine if the program has stopped (in case the currently executed line is the top line).

(3) This variable only reads the data.

[Reference]

M_Wupov, M_Wuprt, M_Wupst

M_Wupov

[Function]

Returns the value of an override (warm-up operation override, unit: %) to be applied to the command speed in order to reduce the operation speed when in the warm-up operation status.

Note: For more information about the warm-up operation mode, see [Page 402, "5.21 Warm-Up Operation Mode"](#) for detail.

[Format]

Example) <Numeric Variable> = M_Wupov [<Mechanism Number>]

[Terminology]

<Numeric Variable> Specifies the numerical variable to assign.

<Mechanism Number> Enter the mechanism number. 1 to 3. If the argument is omitted, 1 is set as the default value.

[Reference Program]

1 M1=M_Wupov(1) ' The value of a warm-up operation override is entered in M1.

[Explanation]

- (1) This is used to confirm the value of an override (warm-up operation override) to be applied to the command speed in order to reduce the operation speed when the robot is in the warm-up operation status (the status in which operation is performed by automatically reducing the speed).
- (2) If the warm-up operation mode is disabled, the MODE switch on the front of the controller is set to "TEACH," or the machine is being locked, the value is always 100.
- (3) If the normal status changes to the warm-up operation status, or the warm-up operation status is set immediately after power on, the value specified in the first element (the initial value of a warm-up operation override) of the WUPOvrd parameter is set as the initial value, and the value of M_Wupov increases according to the operation of the robot. And when the warm-up operation status is canceled, the value of M_Wupov is set to 100.
- (4) The actual override in the warm-up operation status is as follows:
 During joint interpolation operation = (operation panel (T/B) override setting value) x (program override (Ovrd instruction)) x (joint override (JOvrd instruction)) x warm-up operation override
 During linear interpolation operation = (operation panel (T/B) override setting value) x (program override (Ovrd instruction)) x (linear specification speed (Spd instruction)) x warm-up operation override
- (5) This variable only reads the data.

M_Wuprt

[Function]

Returns the time (sec) during which a target axis must operate to cancel the warm-up operation status.

Note: For more information about the warm-up operation mode, see [Page 402, "5.21 Warm-Up Operation Mode"](#) for detail.

[Format]

Example)<Numeric Variable> = M_Wuprt [(<Mechanism Number>)]

[Terminology]

<Numeric Variable> Specifies the numerical variable to assign.

<Mechanism Number> Enter the mechanism number. 1 to 3, If the argument is omitted, 1 is set as the default value.

[Reference Program]

1 M1=M_Wuprt(1) ' The time during which a target axis must operate is entered in M1.

[Explanation]

- (1) This is used to confirm when the warm-up operation status can be canceled after how long more the joint axis specified in the WUPAXIS parameter (warm-up operation mode target axis) operates when the robot is in the warm-up operation status (the status in which operation is performed by automatically reducing the speed).
- (2) If the warm-up operation mode is disabled, 0 is always returned.
- (3) If the normal status changes to the warm-up operation status, or the warm-up operation status is set immediately after power on, the time specified in the first element (the valid time of the warm-up operation mode) of the WUPTIME parameter is set as the initial value, and the value of M_Wuprt decreases according to the operation of the robot. And when the value is set to 0, the warm-up operation status is canceled.
- (4) If a multiple number of target axes in warm-up operation mode exist, the value of the axis with the shortest operation time among them is returned.
For example, when a target axis (A) operates and the warm-up operation status is canceled in remaining 20 seconds (when M_Wuprt = 20), if another target axis (B) that has continuously been stopped changes from the normal status to the warm-up operation status, (B) becomes the axis with the shortest operation time (operation time of 0 sec). Therefore, the time during which (B) must operate (= the valid time of the warm-up operation mode, initial value is 60 sec) becomes the value of this status variable (M_Wuprt = 60).
- (5) This variable only reads the data.

M_Wupst

[Function]

Returns the time (sec) until the warm-up operation status is set again after it has been canceled.

Note: For more information about the warm-up operation mode, see [Page 402, "5.21 Warm-Up Operation Mode"](#) for detail.

[Format]

Example)<Numeric Variable> = M_Wupst [(<Mechanism Number>)]

[Terminology]

<Numeric Variable> Specifies the numerical variable to assign.

<Mechanism Number> Enter the mechanism number. 1 to 3, If the argument is omitted, 1 is set as the default value.

[Reference Program]

1 M1=M_Wupst(1) ' The time until the warm-up operation status is set again is entered in M1.

[Explanation]

- (1) This is used to confirm when the warm-up operation status is set again after how long more the joint axis specified in the WUPAXIS parameter (warm-up operation mode target axis) continues to stop operating while the robot's warm-up operation status (the status in which operation is performed by automatically reducing the speed) is canceled.
- (2) If the warm-up operation mode is disabled, the time specified in the second element (warm-up operation mode resume time) of the WUPTIME parameter is returned.
- (3) If a target axis operates while the warm-up operation status is canceled, the time specified in the second element (warm-up operation mode resume time) of the WUPTIME parameter is set as the initial value, and the value of M_Wupst decreases while the target axis is stopping. And when the value is set to 0, the warm-up operation status is set.
- (4) If a multiple number of target axes exist, the value of the axis that has been stopped the longest among them is returned.
- (5) This variable only reads the data.

P_Base/P_NBase

[Function]

Returns information related to the base conversion data.

P_Base : Returns the base conversion data that is currently being set.

P_NBase : Returns the initial value (0, 0, 0, 0, 0, 0) (0, 0).

[Format]

Example)<Position Variables>=P_Base [(<Mechanism Number>)]

Example)<Position Variables>=P_NBase

[Terminology]

<Position Variables> Specifies the position variable to assign.

<Mechanism Number> Enter the mechanism number. 1 to 3, If the argument is omitted, 1 is set as the default value.

[Reference Program]

1 P1=P_Base ' P1 will contain the base conversion data that is currently being set.

2 Base P_NBase ' Resets the base conversion data to the initial value.

[Explanation]

(1) P_NBase will contain (0, 0, 0, 0, 0, 0) (0, 0).

(2) Be careful when using base conversion since it may affect the teaching data.

(3) Use the Base instruction when changing the base position.

(4) This variable only reads the data.

P_ColDir

[Function]

Return the operation direction of the robot when an impact is detected.

The impact detection function can only be used in certain models (Refer to "[Available robot type]").

[Format]

Example)<Position Variables>=P_ColDir [(<Mechanism Number>)]

[Terminology]

<Position Variables> Specifies the position variable to assign.

<Mechanism Number> Enter the mechanism number. 1 to 3, If the argument is omitted, 1 is set as the default value.

[Reference Program]

Refer to [Page 175](#), " [Reference Program 2]" for "ColChk (Col Check)".

[Explanation]

- (1) This is used to verify the operation direction of the robot in automatic restoration operation after impact detection.
- (2) The operation direction of the robot at the very moment of impact detection is expressed as a ratio using the maximum travel axis as ± 1.0 . Example: If the robot was being operated at a ratio of (X-axis direction:Y-axis direction) = (2:-1)...P_ColDir = (1,-0.5,0,0,0,0)(0,0)
- (3) The posture axis and structural flag are always (*.*.0,0,0,0,0)(0,0).
- (4) A value is calculated when an impact is detected, and then that value is retained until the next impact is detected.
- (5) If an impact is detected when an external object hits the robot in the stationary state, all axes are set to 0.0.
- (6) Because this variable calculates the operation direction based on the target position of an operation instruction, all elements may be set to 0.0 if an impact occurs at a position near the target position.
- (7) This is read only.
- (8) For robots that prohibit the use of impact detection, 0.0 is always returned for all axes.

[Reference]

[ColChk \(Col Check\)](#), [ColLvl \(Col Level\)](#), [M_ColSts](#), [J_ColMxl](#)

[Available robot type]

RV-6SD/6SDL/12SD/12SDL series

P_Curr

[Function]

Returns the current position (X, Y, Z, A, B, C,L1,L2) (FL1, FL2).

[Format]

Example)<Position Variables>=P_Curr [<Mechanism Number>]

[Terminology]

<Position Variables> Specifies the position variable to assign.
<Mechanism Number> Enter the mechanism number. 1 to 3, If the argument is omitted, 1 is set as the default value.

[Reference Program]

1 Def Act 1,M_In(10)=1	GoTo 1000	' Defines interrupt.
2 Act 1=1		' Enables interrupt.
3 Mov P1		
4 Mov P2		
5 Act 1=0		' Disables interrupt.
:		
100 P100=P_Curr		' Loads the current position when an interrupt signal is received.
101 Mov P100,-100		' Moves 100 mm above P100 (i.e, -100 mm in the Z direction of the tool).
102 End		' Ends the program.

[Explanation]

- (1) This can be used to identify the current position.
- (2) This variable only reads the data.

[Reference]

J_Curr

P_Fbc

[Function]

Returns the current position (X,Y,Z,A,B,C,L1,L2)(FL1,FL2) based on the feedback values from the servo.

[Format]

Example)<Position Variables>=P_Fbc [(<Mechanism Number>)]

[Terminology]

<Position Variables> Specifies the position variable to assign.

<Mechanism Number> Enter the mechanism number. 1 to 3, If the argument is omitted, 1 is set as the default value.

[Reference Program]

1 P1=P_Fbc ' P1 will contain the current position based on the feedback.

[Explanation]

(1) Returns the current position based on the feedback values from the servo.

(2) This variable only reads the data.

[Reference]

Torq (Torque),J_Fbc/J_AmpFbc,M_Fbd

P_Safe

[Function]

Returns the safe point (XYZ position of the JSafe parameter).

[Format]

Example)<Position Variables>=P_Safe [(<Mechanism Number>)]

[Terminology]

<Position Variables> Specifies the position variable to assign.

<Mechanism Number> Enter the mechanism number. 1 to 3, If the argument is omitted, 1 is set as the default value.

[Reference Program]

1 P1=P_Safe ' P1 will contain the set safe point being set.

[Explanation]

(1) Returns the XYZ position, which has been converted from the joint position registered in parameter JSafe.

(2) This variable only reads the data.

P_Tool/P_NTool

[Function]

Returns tool conversion data.

P_Tool: Returns the tool conversion data that is currently being set.

P_NTool: Returns the initial value (0,0,0,0,0,0,0)(0,0).

[Format]

Example)<Position Variables>=P_Tool [(<Mechanism Number>)]

Example)<Position Variables>=P_NTool

[Terminology]

<Position Variables> Specifies the position variable to assign.

<Mechanism Number> Enter the mechanism number. 1 to 3, If the argument is omitted, 1 is set as the default value.

[Reference Program]

1 P1=P_Tool ' P1 will contain the tool conversion data.

[Explanation]

(1) P_Tool returns the tool conversion data set by the Tool instruction or the MEXTL parameter.

(2) Use the Tool instruction when changing tool data.

(3) This variable only reads the data.

P_Zero

[Function]

Always returns (0,0,0,0,0,0,0)(0,0).

[Format]

Example)<Position Variables>=P_Zero

[Terminology]

<Position Variables> Specifies the position variable to assign.

[Reference Program]

1 P1=P_Zero '(0,0,0,0,0,0,0)(0,0) is assigned to P1.

[Explanation]

(1) This can be used to initialize the P variable to zeros.

(2) This variable only reads the data.

4.13 Detailed Explanation of Functions

4.13.1 How to Read Described items

[Function]	: This indicates a function of a function.
[Format]	: This indicates how to input the function argument.
[Reference Program]	: An example program using function is shown.
[Terminology]	: This indicates the meaning and range of an argument.
[Explanation]	: This indicates detailed functions and precautions.
[Reference]	: This indicates related function.

4.13.2 Explanation of Each Function

Each variable is explained below in alphabetical order.

Abs

[Function]

Returns the absolute value of a given value.

[Format]

```
<Numeric Variable>=Abs(<Equation>)
```

[Reference Program]

```
1 P2.C=Abs(P1.C)      ' P2.C will contain the value of P1.C without the sign.  
2 Mov P2  
3 M2=-100  
4 M1=Abs(M2)          ' 100 is assigned to M1.
```

[Explanation]

(1) Returns the absolute value (Value with the positive sign) of a given value.

[Reference]

Sgn

Align

[Function]

Positional posture axes (A, B, and C axes) are converted to the closest XYZ postures (0, +/-90, and +/-180). Align outputs numerical values only. The actual operation will involve movement instructions such as the Mov instruction.

[Format]

<Position Variables>=Align(<Position>)

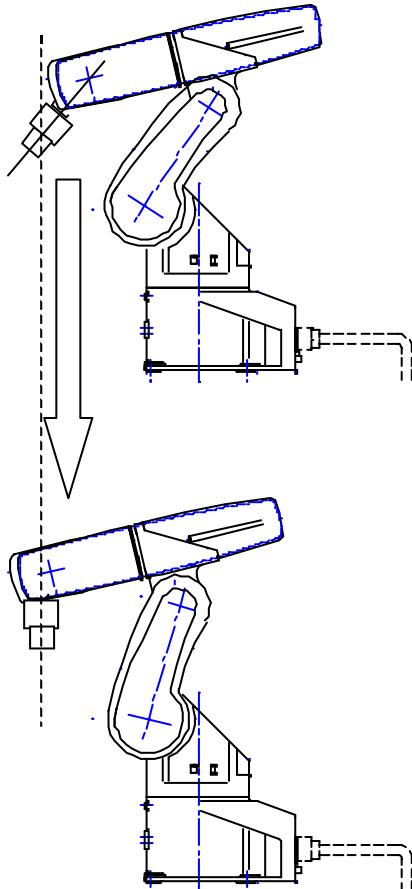
[Reference Program]

```
1 P1=P_Curr
2 P2=Align(P1)
3 Mov P2
```

[Explanation]

- (1) Converts the A, B, and C components of the position data to the closest XYZ postures (0, +/-90, and +/-180).
- (2) Since the return value is of position data type, an error will be generated if the left-hand side is of joint variable type.
- (3) This function cannot be used in vertical multi-joint 5-axes robot.

The following shows a sample case for the axis B.



Asc

[Function]

Returns the character code of the first character in the string.

[Format]

```
<Numeric Variable>=Asc(<Character String Expression>)
```

[Reference Program]

```
1 M1=Asc("A")
```

' &H41 is assigned to M1.

[Explanation]

- (1) Returns the character code of the first character in the string.
- (2) An error will be generated if the string is a null string.

[Reference]

[Chr\\$](#), [Val](#), [Cvi](#), [Cvs](#), [Cvd](#)

Atn/Atn2

[Function]

Calculates the arc tangent.

[Format]

```
<Numeric Variable>=Atn(<Equation>)
<Numeric Variable>=Atn2(<Equation 1>, <Equation 2>)
```

[Terminology]

<Numeric Variable>	Calculates the arc tangent with specified expression, and returns the result. The unit is radian.
<Equation>	Calculated value of delta Y/delta X.
<Equation 1>	delta Y
<Equation 2>	delta X

[Reference Program]

```
1 M1=Atn(100/100)
```

'PI/4 is assigned to M1.

```
2 M2=Atn2(-100,100)
```

'-PI/4 is assigned to M1.

[Explanation]

- (1) Calculates the arc tangent of a given equation. Unit is in radians.
- (2) The range of the returned value for Atn is $-\pi/2 < \text{Atn} < \pi/2$.
- (3) The range of the returned value for Atn2 is $-\pi < \text{Atn} < \pi$.
- (4) If <Equation 2> evaluates to 0, Atn2 will return PI/2 when <Equation 1> evaluates to a positive value and -PI/2 when <Equation 1> evaluates to a negative value.
- (5) In the case of Atn2, it is not possible to describe a function that contains an argument in <Equation 1> and <Equation 2>. If such a function is described, an error will be generated during execution.

NG example
M1=Atn2(Max(MA,MB), 100)

M1=Atn2(Cint(10.2), 100)

[Reference]

[Sin](#), [Cos](#), [Tan](#)

Bin\$

[Function]

Value is converted to a binary string.

[Format]

```
<Character String Variable >=Bin$(<Equation>)
```

[Reference Program]

```
1 M1=&B11111111
2 C1$=Bin$(M1)      ' C1$ will contain the character string of "11111111".
```

[Explanation]

- (1) Value is converted to a binary string.
- (2) If the equation does not evaluate to an integer, the integral value obtained by rounding the fraction will be converted to a binary string.
- (3) Val is a command that performs the opposite of this function.

[Reference]

[Hex\\$](#), [Str\\$](#), [Val](#)

CalArc

[Function]

Provides information regarding the arc that contains the three specified points.

[Format]

```
<Numeric Variable 4> = CalArc(<Position 1>, <Position 2>, <Position 3>,
                               <Numeric Variable 1>, <Numeric Variable 2>, <Numeric Variable 3>,
                               <Position Variables 1>)
```

[Terminology]

<Position 1>	Specifies the starting point of the arc.
<Position 2>	Specifies the passing point of the arc. Same as the three points in the Mvr instruction.
<Position 3>	Specifies the endpoint of the arc.
<Numeric Variable 1>	Radius of the specified arc (in mm) will be calculated and returned.
<Numeric Variable 2>	Central angle of the specified arc (in radians) will be calculated and returned.
<Numeric Variable 3>	Length of the specified arc (in mm) will be calculated and returned.
<Position Variables 1>	The center coordinates of the specified arc (in mm) will be calculated and returned (as a position data type, ABC are all zeros).
<Numeric Variable 4>	Return value 1 : Calculation was performed normally. -1 : Of positions 1, 2, and 3, either two points had the exact same position or all three points were on a straight line. -2 : All three points are at approximately the same position.

[Reference Program]

```
1 M1=CalArc(P1,P2,P3,M10,M20,M30,P10)
2 If M1<>1 Then End      ' Ends if an error occurs.
3 MR=M10                  ' Radius.
4 MRD=M20                 ' Circular arc angle.
5 MARCLEN=M30              ' Circular arc length.
6 PC=P10                  ' Coordinates of the center point.
```

[Explanation]

- (1) Provides information regarding the arc that is determined by the three specified points, position 1, position 2 and position 3.
- (2) If the arc generation and calculation of various values succeeded, 1 will be returned as the return value.
- (3) If some points have the exact same position or if all three points are on a straight line, -1 will be returned as the return value. In such cases, the distance between the starting point and the endpoint will be returned as the arc length, -1 as the radius, 0 as the central angle, and (0, 0, 0) as the center point.
- (4) If circular arc generation fails, -2 will be returned as the return value. If a circular arc cannot be generated, -1, 0, 0 and (0, 0, 0) are returned as the radius, central angle, arc length and center point, respectively.
- (5) It is not possible to describe a function that contains an argument in <position 1>, <position 2>, <position 3>, <numeric variable 1>, <numerical variable 2>, <numeric variable 3> and <position variable 1>. If such a function is described, an error will be generated during execution.

Chr\$

[Function]

Returns the character that has the character code obtained from the specified equation.

[Format]

```
<Character String Variable >=Chr$(<Equation>)
```

[Reference Program]

```
1 M1=&H40
2 C1$=Chr$(M1+1)           ' "A" is assigned to C1$.
```

[Explanation]

- (1) Returns the character that has the character code obtained from the specified equation.
- (2) If the equation does not evaluate to an integer, the character will be returned whose character code corresponds to the integral value obtained by rounding the fraction.

[Reference]

Asc

Cint

[Function]

Rounds the fractional part of an equation to convert the value into an integer.

[Format]

```
<Numeric Variable>=Cint(<Equation>)
```

[Reference Program]

```
1 M1=Cint(1.5)      ' 2 is assigned to M1.
2 M2=Cint(1.4)      ' 1 is assigned to M2.
3 M3=Cint(-1.4)     ' -1 is assigned to M3.
4 M4=Cint(-1.5)     ' -2 is assigned to M4.
```

[Explanation]

- (1) Returns the value obtained by rounding the fractional part of an equation.

[Reference]

[Int](#), [Fix](#)

CkSum

[Function]

Calculates the checksum of the string.

[Format]

```
<Numeric Variable>=CkSum(<Character String>, <Equation 1>, <Equation 2>)
```

[Terminology]

<Character String>	Specifies the string from which the checksum should be calculated.
<Equation 1>	Specifies the first character position from where the checksum calculation starts.
<Equation 2>	Specifies the first character position from where the checksum calculation ends.

[Reference Program]

```
1 M1=CkSum("ABCDEFG",1,3) ' &H41("A")+\&H42("B")+\&H43("C")=&HC6 is assigned to M1.
```

[Explanation]

- (1) Adds the character codes of all characters in the string from the starting position to the end position and returns a value between 0 and 255.
- (2) If the starting position is outside the range of the string, an error will be generated.
- (3) If the end position exceeds the end of the string, checksum from the starting position to the last character in the string will be calculated.
- (4) If the result of addition exceeds 255, a degenerated value of 255 or less will be returned.
- (5) It is not possible to describe a function that contains an argument in <Character String>, <Equation 1> and <Equation 2>. If such a function is described, an error will be generated during execution.

Cos

[Function]

Gives the cosine.

[Format]

```
<Numeric Variable>=Cos(<Equation>)
```

[Reference Program]

```
1 M1=Cos(Rad(60))
```

[Explanation]

- (1) Calculates the cosine of the equation.
- (2) The range of arguments will be the entire range of values that are allowed.
- (3) The range of the return value will be from -1 to 1.
- (4) The unit of arguments is in radians.

[Reference]

[Sin](#), [Tan](#), [Atn/Atn2](#)

Cvi

[Function]

Converts the character codes of the first two characters of a string into an integer.

[Format]

```
<Numeric Variable>=Cvi(<Character String Expression>)
```

[Reference Program]

```
1 M1=Cvi("10ABC")           ' &H3031 is assigned to M1.
```

[Explanation]

- (1) Converts the character codes of the first two characters of a string into an integer.
- (2) An error will be generated if the string consists of one character or less.
- (3) Mki\$ can be used to convert numerical values into a string.
- (4) This can be used to reduce the amount of communication data when transmitting numerical data with external devices.

[Reference]

[Asc](#), [Cvs](#), [Cvd](#), [Mki\\$](#), [Mks\\$](#), [Mkd\\$](#)

Cvs

[Function]

Converts the character codes of the first four characters of a string into a single precision real number.

[Format]

```
<Numeric Variable>=Cvs(<Character String Expression>)
```

[Reference Program]

```
1 M1=Cvs("FFFF")           ' 12689.6 is assigned to M1.
```

[Explanation]

- (1) Converts the character codes of the first four characters of a string into a single-precision real number.
- (2) An error will be generated if the string consists of three character or less.
- (3) Mks\$ can be used to convert numerical values into a string.

[Reference]

[Asc](#), [Cvi](#), [Cvd](#), [Mki\\$](#), [Mks\\$](#), [Mkd\\$](#)

Cvd

[Function]

Converts the character codes of the first eight characters of a string into a double precision real number.

[Format]

```
<Numeric Variable>=Cvd(<Character String Expression>)
```

[Reference Program]

```
1 M1=Cvd("FFFFFFF")      ' +3.52954E+30 is assigned to M1.
```

[Explanation]

- (1) Converts the character codes of the first eight characters of a string into a double precision real number.
- (2) An error will be generated if the string consists of seven character or less.
- (3) Mkd\$ can be used to convert numerical values into a string.

[Reference]

[Asc](#), [Cvi](#), [Cvs](#), [Mki\\$](#), [Mks\\$](#), [Mkd\\$](#)

Deg

[Function]

Converts the unit of angle measurement from radians (rad) into degrees (deg).

[Format]

```
<Numeric Variable>=Deg(<Equation>)
```

[Reference Program]

```
1 P1=P_Curr  
2 If Deg(P1.C) < 170 OR Deg(P1.C) > -150 Then *NOERR  
3 Error(9100)  
4 *NOERR
```

[Explanation]

- (1) Converts the radian value of an equation into degree value.
- (2) When the posture angles of the position data are to be displayed using positional constants, the unit used for ((500, 0, 600, 180, 0, 180) (7, 0)) is Deg. As in the case of P1.C, the unit used will be in radians (rad) when the rotational element of the positional variable is to be referenced directly. Value of P1.C can be handled in Deg. In such case, set parameter "PRGMDeg" to 1.

[Reference]

Rad

Dist

[Function]

Calculates the distance between two points (position variables).

[Format]

```
<Numeric Variable>=Dist(<Position 1>, <Position 2>)
```

[Reference Program]

```
1 M1=Dist(P1,P2)           ' M1 will contain the distance between positions 1 and 2.
```

[Explanation]

- (1) Returns the distance between positions 1 and 2 (in mm).
- (2) Posture angles of the position data will be ignored; only the X, Y, and Z data will be used for calculation.
- (3) The joint variables cannot be used. Trying to use it will result in an error during execution.
- (4) It is not possible to describe a function that contains an argument in <position 1> and <position 2>. If such a function is described, an error will be generated during execution.

Exp

[Function]

Calculates exponential functions. (an equation that uses "e" as the base.)

[Format]

```
<Numeric Variable>=Exp(<Equation>)
```

[Reference Program]

```
1 M1=Exp(2)           ' e2 is assigned to M1.
```

[Explanation]

- (1) Returns the exponential function value of the equation.

[Reference]

Ln

Fix

[Function]

Returns the integral portion of the equation.

[Format]

```
<Numeric Variable>=Fix(<Equation>)
```

[Reference Program]

```
1 M1=Fix(5.5)           ' 5 is assigned to M1.
```

[Explanation]

- (1) Returns the integral portion of the equation value.
- (2) If the equation evaluates to a positive value, the same number as Int will be returned.
- (3) If the equation evaluates to a negative value, then for instance Fix(-2.3) = -2.0 will be observed.

[Reference]

[Cint](#), [Int](#)

Fram

[Function]

Calculates the position data that indicates a coordinate system (plane) specified by three position data. Normally, use Def Plt and Plt instructions for pallet calculation.

[Format]

```
<Numeric Variable 4>=Fram(<Numeric Variable 1>, <Numeric Variable 2>,
                           <Numeric Variable 3>)
```

[Terminology]

<Numeric Variable 1>	This will be the origin of X, Y, and Z of the plane to be specified by three positions. A variable or a constant.
<Numeric Variable 2>	A point on the X axis of the plane to be specified by three positions. A variable or a constant.
<Numeric Variable 3>	A point in the positive Y direction of the X-Y plane on the plane to be specified by three positions. A variable or a constant.
<Numeric Variable 4>	Variable to which the result is assigned. Substitute the structural flag by the value of <position 1>.

[Reference Program]

```

1 Base P_NBase
2 P100=Fram(P1,P2,P3)      ' Create P100 coordinate system based on P1, P2 and P3 positions.
3 P10=Inv(P10)
4 Base P10                  ' Position of P100 will be used as the origin for robot.
:
```

[Explanation]

- (1) This can be used to define the base coordinate system.
 - (2) This creates a plane from the three coordinates X, Y, and Z for the three positions to calculate the position of the origin and the inclination of the plane, and returns the result as a position variable. The X, Y, and Z coordinates of the position data will be identical to that of position variable 1, while A, B, and C will be the inclination of the plane to be specified by the three positions.
 - (3) Since the return value is a position data, an error will be generated if a joint variable is used in the left-hand side.
 - (4) It is not possible to describe a function that contains an argument in <position 1>, <position 2> and <position 3>. If such a function is described, an error will be generated during execution.
- NG example
P10=Fram(FPm(P01,P02,P03), P04, P05)

[Reference]

Relative conversion (* operator). Refer to Page 372, "5.8 About user-defined area".

Hex\$

[Function]

Converts the value of an equation (Between -32768 to 32767) into hexadecimal string.

[Format]

```
<Character String Variable>=Hex$(<Equation> [, <Number of output characters>])
```

[Reference Program]

```
1 C1$=Hex$(&H41FF)      ' "41FF" is assigned to C1$.
2 C2$=Hex$(&H41FF,2)    ' "FF" is assigned to C2$.
```

[Explanation]

- (1) Converts the value of an equation into hexadecimal string.
- (2) If <Number of output characters> is specified, the right most part of the converted string is output for the specified length.
- (3) If the numerical value is not an integer, the integer value obtained by rounding the fraction will be converted into hexadecimal string.
- (4) Val is a command that performs this procedure in reverse.
- (5) If <number of output characters> is specified, it is not possible to describe a function that contains an argument in <Equation>. If such a function is described, an error will be generated during execution.
NG example C1\$=Hex\$(Asc("a"),1)

[Reference]

[Bin\\$](#), [Str\\$](#), [Val](#)

Int

[Function]

Returns the largest integer that does not exceed the value of the equation.

[Format]

```
<Numeric Variable>=Int(<Equation>)
```

[Reference Program]

```
1 M1=Int(3.3)      ' 3 is assigned to M1.
```

[Explanation]

- (1) Returns the largest integer that does not exceed the value of the equation.
- (2) If the equation evaluates to a positive value, the same number as Fix will be returned.
- (3) If the equation evaluates to a negative value, then for instance Fix(-2.3) = -3.0 will be observed.

[Reference]

[Cint](#), [Fix](#)

Inv

[Function]

Obtains the position data of the inverse matrix of the position variable. This is used to perform relative calculation of the positions.

[Format]

<Position Variables>=Inv(<Position Variables>)

[Reference Program]

1 P1=Inv(P2) ' P1 will contain the inverse matrix of P2.

[Explanation]

- (1) Obtains the position data of the inverse matrix of the position variable.
- (2) Joint variables cannot be used as the argument. When a joint variable is used, an error will be generated.
- (3) Since the return value is a position data, an error will be generated if a joint variable is used in the left-hand side.

JtoP

[Function]

Given joint data will be converted into position data.

[Format]

<Position Variables>=JtoP(<Joint Variables>)

[Reference Program]

1 P1=JtoP(J1) ' The position that expresses the J1 (joint type) position using the XYZ type will be assigned to P1.

[Explanation]

- (1) Converts the joint data into the position data.
- (2) Position variables cannot be used as the argument. When a position variable is used, an error will be generated.
- (3) Since the return value is a position data, an error will be generated if a joint variable is used in the left-hand side.

[Reference]

PtoJ

Left\$

[Function]

Obtains a string of the specified length starting from the left end.

[Format]

```
<Character String Variable >=Left$(<Character String>, <Equation>)
```

[Reference Program]

```
1 C1$=Left$("ABC",2)           ' "AB" is assigned to C1$.
```

[Explanation]

- (1) Obtains a string of the specified length starting from the left end.
- (2) An error will be generated if the value is a negative value or is longer than the string.
- (3) It is not possible to describe a function that contains an argument in <Character String> and <Equation>. If such a function is described, an error will be generated during execution.

[Reference]

[Mid\\$](#), [Right\\$](#)

Len

[Function]

Returns the length of the string.

[Format]

```
<Numeric Variable>=Len(<Character String>)
```

[Reference Program]

```
1 M1=Len("ABCDEFG")           ' 7 is assigned to M1.
```

[Explanation]

- (1) Returns the length of the argument string.

[Reference]

[Left\\$](#), [Mid\\$](#), [Right\\$](#)

Ln

[Function]

Returns the natural logarithm. (Base e.)

[Format]

```
<Numeric Variable>=Ln(<Equation>)
```

[Reference Program]

```
1 M1=Ln(2)           ' 0.693147 is assigned to M1.
```

[Explanation]

- (1) Returns the natural logarithm of the value of the equation.
- (2) An error will be generated if the equation evaluates to a zero or a negative value.

[Reference]

[Exp](#), [Log](#)

Log

[Function]

Returns the common logarithm. (Base 10.)

[Format]

```
<Numeric Variable>=Log(<Equation>)
```

[Reference Program]

```
1 M1=Log(2)           ' 0.301030 is assigned to M1.
```

[Explanation]

- (1) Returns the common logarithm of the value of the equation.
- (2) An error will be generated if the equation evaluates to a zero or a negative value.

[Reference]

[Exp](#), [Ln](#)

Max

[Function]

Obtains the maximum value.

[Format]

```
<Numeric Variable>=Max(<Equation 1>, <Equation 2>, ...)
```

[Reference Program]

```
1 M1=Max(2,1,3,4,10,100)           ' 100 is assigned to M1.
```

[Explanation]

- (1) Returns the maximum value among the arbitrary number of arguments.
- (2) The length of this instruction can be up to the number of characters allowed in a single line (123 characters).
- (3) It is not possible to describe a function that contains an argument in <Equation 1>, <Equation 2> and
If such a function is described, an error will be generated during execution.

[Reference]

Min

Mid\$

[Function]

Returns a string of the specified length starting from the specified position of the string.

[Format]

```
<Character String Variable >=Mid$(<Character String>, <Equation 2>, <Equation 3>)
```

[Reference Program]

```
1 C1$=Mid$("ABCDEFG",3,2)           ' "CD" is assigned to C1$.
```

[Explanation]

- (1) A string of the length specified by argument 3 is extracted from the string specified by the first argument starting from the position specified by argument 2 and returned.
- (2) An error will be generated if equation 2 or 3 evaluates to a zero or a negative value.
- (3) An error is generated if the position of the last character to be extracted is larger than the length of the string specified by the first argument.
- (4) It is not possible to describe a function that contains an argument in <Character String>, <Equation 2> and <Equation 3>. If such a function is described, an error will be generated during execution.

[Reference]

[Left\\$](#), [Right\\$](#), [Len](#)

Min

[Function]

Obtains the minimum value.

[Format]

```
<Numeric Variable>=Min(<Equation 1>, <Equation 2>, .....)
```

[Reference Program]

```
1 M1=Min(2,1,3,4,10,100)           ' 1 is assigned to M1.
```

[Explanation]

- (1) Returns the minimum value among the arbitrary number of arguments.
- (2) The length of this instruction can be up to the number of characters allowed in a single line (123 characters).
- (3) It is not possible to describe a function that contains an argument in <Equation 1>, <Equation 2> and If such a function is described, an error will be generated during execution.

[Reference]

Max

Mirror\$

[Function]

Inverts the bit string representing each character code of the string in binary, and obtains the character-coded string.

[Format]

```
<Character String Variable>=Mirror$(<Character String Expression>)
```

[Reference Program]

```
1 C1$=Mirror$("BJ")           ' "RB" is assigned to C1$.
                                ' "BJ" =&H42,&H4A=&B01000010,&B01001010.
                                ' Inverted =&H52,&H42=&B01010010,&B01000010.
                                ' Output ="RB".
```

[Explanation]

- (1) Inverts the bit string representing each character code of the string in binary, and obtains the character-coded string.

Mki\$

[Function]

Converts the value of an equation (integer) into a two-byte string.

[Format]

```
<Character String Variable >=Mki$(<Equation>)
```

[Reference Program]

```
1 C1$=Mki$(20299)           ' "OK" is assigned to C1$.
2 M1=Cvi(C1$)                ' 20299 is assigned to M1.
```

[Explanation]

- (1) Converts the lowest two bytes of the value of an equation (integer) into a strings.
- (2) Use Cvi to convert the string to a value.
- (3) This can be used to reduce the amount of communication data when transmitting numerical data to external devices.

[Reference]

[Asc](#), [Cvi](#), [Cvs](#), [Cvd](#), [Mks\\$](#), [Mkd\\$](#)

Mks\$

[Function]

Converts the value of an equation (single-precision real number) into a four-byte string.

[Format]

```
<Character String Variable >=Mks$(<Equation>)
```

[Reference Program]

```
1 C1$=Mks$(100.1)           '
2 M1=Cvs(C1$)                ' 100.1 is assigned to M1.
```

[Explanation]

- (1) Converts the lowest four bytes of the value of an equation (single-precision real number) into the strings.
- (2) Use Cvs to convert the string to a value.
- (3) This can be used to reduce the amount of communication data when transmitting numerical data to external devices.

[Reference]

[Asc](#), [Cvi](#), [Cvs](#), [Cvd](#), [Mki\\$](#), [Mkd\\$](#)

Mkd\$

[Function]

Converts the value of an equation (double-precision real number) into a eight-byte string.

[Format]

<Character String Variable >=Mkd\$(<Equation>)

[Reference Program]

```
1 C1$=Mkd$(10000.1)
2 M1=Cvd(C1$)           ' 10000.1 is assigned to M1.
```

[Explanation]

- (1) Converts the lowest eight bytes of the value of an equation (single-precision real number) into the strings.
- (2) Use Cvd to convert the string to a value.
- (3) This can be used to reduce the amount of communication data when transmitting numerical data to external devices.

[Reference]

[Asc](#), [Cvi](#), [Cvs](#), [Cvd](#), [Mki\\$](#), [Mki\\$](#)

PosCq

[Function]

Checks whether the given position is within the movement range.

[Format]

<Numeric Variable>=PosCq(<Position Variables>)

[Reference Program]

```
1 M1=PosCq(P1)           ' M1 will contain 1 if the position P1 is within the movement range.
```

[Explanation]

- (1) Check whether the position data given by an argument is within the movement range of the robot. Value 1 will be returned if it is within the movement range of the robot; value 0 will be returned if it is outside the movement range of the robot.
- (2) Arguments must give either the position data type or joint data type.

PosMid

[Function]

Obtain the middle position data when a linear interpolation is performed between two given points.

[Format]

```
<Position Variables>=PosMid(<Position Variables 1>, <Position Variables 2>,<Equation 1>,
<Equation 2>)
```

[Reference Program]

```
1 P1=PosMid(P2,P3,0,0)
```

' The position data (including posture) of the middle point between P2 and P3 will be assigned to P1.

[Explanation]

- (1) Obtain the position data of the middle point when a linear interpolation is performed between two position data.
- (2) The first argument gives the starting point of the linear interpolation, while the second argument gives the endpoint of the linear interpolation.
- (3) The third and fourth arguments correspond to the two TYPE arguments of the Mvs command.
- (4) The arguments for the starting and end points must be positions that allow linear interpolation with the specified interpolation type. For instance, an error will be generated if the structure flags of the starting and end points are different.
- (5) It is not possible to describe a function that contains an argument in <Position Variables 1>, <Position Variables 2>,<Equation 1> and <Equation 2>. If such a function is described, an error will be generated during execution.

PtoJ

[Function]

Converts the given position data into a joint data.

[Format]

```
<Joint Variable>=PtoJ(<Position Variables>)
```

[Reference Program]

```
1 J1=PtoJ(P1)
```

' J1 will contain the value of P1 (XYZ position variable) that has been converted into joint data type.

[Explanation]

- (1) Converts the position data into the joint data.
- (2) Joint variables(J variable) cannot be used as the argument. When a joint variable is used, an error will be generated.
- (3) Since the return value is a joint data, an error will be generated if a position variable is used in the left-hand side.

[Reference]

JtoP

Rad

[Function]

Converts the unit of angle measurement from degrees (deg) into radians (rad).

[Format]

<Numeric Variable>=Rad(<Equation>)

[Reference Program]

1 P1=P_Curr

2 P1.C=Rad(90)

3 Mov P1 ' Moves to P1, which is obtained by changing the C axis of the current position to 90 degrees.

[Explanation]

(1) Converts the degree value of an equation into radian value.

(2) This can be used to assign values to the posture components (ABC) of a position variable or to execute trigonometric functions.

[Reference]

Deg

Rdf1 1

[Function]

Returns the structure flag of the specified position using character data "R"/"L", "A"/"B", and "N"/"F".

[Format]

<Character String Variable >=Rdf1(<Position Variables>, <Equation>)

[Terminology]

<Position Variables> Specifies the position variable from which the structure flag will be extracted.

<Equation> Specifies which structure flag is to be extracted.

0 = "R" / "L", 1 = "A" / "B", 2 = "N" / "F"

[Reference Program]

1 P1=(100,0,100,180,0,180)(7,0)' Since the structure flag 7 (&B111) is RAN,

2 C1\$=Rdf1(P1,1) ' C1\$ will contain "A".

[Explanation]

(1) Of the structure flags in the position data specified by argument 1, the flag specified by argument 2 will be extracted.

(2) This function extracts information from the FL1 element of position data.

(3) It is not possible to describe a function that contains an argument in <Position Variables> and <Equation>. If such a function is described, an error will be generated during execution.

[Reference]

[Rdf1 2](#), [Setfl 1](#), [Setfl 2](#)

Rdf1 2

[Function]

Returns the multiple rotation information of the specified joint axis.

[Format]

```
<Numeric Variable>=Rdf12(<Position Variables>, <Equation>)
```

[Terminology]

<Position Variables>	Specifies the position variable from which the multiple rotation information is to be extracted.
<Equation>	Specifies the value for the joint axis from which the multiple rotation information is to be extracted. (1 through 8)

[Reference Program]

```
1 P1=(100,0,100,180,0,180)(7,&H00100000) '
2 M1=Rdf12(P1,6)                                ' 1 is assigned to M1.
```

[Explanation]

- (1) Of the multiple rotation information of the position data specified by argument 1, the value for the joint axis specified by argument 2 is extracted.
- (2) The range of the return value is between -8 and 7.
- (3) This function extracts information from the FL2 element of position data.
- (4) Structure flag 2 (multiple rotation information) has a 32-bit structure, which contains 4 bits of information per axis for 8 axes.
- (5) When displaying in T/B and the multiple rotation is a negative value, value of -1 to -8 is converted into F to 8 (4-bit signed hexadecimal notation) and displayed.

<Sample display of multiple rotation information in TB>	87654321 axis	<Relationship between display and number of multiple rotations per axis>
When multiple rotation of axis J6 is +1:	FL2=00100000 -2 -1 0 +1 +2
When multiple rotation of axis J6 is -1:	FL2=00F00000 E F 0 1 2

- (6) It is not possible to describe a function that contains an argument in <Position Variables> and <Equation>. If such a function is described, an error will be generated during execution.

[Reference]

[Rdf1 1](#), [Setfl 1](#), [Setfl 2](#), [JRC \(Joint Roll Change\)](#)

Rnd

[Function]

Generates a random number.

[Format]

```
<Numeric Variable>=Rnd(<Equation>)
```

[Terminology]

<Equation>	Specifies the initial value of random numbers. If this value is set to 0, subsequent random numbers will be generated without setting the initial value of random numbers.
<Numeric Variable>	A value in the range of 0.0 to 1.0 will be returned.

[Reference Program]

```

1 DIM MRnd(10)
2 C1=Right$(C_Time,2)      ' Initializes random numbers using the clock.
3 MRndBS=Cvi(C1)          ' in order to obtain different sequence of numbers.
4 MRnd(1)=Rnd(MRndBS)     ' Sets the initial value of random numbers and extracts the first random
                           number.
5 For M1=2 TO 10           ' Obtain other nine random numbers.
6  MRnd(M1)=Rnd(0)
7 Next M1

```

[Explanation]

- (1) Initializes random numbers using the value provided by the argument and extracts a random number.
- (2) If the equation provided as the argument evaluates to 0, initialization of random numbers will not take place and the next random number will be extracted.
- (3) When the same value is used to perform initialization of random numbers, identical random number sequence will be obtained.

Right\$

[Function]

Obtains a string of the specified length starting from the right end.

[Format]

```
<Character String Variable>=Right$(<Character String>, <Equation>)
```

[Reference Program]

```
1 C1$=Right$("ABCDEFG",3)      ' "EFG" is assigned to C1$.
```

[Explanation]

- (1) Obtains a string of the specified length starting from the right end.
- (2) An error will be generated if the value of the second argument is a negative value or is longer than the first string.
- (3) It is not possible to describe a function that contains an argument in <Character String> and <Equation>. If such a function is described, an error will be generated during execution.

[Reference]

[Left\\$](#), [Mid\\$](#), [Len](#)

Setfl 1

[Function]

Changes the structure flag of the specified position using a string (such as "RAN").

[Format]

```
<Position Variables>=Setfl1(<Position Variables>, <Character String>)
```

[Terminology]

<Position Variables> Specifies the position variable whose structure flag is to be changed.

<Character String> Specifies the structure flag to be changed. Multiple flags can be specified.

"R" or "L": Right/Left setting.

"A" or "B": Above/Below setting.

"N" or "F": Nonflip/Flip setting.

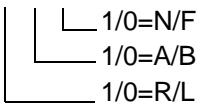
[Reference Program]

```
1 Mov P1
2 P2=Setfl1(P1,"LBF")
3 Mov P2
```

[Explanation]

- (1) Returns the position data obtained by changing the structure flags in the position data specified by argument 1 to flag values specified by argument 2.
- (2) This function changes information from the FL1 element of position data. The content of the position data given by the argument will remain unchanged.
- (3) The structure flag will be specified starting from the last character in the string. Therefore, for instance, if the string "LR" is specified, the resulting structure flag will be "L".
- (4) If the flags are changed using a numerical value, set P1.FL1=7.
- (5) Structure flags may have different meanings depending on the robot model. For details, please refer to "ROBOT ARM SETUP & MAINTENANCE" for each robot.

The structure flag corresponds to 7 in the position constant (100, 0, 300, 180, 0, 180) (7, 0). The actual position is a bit pattern.

7 = & B 0 0 0 0 0 1 1 1


- (6) It is not possible to describe a function that contains an argument in <Position Variables> and <Character String>. If such a function is described, an error will be generated during execution.

[Reference]

[Rdf1 1](#), [Rdf1 2](#), [Setfl 2](#)

Setfl 2

[Function]

Changes the multiple rotation data of the specified position.

[Format]

<Position Variables>=Setfl2(<Position Variables>, <Equation 1>, <Equation 2>)

[Terminology]

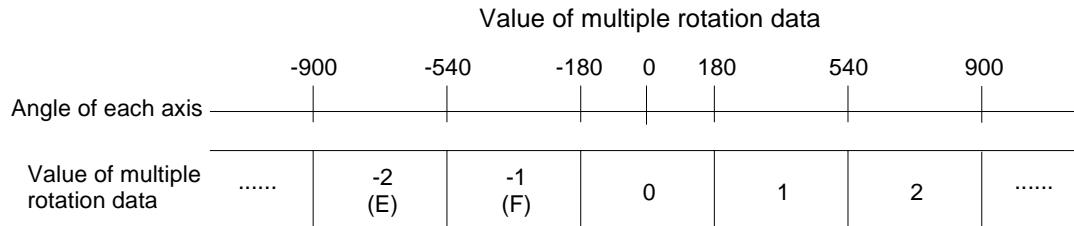
<Position Variables>	Specifies the position variable whose multiple rotation data are to be changed.
<Equation 1>	Specifies the axis number for which the multiple rotation data are to be changed. (1 through 8).
<Equation 2>	Specifies the multiple rotation data value to be changed (-8 through 7).

[Reference Program]

```
1 Mov P1
2 P2=Setfl2(P1,6,1)
3 Mov P2
```

[Explanation]

- (1) Returns the position data obtained by changing the position data's multiple rotation information of the joint axis specified by equation 1 to the value specified by equation 2.
- (2) This function changes information from the FL2 element of position data.
- (3) The content of the position of position variables given by the argument (X, Y, Z, A, B, C, and FL1) will remain unchanged.



- (4) It is not possible to describe a function that contains an argument in <Position Variables>, <Equation 1> and <Equation 2>. If such a function is described, an error will be generated during execution.

[Reference]

[Rdf1 1](#), [Rdf1 2](#), [Setfl 1](#)

SetJnt

[Function]

Sets the value to the joint variable.

[Format]

```
<<Joint Variable>>=SetJnt(<J1 Axis>[,<J2 Axis>[,<J3 Axis>[,<J4 Axis>
[,<J5 Axis>[,<J6 Axis>[,<J7 Axis>[,<J8 Axis>]]]]]))
```

[Terminology]

<Joint Variable> Sets the value to the joint variable.
<J1 Axis>--<J8 Axis> The unit is Rad (the unit is mm for direct-driven axes).

[Reference Program]

```
1 J1=J_Curr
2 For M1=0 to 60 SETP 10
3 M2=J1.J3+Rad(M1)
4 J2=SetJnt(J1.J1,J1.J2,M2)      ' Only for the value of the J3 axis, it is rotated by 10 degrees each
                                   time. The same value is used for the J4 and succeeding axes.
5 Mov J2
6 Next M1
7 M0=Rad(0)
8 M90=Rad(90)
7 J3=SetJnt(M0,M0,M90,M0,M90,M0)
10 Mov J3
```

[Explanation]

- (1) The value of each axis in joint variables can be changed.
- (2) Variable can be described as arguments.
- (3) Arguments can be omitted except for the J1 axis. They can be omitted for all subsequent axes. (Arguments such as SetJnt(10,10,,,10) cannot be described.)
- (4) In an argument, it is not allowed to describe a function with an argument. If described, an error occurs when executed.

[Reference]

SetPos

[Related parameter]

AXUNT, PRGMDEG

SetPos

[Function]

Sets the value to the Position variable

[Format]

```
<<Position Variable>>=SetPos(<X Axis>[,<Y Axis>[,<Z Axis>
[,<A Axis>[,<B Axis>[,<C Axis>[,<L1 Axis>[,<L2 Axis>]]]]]])
```

[Terminology]

<Position Variable>	Sets the value to the Position variable.
<X Axis>-<Z Axis>	The unit is mm.
<A Axis>-<C Axis>	The unit is Rad. (It can be switched to Deg using the PRGMDEG parameter.)
<L1 Axis>-<L2 Axis>	The unit depends on "AXUNT" Parameter.

[Reference Program]

```

1 P1=P_Curr
2 For M1=0 to 100 SETP 10
3 M2=P1.Z+M1
4 P2=SetPos(P1.X, P1.Y, M2)      ' Only for the value of the Z axis, it is rotated by 10 mm each time.
The same value is used for the A and succeeding axes.
5 Mov J2
6 Next M1

```

[Explanation]

- (1) The value of each axis in joint variables can be changed.
- (2) Variable can be described as arguments.
- (3) Arguments can be omitted except for the X axis. They can be omitted for all subsequent axes. (Arguments such as SetPos(10,10,,,10) cannot be described.)
- (4) In an argument, it is not allowed to describe a function with an argument. If described, an error occurs when executed.

[Reference]

SetJnt

[Related parameter]

AXUNT, PRGMDEG

Sgn

[Function]

Checks the sign of the equation.

[Format]

```
<Numeric Variable>=Sgn(<Equation>)
```

[Reference Program]

```
1 M1=-12
2 M2=Sgn(M1)      ' -1 is assigned to M2.
```

[Explanation]

(1) Checks the sign of the equation and returns the following value.

Positive value 1

0 0

Negative value -1

Sin

[Function]

Calculates the sine.

[Format]

```
<Numeric Variable>=Sin(<Equation>)
```

[Reference Program]

```
1 M1=Sin(Rad(60))      ' 0.866025 is assigned to M1.
```

[Explanation]

(1) Calculates the sine to which the given equation evaluates.

(2) The range of values will be the entire range that numerical values can take.

(3) The range of the return value will be from -1 to 1.

(4) The unit of arguments is in radians.

[Reference]

[Cos](#), [Tan](#), [Atn/Atn2](#)

Sqr

[Function]

Calculates the square root of an equation value.

[Format]

```
<Numeric Variable>=Sqr(<Equation>)
```

[Reference Program]

```
1 M1=Sqr(2)           ' 1.414214 is assigned to M1.
```

[Explanation]

- (1) Calculates the square root of the value to which the given equation evaluates.
- (2) An error will be generated if the equation given by the argument evaluates to a negative value.

Strpos

[Function]

Searches for a specified string in a string.

[Format]

```
<Numeric Variable>=Strpos(<Character String 1>, <Character String 2>)
```

[Reference Program]

```
1 M1=Strpos("ABCDEFG","DEF")      ' 4 is assigned to M1.
```

[Explanation]

- (1) Returns the position of the first occurrence of the string specified by argument 2 from the string specified by argument 1.
- (2) An error will be generated if the length of the argument 2 is 0.
- (3) For instance, if argument 1 is "ABCDEFG" and argument 2 is "DEF", 4 will be returned.
- (4) If the search string could not be found, 0 will be returned.
- (5) It is not possible to describe a function that contains an argument in <Character String 1> and <Character String 2>. If such a function is described, an error will be generated during execution.

Str\$

[Function]

Converts the value of the equation into a decimal string.

[Format]

```
<Character String Variable >=Str$(<Equation>)
```

[Reference Program]

```
1 C1$=Str$(123)           ' "123" is assigned to C1$.
```

[Explanation]

- (1) Converts the value of the equation into a decimal string.
- (2) Val is a command that performs this procedure in reverse.

[Reference]

[Bin\\$](#), [Hex\\$](#), [Val](#)

Tan

[Function]

Calculates the tangent.

[Format]

```
<Numeric Variable>=Tan(<Equation>)
```

[Reference Program]

```
1 M1=Tan(Rad(60))           ' 1.732051 is assigned to M1.
```

[Explanation]

- (1) Returns the tangent of the value to which the equation evaluates.
- (2) The range of arguments will be the entire range of values that are allowed.
- (3) The range of return values will be the entire range that numerical values can take.
- (4) The unit of arguments is in radians.

[Reference]

[Sin](#), [Cos](#), [Atn/Atn2](#)

Val

[Function]

Converts the value in the string into a numerical value.

[Format]

```
<Numeric Variable>=Val(<Character String Expression>)
```

[Reference Program]

```
1 M1=Val("15")
2 M2=Val("&B1111")
3 M3=Val("&HF")
```

[Explanation]

- (1) Converts the given character string expression string into a numerical value.
- (2) Binary (&B), decimal, and hexadecimal (&H) notations can be used for the string.
- (3) In the example above, M1, M2 and M3 evaluate to the same value (15).

[Reference]

[Bin\\$](#), [Hex\\$](#), [Str\\$](#)

Zone

[Function]

Checks if the specified position is within the specified area (a rectangular solid defined by two points).

[Format]

```
<Numeric Variable>=Zone(<Position 1>, <Position 2>, <Position 3>)
```

[Terminology]

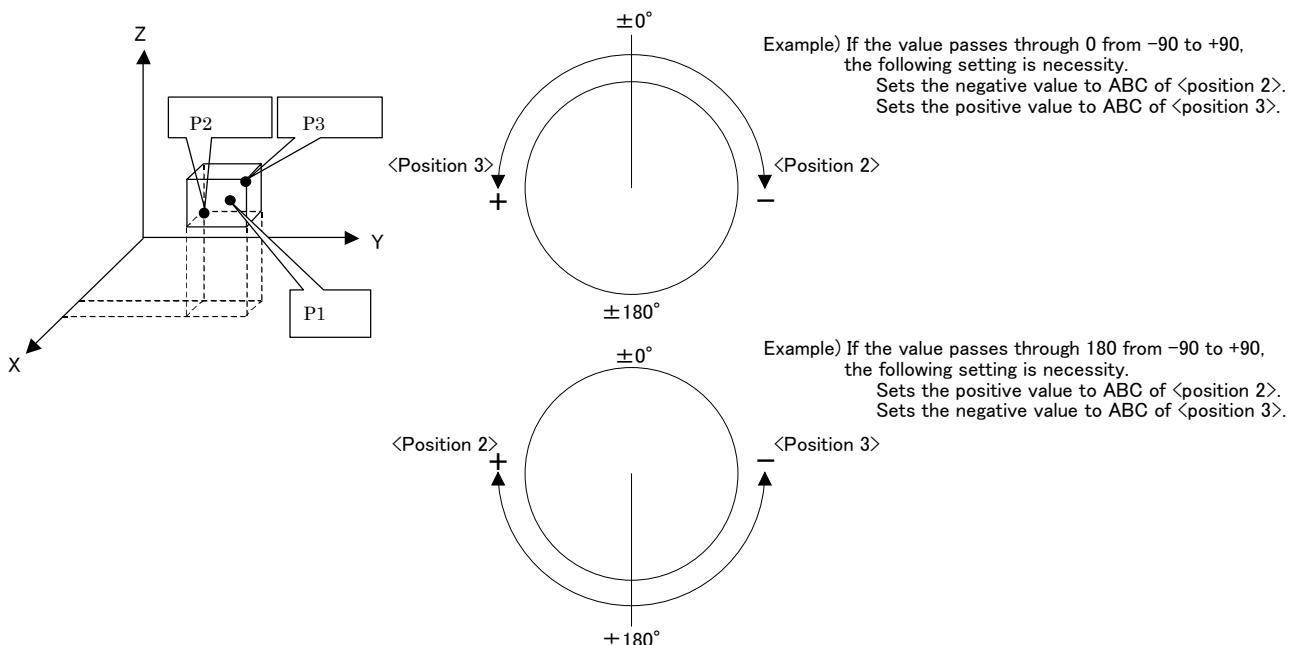
- <Position 1> The position to be checked.
 - <Position 2> The position of the first point that specifies the area.
 - <Position 3> The position of the second point that specifies the area. (diagonal point)
- Positions 1 to 3 set the XYZ coordinates variable system (P variable X, Y, Z, A, B, C, L1 and L2).

[Reference Program]

```
1 M1=Zone(P1,P2,P3)
2 If M1=1 Then Mov P_Safe Else End
```

[Explanation]

- (1) This will check if position 1 is inside the rectangular solid defined by the two points, position 2 and position 3. (The two points will become the diagonal points of the rectangular solid.) If the point is inside the rectangular solid, 1 is returned; otherwise, 0 is returned.
- (2) To check whether position 1 is inside that area, each element of position 1 (X, Y, Z, A, B, C, L1 and L2) will be checked if it is between the values for position 2 and position 3.
- (3) As for the posture angles (A, B, and C), they are checked by rotating in the positive direction from the angle in position 2 to position 3 and by seeing if the target value is inside the swiped range.
Example) If P2.A is -100 and P3.A is +100, if P1.A is 50, the value is within the range. Similar checking will be performed for B and C axes. (Refer to diagram below.)
- (4) For components that are not checked or do not exist, if the unit is in degrees, position 2 will be set to -360 and position 3 will be set to 360. If the unit is in millimeters, position 2 will be set to -10000 and position 3 will be set to 10000.
- (5) It is not possible to describe a function that contains an argument in <Position 1>, <Position 2> and <Position 3>. If such a function is described, an error will be generated during execution.



Zone 2

[Function]

Checks if the specified position is within the specified area (Cylindrical area defined by two points).

[Format]

<Numeric Variable>=Zone2(<Position 1>, <Position 2>, <Position 3>, <Equation>)

[Terminology]

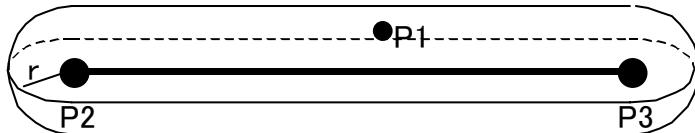
<Position 1>	The position to be checked.
<Position 2>	The position of the first point that specifies the area.
<Position 3>	The position of the second point that specifies the area.
<Equation>	Radius of the hemisphere on both ends.

[Reference Program]

```
1 M1=Zone2(P1,P2,P3,50)
2 If M1=1 Then Mov P_Safe Else End
```

[Explanation]

- (1) This will check if position 1 is inside the cylindrical area (Refer to diagram below) defined by the two points, position 2 and position 3, and the radius represented by the equation. If the point is inside the space, 1 is returned; otherwise, 0 is returned.
- (2) This function checks whether the check position (X, Y, and Z coordinates) is within the specified area, but does not take the posture components into consideration.



- (3) It is not possible to describe a function that contains an argument in <Position 1>, <Position 2>, <Position 3> and <Equation>. If such a function is described, an error will be generated during execution.

Zone3

[Function]

Checks if the specified position is within the specified area (The cube which consists of the three points).

[Format]

```
< Numeric Variable > = Z o n e 3 (<Position 1>, <Position 2>, <Position 3>, <Position 4>,
                                         <Equation W>,<Equation H>,<Equation L>)
```

[Terminology]

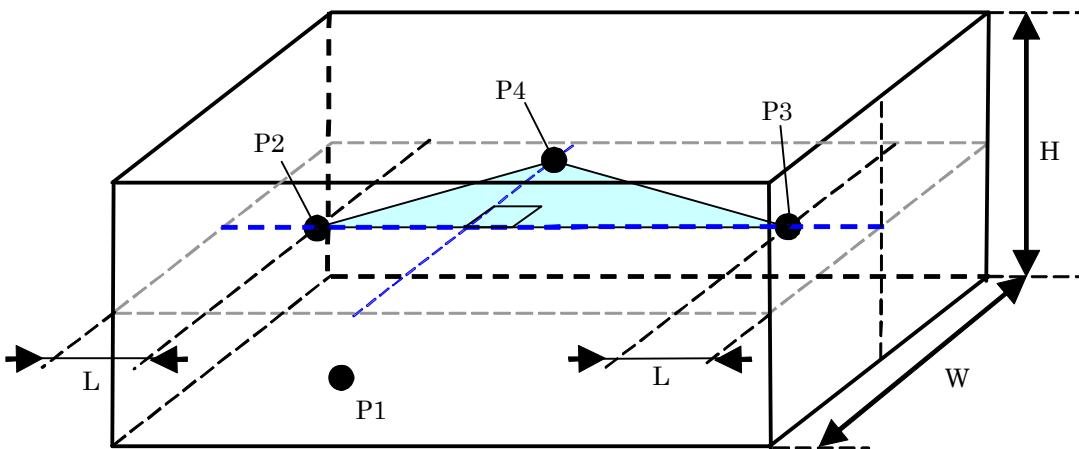
<Position 1>	The position to be checked
<Position 2>	The position of the first point that specifies the area.
<Position 3>	The position of the second point that specifies the area.
<Position 4>	The position of the point of specifying the plane which constitutes the area with <the position 2> and <the position 3>
<Equation W>	Width of the cube which constitutes the area. [mm]
<Equation H>	Height of the cube which constitutes the area. [mm]
<Equation L>	Each depth from <the position 2> and <the position 3> of the cube which constitutes the area. [mm]

[Reference Program]

```
1 M1=Zone3(P1, P2, P3, P4, 100, 100, 50)
2 If M1=1 Then Mov P_Safe Else End
```

[Explanation]

- 1) This will check if position 1 is inside the cube area (Refer to diagram below) defined by the three points, position 2, position 3 and position 4, and the Equation W, Equation H and Equation L.
If the point is inside the space, 1 is returned; otherwise, 0 is returned.
- 2) This function checks whether the check position (X, Y, and Z coordinates) is within the specified area, but does not take the posture components into consideration.



- 3) It is not possible to describe a function that contains an argument in <Position 1>, <Position 2>, <Position 3>, <Position 4>, <Equation W>, <Equation H> and <Equation L>. If such a function is described, an error will be generated during execution.
- 4) If the negative value is inputted into <Equation W> and <Equation H>, the error occurs.
- 5) Since the specified area cannot be created if the same position or the position on the same straight line is inputted into <Position 2>–<Position 4>, return -1, without checking.

By the negative number, <Equation L> returns -1, without checking, if the absolute value is less than the half of the distance for <Position 2> and <Position 3>.

5 Functions set with parameters

This controller has various parameters listed in [Table 5-2](#). It is possible to change various functions and default settings by changing the parameter settings.

No.	Classification	Content	Reference
1	Movement parameter	These parameters set the movement range, coordinate system and the items pertaining to the hand of the robot.	Page 347
2	Signal parameter	These parameters set the items pertaining to signals.	Page 355
3	Operation parameter	These parameters set the items pertaining to the operations of the controller, T/B and so forth.	Page 360
4	Command parameter	These parameters set the items pertaining to the robot language.	Page 363
5	Communication parameter	These parameters set the items pertaining to communications.	Page 367

For the parameters regarding dedicated I/O signals, refer to [Page 428, "6.3 Dedicated input/output"](#). After changing the parameters, make sure to turn the robot controller's power OFF and then turn ON.

Parameter settings will not be in effect until the power is turned on again. For detailed operating method for parameters, refer to [Page 74, "3.14 Operation of maintenance screen"](#).

⚠ CAUTION

When changing parameters, check thoroughly the function and setting values first. Otherwise, the robot may move unexpectedly, which could result in personal injury or property damage.

5.1 Movement parameter

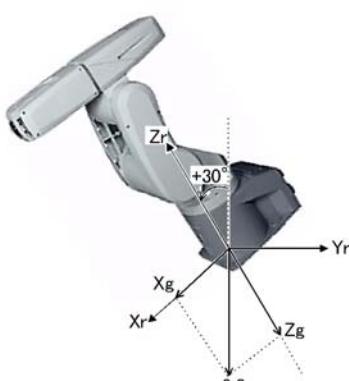
These parameters set the movement range, coordinate system and the items pertaining to the hand of the robot.

Table 5-1:List Movement parameter

Parameter	Parameter name	No. of arrays No. of characters	Details explanation	Factory setting
Joint movement range	MEJAR	Real value 16	Set the overrun limit value for the joint coordinate system. Sets the movement range for each axis. Expanding of the movement range is not recommended, since there is possibility that the robot may strike the mechanical stopper. Set the minus and plus directions. (-J1,+J1,-J2,+J2,.....-J8,+J8) Unit:deg	Setting value for each mechanism
XYZ movement range	MEPAR	Real value 6	Set the overrun limit value for the XYZ coordinate system. The movement range of the robot will be limited based on XYZ coordinate system. This can be used to prevent the robot from striking peripheral devices during manual operation when the robot is installed within the device. Set the minus and plus directions. (-X,+X,-Y,+Y,-Z,+Z) Unit:mm	(-X,+X,-Y,+Y,-Z,+Z)= -10000,10000, -10000,10000, -10000,10000
Standard tool coordinates Refer to "5.6 Standard Tool Coordinates".	MEXTL	Real value 6	Initial values will be set for the hand tip (control point) and the mechanical interface (hand mounting surface). The factory default setting is set to the mechanical interface as the control point. Change this value if a hand is installed and the control point needs to be changed to the hand tip. (This will allow posture control at the hand tip for XYZ or tool jog operation.) (X, Y, Z, A, B, C) Unit: mm, ABC deg.	(X,Y,Z,A,B,C) = 0.0,0.0,0.0,0.0,0.0,0.0
Tool coordinate 1 Refer to "M_Tool"	MEXTL1	Real value 6	If the M_Tool variable is substituted by 1, the tool data can be switched using this parameter value.	(X,Y,Z,A,B,C) = 0.0,0.0,0.0,0.0,0.0,0.0
Tool coordinate 2 Refer to "M_Tool"	MEXTL2	Real value 6	If the M_Tool variable is substituted by 2, the tool data can be switched using this parameter value.	(X,Y,Z,A,B,C) = 0.0,0.0,0.0,0.0,0.0,0.0

Parameter	Parameter name	No. of arrays No. of characters	Details explanation	Factory setting
Tool coordinate 3 Refer to "M_Tool"	MEXTL3	Real value 6	If the M_Tool variable is substituted by 3, the tool data can be switched using this parameter value.	(X,Y,Z,A,B,C) = 0,0,0,0,0,0,0,0,0,0,0,0
Tool coordinate 4 Refer to "M_Tool"	MEXTL4	Real value 6	If the M_Tool variable is substituted by 4, the tool data can be switched using this parameter value.	(X,Y,Z,A,B,C) = 0,0,0,0,0,0,0,0,0,0,0,0
Tool base coordinates Refer to "5.6Standard Tool Coordinates"	MEXBS	Real value 6	Sets the positional relationship between the base coordinate system and the robot coordinate system. The factory default setting is set so that the base coordinate system and the robot coordinate system are identical. This will be set when the coordinate system for the whole device is changed. This parameter does not need to be changed very often. This is set when the coordinate system for the whole device is to be identical. (X, Y, Z, A, B, C) Unit: mm, ABC deg.	(X,Y,Z,A,B,C) = 0,0,0,0,0,0,0,0,0,0,0,0
User area Refer to "5.8About user-defined area"			Designate an area (rectangle defined with two XYZ coordinate points). A signal will be output if that area is outside the movement area (interference), or if the robot's current position is within that area. Up to eight limits can be set using the following four types of parameters. For components that are not checked or do not exist, if the unit is in degrees, set AREA*P1 to -360 and AREA*P2 to 360. If the unit is in mm, set AREA*P1 to -10000 and AREA*P2 to 10000 as the corresponding component.	
	AREA*P1 * is 1 to 8	Real value 8	Designate the first point of the area. (X, Y, Z, A, B, C) Unit: mm, ABC deg.	(X,Y,Z,A,B,C)= 0,0,0,0,0,-360.0,-360.0,-360.0
	AREA*P2 * is 1 to 8	Real value 8	Designate the secand point of the area. (X, Y, Z, A, B, C) Unit: mm, ABC deg.	(X,Y,Z,A,B,C)= 0,0,0,0,0,+360.0,+360.0,+360.0
	AREA*ME * is 1 to 8	Integer 1	Designate the mechanism No. for which the user-defined area is to be validated. The mechanism No. is 1 to 4, but normally 1 is set.	0
	AREA*AT * is 1 to 8	Integer 1	Specify the behavior of the robot when the robot enters the user definition area. (Invalid / In-zone signal output/Error output=0/1/2) Invalid:This function will be invalid. In-zone signal output:The dedicated output signal USRAREA will turn ON. Error output:An error is generated. Posture data will be ignored.	0(Invalid)
	USRAREA		Defines the number of the signal that outputs the status. Refer to Page 428, "6.3 Dedicated input/output"	-1,-1
Free plane limit Refer to "5.9Free plane limit"			This is the overrun limit set on a free plane. Create a plane with three coordinate points, and set the area that does not include the origin as the outside-movement area. Up to eight limits can be set using the following three types of parameters.	
	SFC*P * is 1 to 8	Real value 9	Designate three points for creating the plane. X1,Y1,Z1:Origin position in the plane X2,Y2,Z2:Position on the X-axis in the plane X3,Y3,Z3:Position in the positive Y direction of the X-Y plane in the plane	(X1,Y1,Z1, X2,Y2,Z2, X3,Y3,Z3)=0,0,0, 0,0,0,0,0,0,0,0 .0,0,0
	SFC*ME * is 1 to 8	Integer 1	Designate the mechanism No. for which the free plane limit is to be validated. The mechanism No. is 1 to 3, but normally 1 is set.	0
	SFC*AT * is 1 to 8	Integer 1	Designate the valid/Invalid of the set free plane limit. 0:Invalid 1: Valid (The operable area is the robot coordinate origin side.) -1: Valid (The operable area is the side where the robot coordinate origin does not exist.)	0(Invalid)
Safe point position	JSAFE	Real value 8	Specifies the safe point position. Robot moves to the safe point position if the robot program executes Mov P_Safe instruction or receives input of the SAFEPOS signal, which is an external signal. (J1,J2,J3,J4,J5,J6,J7,J8) Unit:deg	It varies with models.

Parameter	Parameter name	No. of arrays No. of characters	Details explanation	Factory setting
Mechanical stopper origin	MORG	Real value 8	Designate the mechanical stopper origin. (J1,J2,J3,J4,J5,J6,J7,J8) Unit:deg	It varies with models.
User-designated origin	USERORG	Real value 8	Designate the user-designated origin position. This normally does not need to be set. (J1,J2,J3,J4,J5,J6,J7,J8) Unit:deg	It varies with models.
User-designated origin	USERORG	Real value 8	Designate the user-designated origin position. This normally does not need to be set. (J1,J2,J3,J4,J5,J6,J7,J8) Unit:deg	(J1,J2,J3,J4,J5,J6,J7,J8)= 0.0,0.0,0.0,0.0,0.0,0.0, 0.0,0.0,0.0
Select the function of singular point adjacent alarm Refer to Page 391, "5.19 About the singular point adjacent alarm"	MESNGLSW	Integer 1	Designate the valid/invalid of the singular point adjacent alarm. (Invalid/Valid=0/1) When this parameter is set up "VALID", this warning sound is buzzing even if parameter: BZR (buzzer ON/OFF) is set up "OFF".	1(Valid)
Jog setting	JOGJSP	Real value 3	Designate the joint jog and step operation speed. (Inching H, inching L, maximum override.) Inching H: Feed amount when jog speed is set to High Unit: deg. Inching L: Feed amount when jog speed is set to Low Unit: deg. Maximum override: Operates at OP override x maximum override.	Setting value for each mechanism
	JOGPSP	Real value 3	Designate the XYZ jog and step operation speed. (Inching H, inching L, maximum override.) Inching H: Feed amount when jog speed is set to High Unit: deg. Inching L: Feed amount when jog speed is set to Low Unit: deg. Maximum override: Operates at OP override x maximum override. Operation exceeding the maximum speed 250 mm/s cannot be performed.	Setting value for each mechanism
Jog speed limit value	JOGSPMX	Real value 1	Limit the robot movement speed during the teach mode. Unit: mm/s Even if a value larger than 250 is set, the maximum value will be limited to 250.	250.0
Automatic return setting after jog feed at pause Refer to "5.10Automatic return setting after jog feed at pause"	RETPATH	Integer 1	While running a program, if the program is paused by a stop and then the robot is moved by a jog feed for instance, at the time of restart, this setting makes the robot return to the position at which the program was halted before continuing. If this function is disabled, movement instructions will be carried out from the current position until the next point. The robot does not return to the position where the program was halted. 0: Invalid . 1: Return by JOINT interpolation. 2: Return by XYZ interpolation. Note) When returning by XYZ interpolation, carry out shorter circuit movement by 3 axis XYZ interpolation. Note) In the circle interpolation (Mvc, Mvr, Mvr2, Mvr3) command, this function is valid for H4 or later. Moreover, in the circle interpolation command and the Mva command, even if set up with 0, the operation is same as 1.	1.

Parameter	Parameter name	No. of arrays No. of characters	Details explanation	Factory setting										
The gravity direction	MEGDIR	Real value 4	<p>This parameter specifies the direction and magnitude of gravitational acceleration that acts on the robot according to the installation posture for the X, Y, and Z axes of the robot coordinate system, respectively (unit: mm/second²).</p> <p>There are four elements: installation posture, gravitational acceleration in the X axis direction, gravitational acceleration in the Y axis direction, and then gravitational acceleration in the Z axis direction, in this order from the left.</p> <table border="1"> <tr> <td>Installation posture</td><td>Setting value (Installation posture, gravitational acceleration in the X axis direction, gravitational acceleration in the Y axis direction, and then gravitational acceleration in the Z axis direction)</td></tr> <tr> <td>On floor</td><td>(0.0, 0.0, 0.0, 0.0)</td></tr> <tr> <td>Against wall</td><td>(1.0, 0.0, 0.0, 0.0)</td></tr> <tr> <td>Hanging</td><td>(2.0, 0.0, 0.0, 0.0)</td></tr> <tr> <td>Optional posture^{*1}</td><td>(3.0, ***, ***, ***)</td></tr> </table> <p>The example of the setting of gravity acceleration is shown below. Example: If the robot is tilted 30 degrees forward (see the figure below):</p> <p>The direction gravity acceleration of X axis (Xg) = $9.8 \times \sin(30 \text{ degrees}) = 4.9$.</p> <p>The direction gravity acceleration of the Z axis (Zg) = $9.8 \times \cos(30 \text{ degrees}) = 8.5$.</p> <p>Note that the value is set to -8.5 because the direction is opposite to the Z axis of the robot coordinate system.</p> <p>The direction gravity acceleration of the Y axis (Yg) = 0.0</p> <p>Therefore, the set value is (3.0, 4.9, 0.0, and -8.5)</p> 	Installation posture	Setting value (Installation posture, gravitational acceleration in the X axis direction, gravitational acceleration in the Y axis direction, and then gravitational acceleration in the Z axis direction)	On floor	(0.0, 0.0, 0.0, 0.0)	Against wall	(1.0, 0.0, 0.0, 0.0)	Hanging	(2.0, 0.0, 0.0, 0.0)	Optional posture ^{*1}	(3.0, ***, ***, ***)	0.0, 0.0, 0.0, 0.0
Installation posture	Setting value (Installation posture, gravitational acceleration in the X axis direction, gravitational acceleration in the Y axis direction, and then gravitational acceleration in the Z axis direction)													
On floor	(0.0, 0.0, 0.0, 0.0)													
Against wall	(1.0, 0.0, 0.0, 0.0)													
Hanging	(2.0, 0.0, 0.0, 0.0)													
Optional posture ^{*1}	(3.0, ***, ***, ***)													
Hand initial state Refer to "5.13 About default hand status"	HANDINIT	Integer 8	<p>Set the pneumatic hand I/F output for when the power is turned ON.</p> <p>This parameter specifies the initial value when turning ON the power to the dedicated hand signals (900'S) at the robot's tip.</p> <p>To set the initial status at power ON when controlling the hand using general-purpose I/Os (other than 900'S) or CC-Link (6000'S) (specifying a signal other than one in 900'S by the HANDTYPE parameter), do not use this HANDINIT parameter, but use the ORST* parameter.</p> <p>The value set by the ORST* parameter becomes the initial value of signals at power ON.</p>	1,0,1,0,1,0,1,0										
Hand type Refer to "5.12 About the hand type"	HAND-TYPE	Character string 8	<p>Set the single/double solenoid hand type and output signal No. (D:double solenoid, S:single solenoid).</p> <p>Set the signal No. after the hand type.</p> <p>When D900 is set, the signal No. 900 and 901 will be output.</p> <p>In the case of D (double solenoid), please configure the setting so that the signals do not overlap</p>	D900,D902,D904,D906,,,										

Parameter	Parameter name	No. of arrays No. of characters	Details explanation	Factory setting
Hand and work-piece conditions (Used in optimum acceleration/deceleration and impact detection) Refer to "5.18Hand and Workpiece Conditions (optimum acceleration/deceleration settings)"			Set the hand conditions and work conditions for when Oadl ON is set with the program. Up to eight conditions can be set. The condition combination is selected with the LoadSet command.	
	HNDDAT0	Real value 7	Set the initial condition of the hand. (Designate with the tool coordinate system.) Immediately after power ON, this setting value is used. To use the impact detection function during jog operation, set the actual hand condition before using. If it is not set, erroneous detection may occur. (Weight, size X, size Y, Size Z, center of gravity X, center of gravity Y, center of gravity Z) Unit: Kg, mm	RV-3SD/3SDJ/3SDB/3SDJB 3.50, 284.00, 284.00, 286.00, 0.00, 0.00, 75.00 RV-6SD/6SDL 6.00, 213.00, 213.00, 17.00, 0.00, 0.00, 130.00 RV-12SD/12SDL 12.00, 265.00, 265.00, 22.00, 0.00, 0.00, 66.00 RH-6SDH 6.00, 99.00, 99.00, 76.00, 0.00, 0.00, 38.00 RH-12SDH 12.00, 225.00, 225.00, 30.00, 0.00, 0.00, 15.00 RH-18SDH 18.00, 258.00, 258.00, 34.00, 0.00, 0.00, 17.00 Other type are secret.
	HNDDAT* * is 1 to 8	Real value 7	Set the initial condition of the hand. (Designate with the tool coordinate system.) (Weight, size X, size Y, Size Z, center of gravity X, center of gravity Y, center of gravity Z) Unit: Kg, mm	Standard load ,0.0,0.0,0.0,0.0,0.0,0.0
	WRKDAT0	Real value 7	Set the work conditions. (Designate with the tool coordinate system.) Immediately after power ON, this setting value is used. (Weight, size X, size Y, Size Z, center of gravity X, center of gravity Y, center of gravity Z) Unit: Kg, mm	RV-SD/RH-SDH series 0.0,0.0,0.0,0.0,0.0,0.0 Other type are secret.
	WRKDAT* * is 1 to 8	Real value 7	Set the work conditions. (Designate with the tool coordinate system.) (Weight, size X, size Y, Size Z, center of gravity X, center of gravity Y, center of gravity Z) Unit: Kg, mm	0.0,0.0,0.0,0.0,0.0,0.0
	HNDHOLD* * is 1 to 8	Integer 2	Set whether to grasp or not grasp the workpiece when HOpen (or HClose) is executed. (Setting for Open, setting for Close) (No grasp/grasp = 0/1)	0,1
Maximum acceleration/deceleration setting Refer to "5.18Hand and Workpiece Conditions (optimum acceleration/deceleration settings)"	ACCMODE	Integer 1	Sets the initial value and enables/disables the optimum acceleration/deceleration mode. (Invalid/Valid=0/1)	RH-SDH/RV-SD series1 Except the above.....0

Parameter	Parameter name	No. of arrays No. of characters	Details explanation	Factory setting
Optimum acceleration/ deceleration adjustment rate	JADL	Real value 8	<p>Set the initial value (value at power ON) of the acceleration/deceleration adjustment rate (%) during optimum acceleration/deceleration. It is the rate applied to the acceleration/deceleration speed calculated by optimum acceleration/deceleration control. In the RV-SD series, high-speed operation can be performed by setting this value to a larger value. However, if the robot is operated continuously for a certain period of time at high speed, overload and overheat errors may occur. Lower the setting value if such errors occur.</p> <p>In the RV-SD series, the initial values have been set so as to prevent overload and overheat errors from occurring.</p> <p>They are applied to both the deceleration and acceleration speeds.</p> <p>* What is an overload error? An overload error occurs when the load rate reaches a certain value in order to prevent the motor from being damaged by heat from high-speed rotation.</p> <p>* What is an overheat error? An overheat error occurs when the temperature reaches a certain value in order to prevent the position detector from being damaged by heat from high-speed rotation.</p> <p>Note) This function is valid only in the RV-SD series.</p>	<p>RV-3SD/3SDJ/ 3SDB/3SDJB series 100,100,100,100, 100,100,100,100 (%)</p> <p>RV-6SD/12SD 50,50,50,50, 50,50,50,50(%)</p> <p>RV-6SDL/12SDL 35,35,35,35, 35,35,35,35(%)</p>
Impact Detection Note that this parameter cannot be used together with the multi-mechanism control function.	COL	Integer 3	<p>Define whether the impact detection function can/cannot be used, and whether it is enabled/disabled immediately after power ON.</p> <p>Element 1: The impact detection function can (1)/cannot (0) be used.</p> <p>Element 2: Enable (1)/disable (0) as the initial state at automatic operation.</p> <p>Element 3: Enable (1)/disable (0)/NOERR mode (2) during jog operation</p> <p>The NOERR mode does not issue an error even if impact is detected. It only turns off the servo. Use the NOERR mode if it is difficult to operate because of frequently occurred errors when an impact is detected. The specification depends on the setting for jog operation (element 3) in cases other than program operation (including position jump and step feed).</p>	<p>RV-SD series is 0,0,1</p> <p>RH-SDH series is 1,0,1</p>
Detection level	COLLVL	Integer 8	<p>Set the initial value of the detection level (sensitivity) of each joint axis during automatic operation.</p> <p>Setting range: 1 to 500, unit: % * If a value exceeding the setting range is specified, the closest value allowed within the range is used instead.</p>	200,200,200,200, 200,200,200,200
Detection level during jog operation	COLLVLJG	Real value 8	<p>Set the detection level (sensitivity) during jog operation (including pause status) for each joint axis. Unit: %</p> <p>Decrease the value to increase the detection level (sensitivity).</p> <p>Increase the value if an impact detection error occurs even though no impact is detected during jog operation.</p> <p>Setting range: 1 to 500, unit: %</p> <p>* If a value exceeding the setting range is specified, the closest value allowed within the range is used instead.</p>	The standard is 200,200,200,200,200,200,200,200, but it varies with models.

Parameter	Parameter name	No. of arrays No. of characters	Details explanation	Factory setting
Selection of wrist rotation angle (axis A) coordinate system	RCD	Integer 1	<p>Switch the control and display method of the wrist rotation angle (axis A of the XYZ coordinates system) of a vertical 5-axis type robot. This parameter is invalid for robots of other types.</p> <p>2: General angle method</p> <p>Control axis A such that the hand's posture is maintained if the value of axis A is the same before and after an operation. Note that there are cases where the hand's posture cannot be maintained depending on the attitude of the wrist (axis B of the XYZ coordinates system). Under normal circumstances, use this method without changing the setting at shipment from the factory.</p> <p>0/1/3 = General angle method of the E series/joint angle method/old general angle method</p> <p>These options are prepared for the compatibility with programs (position data) created for older models (e.g., RV-E3J, RV-E5NJ). To use programs (position data) created for older models, change the parameter value to the same value as the RCD value specified for the given older model.</p> <p>Note that these methods are not mutually compatible; the postures of the hand in the middle of movement and at the registered position may be different for two different values of this parameter, even if the robot is moving toward the same position data. Make sure to set the same method as when the position data was registered in order to execute the program.</p>	2 (general angle method)
Warm-up operation mode setting	WUPENA	Integer 1	<p>Designate the valid/invalid of the Warm-up operation mode.</p> <p>0: Invalid 1: Valid</p> <p>Note: If a value other than the above is set, everything will be disabled.</p> <p>Note: For multiple mechanisms, this mode is set for each mechanism.</p>	0(Invalid)
Warm-up operation mode target axis	WUPAXIS	Integer 1	<p>Specify the joint axis that will be the target of control in the warm-up operation mode by selecting bit ON or OFF in hexadecimal (J1, J2, ... from the lower bits).</p> <p>Bit ON: Target axis Bit OFF: Other than target axis</p> <p>A joint axis that will generate an excessive difference error when operated at low temperature will be a target axis.</p> <p>Note: If the bit of a non-existent axis is set to ON, it will not be a target axis.</p> <p>Note: If there is no target axis, the warm-up operation mode will be disabled.</p> <p>Note: For multiple mechanisms, this mode is set for each mechanism.</p>	<p>RV-6SD/12SD series :00111000 (J4, J5, J6 axis)</p> <p>RV-3SD/3SDB :0000110</p> <p>RV-3SDJ/3SDJB : 00000110</p> <p>The type other than the above are 0</p>
Warm-up operation mode control time	WUPTIME	Real value 2	<p>Specify the time to be used in the processing of warm-up operation mode. (Valid time, resume time) Unit: min.</p> <p>Valid time: Specify the time during which the robot is operated in the warm-up operation status and at a reduced speed. (Setting range: 0 to 60)</p> <p>Resume time: Specify the time until the warm-up operation status is set again after it has been canceled if a target axis continues to stop. (Setting range: 1 to 1440)</p> <p>Note: If a value outside the setting range is specified, it is processed as if the closest value in the setting range is specified.</p> <p>Note: If the valid time is 0 min, the warm-up operation mode will be disabled.</p> <p>Note: For multiple mechanisms, this mode is set for each mechanism.</p>	1, 60

Parameter	Parameter name	No. of arrays No. of characters	Details explanation	Factory setting
Warm-up operation override	WUPOvrd	Integer 2	<p>Perform settings pertaining to the speed in the warm-up operation status. (Initial value, ratio of value constant time) Unit: %</p> <p>Initial value: Specify the initial value of an override (warm-up operation override) to be applied to the operation speed when in the warm-up operation status. (Setting range: 50 to 100) Ratio of value constant time: Specify the duration of time during which the override to be applied to the operation speed when in the warm-up operation status does not change from the initial value, using the ratio to the valid time. (Setting range: 0 to 50)</p> <p>The correspondence between the values of warm-up operation overrides and the setting values of various elements is shown in the figure below.</p> <p>Note: If a value outside the setting range is specified, it is processed as if the closest value in the setting range is specified. Note: If the initial value of an override is 100%, the warm-up operation mode will be disabled. Note: For multiple mechanisms, this mode is set for each mechanism.</p>	70, 50
Functional setting of compliance error	CMPERR	Integer 1	<p>Setting this parameter prevents errors 2710 through 2740 (errors that occur if the position command generated in compliance control is abnormal) from occurring.</p> <p>1: Enable error generation 0: Disable error generation</p> <p>The contents of applicable errors are as follows: 2710: The displacement from the original position command is too large. 2720: Exceeded the joint limit of the compliance command 2730: Exceeded the speed of the compliance command 2740: Coordinate conversion error of the compliance command</p> <p>If these errors occur, compliance control is not functioning normally. It is thus necessary to re-examine the teaching position and the program content to correct the causes of these errors. Change this parameter value to 0 (disable error generation) only when you can determine that doing so does not cause any operational problem even if the current operation is not suspended by an error.</p>	1 (Enable error generation)

5.2 Signal parameter

These parameters set the items pertaining to signals

Table 5-2:List Signal parameter

Parameter	Parameter name	No. of arrays No. of characters	Details explanation	Factory setting																																							
Dedicated I/O signal			For the parameters of the dedicated I/O signal, refer to Page 428, "6.3 Dedicated input/output".																																								
Stop input B contact designation	INB	Integer 1	Change the dedicated input (stop) between the A contact and B contact. (A contact/B contact = 0/1)	0(A contact)																																							
Reads the program number from the numerical input when the start signal is input.	PST	Integer 1	To select a program from the normal external input signal, set the numerical input signal (IODATA) to the program number, establish the number with the program select signal (PRGSEL), and start with the START signal. If this function is enabled, the program select signal becomes unnecessary, and when the START signal turns ON, the program number is read from the numerical input signal (IODATA). (Function invalid/Valid=0/1)	0(Invalid)																																							
Robot error output	ROBOTERR	Integer 1	<p>Set the error level that opens the EMGOUT connector robot error output terminal.</p> <p>When shipped from the factory, this setting is set so that it will open at any error level.</p> <p>For example, if the warning level is ignored and either or both a low level or high level error occurs, set 3 to open this output terminal.</p> <table border="1" data-bbox="666 961 1150 1410"> <thead> <tr> <th rowspan="2">Setting</th> <th colspan="3">Error Level</th> </tr> <tr> <th>Warning</th> <th>Low</th> <th>High</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>–</td> <td>–</td> <td>–</td> </tr> <tr> <td>1</td> <td>–</td> <td>–</td> <td>Open</td> </tr> <tr> <td>2</td> <td>–</td> <td>Open</td> <td>–</td> </tr> <tr> <td>3</td> <td>–</td> <td>Open</td> <td>Open</td> </tr> <tr> <td>4</td> <td>Open</td> <td>–</td> <td>–</td> </tr> <tr> <td>5</td> <td>Open</td> <td>–</td> <td>Open</td> </tr> <tr> <td>6</td> <td>Open</td> <td>Open</td> <td>–</td> </tr> <tr> <td>7</td> <td>Open</td> <td>Open</td> <td>Open</td> </tr> </tbody> </table> <p>Refer to the following manual for details on the EMGOUT connector. Instruction Manual: Refer to "External Emergency Stop Connection" for ROBOT ARM SETUP & MAINTENANCE.</p>	Setting	Error Level			Warning	Low	High	0	–	–	–	1	–	–	Open	2	–	Open	–	3	–	Open	Open	4	Open	–	–	5	Open	–	Open	6	Open	Open	–	7	Open	Open	Open	7 (Open for any error level)
Setting	Error Level																																										
	Warning	Low	High																																								
0	–	–	–																																								
1	–	–	Open																																								
2	–	Open	–																																								
3	–	Open	Open																																								
4	Open	–	–																																								
5	Open	–	Open																																								
6	Open	Open	–																																								
7	Open	Open	Open																																								
CC-Link error release permission.	E7730	Integer 1	<p>If the controller is used without connecting CC-Link even though it is equipped with the CC-Link option, error 7730 is generated and the controller becomes inoperable. This error cannot be canceled under normal circumstances, but it becomes possible to temporarily cancel the error by using this parameter.</p> <p>(Enable temporary error cancellation/disable error cancellation = 1/0)</p> <p>This parameter becomes valid immediately after the value is changed by the T/B or Personal Computer support software. It is not necessary to turn the power supply off and on again. Note, however, that the value of this parameter returns to 0 again (it is no longer possible to cancel the error) when the power supply is turned off and on because changes of the parameter value are not stored.</p>	0 (disable error cancellation)																																							

Parameter	Parameter name	No. of arrays No. of characters	Details explanation	Factory setting								
Output signal reset pattern Refer to "5.14 About the output signal reset pattern"			Set the operation to be taken when the general-purpose output signal for the Clr command or dedicated input (OUTRESET) is reset. Signals are output in the pattern set here even when the power is turned ON. Set with a 32-bit unit for each signal using the following parameters.(OFF/ON/hold=0/1/*)									
	ORST0	Character string 4	Set the signal No. 0 to 31.	00000000,00000 000,00000000,00 000000								
	ORST32 : ORST8016	Character string 4	Set the signal No. 32 to 63. : Set the signal No. 8016 to 8047	00000000,00000 000,00000000,00 000000 :								
Output reset at reset	SLRSTIO	Integer 1	Designate the function to carry out general-purpose output signal reset when the program is reset. (Invalid/Valid=0/1)	0(Invalid)								
Multi CPU quantity setting (CRnQ-700 only)	QMLTCPUN	Integer 1	At the multi CPU system, set the number of CPU units with which the standard base unit is equipped.	2								
Multi CPU high-speed communication area setting (CRnQ-700 only)	QMLTCPUn n=1 ~ 4	Integer 4	<p>At the multi CPU system, set the number of points performing transmission and receipt between each CPU unit for the high speed communication function between multi CPU nos. 1 to 4.</p> <p>It is necessary to match the parameter settings for all CPUs. An error will occur at the sequencer CPU If the parameter settings do not match, and therefore care should be taken to ensure that the parameter settings for each CPU match.</p> <p>First element: User free area size (k points) Range: 1 to 14 (Max. *) * The max. value will differ based on the number of multi CPUs as shown below.</p> <table border="1"> <thead> <tr> <th>CPU Qty</th> <th>Setting Range</th> </tr> </thead> <tbody> <tr> <td>2</td> <td>0 ~ 14K points</td> </tr> <tr> <td>3</td> <td>0 ~ 13K points</td> </tr> <tr> <td>4</td> <td>0 ~ 12K points</td> </tr> </tbody> </table> <p>Second element: No. of auto refresh points (points) Range: 0 to 14335 The robot CPU does not support auto refresh, and therefore the number of points for auto refresh should always be set to 0.</p> <p>Third element: System area size (K points) Range: 1 or 2</p> <p>Fourth element: Multi CPU synchronous start-up (1: Yes, 2: No) Robot CPUs take some time to start up and therefore the current setting of 1 (synchronous start-up) should not be changed.</p>	CPU Qty	Setting Range	2	0 ~ 14K points	3	0 ~ 13K points	4	0 ~ 12K points	1,0,1,1
CPU Qty	Setting Range											
2	0 ~ 14K points											
3	0 ~ 13K points											
4	0 ~ 12K points											

Parameter	Parameter name	No. of arrays No. of characters	Details explanation	Factory setting
Multi CPU input offset (CRnQ-700 only)	QMLTCPUS	Integer 1	<p>At the CRnQ controller, set the robot input signal offset for the multi CPU.</p> <p>Specify an offset from G10000 in 1K word units, and read as an R/C input from the specified shared memory. Set as required if mixing other iQPlatform compatible CPUs (motion CPU or NCCPU) and you wish to prevent the shared memory used at each CPU from overlapping.</p> <p>Setting range: -1 to 14 (integer value) (-1: Not use / 0 to 14 K words)</p> <p>(A) By setting to -1, the offset will be automatically fixed based on the installed slot. (Compatible with previous versions (N4a and prior).)</p> <p>(B) By setting to 0 to 14, the input offset is set based on the value. * Refer to cases (A) and (B) in “5.2.1 About multi CPU input offsets (CRnQ-700 controller only)” on the following pages.</p> <p>Please note that by connecting multiple robots and setting this parameter to the same value (anything other than -1), it is also possible to input the same signal status from the sequencer to multiple robots almost simultaneously.</p> <p>Refer to the QCPU User’s Manual (Multi CPU System Edition) SH(Name)-080475-F for details on the multi CPU system.</p>	-1

5.2.1 About multi CPU input offsets (CRnQ-700 controller only)

(1) Case (A)

When using no offset for input (Parameter: when QMLTCPUS = -1)

Table 5-3: CPU shared memory and robot I/O signal compatibility

Sequencer (word device)		Robot (bit device)		
Output		Input		Robot CPU No.1 / 10000 to 18191
				Robot CPU No.2 / 10000 to 18191
				Robot CPU No.3 / 10000 to 18191
Input	U3E1¥G10000 to U3E1¥G10511	Output	Robot CPU No.1 / 10000 to 18191	
	U3E2¥G10000 to U3E2¥G10511		Robot CPU No.2 / 10000 to 18191	
	U3E3¥G10000 to U3E3¥G10511		Robot CPU No.3 / 10000 to 18191	

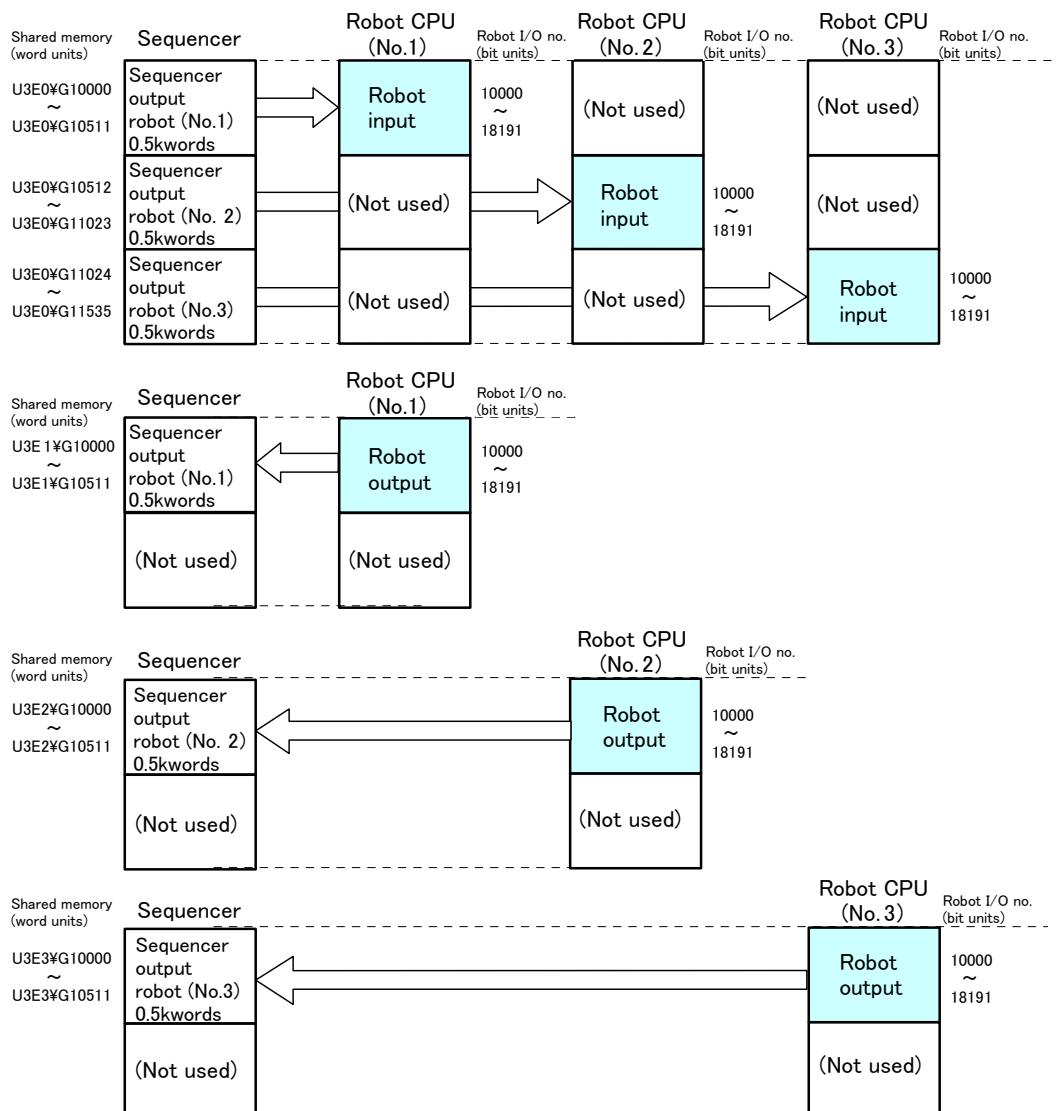


Fig.5-1: CPU shared memory and robot I/O signal compatibility (Case (A))

(2) Case (B)

When using an offset for input (Parameter: when QMLTCPUS = 0 to 14)

(*1) = (Robot CPU No.1 QMLTCPUS) * 1024

(*2) = (Robot CPU No.2 QMLTCPUS) * 1024

(*3) = (Robot CPU No.3 QMLTCPUS) * 1024

Table 5-4:CPU shared memory and robot I/O signal compatibility

Sequencer (word device)		Robot (bit device)	
Output	U3E0¥G10000+(*1) to U3E0¥G10511+(*1)	Input	Robot CPU No.1 / 10000 to 18191
	U3E0¥G10000+(*2) to U3E0¥G10511+(*2)		Robot CPU No.2 / 10000 to 18191
	U3E0¥G10000+(*3) to U3E0¥G10511+(*3)		Robot CPU No.3 / 10000 to 18191
Input	U3E1¥G10000 to U3E1¥G10511	Output	Robot CPU No.1 / 10000 to 18191
	U3E2¥G10000 to U3E2¥G10511		Robot CPU No.2 / 10000 to 18191
	U3E3¥G10000 to U3E3¥G10511		Robot CPU No.3 / 10000 to 18191

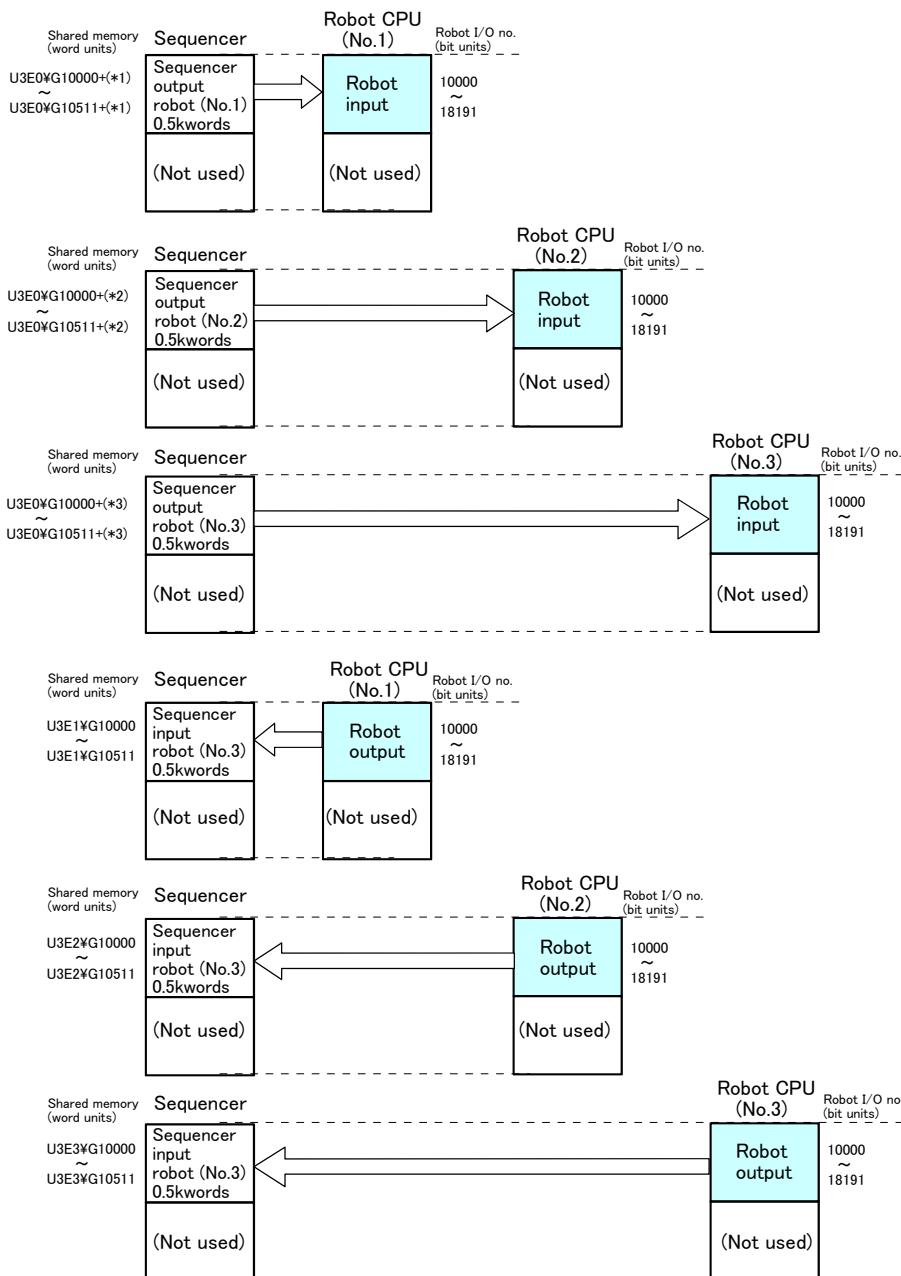


Fig.5-2:CPU shared memory and robot I/O signal compatibility (Case (B))

5.3 Operation parameter

These parameters set the items pertaining to the operations of the controller, T/B and so forth.

Table 5-5:List Operation parameter

Parameter	Parameter name	No. of arrays No. of characters	Details explanation	Factory setting
Buzzer ON/OFF	BZR	Integer 1	Specifies the on/off of the buzzer sound that can be heard when an error occurs in the robot controller. (OFF/ON=0/CP)	1(ON)
Program reset operation rights	PRSTENA	Integer 1	Whether or not the operation right is required for program reset operation (Required/Not required = 0/1) If operation rights are abandoned, program can be reset from anywhere. However, this is not possible under the teaching mode for safety reasons.	0(Required)
Program reset when key switch is switched	MDRST	Integer 1	Program paused status is canceled when the key switch is operated. (Invalid/Valid=0/1)	0(Invalid)
Operation panel display mode .	OPDISP	Integer 1	Setting of the 5-digit LED display when the key switch is changed over 0: Override display takes effect when the key switch is changed over (initial value). 1: The current display mode is maintained even after the key switch is changed over.	0 (Override display)
Program selection rights setting	OPPSL	Integer 1	Specifies the program selection operation rights when the key switch of the operation panel is in AUTO (OP) mode. (External/OP=0/1)	1(OP)
	RMTPSL	Integer 1	Designate the program selection operation rights for the automatic (Ext) mode. (External/OP=0/1)	0(External)
TB override operation rights	OVRDTB	Integer 1	Specifies whether the operation rights are required when changing override from T/B. (Not required/Required = 0/1)	1(Required)
Speed setting during mode change	OVRDMD	Integer 2	Override is set automatically when the mode is changed. First element.....override value when the mode is automatically changed from teaching mode Second element.....Override value when the mode is changed from Auto to Teaching. Current status is maintained if changed to 0.	0,0
Override change operation rights	OVRDENA	Integer 1	Specifies whether the operation rights are required when changing override from the operation panel and external device. (Not required/Required = 0/1)	1(Required)
This parameter switches the access target of a program. Refer to Page 392, "5.20 About ROM operation/high-speed RAM operation function".	ROMDRV	Integer 1	The access target of a program can be switched between RAM and ROM. 0: RAM mode. (Standard mode.) 1: ROM mode. (Special mode.) 2: High-speed RAM mode	0 (RAM mode.)
Copy the information on the RAM to the ROM Refer to Page 392, "5.20 About ROM operation/high-speed RAM operation function".	BACKUP	Character string 1	Copy the program, the parameter, the common variable, and the error log to the ROM from the RAM. Do not change this parameter.	SRAM->FLROM (unchangeable)

Parameter	Parameter name	No. of arrays No. of characters	Details explanation	Factory setting
Restore the information on the ROM to the RAM. Refer to Page 392, "5.20 About ROM operation/high-speed RAM operation function".	RESTORE	Character string 1	Restore the program, the parameter, the common variable, and the error log to the RAM from the ROM. Do not change this parameter.	FLROM->SRAM (unchangeable)
Maintenance forecast	MFENA	Integer 1	This sets whether maintenance forecast is enabled or disabled. 1: Enable 0: Disable Note) This function is limited to the RV-SD/RH-SDH series. This parameter does not take effect on models that do not support the maintenance forecast function.	RV-SD/RH-SDH series1 Except the above.....0
Maintenance forecast execution interval	MFINTVL	Integer 2	This sets the interval of collecting data for maintenance forecast. 1st element: Data collection level -- 1 (lowest) to 5 (highest) 2nd element: Forecast check execution interval (unit: hours)	1(lowest),6(hour)
Maintenance forecast announcement method	MFREPO	Integer 2	This sets the maintenance forecast announcement method. Set 0 in order to stop a warning or signal output. 1st element: 1: Generates a warning, 0: Does not generate a warning 2nd element: 1: Outputs a dedicated signal, 0: Does not output a dedicated signal	0 (Does not generate a warning) , 0 (Does not output a signal)
Resetting Maintenance Forecast Note) When reading this parameter from the teaching pendant, enter all parameter names and then read.	MFGRST	Integer 1	Reset the accumulated data relating to grease in the maintenance forecast function. * When axes generated a warning (numbered in 7530's) that prompts the replenishment of grease in the maintenance forecast function and, as a result, grease was replenished, the data relating to grease accumulated on the controller must be reset. Generally, a reset operation is performed on the Maintenance Forecast screen in Personal Computer Support software. However, if a personal computer cannot be readied, the accumulated data can be reset by entering this parameter from the teaching pendant instead.	0: Reset all axes. 1 to 8: Reset the specification axis.
	MFBRST	Integer 1	Reset the accumulated data relating to grease in the maintenance forecast function. * When axes generated a warning (numbered in 7530's) that prompts the replacement of belt in the maintenance forecast function and, as a result, the belt was replaced, the data relating to the belt accumulated on the controller must be reset. Generally, a reset operation is performed on the Maintenance Forecast screen in Personal Computer Support software . However, if a personal computer cannot be readied, the accumulated data can be reset by entering this parameter from the teaching pendant instead.	0: Reset all axes. 1 to 8: Reset the specification axis.

Parameter	Parameter name	No. of arrays No. of characters	Details explanation	Factory setting
Position Restoration Support	DJNT	Real value 8	The OP correction data obtained by the Position Restoration Support tool is input. Do not change it with any tool other than the Position Restoration Support tool. It can only be referenced on a dedicated parameter screen in the Personal Computer Support software.	It varies with models.
	MEXDTL	Real value 6	The standard tool correction data obtained by the Position Restoration Support tool is input. Do not change it with any tool other than the Position Restoration Support tool.	$(X,Y,Z,A,B,C) =$ 0.0, 0.0, 0.0, 0.0, 0.0, 0.0
	MEXDTL1	Real value 6	The correction data for tool number 1 obtained by the Position Restoration Support tool is input. Do not change it with any tool other than the Position Restoration Support tool.	$(X,Y,Z,A,B,C) =$ 0.0, 0.0, 0.0, 0.0, 0.0, 0.0
	MEXDTL2	Real value 6	The correction data for tool number 2 obtained by the Position Restoration Support tool is input. Do not change it with any tool other than the Position Restoration Support tool.	$(X,Y,Z,A,B,C) =$ 0.0, 0.0, 0.0, 0.0, 0.0, 0.0
	MEXDTL3	Real value 6	The correction data for tool number 3 obtained by the Position Restoration Support tool is input. Do not change it with any tool other than the Position Restoration Support tool.	$(X,Y,Z,A,B,C) =$ 0.0, 0.0, 0.0, 0.0, 0.0, 0.0
	MEXDTL4	Real value 6	The correction data for tool number 4 obtained by the Position Restoration Support tool is input. Do not change it with any tool other than the Position Restoration Support tool.	$(X,Y,Z,A,B,C) =$ 0.0, 0.0, 0.0, 0.0, 0.0, 0.0
	MEXDBS	Real value 6	The correction data for the base obtained by the Position Restoration Support tool is input. Do not change it with any tool other than the Position Restoration Support tool.	$(X,Y,Z,A,B,C) =$ 0.0, 0.0, 0.0, 0.0, 0.0, 0.0

5.4 Command parameter

This parameter sets the items pertaining to the program execution and robot language.

Table 5-6: List Program Execution Related Parameter

Parameter	Parameter name	No. of arrays No. of characters	Details explanation	Factory setting
No. of multi-tasks	TASKMAX	Integer 1	Designate the number of programs to be executed simultaneously.	8
Slot table (Set during multitask operation.)	SLT* * is 1 to 32	Character string 4	<p>Operation conditions for each task slot is set during multitask operations. These are set when the program is reset.</p> <p>Designate the [program name],[operation mode],[starting conditions],[order of priority].</p> <p>Program name: Selected program name. Use uppercase letters when using alphabet. Lowercase characters are not recognized.</p> <p>Operation mode: Continuous/1 cycle = REP/CYC REP:The program will be executed repeatedly. CYC:The program ends after one cycle is completed. (The program does not end if it runs in an endless loop created by a GoTo instruction.)</p> <p>Starting conditions: Normal/Error/Always =START/Error/ALWAYS START:This is executed by the START button on the operation panel or by the start signal. ALWAYS:This is executed immediately after the controller's power is turned on. This program does not affect the status such as startup. To edit a program whose attribute is set to ALWAYS, first cancel the ALWAYS attribute. A program with the ALWAYS attribute is being executed continuously and therefore cannot be edited. Change ALWAYS to START and turn on the controller's power again to stop the constant execution. Error:This is executed when an error is generated. This program does not affect the status such as startup.</p> <p>Programs with ALWAYS or Error set as the starting condition cannot execute the following movement instructions. An error will be generated if any of them is executed. Mov,Mvs,Mvr,Mvr2,Mvr3,Mvc,Mva, DRIVE,GetM,RelM,JRC</p> <p>Order of priority: 1 to 31 (31 is the maximum) This value shows the number of lines to be executed at a time. This has the same meaning as the number of lines in the Priority instruction. For instance, when two slots are used during execution, if SLT1 is set to 1 and SLT2 is set to 2, after one line of program in SLT1 is executed, two lines of program in SLT2 is executed. Therefore, more SLT2 programs will be executed and as a result, priority of SLT2 is higher.</p>	","",REP,START,1

Parameter	Parameter name	No. of arrays No. of characters	Details explanation	Factory setting
Program selection save	SLOTON	Integer 1	<p>This parameter specifies whether or not to store the program name in the SLT1 parameter at program selection, as well as whether or not to maintain the program selection status at the end of cycle operation.</p> <p>(1) Enabling program name storage at program selection (Bit 0, enable/disable storage = 1/0) Enable storage: The name of the current program is stored in the SLT1 parameter at program selection for slot 1. Moreover, the program specified in the SLT1 parameter is selected when the power supply is turned on.</p> <p>Disable storage: The name of the current program is not stored in SLT1 parameter at program selection for slot 1. In the same way as when the storage is enabled, the program specified in the SLT1 parameter is selected when the power supply is turned on.</p> <p>(2) Maintaining program at the end of cycle operation (Bit 1, maintain/do not maintain = 1/0) Maintain: The status of program selection is maintained at the end of cycle operation. The parameter value does not become P.0000.</p> <p>Do not maintain: The status of program selection is not maintained at the end of cycle operation. The parameter value becomes P.0000.</p> <p>Setting values and operations</p> <p>0: Disable storage, do not maintain 1: Enable storage, do not maintain (initial value) 2: Disable storage, maintain 3: Enable storage, maintain</p>	1(Valid)
Setting that allows the execution of X** instructions and Servo instruction in an ALWAYS program. Refer to "5.11 Automatic execution of program at power up"	ALWENA	Integer 1	<p>XRun, XLoad, XStp, XRst, Servo and Reset Err instructions become available in a program whose SLT* parameter is set to "constantly execute" (startup condition is set to ALWAYS).</p> <p>Enable/Disable = 1/0</p>	0(Not allowed)
User base program Refer to "4.3.24 User-defined external variables"	PRGUSR	Character string 1	<p>User base program is a program that is set when user-defined external variables are to be used. In case of DEF number, variable declaration instructions such as INTE and Dim are described.</p> <p>If an array variable is declared in the user base program using the Dim instruction, the same variable name must be redefined using the Dim instruction in the program that uses the user base program. Variables need not be redefined if the variable is not an array.</p>	""(Non)

Parameter	Parameter name	No. of arrays No. of characters	Details explanation	Factory setting
Continue function	CTN	Integer 1	<p>For only the program execution slot 1, the state when the power is turned OFF is held, and the operation can be continued from the saved state when the power is turned ON next. The saved data is the program execution environment (override, execution step line, program variables, etc.), and the output signal state. When this function is valid, if the robot is operated when the power is turned OFF, the robot will start in the standby state when the power is turned ON next. To continue operation, turn the servo power ON, and input start. (Function invalid/Valid=0/1)</p> <p><Precautions></p> <p>(1)As for robots with axes without brakes, the arm may lower due to gravitational weight or rotate itself when the power is turned off. Thus, extra care is necessary when using this function.</p> <p>(2)Program that can continue using the Continuity function is the one loaded in task slot 1. Programs in task slot 2 or subsequent slots will not continue but will restart in program reset state.</p> <p>(3)The following parameters cannot be changed after this function is enabled. Be sure to change them, if necessary, prior to enabling this function.</p> <p>SLTn, SLOTON, TASKMAX.</p> <p>(4)If parameters in the slot table (SLT*) are changed after enabling this function, the changes are not reflected in the slot table. Disable the continue function once, turn the power supply off and then on, and then change parameters in the slot table.</p>	0(Invalid)
JRC command (Multiple rotation function of axes)	JRCEXE	Integer 1	Set the execution status of the JRC instruction.	
	JRCQTT	Real value 8	<p>Set the validity of the JRC command execution.</p> <p>Execution valid/invalid = (1/0)</p> <p>JSet the change amount to increment or decrement with the JRC command in the order of J1, J2, J3 to J8 axes from the head element. The setting is valid only for the user-defined axis, so the J7 and J8 axes will be valid for the robot's additional axis, and a random axis for the mechanism's additional axis.</p> <p>The unit relies on the parameter AXUNT.</p>	<p>0(Execution invalid)</p> <p>JRC execution valid robot 0,0,0,0,360,0, 0 or 0,0,360,0,0,0, 0</p> <p>JRC execution invalid robot 0,0,0,0,0,0,0</p>
	JRCORG	Real value 8	<p>Set the origin coordinate value for executing the JRC O command and setting the origin.</p> <p>This setting is valid only for the user-defined axis.</p> <p>The unit relies on the parameter AXUNT.</p>	0,0,0,0,0,0,0
Setting of additional axis	AXUNT	Integer 16	Set the unit system for the additional axis. Angle(degree)/Length(mm) = 0/1	0,0,0,0,0,0,0, 0,0,0,0,0,0,0
User error setting	UER1 to UER20	Integer 1, Character string 3	<p>Sets the message, cause, and method of recovery for errors from the Error instruction. Maximum of 20 user errors can be set.</p> <p>First element ... error number to set (9000 to 9299 is the available range). The default value 9900 is not available. Change the value before proceeding.</p> <p>Second element ... Error message</p> <p>Third element ... Cause</p> <p>Fourth element ... Method of recovery</p> <p>If a space character is included in the message, enclose the entire message in double quotation marks ("").</p> <p>Example)9000,"Time Out","No Signal","Check Button"</p>	9900,"message","cause","treat"
Unit setting for the rotational element of position data	PRGMDEG	Integer 1	<p>Specifies the unit used for describing the rotational element of position data in the robot program.</p> <p>0:Rad 1:Deg</p> <p>Example)M1=P1.A (Unit for this case is specified.)</p> <p>(Default unit for referencing data components is radian.)</p> <p>The default rotational element for the position constant (P1=(100, 0, 300, 0, 180, 0, 180) (7, 0)) is Deg. This parameter is irrelevant.</p>	0(Rad)

Parameter	Parameter name	No. of arrays No. of characters	Details explanation	Factory setting												
Set the delay time of the GC/GO command and the moving command. *This parameter is valid for using the MOVEMASTER command only.	HANDDLY	Integer 1	<p>The delay time of hand open/close in MOVEMASTER command is the time specified by GP command. (Default value is 0.3 sec.)</p> <p>The delay time of hand open/close can be specified by this parameter.</p> <table border="1"> <thead> <tr> <th>Parameter value</th> <th>Motorized hand</th> <th>Pneumatic hand</th> </tr> </thead> <tbody> <tr> <td>-1</td> <td>When the status of the hand changes, the delay timer specified by GP command is taken.</td> <td>The delay time specified by the GP instruction is stored in this parameter when opening/closing the hand, regardless of whether or not the hand status has changed.</td> </tr> <tr> <td>0</td> <td>No delay</td> <td>No delay</td> </tr> <tr> <td>Value Unit(msec)</td> <td>When the status of the hand changes, the delay timer specified with this parameter is taken.</td> <td></td> </tr> </tbody> </table> <p>The units of the delay time specified by GP command are 1 / 10 seconds.</p> <p>The units of the delay time specified with this parameter are 1 / 1000 seconds (=msec).</p>	Parameter value	Motorized hand	Pneumatic hand	-1	When the status of the hand changes, the delay timer specified by GP command is taken.	The delay time specified by the GP instruction is stored in this parameter when opening/closing the hand, regardless of whether or not the hand status has changed.	0	No delay	No delay	Value Unit(msec)	When the status of the hand changes, the delay timer specified with this parameter is taken.		-1
Parameter value	Motorized hand	Pneumatic hand														
-1	When the status of the hand changes, the delay timer specified by GP command is taken.	The delay time specified by the GP instruction is stored in this parameter when opening/closing the hand, regardless of whether or not the hand status has changed.														
0	No delay	No delay														
Value Unit(msec)	When the status of the hand changes, the delay timer specified with this parameter is taken.															
Robot language setting	RLNG	Integer 1	Select the robot language 2:MELFA-BASIC V 1:MELFA-BASIC IV	1												
Display language <small>Note1)</small>	LNG	Character string 1	<p>Set up the display language. "JPN":Japanese "ENG":English</p> <p>The following language is changed. (1)The display LCD of teaching pendant. (2) Personal computer support software. *alarm message of the robot. *Parameter explanation list. (3)Alarm message that read from the robot with external communication. (Standard RS232C, Extended serial I/F, Ethernet I/F)</p>	The "JPN" is Japanese specification. The "ENG" is English specification.												
Extension of external variable	PRGGBL	-	Sets "1" to this parameter, and turns on the controller power again, then the capacity of each program external variable will double. However, if a variable with the same name is being used as a user-defined external variable, an error will occur when the power is turned ON, and it is not possible to expand. It is necessary to correct the user definition external variable.	0												

Note1) The parameter is set up based on the order specifications before shipment.

Order to dealer when the instruction manual of the other language is necessary.

More, the caution seals that stuck on the robot arm and the controller are made based on the language of the order specification. Use it carefully when selecting the other language.

5.5 Communication parameter

These parameters set the items pertaining to communications

Table 5-7:List Communication parameter

Parameter	Parameter name	No. of arrays No. of characters	Details explanation	Factory setting
Communication setting			Communication environment is set for RS-232C and ethernet interface. However, since RS-232C is used by the PC support software, parameter for RS-232C normally does not need to be changed. Refer to " 5.15About the communication setting(RS-232) " for RS-232 Refer to " 5.16About the communication setting(Ethernet) " for ethernet	
	COMDEV	Character string 8	<p>This configures which lines will be assigned to COM1 and COM2 when using communication lines in the Open instruction in MELFA BASIC V. This parameter must be set if data link (used by the Open instruction) is to be performed.</p> <p>This parameter specifies the device that corresponds to COMn specified in the Open statement in the program (n is between 1 and 8). Parameters are starting from the left COM1, COM2, ..., COM8 in that order.</p> <p>When the data link is applied by ethernet I/F, setting is necessary. OPT11 to OPT19 are allocated. Here, RS-232C of the controller is previously allocated to COM1: .</p> <p>Note)Since the communication interface is not prepared for robot CPU of the notes CRnQ-700 series, and the drive unit, this parameter cannot be used.</p>	"RS232", , , , ,
For RS-232	CBAU232	Integer 1	Baud rate(9600,19200)	9600
	CPRTY232	Integer 1	Parity bit(0: None, 1: Odd, 2: Even)	2 (Even)
	CSTOP232	Integer 1	Stop bit(1,2)	2 (Stop bit)
	CTERM232	Integer 1	End code(0:CR 1:CR+LF)	0 (CR)
	CPRC232	Integer 1	<p>Communication method(protocol)</p> <p>0: For RT ToolBox (Non-procedure)</p> <p>If data link is to be performed (Open, Print and Input instructions are executed from the program) under this setting, the external device must attach three characters "PRN" at the beginning when transmitting data.</p> <p>1: For RT ToolBox (With procedure) PC side must also be changed.</p> <p>2: For data link with the robot program</p> <p>Be advised that under this setting, connection with the RT ToolBox cannot be made.</p>	1
For ethernet	NETIP	Character string 1	IP address of robot controller	192.168.0.1
	NETMSK	Character string 1	Sub-net-mask	255.255.255.0
	NETPORT	Numerical value 10	<p>Port No. Range 0 to 32767</p> <p>For real-time external control functions, Correspond to OPT11 to 19 of COMDEV (OPT11), (OPT12), (OPT13), (OPT14), (OPT15), (OPT16), (OPT17), (OPT18), (OPT19)</p>	10000, 10001, 10002, 10003, 10004, 10005, 10006, 10007, 10008, 10009

Parameter	Parameter name	No. of arrays No. of characters	Details explanation	Factory setting
For ethernet			Protocol 0: No-procedure, 1: Procedure, 2: Data link (1: Procedure has currently no function.) When the data link is applied by ethernet I/F, setting is necessary. Correspond to OPT11 to 19 of COMDEV (OPT11), (OPT12), (OPT13), (OPT14), (OPT15), (OPT16), (OPT17), (OPT18), (OPT19)	
	CPRCE11 CPRCE12 CPRCE13 CPRCE14 CPRCE15 CPRCE16 CPRCE17 CPRCE18 CPRCE19	Numerical value 9		0 0 0 0 0 0 0 0 0
	NETMODE	Numerical value 9	Server designation (1: Server, 0: Client) When the data link is applied by ethernet I/F, setting is necessary. Correspond to OPT11 to 19 of COMDEV (OPT11), (OPT12), (OPT13), (OPT14), (OPT15), (OPT16), (OPT17), (OPT18), (OPT19)	1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1
	NETHSTIP	Numerical value 9	The IP address of the data communication destination server. When the data link is applied by ethernet I/F, setting is necessary. * It is valid if specified as the client by NETMODE only. Correspond to OPT11 to 19 of COMDEV (OPT11), (OPT12), (OPT13), (OPT14), (OPT15), (OPT16), (OPT17), (OPT18), (OPT19)	192.168.0.2 , 192.168.0.3 , 192.168.0.4 , 192.168.0.5 , 192.168.0.6 , 192.168.0.7 , 192.168.0.8 , 192.168.0.9 , 192.168.0.10
	MXTTOUT	Numerical value 1 (0 to 32767)	Timeout time for executing real-time external control command (Multiple of 7.1msec, Set -1 to disable timeout)	-1

5.6 Standard Tool Coordinates

Tools data must be set if the robot's control point is to be set at the hand tip when the hand is installed on the robot. The setting can be done in the following three manners.

- 1) Set in the MEXTL parameter.
- 2) Set in the robot program using the Tool instruction.
- 3) Set a tool number in the M_Tool variable. The values set by the MEXTL1 to 4 parameters are used as tool data.

Refer to [Page 299, "M_Tool"](#).

The default value at the factory default setting is set to zero, where the control point is set to the mechanical interface (flange plane).

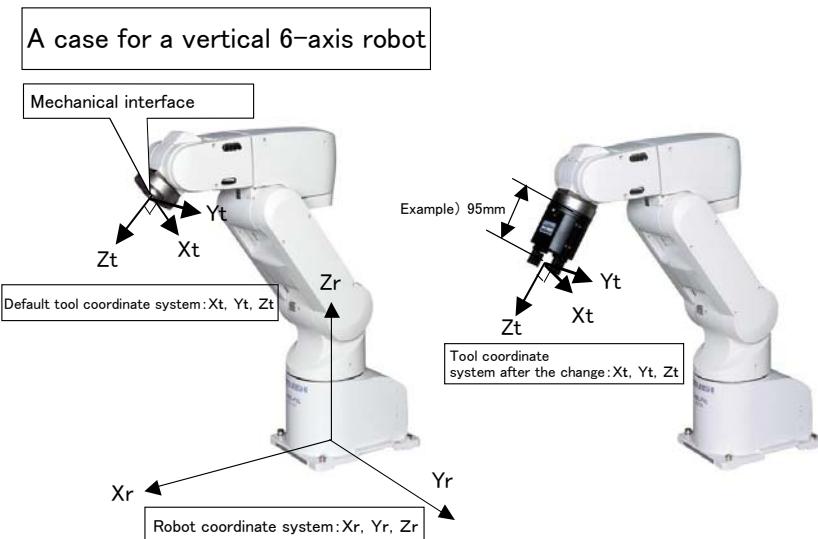
Structure of tools data : X, Y, Z, A, B, C

X, Y and Z axis: Shift from the mechanical interface in the tool coordinate system

A axis :X-axis rotation in the tool coordinate system

B axis :Y-axis rotation in the tool coordinate system

C axis :Z-axis rotation in the tool coordinate system



<A case for a vertical 6-axis robot>

1) Sample parameter setting

Parameter name: MEXTL

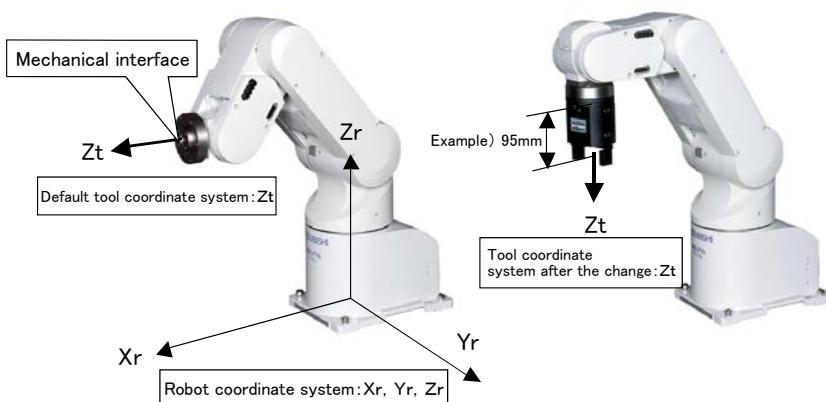
Value: 0, 0, 95, 0, 0, 0

2) Sample Tool instruction setting

1 Tool (0,0,95,0,0,0)

A 6-axis robot can take various postures within the movement range.

A case for a vertical 5-axis robot



<A case for a vertical 5-axis robot>

1) Sample parameter setting

Parameter name: MEXTL

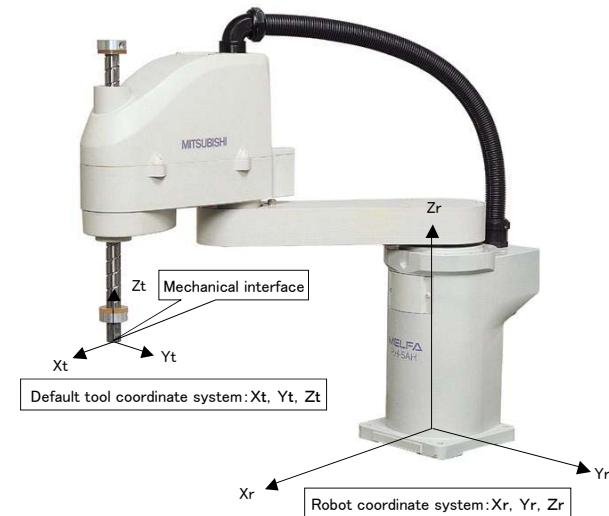
Value: 0, 0, 95, 0, 0, 0

2) Sample Tool instruction setting

1 Tool (0,0,95,0,0,0)

Only the Z-axis component is valid for a 5-axis robot for movement range reasons. Data input to other axes will be ignored.

A case for a horizontal 4-axis robot



<A case for a horizontal 4-axis robot>

1) Sample parameter setting

Parameter name: MEXTL

Value: 0, 0, -10, 0, 0, 0

2) Sample Tool instruction setting

1 Tool (0,0,-10,0,0,0)

Horizontal 4-axis robots can basically offset using parallel shifting. Note that the orientation of the tool coordinate system is set up differently from that of vertical robots.

An axis element of the tool conversion data may or may not be valid depending on the robot model. See [Table 5-8](#) to set the appropriate data.

Table 5-8:Valid axis elements of the tool conversion data depending on the robot model

Type	Number of axis	An axis element of the tool conversion data <small>Note1)</small>					
		X	Y	Z	A	B	C
RV-3SD, RV-6SD, RV-12SD/SDL, RV-18SD	6	O	O	O	O	O	O
RV-3SDJ	5	X	X	O	X	X	X
RH-6SDH/12SDH/18SDH	4	O	O	O	X	X	O

Note1) O: Valid, X: Invalid. This is meaningless and ignored if set., X: The setting value is fixed to 0.

If a value other than 0 is set, operation may be adversely affected.

5.7 About Standard Base Coordinates

When shifting the robot origin to a position other than the center position of the J1 axis of the robot, the conversion is performed using the base coordinate system. The setting will be done from the following two points. When base data is changed, the coordinates of teaching positions will be values based on the base coordinate system.

- 1) Set in the MEXTL parameter.
- 2) Set in the robot program using the Base instruction.

The factory default setting value is set to zero at the base coordinate system position, which is identical to the robot origin.

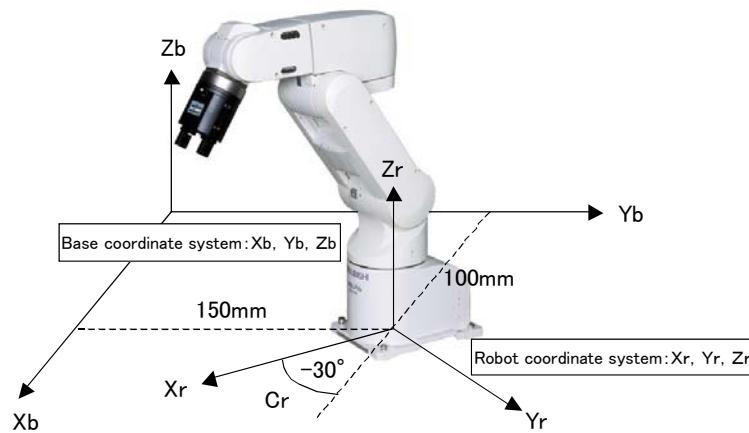
Structure of base coordinate system data: X, Y, Z, A, B, and C

X, Y and Z axis : The position of robot coordinate system from the base coordinate system origin

A axis : X-axis rotation in the base coordinate system

B axis : Y-axis rotation in the base coordinate system

C axis : Z-axis rotation in the base coordinate system



(Example)

1) Sample parameter setting

Parameter name: MEXBS

Value AF100,150,0,0,0,-30

2) Sample Base instruction setting

1 Base (100,150,0,0,0,-30)

Normally, the base coordinate system need not be changed. If you wish to change it, see the sample above when configuring the system. Note that the Base instruction within the robot program may shift the robot to an unexpected position. Exercise caution when executing the instruction.

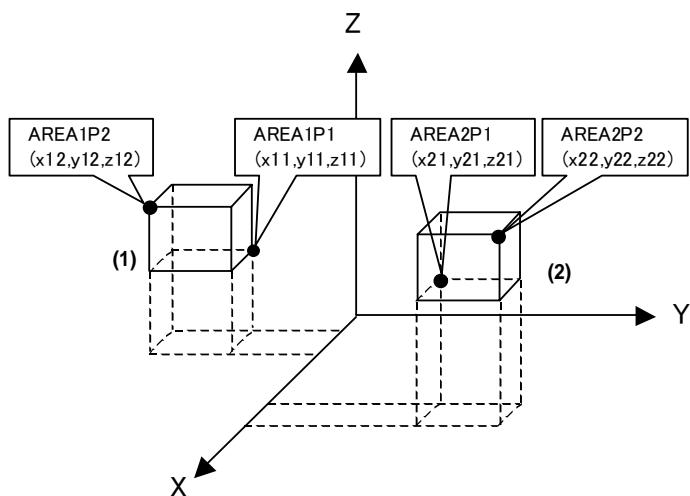
An axis element of the base conversion data may or may not be valid depending on the robot model. See [Table 5-9](#) to set the appropriate data.

Table 5-9:Valid axis elements of the base conversion data depending on the robot model

Type	Number of axis	An axis element of the base conversion data Note1)					
		X	Y	Z	A	B	C
RV-3SQ/3SD RV-6SQ/6SQL/6SD/6SDL RV-12SQ/12SQL/12SD/SDL	6	O	O	O	O	O	O
RV-3SQJ/3SDJ	5	O	O	O	X	X	X
RH-6SDH/12SDH/18SDH	4	O	O	O	X	X	O

Note1) O: Valid, X: Invalid. This is meaningless and ignored if set., X: The setting value is fixed to 0.

5.8 About user-defined area



When operation is performed together with peripheral devices, work area may have to be shared. Under such circumstances, one device must let the other know when it is within the shared area. For this purpose, a robot can be configured to output a signal while it is in a certain area by setting parameters.

For instance, in the diagram to the left, the following parameter setting will output the signal 10 when operating in area (1) and output the signal 11 when operating in area (2).

Similar confirmation is possible using the M_Uar variable if checking within the program. Refer to [Page 300, "M_Uar"](#), [Page 301, "M_Uar32"](#).

Parameter name	Meaning of the value	Value
AREA1P1	Position data for the first point: X,Y,Z,A,B,C,L1,L2	x11, y11, z11, -360, -360, -360,0,0
AREA1P2	Position data for the second point: X,Y,Z,A,B,C,L1,L2	x12, y12, z12, 360, 360, 360,0,0
AREA1ME	Target mechanism number: Usually 1	1
AREA1AT	Invalid/Output signal/Error: 0/1/2	1
AREA2P1	Position data for the first point: X,Y,Z,A,B,C,L1,L2	x21, y21, z21, -360, -360, -360,0,0
AREA2P2	Position data for the second point: X,Y,Z,A,B,C,L1,L2	x22, y22, z22, 360, 360, 360,0,0
AREA2ME	Target mechanism number: Usually 1	1
AREA2AT	Invalid/Output signal/Error: 0/1/2	1
USRAREA	Output signal: starting number, end number	10, 11 (Information regarding whether or not the robot is in AREA1 is output to the signal 10, and whether or not the robot is in AREA2 is output to the signal 11.) Set 10,10 in the case of one area.

*1 Enter the coordinates (x, y, and z) for x11 to z22.

*2 In the setting sample above, since the posture data (A, B, and C) are ignored, a signal will be output regardless of the posture. To set the posture data (A, B, and C), set the values in the AREA*P1 to AREA*P2 direction. (Example: In the case of -10 -> +30, evaluation as in-area occurs in the range from -10 to + 30, but in the case of +30 -> -10 evaluation as in-area occurs in the range from +30 to -10.)

*3 For a non-existent axis of the posture data (for instance the A- and B-axes of horizontal multi-joint type robots), make sure that AREA*P1 is set to -360 and AREA*P2 is set to +360.

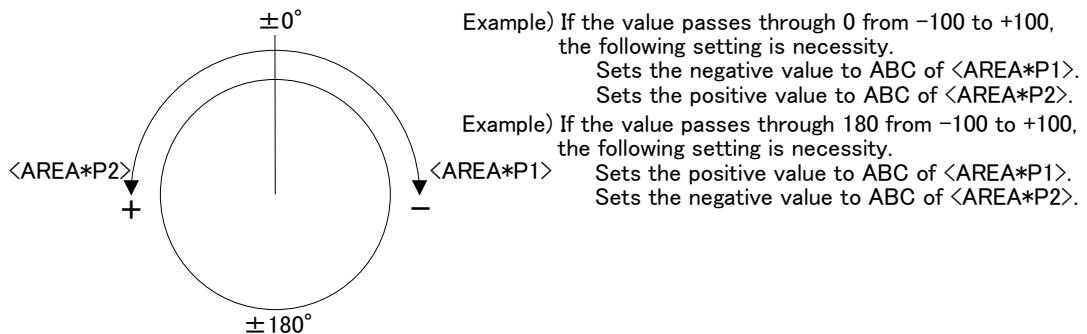
*4 AREA*ME specifies to which mechanism will the area checking configuration apply. Under the standard configuration (one unit is connected), this is set to 1.

*5 AREA*AT specifies the type of area checking. The meaning of the value is shown below.

0 = Does not check, 1 = Outputs a signal, 2 = Generates an error.

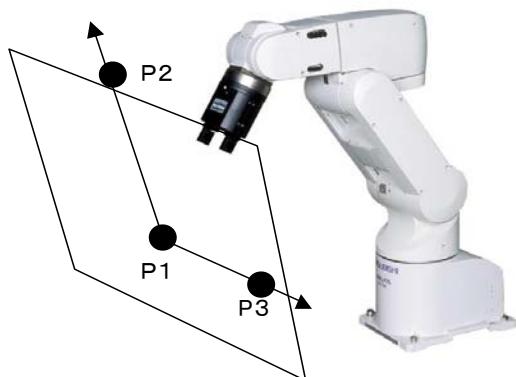
If 2, posture data, L1 and L2 are ignored.

*6 If additional axes are used, set an area for axes L1 and L2 respectively.



5.9 Free plane limit

Defines any plane in the robot coordinate system, determines the front or back of the plane, and generates a free plane limit error.



As can be seen in the diagram to the left, any plane can be defined by three points (P1, P2, and P3), after which an evaluation of which side of the plane it is in (the side that includes the robot origin or the other side) can be performed. This function can be used to prevent collision with the floor or interference with peripheral devices. Maximum of eight planes can be monitored. There is no limit to the plane.

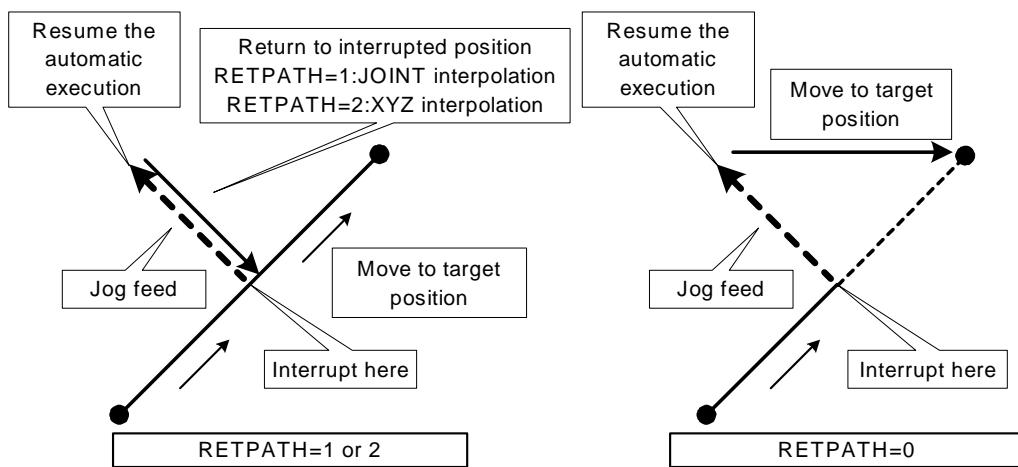
Parameter and value	Explanation
SFCnP(n=1 to 8)	Specifies the 3 points that define the plane. P1 coordinates X1, Y1, and Z1: The origin of the plane P2 coordinates X2,Y2,Z2: A position on the X axis of the plane P3 coordinates X3,Y3,Z3: A position in the positive Y direction of the X-Y plane in the plane
SFCnME(n=1 to 3)	Specifies the mechanism number to which the free plane limit applies. Usually, set up 1. In the case of multiple mechanisms, the mechanism numbers are specified.
SFCnAT(n=1 to 8)	Designate the valid/Invalid of the set free plane limit. 0:Invalid 1: Valid (The operable area is the robot coordinate origin side.) -1: Valid (The operable area is the side where the robot coordinate origin does not exist.)

After setting the parameters above, turn the controller's power ON again. This will allow the generation of free plane limit error when it crosses the plane.

5.10 Automatic return setting after jog feed at pause

This specifies the path behavior that takes place when the robot is paused during automatic operation or during step feed operation, moved to a different position using a jog feed with T/B, and the automatic operation is resumed or the step feed operation is executed again. See the following diagram.

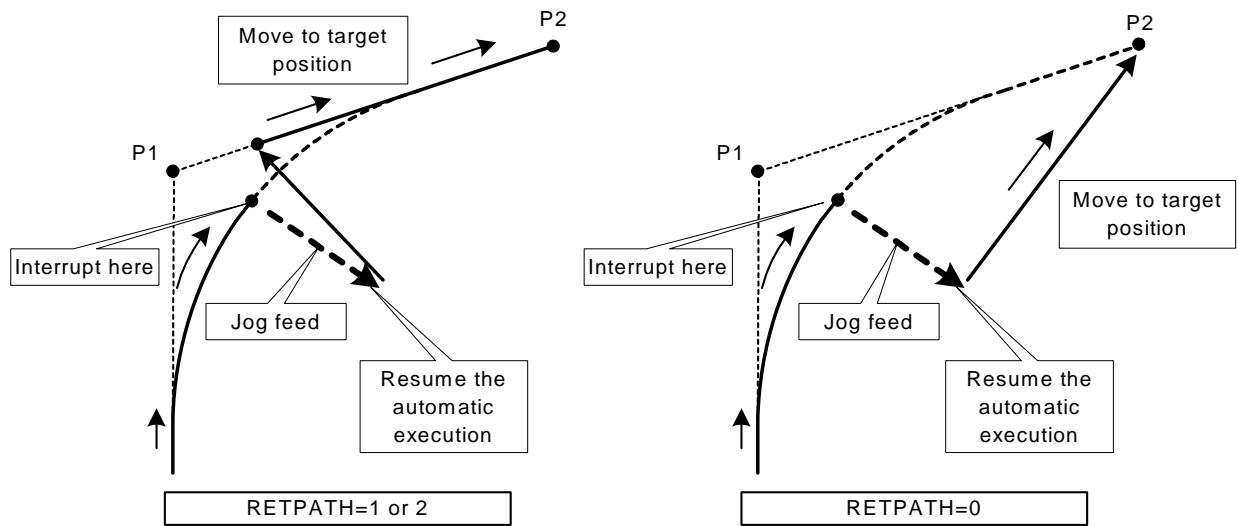
Parameter and value	Description of the operation
RETPATH=1 (Default)	1) Returns to the original position where the pause took place using joint interpolation. 2) Resumes from the line that was paused.
RETPATH=0	Resumes from the line that was paused from the position resulting after the jog operation. Therefore, movement will take place using the interpolation method of the instruction under execution from the current position to the next target position.
RETPATH=2	1) Return by XYZ interpolation to the interrupted position. 2) Resume the interrupted line.



[Caution] If movement other than a joint jog (XYZ, tools, cylindrical, etc.) has been used when the "RETPATH" parameter is set to 1, joint interpolation will be used to return to the original position at the time pause took place. Therefore, be careful not to interfere with peripheral devices.

[Caution] If the parameter "RETPATH" is set to 2 for a robot whose structure data is valid or with multiple rotations, and the robot is moved from a suspended position by joint jog, the robot is moved to a position different from the original structure data and/or multiple-rotation data and may become unable to return to the suspended position. In this case, adjust the position of the robot to the suspended position and resume moving the robot.

If "RETPATH=1 or 2" is set as shown in the figure below, and the robot is operated continuously (continuous path operation) using the Cnt instruction, the robot returns to a position on the travel path from P1 to P2 instead of the suspended position. When "RETPATH=0" is set, the robot moves to the target position from the current position.



5.11 Automatic execution of program at power up

The following illustrates how to automatically run a robot program when the controller's power is turned on. However, since the robot starts operating simply by turning the power on, exercise caution upon using this function.

Related parameters

Parameter and value	Description of the operation
SLT*	Example) SLT2=2,ALWAYS,REP Specifies the program name, start condition, and operation status. The point here is the start condition.
ALWENA	0->7 In the ALWAYS program, it is possible to execute multitask-related instructions such as XRun and XLoad, and also the Servo instruction.

(1) First, create an ALWAYS program and an operating program.

<Program #2, ALWAYS program>

```

1 ' Auto Start Sample Program
2 '
3 ' Execute Program #1 if the key switch is AUTO (Ext.).
4 ' Stop the program and return the execution line to the beginning of the program if the key switch is not AUTO
(Ext.).
5 '
6 If M_Mode<>3 AND (M_Run(1)=1 OR M_Wai(1)=1) Then GoSub *MTSTOP
7 If M_Mode=3 AND M_Run(1)=0 AND M_Wai(1)=0 Then GoSub *MTSTART
8 If M_Mode=2 Then Hlt ' for DEBUG
9 End
10 '
11 *MTSTART
12 XRun 1,"1"
13 RETURN
14 '
15 *MTSTOP
16 XStp 1
17 XRst 1
18 RETURN

```

< Program #1, operating program > (this can be any program)

```

1 'Main Program
2 Servo On
3 M_Out(8)=0
4 Mov P1
5 M_Out(8)=1
6 Mov P2
7 End
P1=(-300.00,-200.00,+200.00,+0.00,+180.00,+0.00)(6,0)
P2=(-300.00,+200.00,+200.00,+0.00,+180.00,+0.00)(6,0)

```

(2) Set the parameter.

Parameter and value	Description of the operation
SLT2	SLT2=2,ALWAYS,REP 'Execute program #2 in ALWAYS mode.
ALWENA	0->7 In the ALWAYS program, it is possible to execute multitask-related instructions such as XRun and XLoad, and also the Servo instruction.

After the setting is complete, turn the controller's power OFF.

(3) Turn the power ON.

In the sample above, after the controller's power is turned on, when the key switch is turned to AUTO (Ext.), program #1 is executed and the robot starts its operation.

5.12 About the hand type

The factory default setting assumes that the double-solenoid type hand will be used. If the single-solenoid type is used or if a general-purpose signal is to be used to control the robot, the HANDTYPE parameter must be set as described below.

Table 5-10:Factory default parameter settings

Parameter name	Value
HANDTYPE	D900,D902,D904,D906, , ,

Note) The default settings are D224, D226, D192 and D194 in the case of the RC-1300G series.

From the left, the values correspond to hand #1, #2, and so on. The default value is shown below.

Hand 1 = accesses signals #900 and #901

Hand 2 = accesses signals #902 and #903

Hand 3 = accesses signals #904 and #905

Hand 4 = accesses signals #906 and #907

The hand numbers 1 through 4 (or 8) will be used as the argument in the hand open/close instructions (HOpen or HClose).

<Setting method>

When a double-solenoid type is used, 'D' must be added in front of the signal number to specify the number.

In the case of double-solenoid type, hand number will be from 1 to 4.

When a single-solenoid type is used, 'S' must be added in front of the signal number to specify the number.

In the case of single-solenoid type, hand number will be from 1 to 8.

<Example>

1) To assign two hands of the double-solenoid type from the general-purpose signal #10

HANDTYPE=D10,D12, , , ,

2) To assign three hands of the double-solenoid type from the general-purpose signal #10

HANDTYPE=S10,S11,S12, , , ,

3) To assign hand 1 to the general-purpose signal #10 as the single-solenoid type while assigning hand

2 to the general-purpose signal #12 as the single-solenoid type

HANDTYPE=D10,S12, , , ,

5.13 About default hand status

The factory default setting is shown below.

Hand type	Status	Status of output signal number		
		Mechanism #1	Mechanism #2	Mechanism #3
When pneumatic hand interface is installed (double-solenoid is assumed)	Hand 1 = Open	900=1 901=0	910=1 911=0	920=1 921=0
	Hand 2 =Open	902=1 903=0	912=1 913=0	922=1 923=0
	Hand 3 =Open	904=1 905=0	914=1 915=0	924=1 925=0
	Hand 4 =Open	906=1 907=0	916=1 917=0	926=1 927=0
When electric-powered hand interface is installed	Hand open	M_Out (9*0) through M_Out (9*7) are used by the system and therefore unavailable to the user. If used, normal opening and closing of the hand will not be possible.		
I/F before interface installation	-	M_Out (9*0) through M_Out (9*7) do not function.		

A single controller can control multiple robots. If pneumatic hand interface is to be used for each robot, the hand output signal number is assigned in the following manner.

Mechanism #1 = #900 to #907 (This will be the case for standard configuration with one unit connected.)

Mechanism #2 = #910 to #917

Mechanism #3 = #920 to #927

When electric-powered hand interface is used, the system will use the 900's using special controls. The users should not access the 900's directly but instead use the hand control instructions or the hand operation from T/B only. If you access the 900's, normal opening and closing of the hand will not be possible.

The default parameters are set as shown below so that all hands start as "Open" immediately after power up.

Parameter name	Signal number	Value
HANDINIT	900, 901, 902, 903, 904, 905, 906, 907	1, 0, 1, 0, 1, 0, 1, 0

The above describes the situation for standard configuration (one unit is connected). When multiple mechanisms are used, specify the mechanism number to set the HANDINIT parameter.

If for instance hand 1 alone needs to be closed when the power is turned ON, the following should be set. Similarly, in the case of electric-powered hand (hand number is fixed to 1), the hand will be closed when the power is turned on if the following configuration is applied.

Parameter name	Signal number	Value
HANDINIT	900, 901, 902, 903, 904, 905, 906, 907	0, 1, 1, 0, 1, 0, 1, 0

[Caution1] If you set the initial hand status to "Open," note that the workpiece may be dropped when the power is turned ON.

[Caution2] This parameter specifies the initial value when turning ON the power to the dedicated hand signals (900's) at the robot's tip.

To set the initial status at power ON when controlling the hand using general-purpose I/Os (other than 900's) or CC-Link (6000's) (specifying a signal other than one in 900's by the HANDTYPE parameter), do not use this HANDINIT parameter, but use the ORST* parameter.

The value set by the ORST* parameter becomes the initial value of signals at power ON.

[Caution3] The RC-1300G series uses #224 to #227 and #192 to #195 for hand input signals. Use the ORST224 and ORST192 parameters, rather than the HANDINIT parameter, to set the initial value when the power supply is turned on.

5.14 About the output signal reset pattern

The factory default setting sets all general-purpose output signals to OFF (0) at power up. The status of general-purpose output signals after power up can be changed by changing the following parameter. Note that this parameter also affects the general-purpose output signal reset operation (called by dedicated I/O signals) and the reset pattern after executing the Clr instruction.

	Parameter name	Value (Values are all set to 0 at the factory default setting.)
Remote I/O	ORST0	Signal number 0-----7 8-----15 16-----23 24-----31 00000000, 00000000, 00000000, 00000000
	ORST32	32-----40 41-----49 50-----57 58-----66 (Same as above) 00000000, 00000000, 00000000, 00000000
	ORST64	00000000, 00000000, 00000000, 00000000
	ORST96	00000000, 00000000, 00000000, 00000000
	ORST128	00000000, 00000000, 00000000, 00000000
	ORST160	00000000, 00000000, 00000000, 00000000
	ORST192	00000000, 00000000, 00000000, 00000000
	ORST224	00000000, 00000000, 00000000, 00000000
PROFIBUS option	ORST2000	00000000, 00000000, 00000000, 00000000
	ORST2032	00000000, 00000000, 00000000, 00000000
	ORST2064	00000000, 00000000, 00000000, 00000000
	ORST2096	00000000, 00000000, 00000000, 00000000
	ORST2128	00000000, 00000000, 00000000, 00000000
	ORST2160	00000000, 00000000, 00000000, 00000000
	ORST2192	00000000, 00000000, 00000000, 00000000
	ORST2224	00000000, 00000000, 00000000, 00000000
	ORST2256	00000000, 00000000, 00000000, 00000000
	ORST2288	00000000, 00000000, 00000000, 00000000
	:	:
	:	:
	:	:
CC-Link option	ORST5008	00000000, 00000000, 00000000, 00000000
	ORST5040	00000000, 00000000, 00000000, 00000000
	ORST6000	00000000, 00000000, 00000000, 00000000
	ORST6032	00000000, 00000000, 00000000, 00000000
	ORST6064	00000000, 00000000, 00000000, 00000000
	ORST6096	00000000, 00000000, 00000000, 00000000
	ORST6128	00000000, 00000000, 00000000, 00000000
	ORST6160	00000000, 00000000, 00000000, 00000000
	ORST6192	00000000, 00000000, 00000000, 00000000
	ORST6224	00000000, 00000000, 00000000, 00000000
	ORST6256	00000000, 00000000, 00000000, 00000000
	ORST6288	00000000, 00000000, 00000000, 00000000
	:	:
	:	:
	:	:
	ORST7984	00000000, 00000000, 00000000, 00000000
	ORST8016	00000000, 00000000, 00000000, 00000000

	Parameter name	Value (Values are all set to 0 at the factory default setting.)
PLC link Note1)	ORST10000	信号番号 10000—10007 10008—10015 10016—10023 10024—10031 00000000、 00000000、 00000000、 00000000
	ORST10032	10032—10039 10040—10047 10048—10055 10056—10063 00000000、 00000000、 00000000、 00000000
	ORST10064	00000000、 00000000、 00000000、 00000000
	ORST10096	00000000、 00000000、 00000000、 00000000
	ORST10128	00000000、 00000000、 00000000、 00000000 00000000、 00000000、 00000000、 00000000 00000000、 00000000、 00000000、 00000000 00000000、 00000000、 00000000、 00000000 00000000、 00000000、 00000000、 00000000
	ORST18160	00000000、 00000000、 00000000、 00000000

Note1) PLC link is for the CRnQ Series only.

The value corresponds to bits from the left.

Setting is "0", "1", or "*".

"0" = Set to off

"1" = Set to on

"*" = Maintain status with no change. (Set to off at power up.)

For instance, if you want to always turn ON immediately after power up 10138, 10139, 10140, 10160, 10161 and 10168 of the general-purpose signals, the robot should be set to the configuration shown below.

Parameter name	Value
ORST10128	10128—10135 10136—10143 10144—10151 10152—10159 00000000、 00000000、 00000000、 00000000.....At the factory default setting 00000000、 <u>00111000</u> 、 00000000、 00000000.....Setting value
ORST10160	10160—10167 10168—10175 10176—10183 10184—10191 00000000、 00000000、 00000000、 00000000.....At the factory default setting <u>11000000</u> 、 <u>10000000</u> 、 00000000、 00000000.....Setting value

In addition to the above, to make 10148, 10149 and 10150 retain their individual on/off status upon a general-purpose output signal reset, the robot should be set to the configuration shown below.

Parameter name	Value
ORST10128	00000000、 00111000、 0000 <u>**</u> 0、 00000000
ORST10160	<u>11000000</u> 、 <u>10000000</u> 、 00000000、 00000000

In the case above, general-purpose signals 10148, 10149, 10150 will start up as 0 (off) after a power up.

The setting cannot be made in such a way that will turn the signal to 1 (on) after power up and will retain the current status upon a general-purpose output signal reset.

[Caution] When editing the parameters, do not enter an incorrect number of zeros. If the number of zeros is incorrect, an error is generated next time the power is turned on.

5.15 About the communication setting(RS-232)

(1) Overview

The controller of CRnD series has RS-232 of the one port as standard.

Although it usually connects with the personal computer and the standard RS-232 port is used for transmission of the robot program, and debugging using RT-ToolBox of the option, it sets the parameter and can use it for communication of the data with external equipment. Carry out communication by the commands (Open/Close/Print/Input etc.) of the communication relation on the robot program. Call this the data link.

The controller cannot be controlled from external devices such as a PC (i.e., automatic execution or status monitoring). If this is necessary, contact the dealer or branch from where the robot has been purchased for further consultation.

Example of standard RS-232C port usage

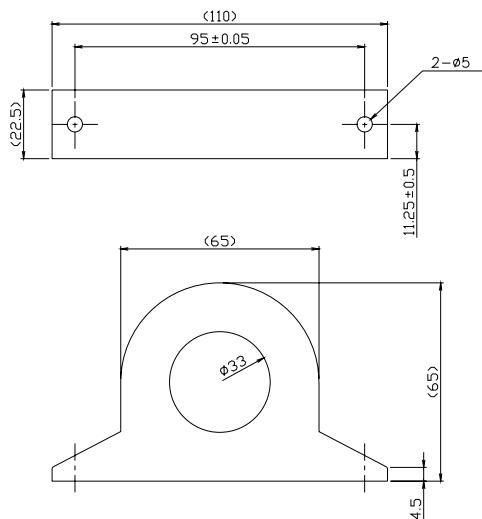
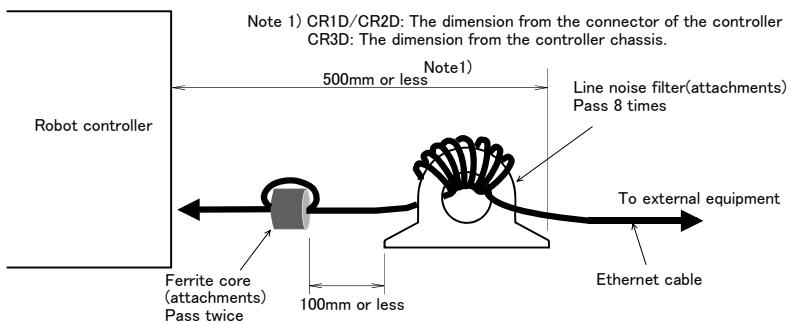


Use of PC with the support software

5.16 About the communication setting(Ethernet)

In the CE Marking specification, Please use Ethernet interface by installing the line noise filter and the ferrite core to the Ethernet cable. The line noise filter and the ferrite core are attachments.

The installation conditions of the ferrite core and the line noise filter



Outside dimension of the line noise filter

Fig.5-3 : Installation to the Ethernet cable of ferrite core and filter (CE Marking)

5.16.1 Details of parameters

(1) NETIP (IP address of robot controller)

The IP address of the robot controller is set. IP address is like the address of the mail.

The format of IP address is composed of 4 numbers of 0 to 255 and the dot (.) between the numbers.

For example, it is set as 192.168.0.1.

If the controller and network personal computer are directly connected to each other one-to-one, it is allowed to set default value (a random value) but if it is connected to the local area network (LAN), IP address must be set as instructed by the manager of customer's LAN system.

If any IP addresses are overlapped, the function will not properly operate. Therefore, take care to prevent it from being overlapped with another during setting.

The personal computer used for communication with the robot controller must be located on the same network.

(2) NETMSK (sub-net-mask)

Set the sub-net-mask of the robot controller. Among the IP addresses, the sub-net-mask is set to define the sub-net-work.

The format of the sub-net-mask is composed of 4 numbers of 0 to 255 and the dot (.) between the numbers.

For example, it is set as 255.255.255.0 or 255.255.0.0.

As usual, it is allowed to set default value. If it is connected to the local area network (LAN), the sub-net-mask must be set as instructed by the manager of customer's LAN system.

(3) NETPORT (port No.)

The port No. of the robot controller is set. The port No. is like the name of the mail.

For the nine elements, the port numbers are each expressed with a value.

The first element (element No. 1) is used for real-time control.

The second to ninth elements (elements No. 2 to 9) are used for the support software or data link.

Normally, the default value does not need to be changed. Make sure that the port numbers are not duplicated.

(4) CRRCE11 to 19 (protocol)

When using the data link function, the setup is necessary.

Sets the protocol (procedure) for communication. The protocol has three kinds of no-procedure, procedure and data link.

0... No-procedure: The protocol is applied to use the personal computer Support Software .

1... Procedure: Reserved. (Since it is not any function, don't set it by mistake.)

2... Data link: The protocol is used to use OPEN/INPUT/PRINT commands for communication.

(5) COMDEV (Definition of devices corresponding to COM1: to 8)

When using the data link function, the setup is necessary.

Definition of device corresponding to COM1: to 8 is set. COM1: to 8 is used for OPEN command of the robot program.

Be sure to set it only when the data link is specified on setting of the protocol (CPRCE11 to 19).

The setting values of the Ethernet interface option correspond to the port Nos. which are set at the parameter NETPORT.

* In the following parameters NETPORT (n) and COMDEV(n), n indicates the element No. of that parameter.

n	The device name set up by COMDEV(n)	Port number
1	OPT11	The port number specified by NETPORT(2)
2	OPT12	The port number specified by NETPORT(3)
3	OPT13	The port number specified by NETPORT(4)
4	OPT14	The port number specified by NETPORT(5)
5	OPT15	The port number specified by NETPORT(6)
6	OPT16	The port number specified by NETPORT(7)
7	OPT17	The port number specified by NETPORT(8)
8	OPT18	The port number specified by NETPORT(9)
9	OPT19	The port number specified by NETPORT(10)

For example, if the port No. specified at NETPORT(4) is allocated to the data link of COM:3, the following will be applied.

COMDEV(3) = OPT13 * OPT13 is set at 3rd element of COMDEV.

CPRCE13 = 2 * Set up as a data link.

(6) NETMODE (server specification).

Set up, when using the data link function.

Set the TCP/IP communication in the data link function of the robot controller as the server or the client.

It is necessary to change with the application of the equipment connected to the robot controller.

This function corresponds to the software version H7 or later.

In the version older than H7, the robot controller operates only as a server.

(7) NETHSTIP (The IP address of the server of the data communication point).

Set up, when using the robot controller as a client by the data link function.

Specify the IP address of the partner server which the robot controller connects by the data link function.

Set up, when only set the robot controller to the client by server specification of NETMODE.

(8) MXTTOUT (Timeout setting for executing real-time external control command)

This is changed when using real-time external control command and setting the timeout time for communication with the robot controller.

Set a multiple of the approx. 7.11msec control cycle.

When the real-time external control command is executed, the timeout time during which no communication data is received by the robot controller from the personal computer is counted up. If the count reaches the value set in MXTTOUT, the operation will stop with the error (#7820). For example, to generate an error when there is no communication for approx. 7 seconds, set 1000.

This setting is set to -1 (timeout disabled) as the default.

5.16.2 Example of setting of parameter 1 (When the Support Software is used)

The setting example to use the Support Software is shown below.

Set the parameters for the robot controller, and the network for the personal computer OS being used.

IP address of robot controller	192.168.0.20
IP address of personal computer	192.168.0.2
Port No. of robot controller 10001	10001

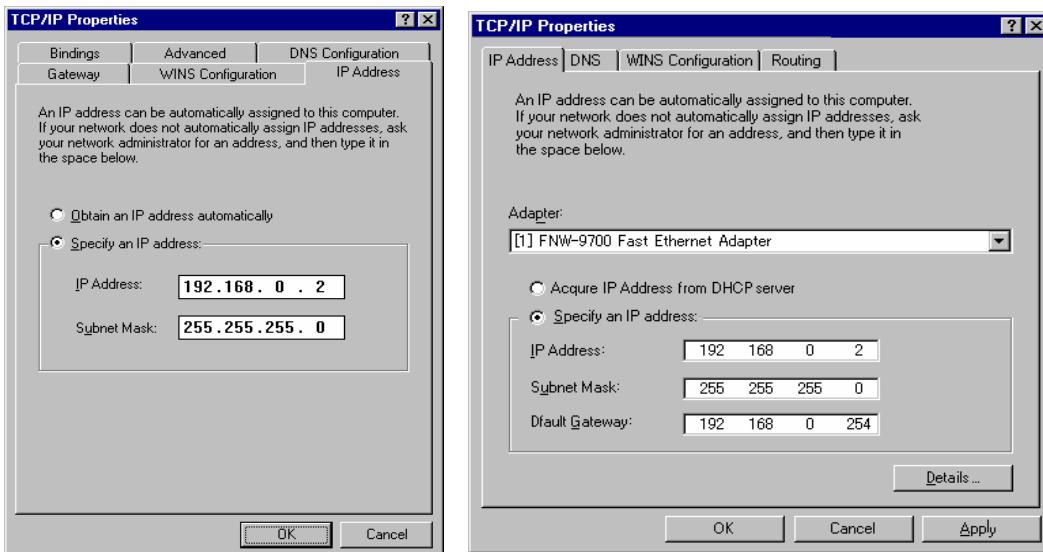
Set the robot controller parameters as shown below.

If the default settings are to be used, the parameters do not need to be changed.

Parameter name to be changed	Before/after change	Parameter value
NETIP	Before	192.168.0.20
	After	192.168.0.20 (With the default value.)
NETPORT	Before	10001
	After	10001 (With the default value.)

Next, set the personal computer IP address to 192.168.0.2. Set this value on the Network Properties screen.

Windows 95 (lower left screen), Windows NT4.0 (lower right screen)



The personal computer IP address is set with the Windows TCP/IP Property Network setting (property in network computer). Because the set-up screen differs with versions of Windows, refer to the manuals enclosed with Windows, etc., for details on setting this address.

Refer to the instruction manuals enclosed with the personal computer support software for details on setting and using the personal computer support software.

5.16.3 Example of setting of parameter 2-1

(When the data link function is used: When the controller is the server)

Shows the example of the setting, when the controller is server by the data link function.

List Example of conditions 2-1

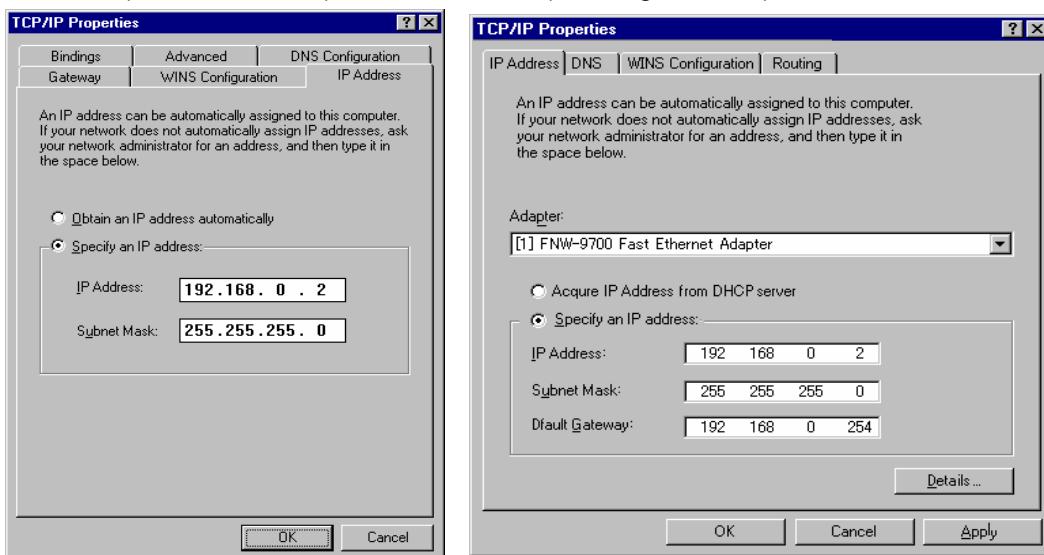
Robot controller IP address	192.168.0.20
Personal computer IP address	192.168.0.2
Robot controller port No.	10003
Communication line No. Open command COM No.	COM3:

List Example of parameter changes 2-1

Parameter name to be changed	Before/after change	Parameter value
NETIP	Before	192.168.0.20
	After	192.168.0.20 (With the default value.)
NETPORT	Before	10000, 10001, 10002, 10003, 10004, 10005, 10006, 10007, 10008, 10009
	After	〃 (Default value)
CPRCE13	Before	0
	After	2
COMDEV	Before	RS232, , , , ,
	After	RS232, , OPT13, , , ,

Next, set the personal computer IP address to 192.168.0.2. Set this value on the Network Properties screen.

Windows 95 (lower left screen), Windows NT4.0 (lower right screen)



The personal computer IP address is set with the Windows TCP/IP Property Network setting (property in network computer). Because the set-up screen differs with versions of Windows, refer to the manuals enclosed with Windows, etc., for details on setting this address.

Refer to the instruction manuals enclosed with the personal computer support software for details on setting and using the personal computer support software.

5.16.4 Example of setting parameters 2-2

(When the data link function is used: When the controller is the client)

Shows the example of the setting, when the controller is client by the data link function.

List Example of conditions 2-2

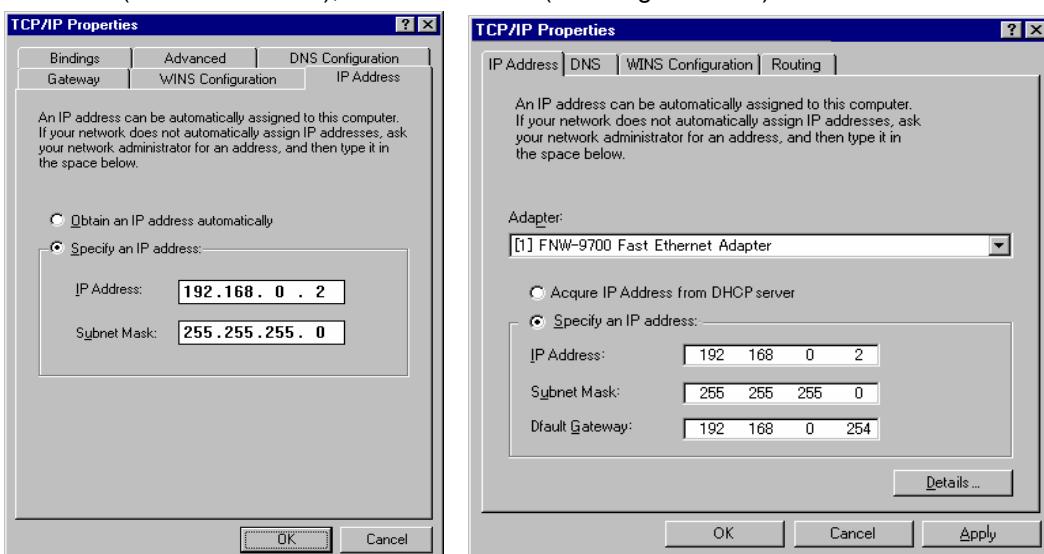
Robot controller IP address	192.168.0.20
Personal computer IP address	192.168.0.2
Robot controller port No.	10003
Communication line No. Open command COM No.	COM3:

List Example of parameter changes 2-2

Parameter name to be changed	Before/after change	Parameter value
NETIP	Before	192.168.0.20
	After	192.168.0.20 (With the default value.)
NETPORT	Before	10000, 10001, 10002, 10003, 10004, 10005, 10006, 10007, 10008, 10009
	After	“ (Default value)
CPRCE13	Before	0
	After	2
COMDEV	Before	RS232, , , , ,
	After	RS232, , OPT13, , , ,
NETMODE	Before	1,1,1,1,1,1,1,1
	After	1,1,0,1,1,1,1,1
NETHSTIP	Before	192.168.0.2, 192.168.0.3, 192.168.0.4, 192.168.0.5, 192.168.0.6, 192.168.0.7, 192.168.0.8, 192.168.0.9, 192.168.0.10
	After	192.168.0.2, 192.168.0.3, 192.168.0.2, 192.168.0.5, 192.168.0.6, 192.168.0.7, 192.168.0.8, 192.168.0.9, 192.168.0.10

Next, set the personal computer IP address to 192.168.0.2. Set this value on the Network Properties screen.

Windows 95 (lower left screen), Windows NT4.0 (lower right screen)



The personal computer IP address is set with the Windows TCP/IP Property Network setting (property in network computer). Because the set-up screen differs with versions of Windows, refer to the manuals enclosed with Windows, etc., for details on setting this address.

Refer to the instruction manuals enclosed with the personal computer support software for details on setting and using the personal computer support software.

5.16.5 Example of setting parameters 3

(for using the real-time external control function)

An example of the settings for using the real-time external control function is shown below.

List Example of conditions 3

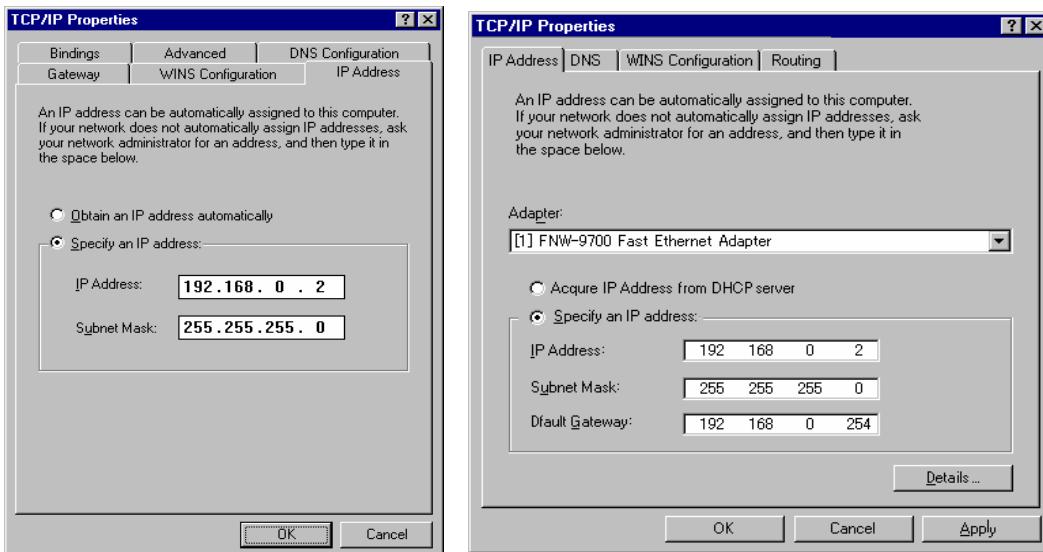
Robot controller IP address	192.168.0.20
Personal computer IP address	192.168.0.2
Robot controller port No.	10000

List Example of parameter changes 3

Parameter name to be changed	Before/after change	Parameter value
NETIP	Before	192.168.0.20
	After	192.168.0.20 (With the default value.)
NETPORT	Before	10000, 10001, 10002, 10003, 10004, 10005, 10006, 10007, 10008, 10009
	After	" (Default value)
MXTTOUT	Before	-1
	After	" (Default value)
MXTCOM1	Before	192.168.0.2
	After	" (Default value)

Next, set the personal computer IP address to 192.168.0.2. Set this value on the Network Properties screen.

Windows 95 (lower left screen), Windows NT4.0 (lower right screen)



The personal computer IP address is set with the Windows TCP/IP Property Network setting (property in network computer). Because the set-up screen differs with versions of Windows, refer to the manuals enclosed with Windows, etc., for details on setting this address.

Refer to the instruction manuals enclosed with the personal computer support software for details on setting and using the personal computer support software.

5.17 Connection confirmation

Before use, confirm the following items again.

Connection confirmation

No.	Confirmation item	Check
1	Is the teaching pendant securely fixed?	
2	Is the Ethernet cable properly connected between the controller and personal computer?	
3	Is any proper Ethernet cable used? (This cross cable is used to connect the personal computer and controller one-on-one. When using a hub with LAN, use a straight cable.)	
4	Is the parameter of the controller properly set? (Refer to 2.3 in this manual.)	
5	Is the power supply of the controller turned off once after the parameter is set?	

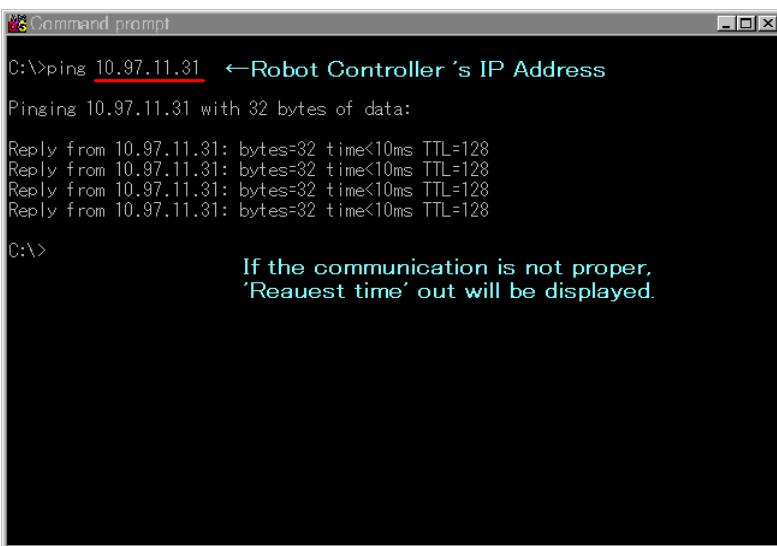
5.17.1 Checking the connection with the Windows ping command

The method for checking the connection with the Windows ping command is shown below.

Start up the " MS-DOS Prompt " from the Windows " Start " - " Programs " menu, and designate the robot controller IP address as shown below.

If the communication is normal, " Reply from ... " will appear as shown below.

If the communication is abnormal, " Request time out " will appear.



```

Command prompt
C:\>ping 10.97.11.31 ←Robot Controller's IP Address
Pinging 10.97.11.31 with 32 bytes of data:
Reply from 10.97.11.31: bytes=32 time<10ms TTL=128

C:\> If the communication is not proper,
      'Request time' out will be displayed.
  
```

5.18 Hand and Workpiece Conditions (optimum acceleration/deceleration settings)

Optimum acceleration/deceleration control allows the optimum acceleration/deceleration to be performed by LoadSet and Oadl instructions automatically in response to the load at the robot tip. The following parameters must be set correctly in order to obtain the optimum acceleration/deceleration.

This parameter is also used in the impact detection function installed in the RV-SD/RH-SDH series.

When using the impact detection function during jog operation, set HNDDAT0 and WRKDAT0 correctly.

The factory default setting is as follows.

Parameter	Value
setting the hand conditions	HNDDAT0 It varies with models.
	HNDDAT1 Maximum load, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0
	HNDDAT2 Maximum load, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0
	HNDDAT3 Maximum load, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0
	HNDDAT4 Maximum load, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0
	HNDDAT5 Maximum load, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0
	HNDDAT6 Maximum load, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0
	HNDDAT7 Maximum load, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0
setting the workpiece conditions	HNDDAT8 Maximum load, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0
	WRKDAT0 It varies with models.
	WRKDAT1 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0
	WRKDAT2 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0
	WRKDAT3 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0
	WRKDAT4 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0
	WRKDAT5 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0
	WRKDAT6 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0
	WRKDAT7 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0
	WRKDAT8 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0

Parameter values define, from the left in order, weight, size X, Y, and Z, and center of gravity X, Y, and Z. Up to eight hand conditions and eight workpiece conditions can be set. For the size of a hand, enter the length of a rectangular solid that can contain a hand. Optimal acceleration/deceleration will be calculated from the hand condition and the workpiece condition specified by a LoadSet instruction.

Parameter	Value(Factory default)
HNDHOLD1	0, 1
HNDHOLD2	0, 1
HNDHOLD3	0, 1
HNDHOLD4	0, 1
HNDHOLD5	0, 1
HNDHOLD6	0, 1
HNDHOLD7	0, 1
HNDHOLD8	0, 1

Parameter values that define grasping or not grasping is shown from the left for cases where the hand is open or closed.

"0" = Set to not grasping

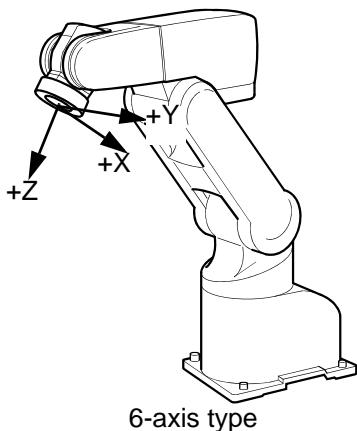
"1" = Set to grasping

Depending on the hand's open/close status, optimum acceleration/deceleration calculation will be performed for either hand-alone condition or hand-and-workpiece condition.

The hand's open/close status can be changed by executing the HOpen/HClose instruction.

The coordinate axes used as references when setting the hand and workpiece conditions are shown below for each robot model. The references of the coordinate axes are the same for both the hand and workpiece conditions. Note that all the sizes are set in positive values.

*Vertical type



6-axis type

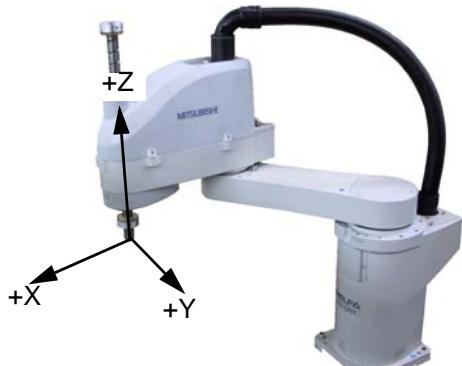
Definitions of Coordinate Axes

The tool coordinate is used for the coordinate axes.

Axes that must be set:

Only the X, Y and Z elements of the center of gravity and the X, Y and Z elements of the size must be set.

*Horizontal type



Definitions of Coordinate Axes

In the coordinate system with the tip of the J4 axis as the origin:

Z axis: The upward direction is positive.
X axis: The direction of extension in the arm orientation is positive.

Y axis: A right hand coordinate system

Axes that must be set:

Only the X element of the center of gravity and the X and Y elements of the size must be set.

5.19 About the singular point adjacent alarm

When a robot having a singular point is being operated using a T/B, a singular point adjacent alarm is generated to warn the operators of the robot if the control point of the robot approaches a singular point. Even if an alarm is generated, the robot continues to operate as long as it can perform operation unless operation is suspended. Also, an alarm is reset automatically when the robot moves away from a singular point. The following describes the details of the singular point adjacent alarm.

(1) Operations that generate an alarm

An alarm is generated if the control point of a robot approaches a singular point while a robot is performing any of the following operations using the T/B.

- 1) Jog operation (other than in joint jog mode)
- 2) Step feed and step return operations
- 3) MS position moving operation
- 4) Direct execution operation

If the robot approaches a singular point by any of the operations listed above, the buzzer of the controller keeps buzzing (continuous sound). However, the STATUS. NUMBER display on the operation panel does not change.

Also, in the case of "[1] Jog operation (other than joint jog mode)" above, a warning is displayed on the T/B screen together with the sound of the buzzer.

(2) Operations that do not generate an alarm

No alarm is generated when a robot is performing any of the operations listed below even if the control point of the robot approaches a singular point.

- 1) Additional axis jog operation initiated in joint jog mode using the T/B
- 2) When the joint interpolation instruction is executed even by an operation from the T/B
(Execution of the Mov instruction, MO position moving operation)
- 3) When the program is running automatically
- 4) Jog operation using dedicated input signals (such as JOGENA and JOGM)
- 5) When the robot is being operated using external force by releasing the brake
- 6) When the robot is stationary

5.20 About ROM operation/high-speed RAM operation function

Because the ROM operation /high-speed RAM function has some restrictions on program operation and data retention, please use it after thoroughly understanding the specifications.

(1) Overview

Initially, the robot programs are saved in the RAM (SRAM) that is backed up in the battery.

By saving the robot programs in Flash ROM (FLROM), a loss of files due to the depletion of the backup battery, damage to the programs due to unexpected power shutoff (including momentary power failure) during a file access operation, or changes or deletion of the programs and position data due to an erroneous operation.

By changing the parameter values, the access target of the programs can be switched between ROM and RAM. Once the access target of the programs is switched to ROM, it is referred to as in or during the ROM operation.

Table 5-11:ROM operation/high-speed RAM parameter list

Parameter	Description and value
ROMDRV	Switches the access target of the programs between ROM and RAM. 0 = RAM mode (initial value) 1 = ROM mode 2 = High-speed RAM mode (high-speed RAM operation : DRAM memory is used.)
BACKUP	Copies programs, parameters, common variables and error logs from the RAM area into the ROM area. SRAM -> FLROM (fixed) * If this processing is canceled while being executed, "CANCEL" is displayed in the value field.
RESTORE	Rewrites programs, parameters, common variables and error logs in the ROM area into the RAM area. FLROM -> SRAM (fixed) * If this processing is canceled while being executed, "CANCEL" is displayed in the value field.

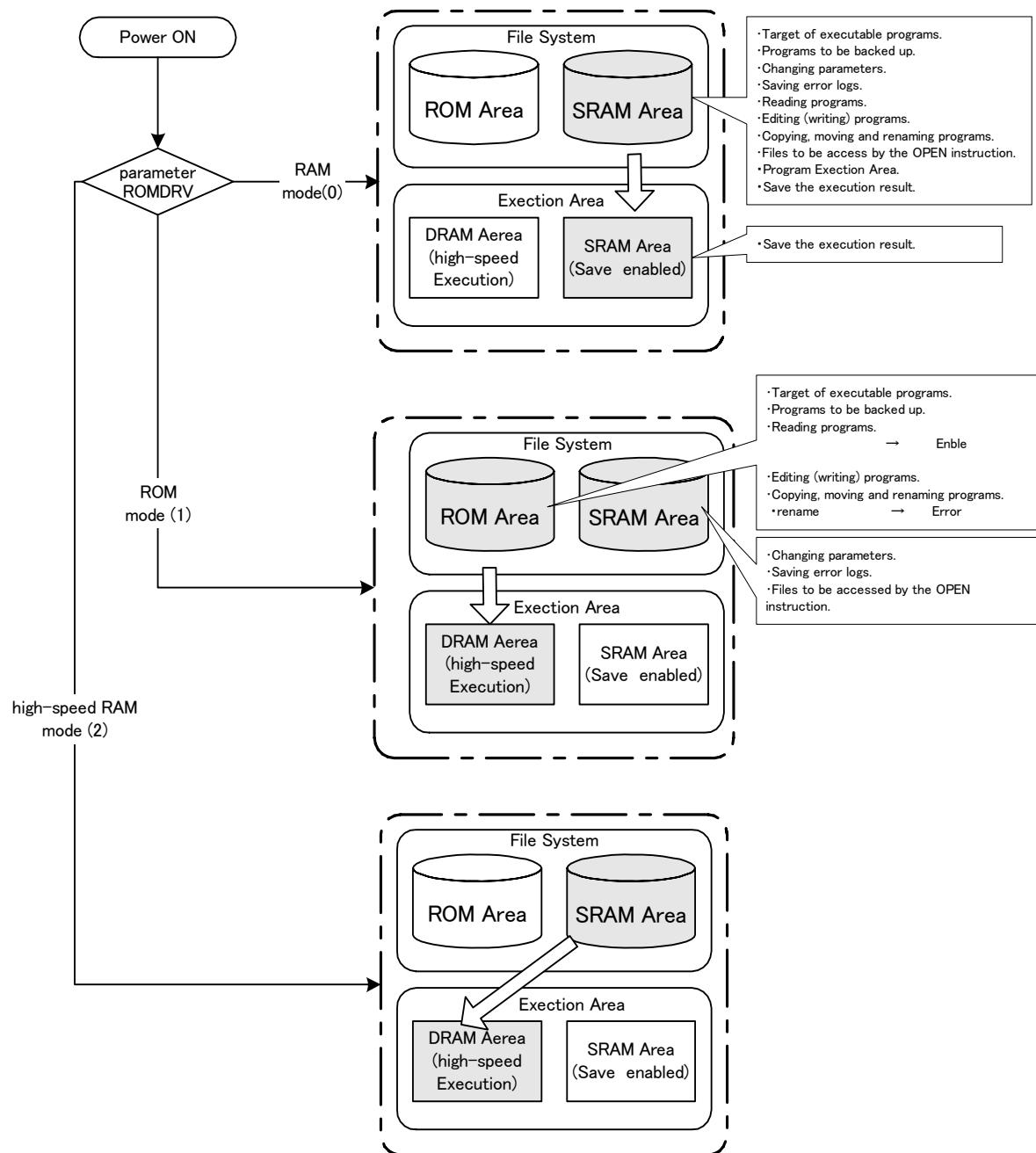
Table 5-12:Relationship between the role of each memory and the ROMDRV parameter

Memory type	Feature	ROMDRVparameter		
		0(RAM mode)	1(ROM mode)	2(High-speed RAM mode)
DRAM	High-speed execution possible Execution of programs that are erased when the power is OFF		Execution of programs (Discard the execution result)	Execution of programs (Discard the execution result)
SRAM	Not erased by power OFF Erased when a battery is consumed. Read/write enabled	Execution of programs (Save the execution result) Program management operation Read/write system data Read/write common variables Read/write programs	Read/write system data	Program management operation Read/write system data Read/write common variables Read/write programs
ROM	Not erased by power OFF Not Erased when a battery is consumed. Read only enabled		(Program management operation disabled) Read/write common variables Read/write programs	

Caution 1)"Save Parameters" and "Save Error Log" in order to save system data. The data files to be read or written by the programs (Open/Print/Input) are included.

Caution 2)Program management operation refers to the operations, such as copying, deleting and renaming the programs in the controller, by using the T/B and personal computer support software.

Table 5-13:ROM operation/high-speed RAM operation function image



DRAM is used as execution memory during ROM operation/high-speed RAM operation; it can perform language processing at a maximum speed of about 1.2 times faster than that of SRAM memory used for normal RAM operation. (The speed varies depending on the contents of each program.)

Note that the operations of the robot, such as program execution and step operation, can be performed similar to RAM operations (when starting in the RAM mode); however, there are restrictions on some operations. Please refer to the following precautions.

Precautions

* About variables

Variables may be changed by executing programs during the ROM operation/high-speed RAM operation; however, the changed values will be discarded when the controller power is turned off. The following lists the handling of variables during the ROM operation.

Variable <small>Note1)</small>	In ROM operation	In high-speed RAM operation	In RAM operation
Local variable	<p>The values of local variables are retained during program operation; however, they will be discarded when switching programs by the OP or external I/O signal, as well as when the power is turned off. The values of variables in the program called by the CallP instruction will be discarded when they return to the called program.</p> <p>The values of variables in a program called by a CallP instruction are discarded upon returning to the calling program.</p>	<p>The values of variables used in a program being executed when the power was shut down are discarded.</p> <p>They are saved when a program is selected or a CallP instruction finishes.</p>	<p>The values of variables are retained even after the power is turned off. <small>Note2)</small></p>
Program external variable	<p>The values of variables are retained until the power is turned off. (They will not be discarded by switching programs. The contents of changes will be discarded when the power is turned off.)</p>	<p>The values of variables are retained as they are even after the power is shut down.</p>	<p>The values of variables are retained even after the power is turned off.</p>
User-defined external variable		<p>The contents of changes are discarded when the power is shut down.</p>	

Note1)There are numeric value variables, character string variables, position variables and joint variables.

Note2)However, if a program is rewritten by using PC support software, the values of local variables used by programs will be discarded.

⚠ CAUTION

In the case of high-speed RAM operation and RAM operation, program external variables are maintained even after the power supply is turned off. Note, however, that variable values are not maintained in the case of power off immediately after changing the setting value of the ROMDRV parameter.

* Changing variables during program execution

If the execution of a program is aborted during the ROM operation and "Variable Monitor" (refer to [Page 63, "3.13 Operating the monitor screen".](#)) of the T/B is used, the program cannot be resumed. Although the stop lamp stays lit, if the program is executed, the program will be executed from the first line. Be careful because peripheral devices may interfere with the robot.

The values of variables cannot be changed by using "Variable Monitor" (refer to [Page 63, "3.13 Operating the monitor screen".](#)) of the T/B during the ROM operation. Program Monitor (watch function) of PC support software can be used to change the values of variables in task slots other than the editing slot.

* About programs

The target of program editing also becomes the ROM area. The programs in the controller is placed in the protected state (protect ON), and they cannot be canceled during the ROM operation. Once the ROM operation is switched to the RAM operation, the protect information reverts to the state set during the RAM operation.

Programs may be read during the ROM operation, but they cannot be written. Similarly, programs can neither be copied nor renamed.

* About parameters

Parameters and error log files are always saved in the RAM area regardless of switching between the ROM operation and the RAM operation. However, the RLNG parameter (for switching the robot language, refer to [Page 366, " RLNG".](#)) cannot be changed during the ROM operation.

* About backup

During the ROM operation, programs are backed up from the ROM area, and parameters and error log files are backed up from the RAM area.

* About direct execution

While in the ROM operation, local variables cannot be rewritten by direct execution.

* About the continue function

While in the ROM operation, the continue function is disabled even if it is set.

The continue function saves the execution status at the time of power OFF, and starts operating from the saved status the next time the power is turned on.

* About extension memory

When extension memory is installed or removed during the ROM operation, an error will occur. Install or remove extension memory only after switching to the RAM operation.

* About operating times

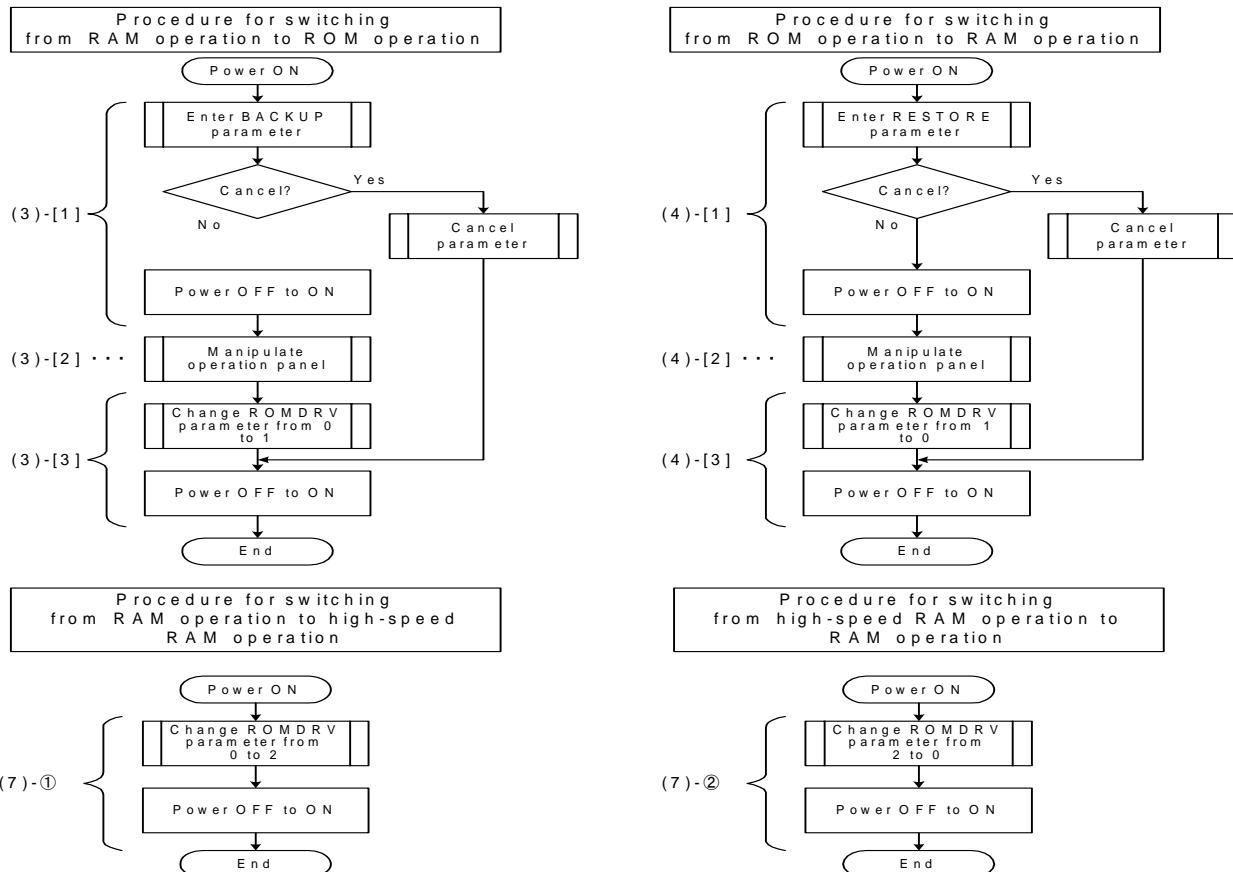
The operating times (power ON time and remaining battery time) are updated regardless of switching between the ROM and RAM operations.

* About production information

The production information monitor (program operation count, cycle time, etc.) of Personal Computer support software is not added or updated during the ROM operation.

(2) Procedures for switching between ROM and RAM

RAM operation, The following shows the procedures for switching ROM operation, RAM operation and high-speed RAM operation:



For more information about the operating procedure of each of the above, see the following pages.

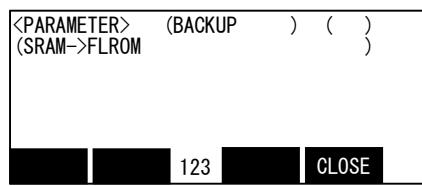
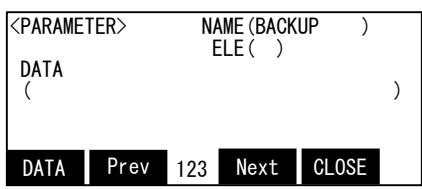
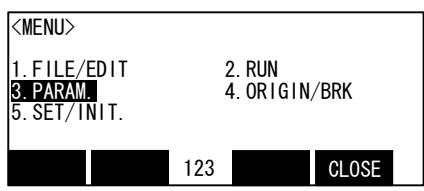
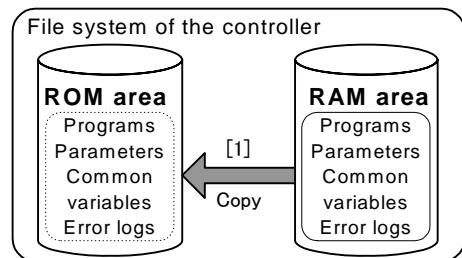
(3) Switching to the ROM operation

Use the following procedure (steps [1] to [3]) to switch to the ROM operation.

[1] Prepare to copy the information in the RAM area into the ROM area.

The programs created before the RAM operation was switched to the ROM operation are saved in the RAM area of the file system of the controller. First, copy these programs into the ROM area using the following procedure.

After the programs in the ROM area are cleared once, they are copied from the RAM area.



Return to RAM operation

- 1) Display the parameter setting screen from the maintenance screen.
- 2) Enter "BACKUP" in the parameter name field, and press the [EXE] key.
- 3) When "SRAM->FLROM" is displayed in the data field, press the [EXE] key again as is.
* Do not change the content of the data field.

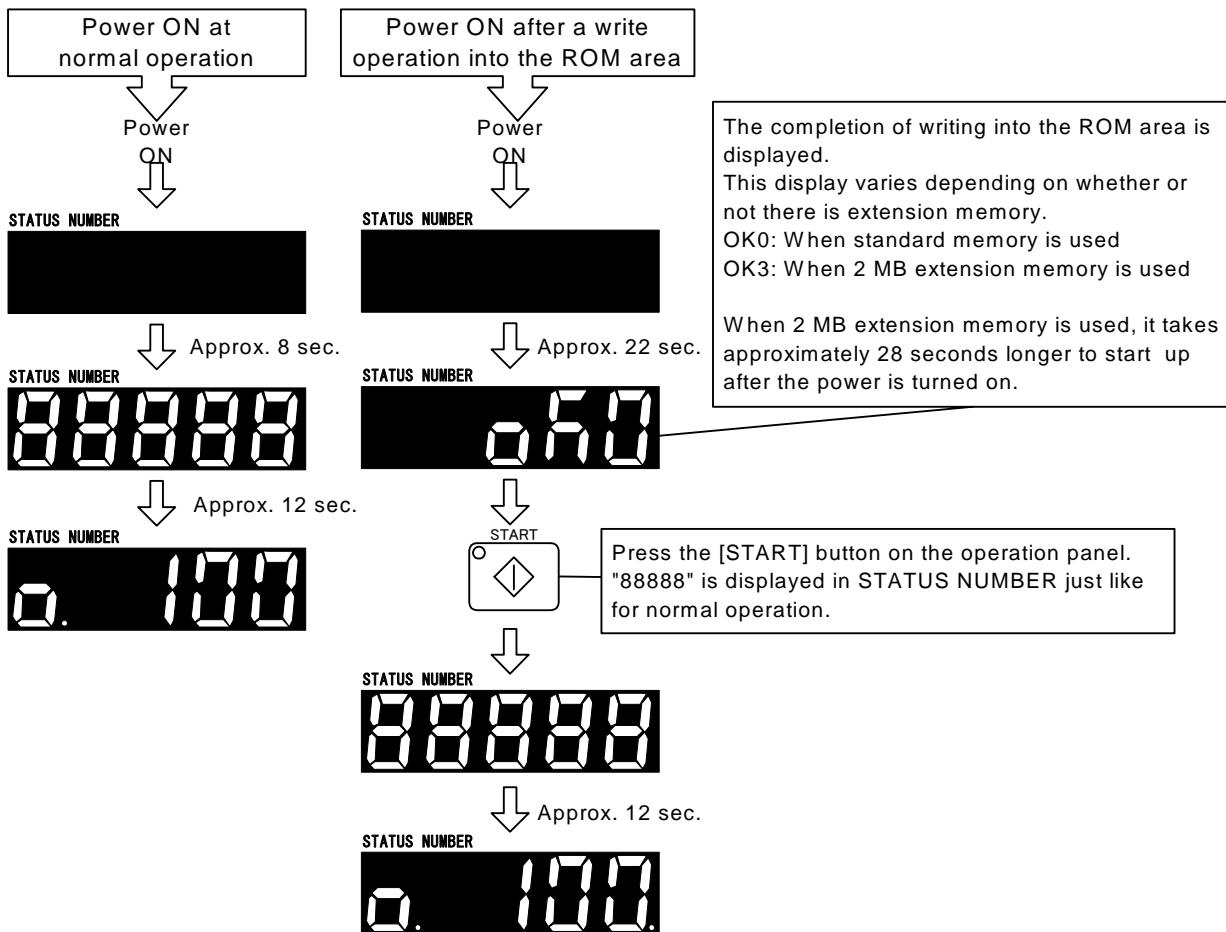
A self-check is performed to check whether or not the programs are normal prior to writing them into the ROM area. The ALWAYS program is automatically stopped during the self-check. When the program check is complete, the cursor moves to the parameter name field. If any abnormality is found during the program check, an error is output and the date of ROM write operation is registered in an error log.

[Cancel Operation]

When the function key corresponding to the "CLOSE" is pressed then data is not changed.

- 4) Be sure to shut off the power here. Also, be sure to shut off the power during [Cancel Operation] in step 3) above. Copy to the ROM area is performed the next time the power is turned on. If the power is not shut off after the operation in step 3), data will not be properly copied into the ROM area.
- 5) Turn on the power to the controller.
After the power is turned on, "OK" is displayed in "STATUS NUMBER" on the operation panel. After verifying that "OK" is displayed, press the [START] button on the operation panel. (The following page describes the detail of this operation.)

[2] Execute a copy operation by manipulating the operation panel.



Caution

If switching from the RAM operation to the ROM operation is performed without copying any program into the ROM area, there would be no program to execute, or a program in which the corrected content has not been reflected would be executed. Therefore, be sure to perform the program copy operation described above.

[3] Change the parameter value and switch to the ROM operation.

<PARAMETER>		NAME (ROMDRV)		
DATA		(0)		
DATA	Prev	123	Next	CLOSE

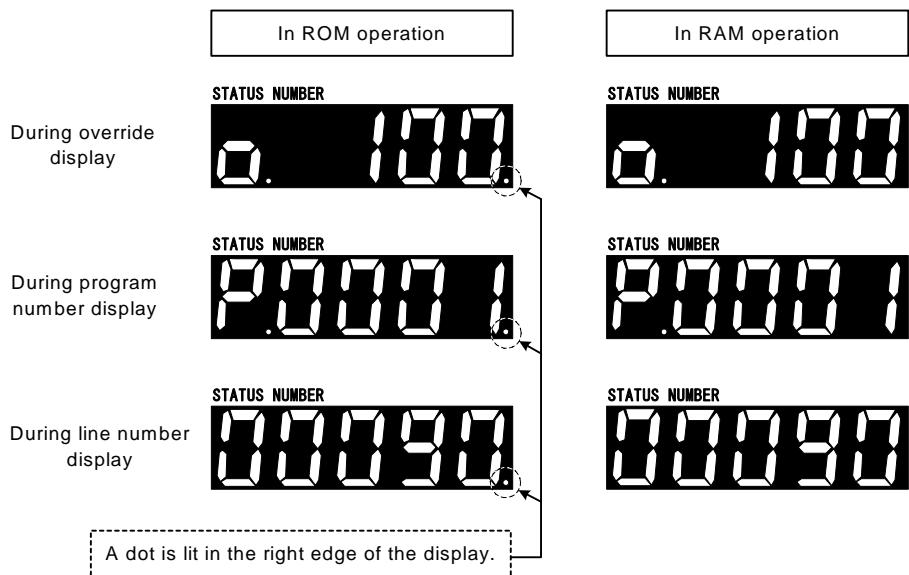
Change the value of the ROMDRV parameter from 0 to 1.

After changing it, be sure to shut off the power, and turn it on again.

<PARAMETER>		NAME (ROMDRV)		
DATA		(1)		
DATA	Prev	123	Next	CLOSE

(4) Display during the ROM operation

A dot is lit in the right edge of "STATUS NUMBER" on the operation panel during the ROM operation.



(5) Program editing during the ROM operation

It is possible to read the programs in the controller during the ROM operation; however, if a rewrite operation is performed, an error will occur. To edit a program, cancel the ROM operation (change the value of the ROMDRV parameter from 1 to 0, and reset the power to the controller) first, and then edit it. To switch back to the ROM operation after the completion of program editing, be sure to perform the operation "Copy Programs to ROM Area."

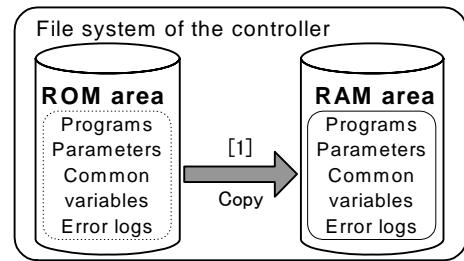
(6) Switching to the RAM operation

Use the following procedure (steps [1] to [3]) to switch to the RAM operation.

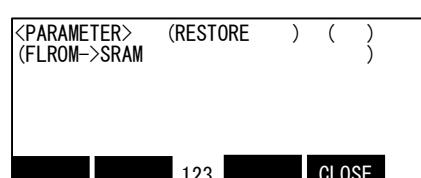
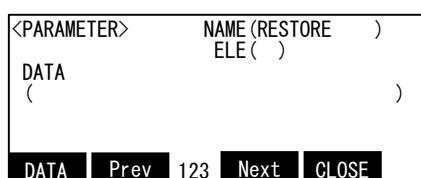
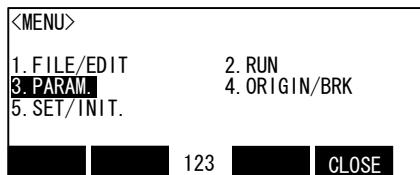
[1] Prepare to write the information in the ROM area back to the RAM area.

Write the programs and parameters written into the ROM area when the RAM operation was switched to the ROM operation back into the RAM area.

At this point, after the information (programs, parameters, values of common variables, and error logs) in the RAM area is cleared once, restore processing is performed from the ROM area.



If the information in the RAM area is restored into the ROM area, all the contents of the parameters changed during the ROM operation, the values of common variables, and logs of errors occurred are discarded. If any parameter was changed during the ROM operation, change the parameter again after switching to the RAM operation is complete.



Power shutoff

Return to RAM operation

1) Display the parameter setting screen from the menu screen.

2) Enter "RESTORE" in the parameter name field, and press the [EXE] key.

3) When "FLROM->SRAM" is displayed in the data field, press the [EXE] key again as is.

* Do not change the content of the data field.

Prepare to restore the information back into the RAM area. At this point, the ALWAYS program does not stop. When the preparation is complete, the cursor moves to the parameter name field.

[Cancel Operation]

When the function key corresponding to the "CLOSE" is pressed then data is not changed.

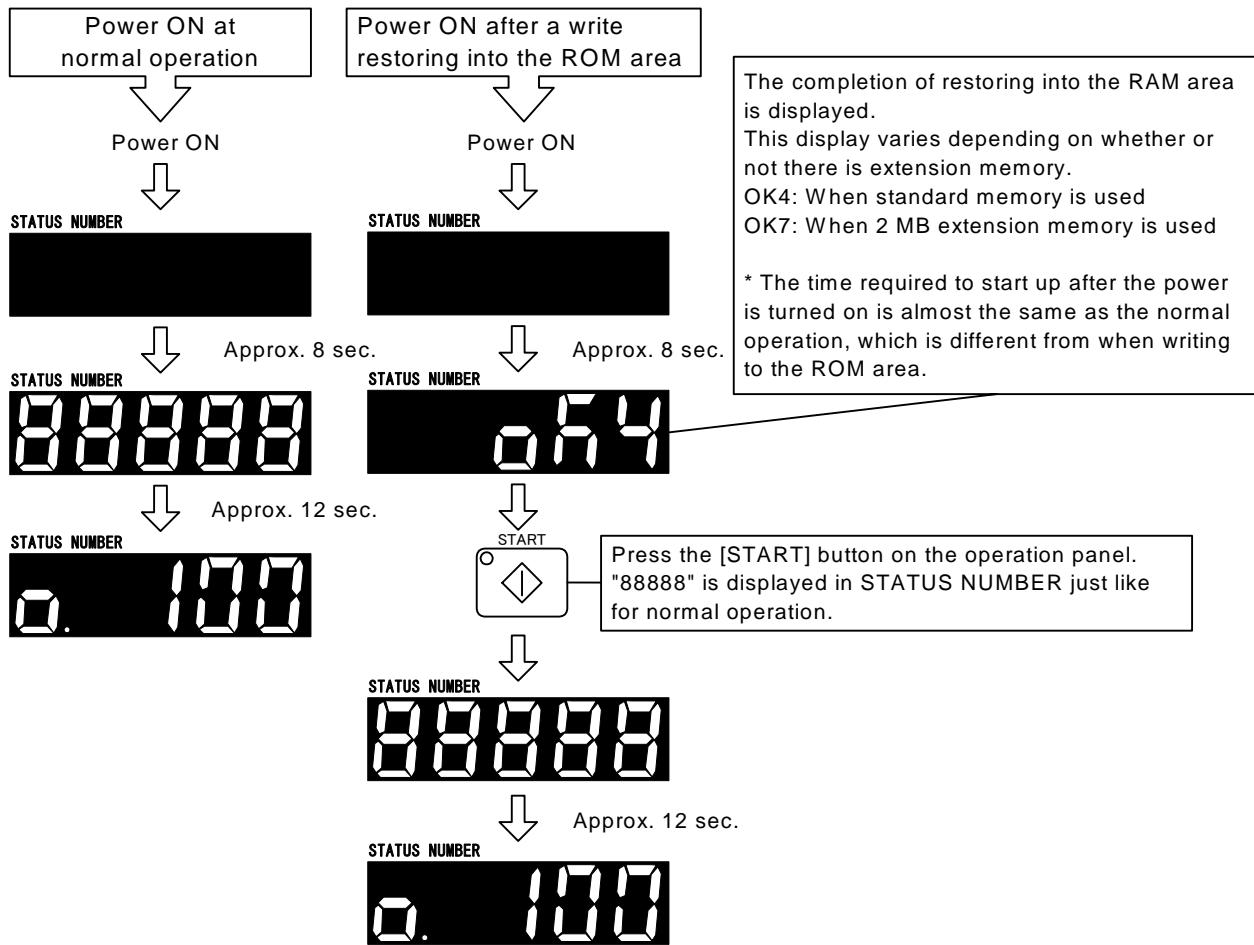
4) in the data field, press the [EXE] key again. The cursor moves to the parameter name field.

5) Be sure to shut off the power here. Also, be sure to shut off the power during [Cancel Operation] in step 3) above. Copy to the RAM area is performed the next time the power is turned on. If the power is not shut off after the operation in step 3), data will not be properly copied into the RAM area.

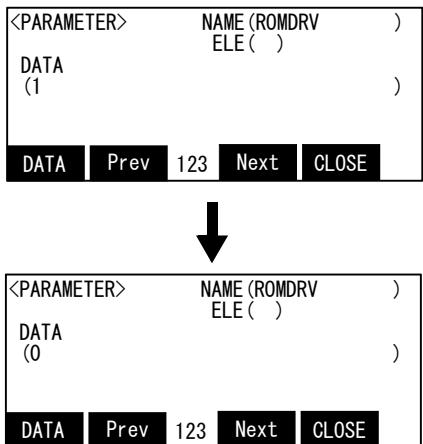
6) Turn on the power to the controller.

After the power is turned on, "OK" is displayed in "STATUS NUMBER" on the operation panel. After verifying that "OK" is displayed, press the [START] button on the operation panel. (The following page describes the detail of this operation.)

[2] Execute a restore operation by manipulating the operation panel



[3] Change the parameter value and switch to the RAM operation.



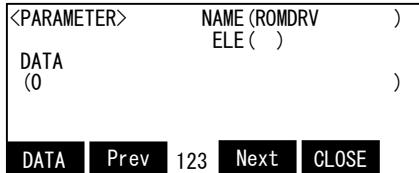
Change the value of the ROMDRV parameter from 1 to 0.

After changing it, be sure to shut off the power, and turn it on again.

(7) Switching to the high-speed RAM operation(DRAM operation)

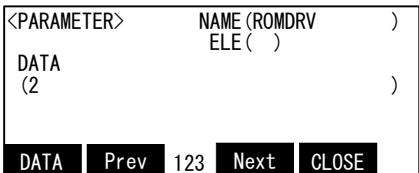
Use the following procedure to switch to the high-speed RAM operation.

[1]Change the applicable parameter and switch to high-speed RAM operation (DRAM operation).

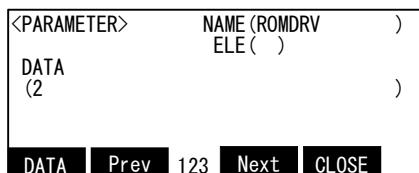


Change the value of the ROMDRV parameter from 0 to 2.

After changing it, be sure to shut off the power, and turn it on again.

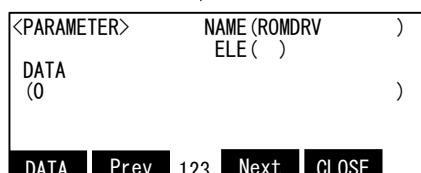


[2]Change back the parameter and return to RAM operation.



Change the value of the ROMDRV parameter from 2 to 0.

After changing it, be sure to shut off the power, and turn it on again.



5.21 Warm-Up Operation Mode

(1) Functional Overview

The acceleration/deceleration speed and servo system of Mitsubishi robots are adjusted so that they can be used with the optimum performance in a normal temperature environment. Therefore, if robots are operated in a low temperature environment or after a prolonged stop, they may not exhibit the intrinsic performance due to change in the viscosity of grease used to lubricate the parts, leading to deterioration of position accuracy and a servo error such as an excessive difference error. In this case, we ask you to operate robots in actual productions after conducting a running-in operation (warm-up operation) at a low speed. To do so, a program for warm-up operation must be prepared separately.

The warm-up operation mode is the function that operates the robot at a reduced speed immediately after powering on the controller and gradually returns to the original speed as the operation time elapses. This mode allows you to perform a warm-up operation easily without preparing a separate program. If an excessive difference error occurs when operating the robot in a low temperature environment or after a prolonged stop, enable the warm-up operation mode.

*To Use the Warm-Up Operation Mode

To use the warm-up operation mode, specify 1 (enable) in the WUPENA parameter and power on the controller again.

Note: To use the warm-up operation mode on the robot other than the RV-SD series, it is necessary to specify a joint axis to be the target of the warm-up operation mode in the WUPAXIS parameter, other than the WUPENA parameter. For more information, see "[\(2\)Function Details](#)"

*When the Warm-Up Operation Mode Is Enabled

When the warm-up operation mode is enabled, powering on the controller enters the warm-up operation status (the speed is automatically reduced). In the warm-up operation status, the robot operates at a speed lower than the specified operation speed, then gradually returns to the specified speed as the operation time of a target axis elapses. The ratio of reducing the speed is referred to as the warm-up operation override. When this value is 100%, the robot operates at the specified speed. In parameter setting at shipment from the factory, the value of a warm-up operation override changes as shown in the [Fig. 5-4](#) below according to the operation time of a target axis.

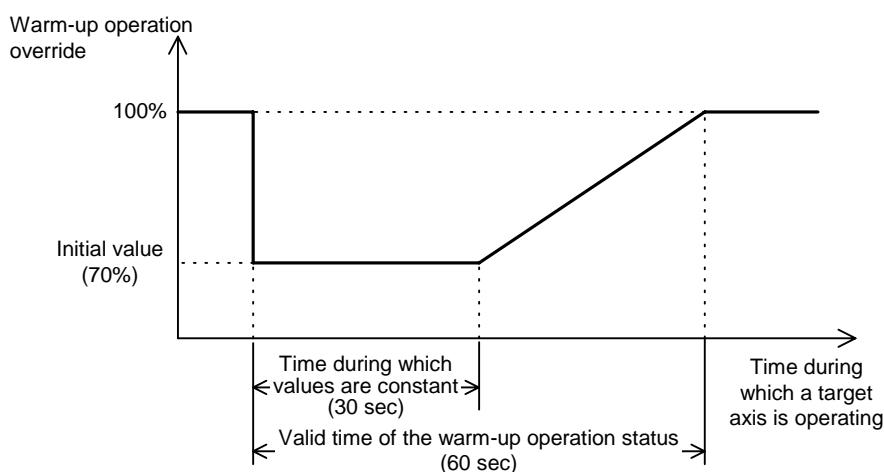


Fig.5-4:Changes in Warm-Up Operation Override

⚠ CAUTION

Even in the warm-up operation status, the robot does not decrease its speed if the MODE switch on the controller's front panel is set to "TEACH," for a jog operation or for an operation by real-time external control (MXT instruction), and operates at the originally specified speed.

⚠ CAUTION

In the warm-up operation status, because the robot operates at a speed lower than the originally specified speed, be sure to apply an interlock with peripheral units.

⚠ CAUTION

If the operating duty of the target axis is low, a servo error such as an excessive difference error may occur even when the warm-up operation mode is enabled.

In such a case, change the program, and lower the speed as well as the acceleration/deceleration speed.

Also, when STATUS NUMBER on the controller's front panel is set to override display in the warm-up operation status, an underscore (_) is displayed in the second digit from the left so that you can confirm the warm-up operation status.

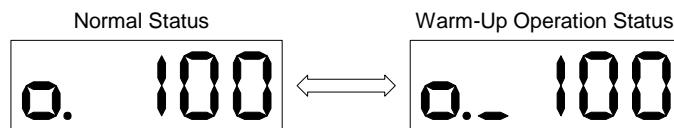


Fig.5-5:Override Display in the Warm-Up Operation Status

When a target axis operates and the warm-up operation status is canceled, the robot operates at the specified speed. Note that the joint section cools down at a low temperature if the robot continues to stop after the warm-up operation status is canceled. Therefore, if a target axis continues to stop for a prolonged period (the setting value at shipment from the factory is 60 min), the warm-up operation status is set again and the robot operates at a reduced speed.

Note 1: When powering off the controller and then powering on again, if the power-off period is short, the temperature of the robot's joint section does not decrease too much. Therefore, when powering off the controller and then powering on again after the warm-up operation status is canceled, if the power-off period is short, the robot starts in the normal status instead of the warm-up operation status.

Note 2: A target axis refers to the joint axis that is the target of control in the warm-up operation mode. It is the joint axis specified in the WUPAXIS parameter.

(2) Function Details

1)Parameters, Dedicated I/O Signals and Status Variables of the Warm-Up Operation Mode

The following parameters, dedicated I/O signals and status variables have been added in the warm-up operation mode. Refer to [Page 347, "5.1 Movement parameter"](#), [Page 428, "6.3 Dedicated input/output"](#) and [Page 87, "4 MELFA-BASIC V"](#) for details.

Table 5-14:Parameter List of the Warm-Up Operation Mode

Parameter name	Description and value
WUPENA	Designate the valid/invalid of the Warm-up operation mode. 0:Invalid/ 1: Valid
WUPAXIS	Specify the joint axis that will be the target of control in the warm-up operation mode by selecting bit ON or OFF in hexadecimal (J1, J2, Ac from the lower bits). Bit ON: Target axis/ Bit OFF: Other than target axis
WUPTIME	Specify the time (unit: min.) to be used in the processing of warm-up operation mode. Specify the valid time in the first element, and the resume time in the second element. Valid time: Specify the time during which the robot is operated in the warm-up operation status and at a reduced speed. (Setting range: 0 to 60) Resume time: Specify the time until the warm-up operation status is set again after it has been canceled if a target axis continues to stop. (Setting range: 1 to 1440)
WUPOvrd	Perform settings pertaining to the speed in the warm-up operation status. Specify the initial value in the first element, and the value constant time in the second element. The unit is % for both. Initial value: Specify the initial value of an override (warm-up operation override) to be applied to the operation speed when in the warm-up operation status. (Setting range: 50 to 100) Ratio of value constant time: Specify the duration of time during which the override to be applied to the operation speed when in the warm-up operation status does not change from the initial value, using the ratio to the valid time. (Setting range: 0 to 50)

Table 5-15:Dedicated I/O Signal List of Warm-Up Operation Mode

Parameter name	Class	Function
MnWUPENA (n=1 to 3) (Operation right required)	Input	Enables the warm-up operation mode of each mechanism. (n: FMechanism No.)
	Output	Outputs that the warm-up operation mode is currently enabled. (n: FMechanism No.)
MnWUPMD(n=1 to 3)	Output	Outputs that the status is the warm-up operation status, and thus the robot will operate at a reduced speed. (n: FMechanism No.)

Table 5-16:Status Variable of Warm-Up Operation Mode

Status variable	Function
M_Wupov	Returns the value of an override (warm-up operation override) to be applied to the command speed in order to reduce the operation speed when in the warm-up operation status.
M_Wuprt	Returns the time during which a target axis in the warm-up operation mode must operate to cancel the warm-up operation status.
M_Wpst	Returns the time until the warm-up operation status is set again after it has been canceled.

2) To Use the Warm-Up Operation Mode

To use the warm-up operation mode, enable its function with parameters. The function can also be enabled or disabled with a dedicated input signal.

*Specifying with a Parameter

To enable the warm-up operation mode with a parameter, set 1 in the WUPENA parameter. After changing the parameter, the warm-up operation mode is enabled by powering on the controller again. In the following cases, however, the warm-up operation mode will not be enabled even if 1 is set in the WUPENA parameter.

- When 0 is set in the WUPAXIS parameter (a target axis in the warm-up operation mode does not exist)
- When 0 is set in the first element of the WUPTIME parameter (the warm-up operation status period is 0 min)
- When 100 is set in the first element of the WUPOvrd parameter (the speed is not decreased even in the warm-up operation status)

When using the warm-up operation mode, change these parameters to appropriate setting values.

Note: For robots other than the RV-SD series, the setting value of the WUPAXIS parameter at shipment from the factory has been set to 0. When using the warm-up operation mode, specify a target axis in the warm-up operation mode (the joint axis to be the target of control in the warm-up operation mode; for example, a joint axis that generates an excessive difference error when operating in a low temperature environment).

*Switching with a Dedicated Input Signal

By assigning the MnWUPENA (n = 1 to 3: mechanism number) dedicated input signal, the warm-up operation mode can be enabled or disabled without powering on the controller again. Also, the current enable/disable status can be checked with the MnWUPENA (n = 1 to 3: mechanism number) dedicated output signal.

Note 1: In order for the dedicated input signal above to function, it is necessary to enable the warm-up operation mode in advance by setting the parameters described previously.

Note 2: This dedicated input signal requires the operation right of external I/O. Also, no input is accepted during operation or jog operation.

Note 3: The enable/disable status specified by this dedicated input signal is held even after the control right of external I/O is lost.

3) When the Warm-Up Operation Mode Is Enabled

When the warm-up operation mode is enabled, powering on the controller enters the warm-up operation status.

In the warm-up operation status, the robot operates at a speed lower than the actual operation speed by applying a warm-up operation override to the specified speed. The operation speed is gradually returned to the specified speed as the operation time of a target axis elapses. When the warm-up operation status is canceled, the robot will start operating at the specified speed.

*Initial Status Immediately After Power On

When the warm-up operation mode is enabled, powering on the controller enters the warm-up operation status.

However, when powering off the controller and then powering on again after the warm-up operation status is canceled, if the power-off period is short, the robot starts in the normal status instead of the warm-up operation status as the temperature of the robot's joint section has not been lowered much from power-off. To be specific, the robot starts in the normal status if the following condition is satisfied:

Condition: The robot starts in the normal status if the time during which a target axis continues to stop from the cancellation of the warm-up operation status to powering on is shorter than the time specified in the second element of the WUPTIME parameter (the resume time of the warm-up operation status).

Note that if the warm-up operation mode is switched to be enabled with the MnWUPENA (n = 1 to 3: mechanism number) dedicated input signal, the warm-up operation status is always set.

*Methods to Check the Warm-Up Operation Status

Whether the current status is the warm-up operation status or normal status can be checked in the following three methods:

- Checking with STATUS NUMBER on the controller's front panel

The current status can be checked by setting STATUS NUMBER to override display. In the warm-up operation status, an underscore (_) is displayed in the second digit from the left.

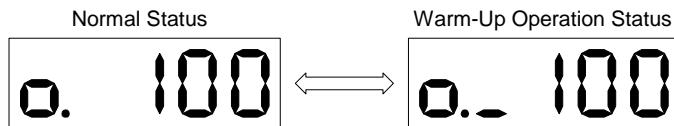


Fig.5-6:Override Display in the Warm-Up Operation Status

- Checking with a status variable

The current status can be checked by monitoring the value of the M_Wupov status variable (the value of a warm-up operation override). In the normal status, the value of M_Wupov is set to 100%; in the warm-up operation status, it is below 100%.

- Checking with a dedicated output signal

In the warm-up operation status, the MnWUPMD (n = 1 to 3: mechanism number) dedicated output signal is output.

*Switching Between the Normal Status and the Warm-Up Operation Status

When in the warm-up operation status, if a target axis in the warm-up operation mode continues operating and its operation time exceeds the valid time of the warm-up operation status, the warm-up operation status is canceled and the normal status is set. Thereafter, if the robot continues to stop, the joint section is cooled down in a low temperature environment. When a target axis continues to stop over an extended period of time and the resume time of the warm-up operation status is exceeded, the normal status switches to the warm-up operation status again.

- Canceling the warm-up operation status

If the accumulated time a target axis has operated exceeds the valid time of the warm-up operation status, the warm-up operation status is canceled and the normal status is set. Specify the valid time of the warm-up operation status in the first element of the WUPTIME parameter. (The setting value at shipment from the factory is 1 min.) If a multiple number of target axes exist, the warm-up operation status is canceled when all target axes exceed the valid time. Note that, with the M_Wuprt status variable, you can check when the warm-up operation status will be canceled after how much more time a target axis operates.

- Switching from the normal status to the warning-up operation status

If the time during which a target axis continues to stop exceeds the resume time of the warm-up operation status, the normal status switches to the warm-up operation status. Specify the resume time of the warm-up operation status in the second element of the WUPTIME parameter. (The setting value at shipment from the factory is 60 min.)

If a multiple number of target axes exist, the warm-up operation status is set when at least one of the axes exceeds the resume time of the warm-up operation status.

Note that, with the M_Wupst status variable, you can check when the status is switched to the warm-up operation status after how much more time a target axis continues to stop.

Note: If a target axis is not operating even when the robot is operating, it is determined that the target axis is stopping.

The following Fig. 5-7 shows an example of a timing chart for switching from the normal status to the warm-up operation status.

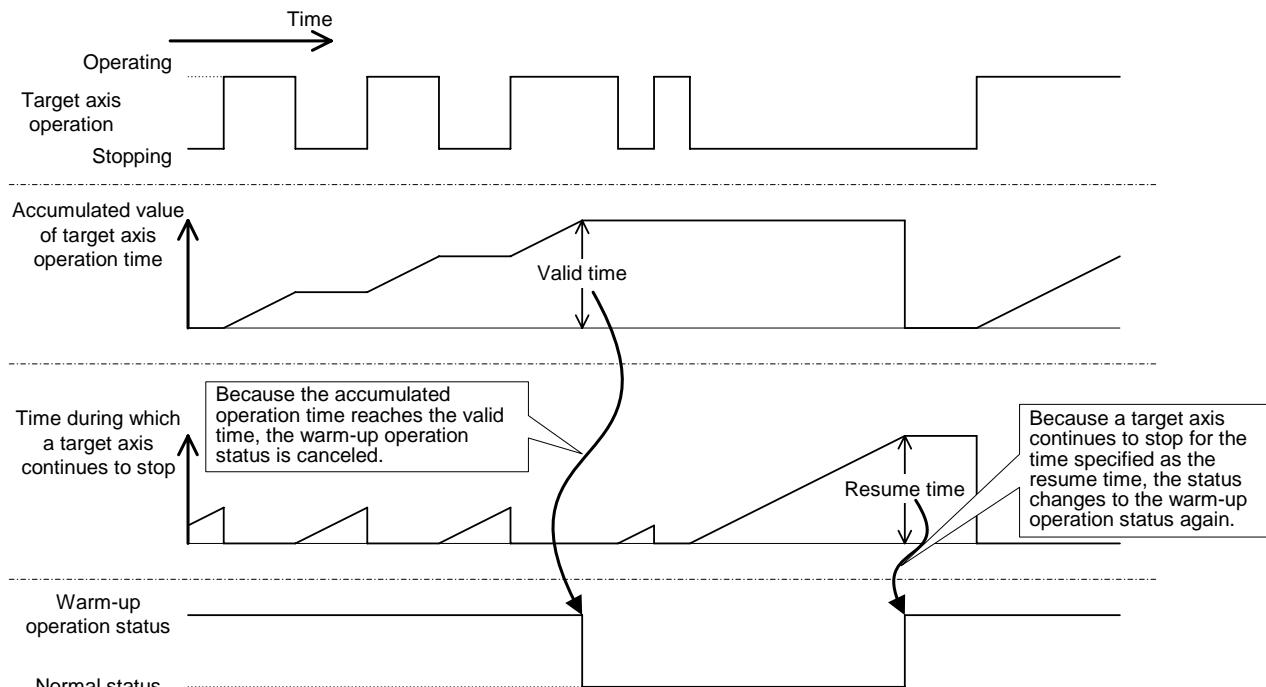


Fig.5-7:Example of Switching Between the Normal Status and the Warm-Up Operation Status

*Warm-Up Operation Override Value

An override to be applied to the operation speed in order to reduce the speed in the warm-up operation status is referred to as the warm-up operation override. The warm-up operation override changes as shown in the figure below according to the time during which a target axis operates, and is immediately reflected in the operation of the robot. Specify the initial value of the warm-up operation override and the ratio of the time during which the override does not change in relation to the valid time of the warm-up operation status using the WUPOVrd parameter. (The initial value is 70% and the ratio is 50% (= 30 sec) in the settings at shipment from the factory.)

These values can be checked with the M_Wupov status variable.

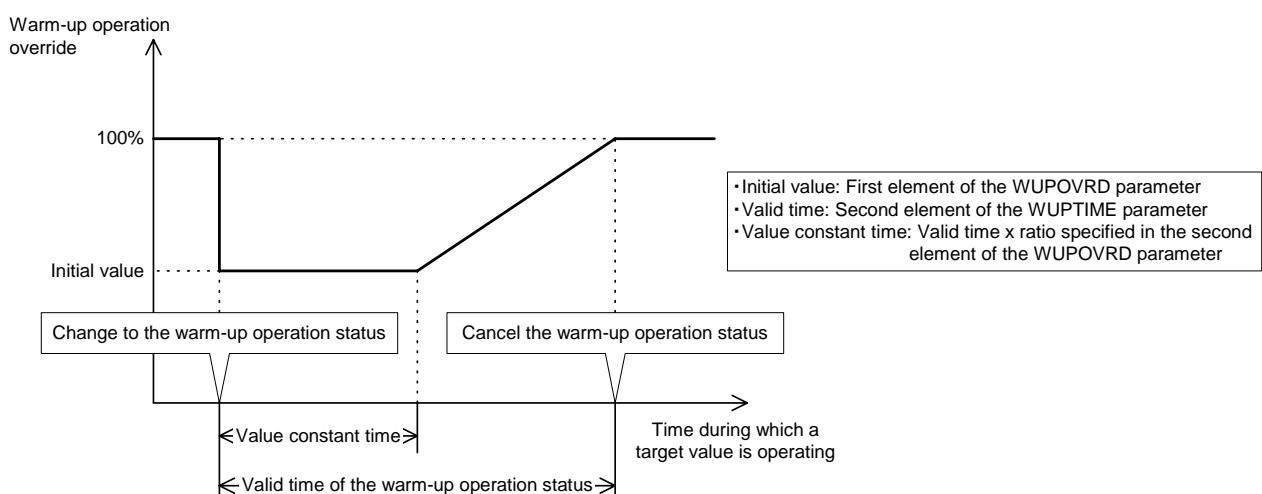


Fig.5-8:Changes in Warm-Up Operation Override

Note that the actual override in the warm-up operation status is as follows:

- During joint interpolation operation = (operation panel (T/B) override setting value) x (program override (Ovrd instruction)) x (joint override (JOvrd instruction)) x warm-up operation override
- During linear interpolation operation = (operation panel (T/B) override setting value) x (program override (Ovrd instruction)) x (linear specification speed (Spd instruction)) x warm-up operation override

Note 1:If the MODE switch on the controller's front panel is set to "TEACH," or for a jog operation or an operation by real-time external control (MXT instruction), the warm-up operation override is not reflected and the robot operates at the originally specified speed.

Note 2:In the warm-up operation status, because the robot operates at a speed lower than the originally specified speed, be sure to apply an interlock with peripheral units.

Note 3:If a multiple number of target axes exist, the warm-up operation override is calculated using the minimum operation time among the target axes. If a certain target axis does not operate and the value of the M_Wuprt status variable does not change, the value of the warm-up operation override does not change regardless how much other target axes operate.

Also, the value may return to the initial value before reaching 100% depending on whether each target axis is operating or stopping.

For example, when the value of a warm-up operation override is larger than the initial value, if a certain target axis switches from the normal status to the warm-up operation status, the operation time of that axis becomes the smallest (the operation time is 0 sec) and the warm-up operation override returns to the initial value.

(3) If alarms are generated

- 1) An excessive difference error occurs even if operating in the warm-up operation status.
 - If an error occurs when the warm-up operation override is set to the initial value, decrease the value of the initial value (the first element of the WUPOvrd parameter).
 - If an error occurs while the warm-up operation override is increasing to 100%, the valid time of the warm-up operation status or the value constant time may be too short. Increase the value of the first element of the WUPTIME parameter (valid time) or the second element of the WUPOvrd parameter (value constant time ratio).
 - If an error cannot be resolved after taking the above actions, change the operation program, and lower the speed and/or the acceleration/deceleration speed.
- 2) An excessive difference error occurs if the warm-up operation status is canceled.
 - Increase the value of the first element of the WUPTIME parameter, and extend the valid time of the warm-up operation status.
 - Check to see if the robot's load and the surrounding temperature are within the specification range.
 - Check whether the target axis continues to stop for an extended period of time after the warm-up operation status has been canceled. In such a case, decrease the value of the second element of the WUPTIME parameter, and shorten the time until the warm-up operation status is set again.
 - If an error cannot be resolved after taking the above actions, change the operation program, and reduce the speed and/or the acceleration/deceleration speed.
- 3) The warm-up operation status is not canceled at all.
 - Check the setting value of the WUPAXIS parameter to see if a joint section that does not operate at all is set as a target axis in the warm-up operation mode.
 - Check to see if a target axis has been stopping longer than the resume time (the second element of the WUPTIME parameter) of the warm-up operation status.
 - Check to see if an operation is continuing at an extremely low specified speed (about 3 to 5% in override during joint interpolation). If the specified speed is low, there is no need to use the warm-up operation mode. Thus, disable the warm-up operation mode.

5.22 About singular point passage function

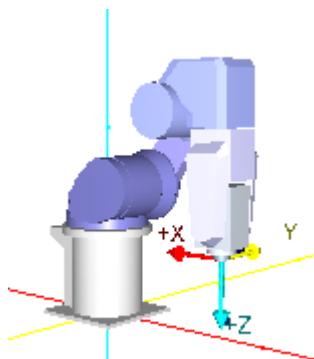
(1) Overview of the function

Mitsubishi's robots calculate linear interpolation movement and store teaching positions as position data in the XYZ coordinates system. In the case of a vertical 6-axis robot, for example, the position data is expressed using coordinate values of the robot's X, Y, Z, A, B and C axes, but the robot can be in several different postures even if the position data is the same. For this reason, the robot's position can be selected among the possible options using the coordinate values and the structure flag (a flag indicating the posture). However, there can be an infinite number of combinations of angles that a particular joint axis can take. Even if the structure flag is used, at the positions where this flag is switched, it may not be possible to operate the robot with the desired position and posture (for example, in the case of a vertical 6-axis robot, axes J4 and J6 are not uniquely determined when axis J5 is 0 degree). Such positions are called singular points, and they cannot be reached through XYZ jog and linear interpolation-based operation. To avoid this problem in the past, the operation layout had to be designed such that no singular points would exist in the working area, or the robot had to be operated using joint interpolation if it could not avoid passing a singular point. The singular point passage function allows a robot to pass singular points through XYZ jog and linear interpolation, which helps increasing the degree of freedom for the layout design by enlarging the working area by linear interpolation.

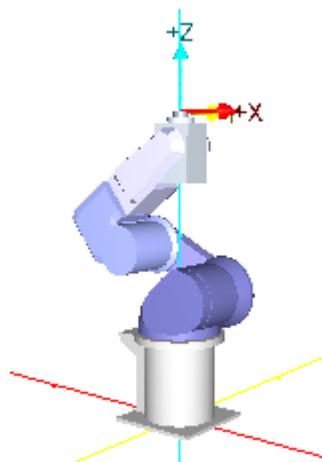
*Positions of singular points that can be passed

The positions of singular points that the singular point passage function allows the robot to pass are as follows.

In the case of vertical 6-axis robots

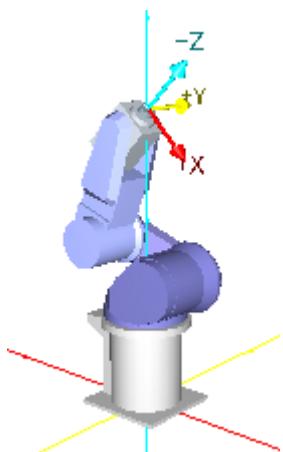


<1> Positions where axis J5 = 0 degree
In these positions, the structure flag switches between NonFlip and Flip.



<2> Positions where the center of axis J5 coincides with the rotation axis of axis J1
In these positions, the structure flag switches between Right and Left.

In the case of vertical 5-axis robots

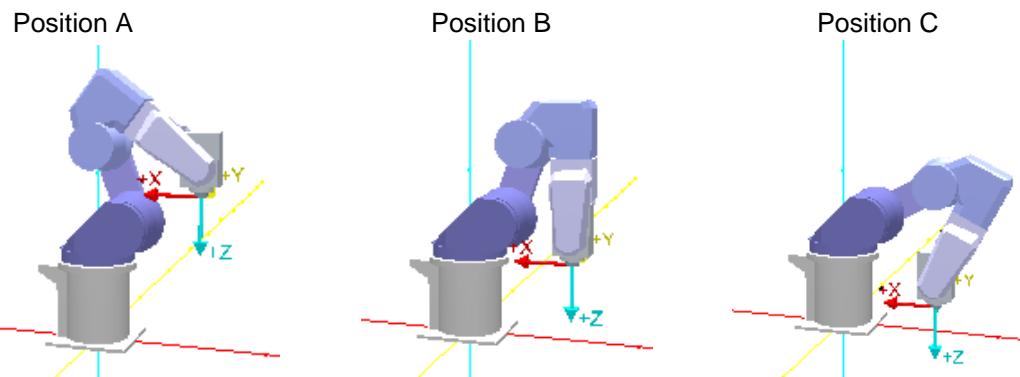


<1> Positions where the center of mechanical interface coincides with the rotation axis of axis J1
In these positions, the structure flag switches between Right and Left.

*Operation when the singular point passage function is valid

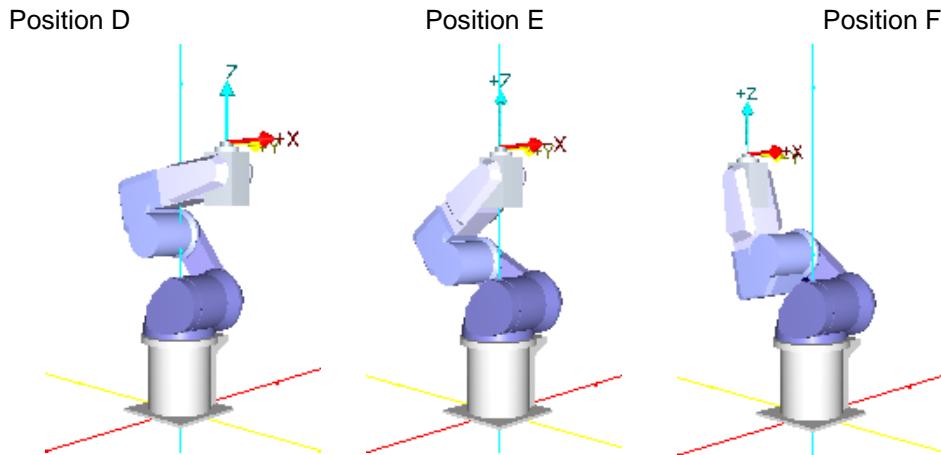
When the singular point passage function is made valid, the robot can move from position A to position C via position B (the position of a singular point) and vice versa through XYZ jog and linear interpolation operation. In this case, the value of the structure flag switches before and after passing position B.

If the singular point passage function is invalid (or not supported), the robot stops before moving from position A to position B, as an error occurs. The robot stops immediately before position B in the case of XYZ jog operation.

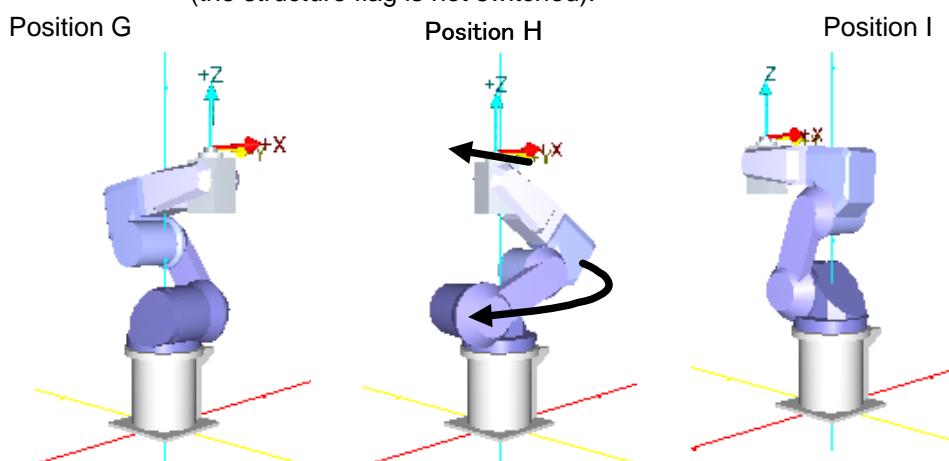


The robot can pass a singular point when the robot's motion path passes through the singular point. If the motion path does not go through the singular point (passes near the singular point), the robot continues operation without switching the value of the structure flag.

- Positions D -> E -> F: The robot's motion path passes through a singular point (the structure flag switches before and after position E).



- Positions G -> H -> I: The robot's motion path passes near a singular point (the structure flag is not switched).



CAUTION When passing near a singular point, the robot may rotate in a wide circle as in the case of position H in the figure above. Be sure to keep an eye on the robot and avoid getting in the way when working near the robot, such as when teaching positions.

*How to use the singular point passage function

In order to use the singular point passage function in jog operation, specify 1 (valid) for parameter FSPJOGMD and turn the power supply to the controller off and on again. To use the function in automatic operation, specify 2 for constant 2 in the TYPE specification of the interpolation instruction.

*Limitations

There are the following limitations to the use of the singular point passage function.

- (1) The singular point passage function cannot be used if additional axes are used for multiple mechanisms.
- (2) The singular point passage function cannot be used if synchronization control is used for additional axes of a robot.
- (3) The singular point passage function cannot be used if the compliance mode is valid.
- (4) The singular point passage function cannot be used if the collision detection function is valid.
- (5) The information collection level of the maintenance forecast function must be set to level 1 (factory setting).

(2) Singular point passage function in jog operation

In case of jog operation, the singular point passage function is specified to be valid (1) or invalid (0) by parameter FSPJOGMD.

FSPJOGMD	XYZ jog	Tool jog	3-axis XYZ jog	CYLINDER jog	JOINT jog
0 (Factory setting)	Same as in the past	Same as in the past	Same as in the past	Same as in the past	Same as in the past
1	Singular point passage XYZ jog	Singular point passage Tool jog	Same as in the past	Same as in the past	Same as in the past

- 1) For robots that cannot use the singular point passage function, changing the setting value of parameter FSPJOGMD has no effect; the same operation as in the past is performed (the models supporting the singular point passage function are the RV-3SD/3SDJ/6SD/6SDL/12SD/12SDL/18SD series).
- 2) It is not possible to specify multiple axes to perform jog operation at the same time when passing a singular point. If it is attempted to operate an axis while another axis is operating, the operation is ignored.
- 3) A singular point adjacent alarm is generated if the robot comes near a singular point when performing jog operation using a T/B. See [Page 391, "5.19 About the singular point adjacent alarm"](#).
- 4) The specification of parameter FSPJOGMD is reflected in jog operation via dedicated input signals as well.

(3) Singular point passage function in position data confirmation (position jump)

The specification of parameter FSPJOGMD is also reflected in position data confirmation (position jump).

FSPJOGMD	MO position movement	MS position movement
0 (Factory setting)	Same as in the past	Same as in the past
1	Same as in the past	Singular point passage position movement



CAUTION If an interpolation instruction (e.g., Mvs P1) is executed directly from T/B when parameter FSPJOGMD is set to 1 (valid), the robot operates with the singular point passage function enabled even if the function is not made valid by the TYPE specification.

(4) Singular point passage function in automatic operation

In order to use the singular point passage function in automatic operation, make the function valid in the TYPE specification for each target interpolation instruction.

TYPE (Type)

[Function]

Specify the singular point passage function in the TYPE specification of an interpolation instruction. The interpolation instructions that support this function are linear interpolation (Mvs), circular interpolation (Mvr, Mvr2 and Mvr3).

[Format]

TYPE[]<Constant 1>, <Constant 2>

[Terminology]

<Constant 1> 0/1 : Short cut/detour

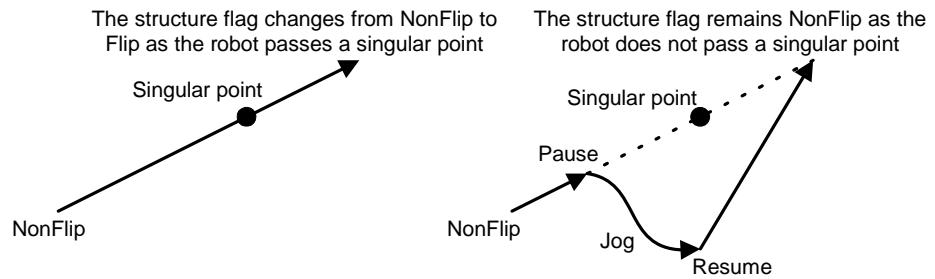
<Constant 2> 0/1/2 : Equivalent rotation/3-axis XYZ/singular point passage

[Reference Program]

- | | |
|-------------------------|--|
| 1 Mvs P1 TYPE 0,2 | ' Perform linear interpolation from the current position to P1 with the singular point passage function enabled. |
| 2 Mvr P1,P2,P3 TYPE 0,2 | ' Perform circular interpolation from P1 to P3 with the singular point passage function enabled. |

[Explanation]

- (1) A runtime error occurs if 2 is specified for constant 2 for robots that do not support the singular point passage function.
- (2) The structure flag is not checked between the starting point and the end point if the singular point passage function is specified. Moreover, since the structure flag of the target position cannot be identified, the movement range is not checked for the target position and intermediate positions before the start of operation.
- (3) If a speed is specified with the Spd instruction, the specified speed is set as the upper limit and the robot automatically lowers the speed down to the level where a speed error does not occur near a singular point.
- (4) The optimal acceleration/deceleration is not applied for interpolation instructions for which the singular point passage function is specified; the robot operates with a fixed acceleration/deceleration. At this point, if the acceleration time and the deceleration time are different due to the specification of the Accel instruction, the longer time is used for both acceleration and deceleration.
- (5) The specification of the Cnt instruction is not applied to interpolation instructions for which the singular point passage function is specified; the robot operates with acceleration/deceleration enabled.
- (6) If the current position and the starting point position are different when a circular interpolation instruction is set to be executed, the robot continues to operate until the starting point using 3-axis XYZ linear interpolation, even if the singular point passage function is specified in the TYPE specification.
- (7) If an interpolation for which the singular point passage function is specified is paused and the operation is resumed after jog movement, the robot moves to the position at which the operation was paused according to parameter RETPATH. If parameter RETPATH is set to 0 (invalid: do not return to the paused position), the structure flag is not switched unless the motion path after resuming the operation does not pass a singular point as in the figure below. Thus, the posture of the robot at the completion of interpolation may be different from the case where the operation is not paused.



- (8) If the singular point passage function is specified, the operation speed may be lowered compared to normal linear interpolation, etc. Moreover, the singular point passage function may affect the execution speed of programs as the function involves complicated processing. Specify the singular point passage function only for interpolation instructions where the function is required.

5.23 About the impact detection function

(1) Overview of the function

When the robot is operated to perform various tasks, it may interfere with workpieces and peripheral devices due to operation mistakes of operators, errors in operation programs and so on. Conventionally, in such cases, the robot would be stopped by protection functions (such as excessive error detection) of servos that control the motor drive of the robot to prevent damage to the robot hands and arms, workpieces and peripheral devices. However, because the robots operate at higher speeds and with larger motors, it becomes difficult to prevent damage solely by the servo protection functions if the load applied at interference increases. The impact detection function detects interferences at higher sensitivity than the servo's conventional protection functions and stops the robot more quickly in order to avoid damage.

⚠ WARNING

Even if the impact detection function is enabled, it is not possible to prevent injury to operators in case they get hit by moving robots. The prescribed safety rules must always be observed in all cases, whether the impact detection function is enabled or disabled.

⚠ CAUTION

Even if the impact detection function is enabled, it is not possible to prevent damage to robots, hands and workpieces due to interference with peripheral devices completely. As a general rule, pay sufficient attention to avoid interference with peripheral devices when operating and handling robots.

*Interference detection principle

If a robot interferes with peripheral devices, the actual position does not follow the position instruction of each joint axis and greater torque is generated due to the feedback control of a servo. Unless the interference is ended, the generated torque will increase further and become much larger than when there is no interference.

The impact detection function detects interferences using such servo characteristics. First, the torque required for each joint axis is estimated based on the current position instruction and load setting. Next, the values are compared with the actually generated torques for each axis one by one. If the difference exceeds the allowable range (detection level), the function judges that an interference occurred. It immediately turns the servo off and stops the robot.

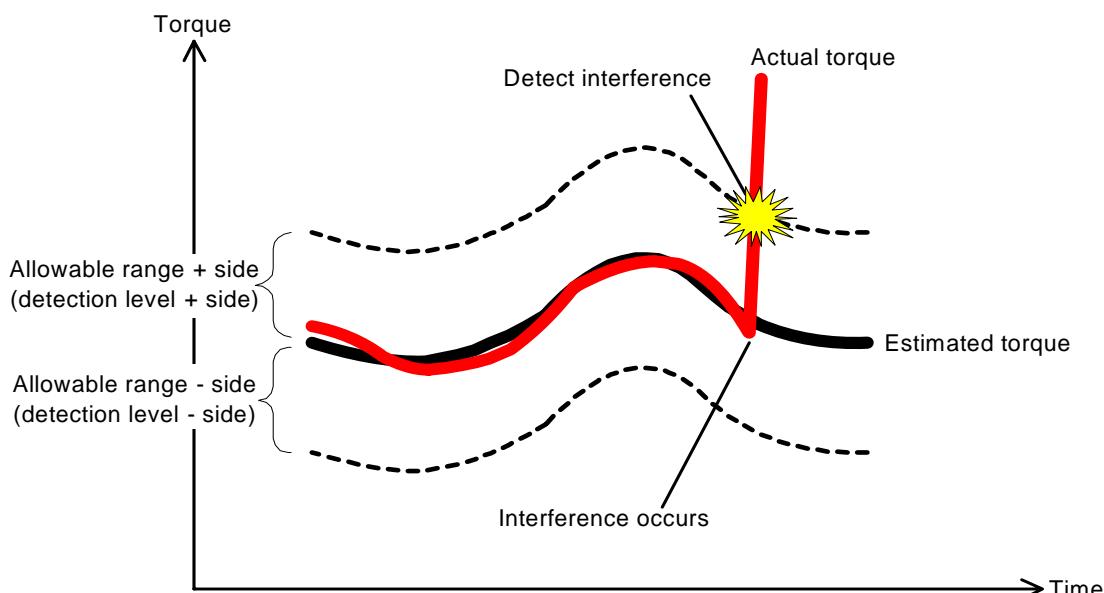


Fig.5-9:Interference detection principle

(2) Applicable models

The impact detection function is available for the following models.

Table 5-17:Models available with impact detection function

Available models
RV-3SD/3SJD/6SD/6SDL/12SD/12SDL series RH-6SDH/12SDH/18SDH series

(3) Related parameters

The following parameters are related to the impact detection function. Refer to [Page 347, "5.1 Movement parameter"](#) and [Page 389, "5.18 Hand and Workpiece Conditions \(optimum acceleration/deceleration settings\)"](#) for the detailed explanation of these parameters.

Table 5-18:Parameters related to the impact detection function

Parameter name	Description and value	Setting value at shipment
COL	Define whether to enable or disable the impact detection function as well as whether it is valid or invalid immediately after turning the power supply on. Element 1: Specify whether to enable (1) or disable (0) the impact detection function Element 2: Specify the initial state in program operation. Enable (1)/disable (0) Element 3: Specify whether the function is enabled or disabled at jog operation. Enabled (1)/disabled (0)/NOERR mode (2)	RV-SD series 0,0,1 RH-SDH series 1,0,1
COLLVL	Set the initial value of the detection level (sensitivity) of each joint axis at program operation. This value is a scaling factor that amplifies the detection level standard value prescribed in the impact detection function. The smaller the value, the higher the detection level. Setting range: 1 to 500, unit: %	The setting varies depending on the model.
COLLVLJG	Set the detection level (sensitivity) of each joint axis at jog operation (including pause status). This value is a scaling factor that amplifies the detection level standard value prescribed in the impact detection function. The smaller the value, the higher the detection level. Setting range: 1 to 500, unit: %	The setting varies depending on the model.
HNDDAT* * is 1 to 8	Set the hand conditions (via tool coordinates). HNDDAT0 is employed as the initial condition immediately after turning the power supply on. (Weight, size X, size Y, size Z, center of gravity X, center of gravity Y, center of gravity Z) Unit: kg, mm	The setting varies depending on the model.
WRKDAT* * is 1 to 8	Set the workpiece conditions (via tool coordinates). WRKDAT0 is employed as the initial condition immediately after turning the power supply on. (Weight, size X, size Y, size Z, center of gravity X, center of gravity Y, center of gravity Z) Unit: kg, mm	0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0
HNDHOLD* * is 1 to 8	Specify whether to grab (1) or not grab (0) workpieces when the HOpen and HClose instructions are executed. Element 1: Specify the status when the HOpen instruction is executed. Element 2: Specify the status when the HClose instruction is executed.	0,1

(4) How to use the impact detection function

To use the impact detection function, first specify "Enable (1)" for element 1 of the COL parameter and turn on the power supply to the control again. Next, make settings for the impact detection function (specify to enable/disable the function and the detection level) for jog operation and program operation, respectively. (Refer to [Page 415, "Table 5-18: Parameters related to the impact detection function"](#) as well.)

1) How to use the function during jog operation

During jog operation, all the settings for the impact detection function are made via parameters. For this reason, if settings such as enabled/disabled are changed while the power supply to the controller is turned on, the changes are not reflected until the power supply is turned on again the next time. Table 5-19 lists parameters used when setting the impact detection function for jog operation.

Table 5-19:Parameters set for the impact detection function used during jog operation

Parameter name	Description and value	Setting value at shipment
COL	Define whether to enable or disable the impact detection function as well as whether it is valid or invalid immediately after turning the power supply on. Element 1: Enables (1) the impact detection function (enable (1)/disable (0)) Element 3: Specify whether the function is enabled or disabled at jog operation. Enabled (1)/disabled (0)/NOERR mode (2)	RV-SD series 0,0,1 RH-SDH series 1,0,1
COLLVLJG	Set the detection level (sensitivity) of each joint axis at jog operation (including pause status). (Reference) In the case of RV-3SD The initial setting is 100, 100, 100, 100, 100, 100, 100, 100.	The setting varies depending on the model.
HNDDAT0	Set the hand conditions (via tool coordinates). (Reference) In the case of RV-3SD The initial setting is 3.5,284.0,284.0,286.0,0.0,0.0,75.0.	The setting varies depending on the model.
WRKDAT0	Set the workpiece conditions (via tool coordinates).	0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0

*Adjustment of impact detection level

The detection level (sensitivity) at jog operation is set relatively low. If a higher detection level is required, use the COLLVLJG parameter to adjust the level. Be sure to set the HNDDAT0 and WRKDAT0 parameters properly as well in order to estimate the torque accurately.

Point

If the detection level is set too high (the setting value is too small), interference may be detected erroneously depending on the robot position and posture. In such cases, lower the detection level (make the setting value larger) before using.

*Behavior when interference is detected

If an interference with peripheral devices or similar is detected during jog operation, an error numbered in the 1010's (the least significant digit is the axis number) is generated and the robot is stopped as the servo is turned off. If the robot is in the NOERR mode (2 is specified for element 3 of the COL parameter), no error is generated, but the robot stops as the servo is turned off (an error numbered in the 1010's will be recorded in the error history, however).

*Operation after interference

If the servo is turned on while a hand or arm is in contact with peripheral devices or similar, the impact is detected again, which may prevent the servo from being turned on. If an error occurs repeatedly when attempting to turn the servo on, move the arm by releasing the brake once or perform jog operation by referring to [Page 51, "3.10 Operation to Temporarily Reset an Error that Cannot Be Canceled"](#) to ensure that there is no interference.

2)How to use the function at program operation

The initial state of the impact detection function at program operation is specified by a parameter. In practice, however, the function is used by changing the setting in a program using a MELFA-BASIC V instruction. The parameters for setting the initial state and instructions related to the impact detection function are shown in the table below. Refer to [Page 153, "4.11 Detailed explanation of command words"](#) and [Page 266, "4.12 Detailed explanation of Robot Status Variable"](#) for the detailed explanation of the instructions.

Table 5-20:Parameters to be set for the impact detection function at program operation.

Parameter name	Description and value	Setting value at shipment
COL	Define whether to enable or disable the impact detection function as well as whether it is valid or invalid immediately after turning the power supply on. Element 1: Enables (1) the impact detection function (enable (1)/disable (0)) Element 2: Set enable (1) as the initial state of the impact detection function at program operation (enable (1)/disable (0)).	RV-SD series 0,0,1 RH-SDH series 1,0,1
COLLVL	Set the detection level (sensitivity) of each joint axis at jog operation (including pause status). (Reference) In the case of RV-3SD The initial setting is 100, 100, 100, 100, 100, 100, 100, 100, 100.	The setting varies depending on the model.
HNDDAT* * is 1 to 8	Set the hand conditions (via tool coordinates). (Reference) In the case of RV-3SD The initial setting is 3.5,284.0,284.0,286.0,0.0,0.0,0.750.	The setting varies depending on the model.
WRKDAT* * is 1 to 8	Set the workpiece conditions (via tool coordinates).	0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0
HNDHOLD* * is 1 to 8	Specify whether to grab (1) or not grab (0) workpieces when the HOpen and HClose instructions are executed.	0,1

Table 5-21:MELFA-BASIC V instructions and status variables used in the impact detection function at program operation

Instruction/ Status variable	Description
ColChk	Enables or disables the impact detection function or specifies the NOERR mode. Example: ColChk ON 'Enable the impact detection function.
COLLVL	Specifies the detection level (sensitivity) of the impact detection function for each joint axis. This value is a scaling factor that amplifies the detection level standard value prescribed in the impact detection function (unit: %). Example: COLLVL 80, 120, 120, 120, 50, 80, 'Specify the detection levels of axes J1 to J6.
LoadSet	Specifies the hand and workpiece conditions. Use this instruction when the hand to be used or workpieces to be grabbed are changed during program operation. Example: LoadSet 1, 0 'Specify conditions of the HNDDAT1 and WRKDAT0 parameters.
J_ColMxl	Returns the maximum difference value between the estimated torque and actual torque by converting it to the detection level. It is referenced when adjusting the arguments of the COLLVL instruction (unit: %).
M_ColSts	Returns 1 when an interference is detected. It is used as interrupt condition in the NOERR mode.
P_ColDir	Returns the robot operation direction (operation ratios in the X, Y and Z directions) when an interference is detected. It is used in retreat operation in the NOERR mode.

Point

If the impact detection function is enabled for the entire program, the probability of erroneous detection becomes higher accordingly. Hence, the detection level must be lowered in order to eliminate erroneous detection. As a result, the interference detection sensitivity may be lowered for operations for which impact detection is required. Thus, it is a good idea to use the impact detection function only for operations that may cause interference, so that the detection sensitivity may be kept high when in use.

Point

If the impact detection function is enabled, the execution time (tact time) may become longer depending on the program. In order to reduce influence on the tact time, use the impact detection function only for operations that may cause interference, rather than enabling the function for the entire program.

***Adjustment of impact detection level**

Adjust the detection level (sensitivity) at program operation according to the robot operation. As a reference, an example of adjustment procedure is shown below. be sure to set the workpiece condition and hand condition properly as well in order to estimate the torque accurately.

***Behavior when interference is detected**

If an interference with peripheral devices or similar is detected during program operation, an error numbered in the 1010's (the least significant digit is the axis number) is generated and the robot is stopped as the servo is turned off. If the robot is in the NOERR mode, no error is generated, but the robot stops as the servo is turned off (an error numbered in the 1010's will be recorded in the error history, however).

Table 5-22:Example of detection level adjustment procedure at program operation

Step	Description
1	Add the COLLVL and ColChk instructions before and after operations for which the impact detection function is used.
2	Set the detection level low (the argument of the COLLVL instruction is set to a large value such as 300) in order to prevent erroneous detection of interference.
3	Run the program and monitor the value of J_ColMxl in the target operation. Note that the value may fluctuate; repeat the target operation several times and record the J_ColMxl value each time.
4	Obtain the maximum value for each joint axis from multiple J_ColMxl values and add some margin (e.g., 20%) to the value. Then set this value as the argument of the COLLVL instruction.
5	Set the value obtained in step 4 to the COLLVL instruction and run the program to check that no erroneous detection occurs at the operation for which the impact detection function is used. If an interference is erroneously detected, gradually increase the value of the argument of the COLLVL instruction to lower the detection level until no erroneous detection occurs.

Point

When the operation speed is changed, it may become necessary to change the detection level. Operate the robot at the actual operation speed and then adjust the detection level.

Point

If the impact detection function is used for multiple robots, it may become necessary to adjust the detection level for each robot even for the same operation, due to individual differences of robots due to differences in motor characteristics and usage environment. Note also that if there are several robot models, the detection level must be adjusted for each robot.

*Program example

This program moves the robot to a retreat position by interrupt processing if an interference is detected.

```

10 Def Act 1,M_ColSts(1)=1 GoTo *HOME,S      ' Define processing to be executed if an interference is detected
20 Act 1=1                                     by interruption.

30 COLLVL 80,120,120,100,80,80,,            ' Set the detection level.
40 ColChk ON,NOERR                           ' Enable the impact detection function in the NOERR mode.

50 Mov P1
60 Mov P2                                     ' Jump to the interrupt processing if an interference is detected
                                               while executing step 50 to 80.

70 Mov P3
80 Mov P4
90 ColChk OFF                                ' Disable the impact detection function.

100 Act 1=0

.
.
.
1000 *HOME
1010 ColChk OFF
1020 Servo On
1030 PESC=P_ColDir(1)*(-5)                   ' Interrupt processing when an interference is detected
                                               'Disable the impact detection function.

1040 PDst=P_Fbc(1)+PESC
1050 Mvs PDst
1060 Error 9100                               ' Turn the servo on.
                                               ' Calculate the retreat amount (reverse operation of approxi-
                                               'mately 5 mm).
                                               ' Create a retreat position.
                                               ' Move to the retreat position.
                                               ' Pause the operation by generating user-defined L-level error.
.
.
.

```

3) Supplement

*Distinction between jog operation and program operation

The robot operation speed and tasks are quite different at jog operation and program operation. The settings for these operations are thus made independently in order to optimize the impact detection function for each type of operation. Here, the terms "at jog operation" and "at program operation" refer to the following.

At jog operation: During jog operation or during pause of automatic operation

At program operation: During automatic operation, during step feed/
return operation or during position data check operation

When these operations are executed, the status switches as shown in Fig. 5-10.

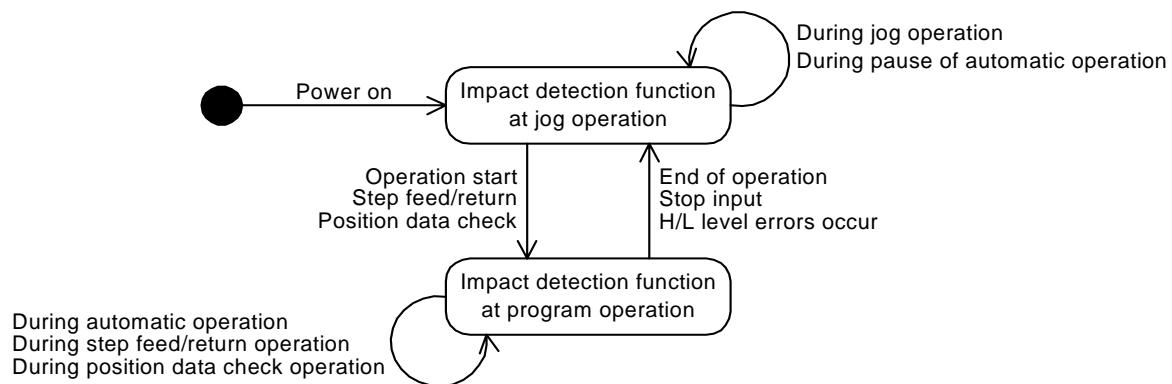


Fig.5-10:State transition diagram illustrating switch between program operation and jog operation

Thus, if the impact detection function at jog operation is enabled, for example, then even if the impact detection is set to be disabled in program operation, the setting is switched to that at jog operation if the stop button is pressed to pause the operation and the impact detection is enabled.

*Impact detection function while servo off

The impact detection function is temporarily disabled while the servo is turned off at both jog operation and program operation.

6 External input/output functions

6.1 Types

- (1) Dedicated input/output These are I/O signals that indicate the status of remote commands such as robot program execution and stoppage, information during execution and the servo power status and so on.
 Assign functions to each I/O signal. Functions can be assigned either by setting used signal numbers to each dedicated parameter (Refer to [Page 428, "6.3 Dedicated input/output"](#).) or by an emergency stop output (Refer to [Page 447, "6.6 Emergency stop input"](#).)
- (2) General-purpose input/output These signals are used for communication with the sequencer and so at the robot program. This is used at such times as when reading positioning signals from peripheral equipment and when checking the robot position.
- (3) Hand input/output These are control signals for the hand and are used for reading hand open and close instructions and information from sensors attached to the hand. These signals can be controlled at the user program and are wired up to near the tip of the hand. (Hand output signals are optional.)

Table 6-1:Overall I/O signal map

Item	I/O signal no.	Usage method
Hand input/output	9 0 0 to 9 0 7	Reference/substitution with M_In, M_Inb, M_Inw, M_Out, M_Outb, M_Outw variables Also possible with HOpen, HClose commands. Example) If M_In(900) Then M_Out(900) = 1
Sequencer link input/output	10000 to 18191	Reference/substitution with M_In, M_Inb, M_Inw, M_Out, M_Outb, M_Outw variables Example) If M_In(10080)=1 Then M_Out(10080) = 1 Note: It is not possible to output using M_Out, M_Outb, or M_Outw variables for signals to which dedicated outputs have been assigned.

6.2 Sequencer link I/O function

This function is only valid on the CRnQ-700 Series drive unit. The QnUD(H)CPU (hereafter referred to as sequencer CPU) and Q172DRCPU (hereafter referred to as robot CPU) use shared memory between CPUs, and communication via a system ladder program. The shared memory “high-speed communication area between multi CPUs ^{*1}” is used for communication. The robot CPU uses signal numbers from 10000 to 18191 for both input and output signals.

6.2.1 Parameter setting

It is necessary to set multi CPU related parameters for both the sequencer CPU and robot CPU In order to use the sequencer link function.

For the robot CPU, use RT ToolBox or a teaching box (R32TB, R56TB) to set the parameters, and for the sequencer CPU, use GX-Developer. Refer to the operation manual for each setting tool for further details.

(1) Sequencer CPU parameter setting

Use GX-Developer to perform multi CPU parameter settings.

1) CPU quantity

At the multi CPU system, set the number of CPU units with which the standard base unit is equipped.

2) Synchronous start-up between multi CPUs

It takes the robot CPU system several seconds to start up from the time the power is turned ON. It is therefore recommended that synchronous start-up be set (check box selected) at the multi CPU system.

3) High-speed communication area between multi-CPUs setting

Set the number of points in K word units. The robot CPU uses only 1K word or less and therefore 1K word should be set.

A user free area and auto refresh area can be set for the high-speed communication area between multi CPUs, however, the robot CPU (Q172DRCPU) does not support the auto refresh area, and therefore the number of points for the auto refresh area should always be set to 0. In addition, please refer to the instructions manual of each CPU for the setup of the CPUs other than robot CPU.

(2) Robot CPU parameter setting

Use RT ToolBox to perform multi CPU parameter settings.

Table 6-2:Robot CPU parameter settings

Parameter name	Details	Factory setting
QMLTCPUN	Multi CPU quantity setting At the multi CPU system, set the number of CPU units with which the standard base unit is equipped. Range: 1 to 4	2

*1)Refer to the QCPU manual (QCPU User’s Manual, Multi CPU System Edition) for details of multi CPUs and the high-speed communication area between multi CPUs.

Parameter name	Details	Factory setting								
QMLTCPUn n = 1 to 4	<p>Multi CPU high-speed communication area setting (n = 1 to 4) At the multi CPU system, set the number of points performing transmission and receipt between each CPU unit for the high speed communication function between multi CPU nos. 1 to 4. It is necessary to match the parameter settings for all CPUs. An error will occur at the sequencer CPU if the parameter settings do not match, and therefore care should be taken to ensure that the parameter settings for each CPU match.</p> <p>First element: User free area size (k points) Range: 1 to 14 (Max.)</p> <p>Table 6-3:Setting range by number of CPU</p> <table border="1"> <thead> <tr> <th>CPU quantity</th> <th>Setting range</th> </tr> </thead> <tbody> <tr> <td>2</td> <td>0 to 14K point</td> </tr> <tr> <td>3</td> <td>0 to 13K point</td> </tr> <tr> <td>4</td> <td>0 to 12K point</td> </tr> </tbody> </table> <p>Second element: No. of auto refresh points (points) Range: 0 to 14335 However, the robot CPU does not support auto refresh, and therefore the number of points for the robot CPU auto refresh should always be set to 0.</p> <p>Third element: System area size (K points) Range: 1 or 2</p> <p>Fourth element: Multi CPU synchronous start-up (1: Yes, 2: No) Robot CPUs take some time to start up and therefore the current setting of 1 (synchronous start-up) should not be changed. Make the same settings for all CPUs.</p>	CPU quantity	Setting range	2	0 to 14K point	3	0 to 13K point	4	0 to 12K point	1, 0, 1, 1
CPU quantity	Setting range									
2	0 to 14K point									
3	0 to 13K point									
4	0 to 12K point									

◇◆◇ Applicable Multi CPUs ◇◆◇

Multi CPUs are the following iQ Platform compatible CPUs. (Current as of September, 2007)

CPU type	Model	Remarks
Sequencer CPU	Q03UDCPU / Q04UDHCPU / Q06UDHCPU	The first CPU must be a sequencer CPU.
Robot CPU	Q172DRCPU	
Motion CPU	Q172DCPU / Q173DCPU	
NC CPU	Q172NCCPU	

For the robot CPU, use RT ToolBox or a teaching box (R32TB, R56TB) to set the parameters, for the sequencer CPU, use GX-Developer, for the motion controller CPU, use MT Developer, and for the NC CPU, use Remote Monitor Tool and so on. Refer to the operation manual for each setting tool for further details.

6.2.2 CPU shared memory and robot I/O signal compatibility

At the sequencer CPU, the CPU shared memory is accessed like U3E0¥G10000. The robot CPU No.n CPU shared memory accesses like U3En¥G10000.

(n = 1 to 3, Up to a maximum of three robot CPUs can be used.)

The robot CPU I/O signal numbers are all from 10000 to 18191.

Word devices are used at the sequencer side and bit devices are used at the robot side, and therefore caution is advised.

Please note that the CPU shared memory and robot I/O signal compatibility is as shown in the following table and cannot be changed.

Table 6-4:CPU shared memory and robot I/O signal compatibility

Sequencer (word device)		Robot (bit device)	
Output	U3E0¥G10000 to U3E0¥G10511	Input	Robot CPU No.1 / 10000 to 18191
	U3E0¥G10512 to U3E0¥G11023		Robot CPU No.2 / 10000 to 18191
	U3E0¥G11024 to U3E0¥G11535		Robot CPU No.3 / 10000 to 18191
Input	U3E1¥G10000 to U3E1¥G10511	Output	Robot CPU No.1 / 10000 to 18191
	U3E2¥G10000 to U3E2¥G10511		Robot CPU No.2 / 10000 to 18191
	U3E3¥G10000 to U3E3¥G10511		Robot CPU No.3 / 10000 to 18191

6.2.3 Sequence ladder example

The following is an example in which the X0 “Enable robot operation permissions” button at the operation panel is turned ON and the robot operation permissions enabled status is output to the Y20 “Robot operation permissions enabled lamp” at the operation panel. The multi CPU configuration is comprised of a sequencer QnUD(H)CPU for the first multi CPU, and a robot Q172DRCPU for the second multi CPU.

[Explanation]

<0 to 16th row>

M100 to M131 is written to the U3E0¥G10000 and U3E0¥G10001 shared device memory, and this represents the input from the sequencer to the robot. The U3E1¥G10000 and U3E1¥G10001 shared device memory is read to the bit devices for M200 to M231, and this represents the output from the robot to the sequencer.

<17 to 22nd row>

By turning X0 ON, M105 turns ON and the sequencer U3E0¥G10000 bit 5 corresponding to M105 turn ON. Consequently, robot input 10005 turns ON, and the operation permissions assigned with the dedicated input signal are enabled.

When operating permissions are enabled, robot output 10005 assigned with the dedicated output signal turns ON, and the robot U3E1¥G10000 bit 5 turns ON. Consequently, the sequencer M205 corresponding to U3E1¥G10000 bit 5 turns ON, and Y20 turns ON.

Please note that bit device M201 (U3E0¥G10000 bit 1 / in other words robot output 10001) in this example indicates controller power ON complete (A signal indicating that external input signals can be received is output.)

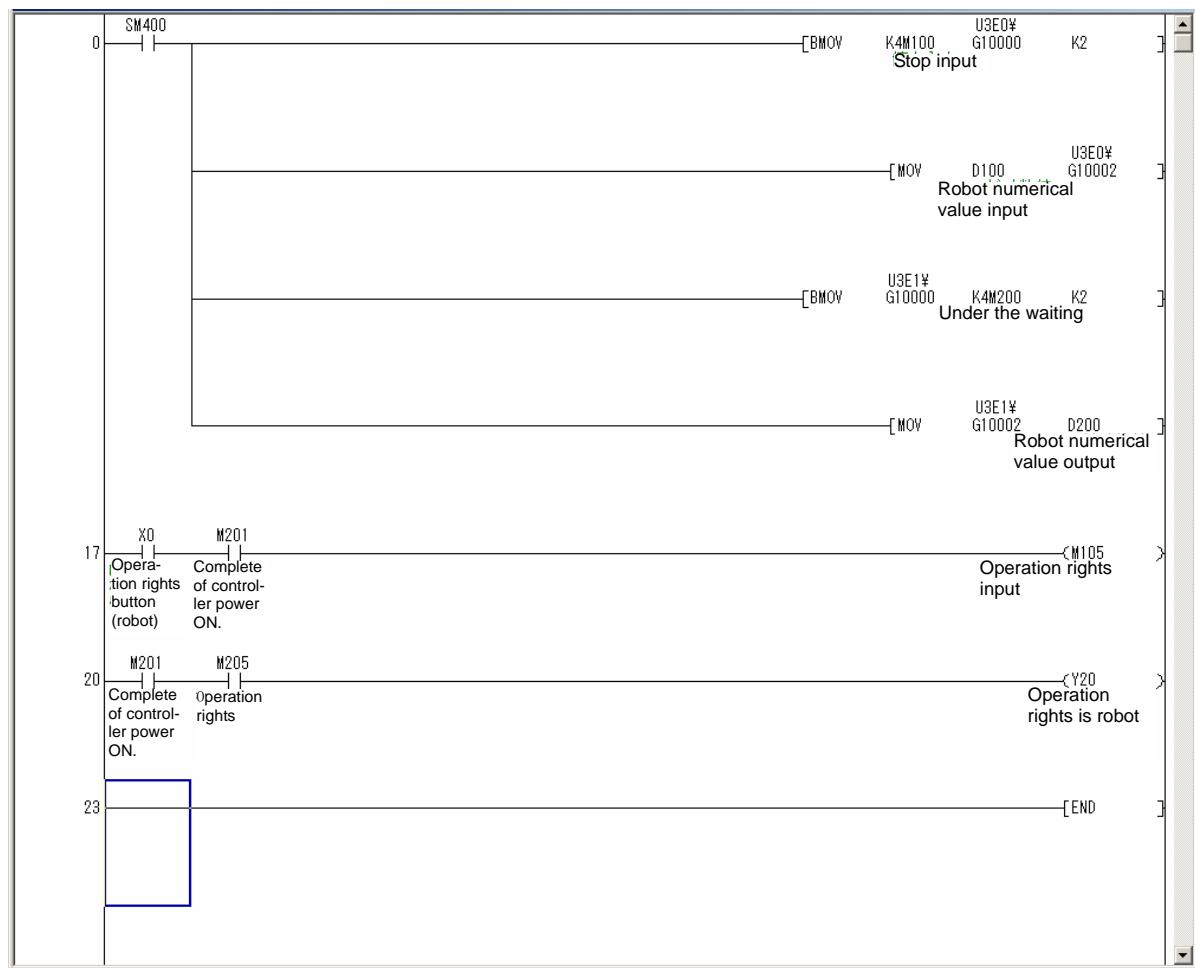


Fig.6-1:Sequence ladder example

6.2.4 Assignment of the dedicated I/O signal. (at factory shipping)

Assignment of the dedicated I/O signal at factory shipments is shown in [Table 6-5](#).

Table 6-5:Assignment of the dedicated I/O signal. (at factory shipping)

Parameter name	Input signal name (*: Operation rights is necessity)	Output signal name	Input	Output	G device Note1)
STOP	Stop input (assignment change is impossible)	Pausing output (assignment change is impossible)	10000	10000	G10000
RCREADY	–	Controller power ON ready	–	10001	
ATEXTMD	–	Remote mode output	–	10002	
TEACHMD	–	Teaching mode output	–	10003	
ATTOPMD	–	Teaching mode output	–	10004	
IOENA	Operation rights input signal	Operation rights output signal	10005	10005	
START	Start input (*)	Operating output	10006	10006	
STOPSTS	–	Stop signal input	–	10007	
SLOTINIT	Program rese t(*)	Program selection enabled output	10008	10008	
ERRRESET	Error reset input signal	Error occurring output signal	10009	10009	
SRVON	Servo ON input signal (*)	In servo ON output signal	10010	10010	
SRVOFF	Servo OFF input signal	Servo ON disable output signal	10011	10011	
CYCLE	Cycle stop input signal	In cycle stop operation output signal	10012	10012	
SAFEPOS	Safe point return input signal (*)	In safe point return output signal	10013	10013	
BATERR	–	Battery voltage drop	–	10014	
OUTRESET	General-purpose output signal reset (*)	–	10015	–	
HLVLERR	–	High level error output signal	–	10016	G1001
LLVLERR	–	Low level error output signal	–	10017	
CLVLERR	–	Warning level error output signal	–	10018	
EMGERR	–	Emergency stop output signal	–	10019	
PRGSEL	Program selection input signal (*)	–	10020	–	
OVRDSEL	Override selection input signal (*)	–	10021	–	
PRGOUT	Program No. output request	Program No. output signal	10022	10022	
LINEOUT	Line No. output request	Line No. output request	10023	10023	
OVRDOUT	Override value request	Override value output signal	10024	10024	
ERROUT	Error No. output request	Error No. output signal	10025	10025	
–	–	–	–	–	
–	–	–	–	–	
–	–	–	–	–	
–	–	–	–	–	
–	–	–	–	–	
–	–	–	–	–	
IODATA	Numeric value input 0	Numeric value output 0	10032	10032	G10002
	Numeric value input 1	Numeric value output 1	10033	10033	
	Numeric value input 2	Numeric value output 2	10034	10034	
	Numeric value input 3	Numeric value output 3	10035	10035	

Parameter name	Input signal name (*: Operation rights is necessity)	Output signal name	Input	Output	G device Note1)
I0DATA	Numeric value input 4	Numeric value output 4	10036	10036	G10002
	Numeric value input 5	Numeric value output 5	10037	10037	
	Numeric value input 6	Numeric value output 6	10038	10038	
	Numeric value input 7	Numeric value output 7	10039	10039	
	Numeric value input 8	Numeric value output 8	10040	10040	
	Numeric value input 9	Numeric value output 9	10041	10041	
	Numeric value input 10	Numeric value output 10	10042	10042	
	Numeric value input 11	Numeric value output 11	10043	10043	
	Numeric value input 12	Numeric value output 12	10044	10044	
	Numeric value input 13	Numeric value output 13	10045	10045	
	Numeric value input 14	Numeric value output 14	10046	10046	
	Numeric value input 15	Numeric value output 15	10047	10047	
HNDCNTL1	–	Hand output signal state 900	–	10048	G10003
	–	Hand output signal state 901	–	10049	
	–	Hand output signal state 902	–	10050	
	–	Hand output signal state 903	–	10051	
	–	Hand output signal state 904	–	10052	
	–	Hand output signal state 905	–	10053	
	–	Hand output signal state 906	–	10054	
	–	Hand output signal state 907	–	10055	
HNDSTS1	–	Hand input signal state 900	–	10056	G1004
	–	Hand input signal state 901	–	10057	
	–	Hand input signal state 902	–	10058	
	–	Hand input signal state 903	–	10059	
	–	Hand input signal state 904	–	10060	
	–	Hand input signal state 905	–	10061	
	–	Hand input signal state 906	–	10062	
	–	Hand input signal state 907	–	10063	
USRAREA	–	User-designated area 8-points 1	–	10064	G1004
	–	User-designated area 8-points 2	–	10065	
	–	User-designated area 8-points 3	–	10066	
	–	User-designated area 8-points 4	–	10067	
	–	User-designated area 8-points 5	–	10068	
	–	User-designated area 8-points 6	–	10069	
	–	User-designated area 8-points 7	–	10070	
	–	User-designated area 8-points 8	–	10071	

Note1) The address of the multi-CPU share device. (Address seen from the sequencer CPU side)

6.2.5 Comparison of the I/O point of the CRnQ700 and the CRn500 series

Comparison of the I/O point with the CRn500 series (our company previous series) is shown in [Table 6-6](#).

Table 6-6:Comparison of the I/O point

Item		CRnQ700 series			CRn500 series		
		Sequencer link	Remote I/O	CC-Link (option)	PROFIBUS (option)		
I/O point	Bit	8192/8192	256/256	126/126 max.	3082/3082 max.		
	Register	0/0		At four-station occupancy The number sum total of I/O data 192 words (the input or output data number is 122 words max.)			
			0/0	16/16 max. At four-station occupancy	0/0		

6.3 Dedicated input/output

The functions shown in [Table 6-7](#) are available for the dedicated input/output signals. These are used by the parallel input/output unit by assigning the signal No. in the parameter.

The signal No. is assigned by the signal No. used in the order of "input signal" and "output signal" in each parameter. Refer to [Page 74, "3.14 Operation of maintenance screen"](#) for details on setting the parameters. If a "-1" is designated for the assigned signal No., that signal will be invalidated.

The I/O parameters can be set on the T/B parameter screen or by using the maintenance tool of the PC support software (optional).

To use the dedicated I/O signals, set the key switch on the operation panel to AUTO (Ext.) beforehand.

Table 6-7:Table of dedicated input/output

Parameter name	Class	Name	Function	Signal level Note1)	Factory shipment signal number. Input, output	
					CRnQ	CRnD
RCREADY	Input	-	-	-	-1(No meaning), 10001	-1(No meaning), -1
	Output	Controller power ON ready	Outputs that the power has been turned ON and that the external input signal can be received.	-	-	-
ATEXTMD	Input	-	-	-	-1(No meaning), 10002	-1(No meaning), -1
	Output	Remote mode output	This output indicates that the key switch on the operation panel is set to AUTO (Ext.), which is a remote operation mode. This signal must be turned ON before any control tasks using I/O signals can be performed.	-	-	-
TEACHMD	Input	-	-	-	-1(No meaning), 10003	-1(No meaning), -1
	Output	Teaching mode output	This output indicates that the key switch on the operation panel is set to Teaching mode.	-	-	-
ATTOPMD	Input	-	-	-	-1(No meaning), 10004	-1(No meaning), -1
	Output	Automatic mode output	This output indicates that the key switch on the operation panel is set to AUTO (OP),	-	-	-
IOENA	Input	Operation rights input signal	Sets the validity of the operation rights for the external signal control.	Level	10005, 10005	5, 3
	Output	Operation rights output signal	Outputs the operation rights valid state for the external signal control. The operation right is given when the operation right input signal is ON, the mode switch is set to AUTO (Ext.), and there is no other device that currently has the operation right.	-	-	-
START (Operation right required)	Input	Start input	This input starts a program. To start a specific program, select the program using the program selection signal "PRGSEL" and numerical input "IODATA," and then input the start signal. Note that when the parameter "PST" is enabled, the system reads the program number from the numerical input (IODATA) and starts the corresponding program (i.e., program selection becomes no longer necessary). All task slots are executed during multitask operation. However, slots whose starting condition is set to ALWAYS or Error via a parameter "SLT***" will not be executed.	Edge	10006,	3,
	Output	Operating output	This output indicates that a program is being executed. During multitask operation, this signal turns ON when at least one task slot is operating. However, slots whose starting condition is set to ALWAYS or Error via a parameter "SLT***" will not be executed.	-	10006	0

Parameter name	Class	Name	Function	Signal level Note1)	Factory shipment signal number. Input, output	
					CRnQ	CRnD
STOP	Input	Stop input	This input stops the program being executed. (This does not apply to slots whose starting condition is set to ALWAYS or Error.) The stop input signal No. is fixed to 0, and cannot be changed. All task slots are stopped during multitask operation. However, slots whose starting condition is set to ALWAYS or Error via a parameter "SLT**" will not be executed. Contacts A and B may be changed using the parameter INB.	Level	10000 (Cannot change),	0(Cannot change),
		Pausing output	This output indicates that the program is paused. Turns ON when there is not slot multitask running, and at least one slot is pausing. However, slots whose starting condition is set to ALWAYS or Error via a parameter "SLT**" will not be executed.		10000	-1
STOP2	Input	Stop input	This input stops the program being executed. (The specification is the same as for the STOP parameter.) Unlike the STOP parameter, signal numbers can be changed.	Level	-1,	-1
		Pausing output	This output indicates that the program is paused. (The specification is the same as for the STOP parameter.)		-1	-1
STOPSTS	Input	-	-		-1(No meaning),	-1(No meaning),
	Output	Stop signal input	Outputs that the stop is being input. (Logical ADD of all devices.)		10007	-1
SLOTINIT (Operation right required)	Input	Program reset	This input cancels the paused status of the program and brings the executing line to the top. Executing a program reset makes it possible to select a program. In the multitask mode, the program reset is applied to all task slots. However, slots whose starting condition is set to ALWAYS or Error via a parameter "SLT**" will not be executed.	Edge	10008,	-1,
		Program selection enabled output	Outputs that in the program selection enabled state. Turns ON when program are not running or pausing. In multitask operation, this output turns ON when all task slots are neither operating nor paused. However, slots whose starting condition is set to ALWAYS or Error via a parameter "SLT**" will not be executed.		10008	-1
ERRRESET	Input	Error reset input signal	Releases the error state.	Edge	10009,	2,
	Output	Error occurring output signal	Outputs that an error has occurred.		10009	2
SRVON (Operation right required)	Input	Servo ON input signal	This input turns ON the servo power supply for the robot. With a multi-mechanism configuration, the servo power supplies for all mechanisms will be turned ON.	Edge	10010,	4,
		In servo ON output signal	This output turns ON when the servo power supply for the robot is ON. If the servo power supply is OFF, this output also remains OFF. With a multi-mechanism configuration, this output turns ON when the servo of at least one mechanism is ON.		10010	1

Parameter name	Class	Name	Function	Signal level Note1)	Factory shipment signal number. Input, output	
					CRnQ	CRnD
SRVOFF	Input	Servo OFF input signal	This input turns OFF the servo power supply for the robot.(Applicable to all mechanisms) The servo cannot be turned ON while this signal is being input.	Level	10011,	1,
	Output	Servo ON disable output signal	This output indicates a status where the servo power supply cannot be turned ON. (Echo back)			-1
AUTOENA	Input	Automatic operation enabled input	Disables automatic operation when inactive. If this signal is inactive, and the AUTO mode is entered, E5010 will occur. This input is used to interlock the operations via the operation panel with the I/O signals. Use of this input is not a requirement.		-1,	-1,
	Output	Automatic operation enabled output	Outputs the automatic operation enabled state.		-1	-1
CYCLE	Input	Cycle stop input signal	Starts the cycle stop.	Edge	10012,	-1,
	Output	In cycle stop operation output signal	Outputs that the cycle stop is operating. Turns OFF when the cycle stop is completed.			-1
MELOCK (Operation right required)	Input	Machine lock input signal	Sets/releases the machine lock state for all mechanisms. This can be set or released when all slots are in the program selection state. Signal level will be set to Level when program selection is enabled.		-1,	-1,
	Output	In machine lock state output signal	Outputs the machine lock state. This turns On when at least one mechanism is in the machine lock state. During the machine lock state, the robot will not move, and program operation will be enabled.		-1	-1
SAFEPOS (Operation right required)	Input	Safe point return input signal	Requests the safe point return operation. This signal initiates a joint interpolation movement to the position set by the parameter "JSAFE." The speed is determined by the override setting. Be careful not to interfere with peripheral devices.	Edge	10013,	-1,
	Output	In safe point return output signal	Outputs that the safe point return is taking place.			-1
BATERR	Input	-	-		-1(No meaning),	-1(No meaning),
	Output	Battery voltage drop	Outputs that the controller battery voltage is low. The output is turned off when the controller power supply is reconnected after the battery replacement. *The cumulative time where the controller power supply is turned off exceeds 14600 hours. The output is turned off if the battery depletion time is reset.		10014	-1
OUTRESET (Operation right required)	Input	General-purpose output signal reset	Resets the general-purpose output signal. The operation at the input is set with parameters ORST0 to ORST224.	Edge	10015,	-1,
	Output	-	-			-1(No meaning)
HLVLERR	Input	-	-		-1(No meaning),	-1(No meaning),
	Output	High level error output signal	Outputs that a high level error is occurring.		10016	-1
LLVLERR	Input	-	-		-1(No meaning),	-1(No meaning),
	Output	Low level error output signal	Outputs that a low level error is occurring.		10017	-1
CLVLERR	Input	-	-		-1(No meaning),	-1(No meaning),
	Output	Warning level error output signal	Outputs that a warning level error is occurring.		10018	-1
EMGERR	Input	-	-		-1(No meaning),	-1(No meaning),
	Output	Emergency stop output signal	Outputs that an emergency stop is occurring.		10019	-1

Parameter name	Class	Name	Function	Signal level Note1)	Factory shipment signal number. Input, output	
					CRnQ	CRnD
SnSTART (n=1 to 32) (Operation right required)	Input	Slot n start input	Starts each slot. n=1 to 32	Edge	-1,	-1,
	Output	Slot n in operation output	Outputs the operating state for each slot. n=1 to 32		-1	-1
SnSTOP (n=1 to 32)	Input	Slot n stop input	Outputs the operating state for each slot. n=1 to 32	Level	-1,	-1,
	Output	Slot n in pausing output	Outputs that each slot and program is temporarily stopped. n=1 to 32		-1	-1
MnSRVOFF (n=1 to 3)	Input	Mechanism n servo OFF input signal	This signal turns OFF the servo for each mechanism. n=1 to 3 The servo cannot be turned ON while this signal is being input.	Level	-1,	-1,
	Output	Mechanism n servo ON disabled output signal	Outputs the servo ON disabled state. (Echo back)			-1
MnSRVON (n=1 to 3) (Operation right required)	Input	Mechanism n servo ON input signal	Turns the servo for each mechanism ON. n=1 to 3	Edge	-1,	-1,
	Output	Mechanism n in servo ON output signal.	Turns the servo for each mechanism ON. n=1 to 3			-1
MnMELOCK (n=1 to 3) (Operation right required)	Input	Mechanism n machine lock input signal	Sets/releases the machine lock state for each mechanism. n=1 to 3	Level	-1,	-1,
	Output	Mechanism n in machine lock output signal	Outputs that the machine lock state is entered. n=1 to 3			-1
PRGSEL (Operation right required)	Input	Program selection input signal	Designates the setting value for the program No. with numeric value input signals. The program for slot 1 is selected. Output this signal when at least 30 ms has elapsed following the start of output to the numerical input (IODATA). This signal should also be output to the robot for at least 30 ms.	Edge	10020	-1,
	Output	-	-			
OVRDSEL (Operation right required)	Input	Override selection input signal	Designates the setting value for the override with the numeric value input signals. Output this signal when at least 30 ms has elapsed following the start of output to the numerical input (IODATA). This signal should also be output to the robot for at least 30 ms.	Edge	10021	-1,
	Output	-	-			
IODATA	Input	Numeric value input (Start bit number, end bit number)	Numerical values are read as binary values. *Program number (Read by the PRGSEL) If the parameter "PST" is enabled, it is read by the start signal. *Override (Read by the OvrdsSEL) The bit width can be set arbitrarily. However, the accuracy of output values cannot be guaranteed when they exceed the set bit width. Output this input to the robot for at least 30 ms before inputting the PRGSEL or other setting signals.	Level	Note2) 10032(Start bit), 10047(End bit),	Note2) -1(Start bit), -1(End bit),
	Output	Numeric value output (Start bit number, end bit number)	Numerical values are output as binary values. *Program number (Output by the PRGOUT), *Override (Output by the OvrdsOUT), *Outputs the line number (output by the LINE-OUT) *Error number (output by the ERROUT). The bit width can be set arbitrarily. However, the accuracy of output values cannot be guaranteed when they exceed the set bit width. Read this signal when at least 30 ms has elapsed following the start of input of a program number (PRGOUT) or other signal to the robot.		10032(Start bit), 10047(End bit)	-1(Start bit), -1(End bit)

Parameter name	Class	Name	Function	Signal level Note1)	Factory shipment signal number. Input, output	
					CRnQ	CRnD
PRGOUT	Input	Program No. output request	The program number for task slot 1 is output to the numerical output (IODATA). After the start of inputting this signal to the robot, wait at least 30 ms before reading the numerical output (IODATA) signal.	Edge	10022,	-1,
	Output	Program No. output signal	The "program number output in progress" status is output to the numerical output.		-1	-1
LINEOUT	Input	Line No. output request	The line number for task slot 1 is output to the numerical output (IODATA). After the start of inputting this signal to the robot, wait at least 30 ms before reading the numerical output (IODATA) signal.	Edge	10023,	-1,
	Output	Line No. output request	The "line number output in progress" status is output to the numerical output.		-1	-1
OVRDOUT	Input	Override value request	The OP override is output to the numerical output (IODATA). After the start of inputting this signal to the robot, wait at least 30 ms before reading the numerical output (IODATA) signal.	Edge	10024,	-1,
	Output	Override value output signal	The "override output in progress" status is output to the numerical output.		-1	-1
ERROUT	Input	Error No. output request	The error number is output to the numerical output (IODATA). After the start of inputting this signal to the robot, wait at least 30 ms before reading the numerical output (IODATA) signal.	Edge	10025,	-1,
	Output	Error No. output signal	The "error number output in progress" status is output to the numerical output.		-1	-1
JOGENA (Operation right required)	Input	Jog valid input signal	Jogs the designated axis in the designated mode. Operation takes place while this signal is ON.	Level	-1,	-1,
	Output	Jog valid output signal	Outputs that the jog operation is entered.		-1	-1
JOGM	Input	Jog mode input (start No., end No.)	Designates the jog mode. 0/1/2/3/4 = Joint, XYZ, cylindrical, 3-axis XYZ, tool	Level	Note3) -1(Start bit), -1(End bit),	Note3) -1(Start bit), -1(End bit),
	Output	Jog mode output (start No., end No.)	Outputs the current jog mode.		-1(Start bit), -1(End bit)	-1(Start bit), -1(End bit)
JOG+	Input	Jog feed plus side for 8-axes (start No., end No.)	Designates the jog operation axis. JOINT jog mode: J1, J2, J3, J4, J5, J6, J7 and J8 axes from the start number. XYZ jog mode: X, Y, Z, A, B, C, L1 and L2 axes from the start number. CYLINDER jog mode: X, Éy, Z, A, B, C, L1 and L2 axes from the start number. 3-axis XYZ jog mode: X, Y, Z, J4, J5 and J6 axes from the start number. Tool jog mode: X, Y, Z, A, B and C axes from the start number.	jiku	Note4) -1, -1	Note4) -1, -1
	Output	-	-			
JOG-	Input	Jog feed minus side for 8-axes (start No., end No.)	Designates the jog operation axis. JOINT jog mode: J1, J2, J3, J4, J5, J6, J7 and J8 axes from the start number. XYZ jog mode: X, Y, Z, A, B, C, L1 and L2 axes from the start number. CYLINDER jog mode: X, Éy, Z, A, B, C, L1 and L2 axes from the start number. 3-axis XYZ jog mode: X, Y, Z, J4, J5 and J6 axes from the start number. Tool jog mode: X, Y, Z, A, B and C axes from the start number.	Level	Note4) -1, -1	Note4) -1, -1
	Output	-	-			

Parameter name	Class	Name	Function	Signal level Note1)	Factory shipment signal number. Input, output	
					CRnQ	CRnD
JOGNER (Operation right required)	Input	Errors during jog operation Temporarily ignoring input signal	Temporarily ignores errors that cannot be reset during jog operation.	Level	-1,	-1,
	Output	Errors during jog operation Temporary ignoring output signal	Outputs that the error is being ignored temporarily. * This signal is applicable to only machine 1.		-1	-1
HNDCNTLn (n=1 to 3)	Input	-	-			
	Output	Mechanism n hand output signal state (start No., end No.)	Outputs the hand output(n=1) 900 to 907 state. Outputs the hand output(n=2) 910 to 917 state. Outputs the hand output(n=3) 920 to 927 state. Example) To output the four points from 900 through 903 to general-purpose output signals 3, 4, 5 and 6, set the HNDCNTL1 to (3, 6).		HNDCNTL1 10048(Start bit), 10055(End bit)	-1(Start bit), -1(End bit)
HNDstSn (n=1 to 3)	Input	-	-			
	Output	Mechanism n hand input signal state (start No., end No.)	Outputs the hand input(n=1) 900 to 907 state. Outputs the hand input(n=2) 910 to 917 state. Outputs the hand input(n=3) 920 to 927 state. Example) To output the four points from 900 through 903 to general-purpose output signals 3, 4, 5 and 6, set the HNDCNTL1 to (3, 6).		HNDSTS1 10056(Start bit), 10063(End bit)	-1(Start bit), -1(End bit)
HNDERRn (n=1 to 3)	Input	Mechanism n hand error input signal	Requests the hand error occurrence. A LOW level error (error number 30) will be generated.	Level	-1,	-1,
	Output	Mechanism n hand error output signal	Outputs that a hand error is occurring.		-1	-1
AIRERRn (n=1 to 5)	Input	Mechanism n pneumatic pressure error input signal	Request the pneumatic pressure error occurrence. A LOW level error (error number 31) will be generated.	Level	-1,	-1,
	Output	Mechanism n pneumatic error output signal	Outputs that a pneumatic pressure error is occurring.		-1	-1
USRAREA Refer to Page 372, "5.8 About user-defined area"	Input	-	-		Note5)	Note5)
	Output	User-designated area 8-points (start No., end No.)	Outputs that the robot is in the user-designated area. The output is made sequentially for areas 1, 2 and 3, as designed from the one closest to the start number. The area is set with parameters AREA1P1, AREA1P2 to AREA8P1 and AREA8P2. Setting example) When USRAREA is used as an example: If only area 1 is used, USRAREA: 8, 8 Setting valid If only area 1,2 is used, USRAREA: 8, 9 Setting valid USRAREA:-1,-1 to Setting invalid USRAREA: 8,-1 to Setting invalid(No Error) USRAREA:-1,8 to Setting invalid(No Error) USRAREA:9,8 to Setting invalid(Error L6643)		10064(Start bit), 10071(End bit)	-1(Start bit), -1(End bit)
MnPTEXC (n=1 to 3)	Input	-	-		-1(No meaning),	-1,(No meaning)
	Output	Warning for maintenance parts replacement time	This output notifies that the replacement time of maintenance parts has been reached.	Level	-1	-1
MnWUPENA (n=1 to 3) (Operation right required)	Input	Mechanism n warm-up operation mode enable input signal	Enables the warm-up operation mode of each mechanism. (n=1 to 3) Note: To switch the warm-up operation mode from enable to disable or vice versa using this input signal, it is necessary to enable the warm-up operation mode with the WUPENA parameter, etc. If the warm-up operation mode has been disabled with a parameter, inputting this input signal will not enable the mode.	Level	-1,	-1,
	Output	Mechanism n warm-up operation mode output signal	Outputs that the warm-up operation mode is currently enabled. (n=1 to 3)		-1	-1

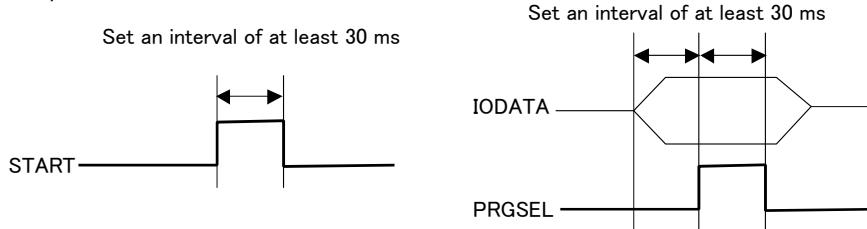
Parameter name	Class	Name	Function	Signal level Note1)	Factory shipment signal number. Input, output	
					CRnQ	CRnD
MnWUPMD (n=1 to 3)	Input	-	-		-1(No meaning),	-1(No meaning),
	Output	Mechanism n warm-up operation status output signal	Outputs that the status is the warm-up operation status, and thus the robot will operate at a reduced speed. (n=1 to 3)		-1	-1

Note 1) The meanings of the signal level are explained below.

Level: The designated function is validated when the signal is ON, and the function is invalidated when the signal is OFF. Make sure the signal is turned ON for at least 30 ms.

Edge: The designated function is validated when the signal changes from the OFF to ON state, and the function maintains the original state even when the signal returns to the OFF state. .

Example)



Note 2) Set in the order of input start No., input end No., output start No. and output end No.

When using as the input or output of an actual value, use from the start No. to the end No., and express as a binary. The start No. indicates the low-order bit, and the end No. indicates the high-order bit. Set only the numbers required to express the value.

For example, when using for program selection and only programs 1 to 6 are available, the expression can be created by setting 3 bits. Up to 16 bits can be set.

Assignment examples are shown below.

Example) To set the start input signal in general-purpose input 10016, and the operating output signal in general-purpose output 10026.

Parameter START = {10016, 10026}

Example) When setting 4 bits of numerical input to general-purpose inputs 10027 to 10030, and 5 bits of numerical output to general-purpose outputs 10027 to 10031.

Parameter IODATA = {10027, 10030, 10027, 10031}

Note 3) Set in the order of input start No., input end No., output start No. and output end No.

When using as the actual jog mode, use from the start No. to the end No., and express as a binary. The start No. indicates the low-order bit, and the end No. indicates the high-order bit. Set only the numbers required to express the value.

For example, when using only the joint mode and XYZ mode, the expression can be created by setting 2 bits. Up to 3 bits can be set.

Note 4) They are in the order of an input starting number and then an input end number. Specify the J1/X axis for the input starting number and the J8/L2 axis for the input end number at its maximum.

For example, when using a 6-axis robot, only 6 bits need to be set.

Even if using a 4-axis robot, when using the XYZ mode, the C axis is required, so 6 bits must be set. Up to 8 bits can be set.

Note 5) Set in the order of output start No. and output end No. The start number specifies area 1, while the end number specifies area 32 in the largest configuration.

For example, setting 2 bits will suffice if only two areas are used. A maximum of 32 bits can be set.

6.4 Enable/disable status of signals

Note that depending on the input signal type, the function may not occur even if the target signal is input depending on the robot state at that time, such as during operation or when stop is input.

The relation of the robot status to the input signal validity is shown below.

Table 6-8:Validity state of dedicated input signals

Parameter name	Name	Validity of symbol on left according to robot states.
SLOTINIT	Program reset	These do not function in the operation state (when START output is ON).
SAFEPOS	Safe point return input	
OUTRESET	General-purpose output signal reset	
MnWUPENA	Mechanism n warm-up operation mode enable input	
START SnSTART (n=1 to 32)	Start input	These function only when the external input/output has the operation rights (when IOENA output is ON).
SLOTINIT	Program reset	
SRVON MnSRVON (n=1 to 3)	Servo ON input	
MELOCK MnMELOCK (n=1 to 3)	Machine lock input	
SAFEPOS	Safe point return input	
PRGSEL	Program selection input	
OvrdSEL	Override selection input	
JOGENA	Jog enable input	
MnWUPENA	Mechanism n warm-up operation mode enable input	
START	Start input	These do not function in the stop input state (when STOPSTS is ON).
SLOTINIT	Program reset	
SAFEPOS	Safe point return input	
JOGENA	Jog enable input	This does not function in the servo OFF input state.
SRVON	Servo ON input	
MELOCK	Machine lock input	
PRGSEL	Program selection input	The signal does not function during pause status (STOP output is on).

6.5 External signal timing chart

6.5.1 Individual timing chart of each signal

(1) RCREADY (Controller's power ON completion output)

<Output>
Power ON (RCREADY) | _____ (Indicates the status in which the controller can receive signals.)

(2) ATEXTMD (Remote mode output)

<Output>
Remote mode output (ATEXTMD) | _____ (Indicates when the key switch on the operation panel is "Auto (Ext)")

(3) TEACHMD (Teach mode output)

<Output>
Teach mode output (TEACHMD) | _____ (Indicates when the key switch on the operation panel is "TEACH.")

(4) ATTOPMD (Auto mode output)

<Output>
Auto mode output (ATTOPMD) | _____ (Indicates when the key switch on the operation panel is "Auto (Op.)")

(5) IOENA (Operation right input signal/operation right output signal)

<Input>
Operation right input (IOENA) | _____
<Output>
Operation right output (IOENA) | _____

(6) START (Start input/operating output)

<Input>
Start input (START) | _____
<Output>
Operating output (START) | _____

When the STOP signal, or the emergency stop or other signal was input, or after the completion of the CYCLE signal

(7) STOP (Stop input/aborting output)

<Input>
Stop input (STOP) | _____
<Output>
Aborting output (STOP) | _____

When the START, SnSTART or SLOTINIT signal was input

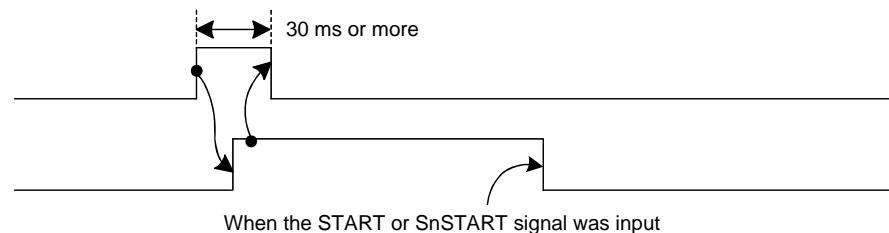
(8) STOPSTS (Output during stop signal input)

<Output>
During stop signal input (STOPSTS) | _____ (Indicates that the STOP is being input.)

(9) SLOTINIT (Program reset input/program selectable output)

<Input>
Program reset (SLOTINIT)

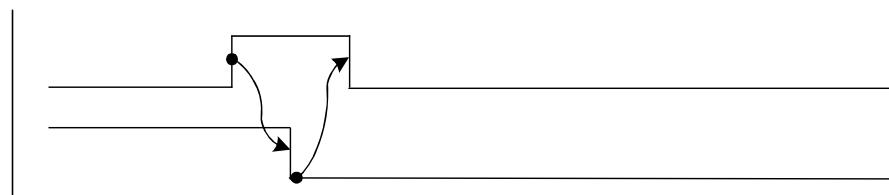
<Output>
Program selectable output
(SLOTINIT)



(10) ERRRESET (Error reset input/output during error occurrence)

<Input>
Error reset input (ERRRESET)

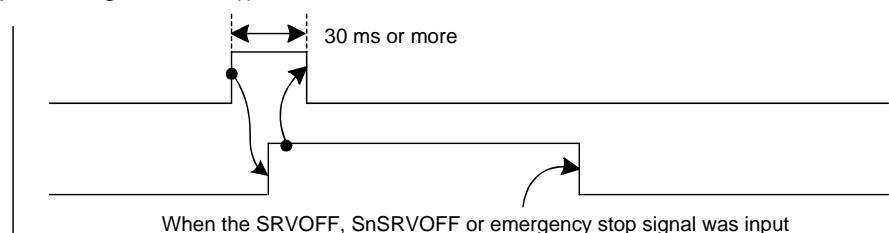
<Output>
Output during error occurrence
(ERRRESET)



(11) SRVON (Servo ON input/output during servo ON))

<Input>
Servo ON input (SRVON)

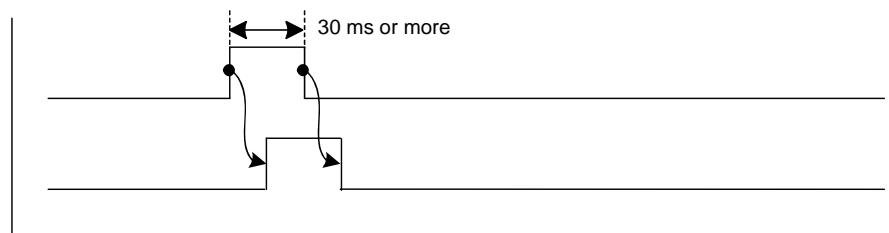
<Output>
Output during servo ON (SRVON)



(12) SRVOFF (Servo OFF input/servo ON disable output)

<Input>
Servo OFF input (SRVOFF)

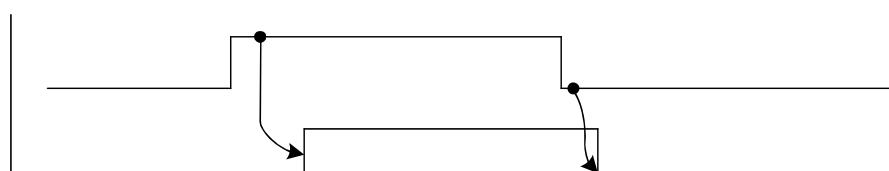
<Output>
Servo ON disable output (SRVOFF)



(13) AUTOENA (Auto operation input/auto operation enable output)

<Input>
Auto operation enable input
(AUTOENA)

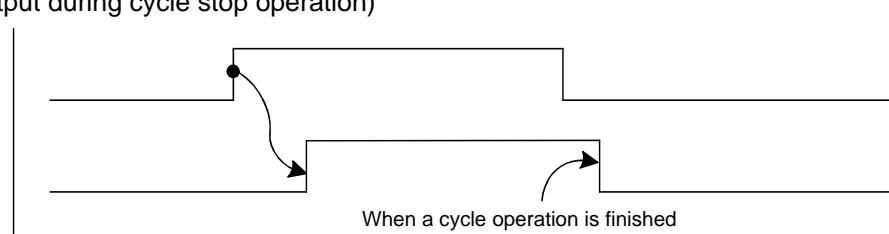
<Output>
Auto operation enable output
(AUTOENA)



(14) CYCLE (Cycle stop input/output during cycle stop operation)

<Input>
Cycle stop input (CYCLE)

<Output>
Output during cycle stop operation
(CYCLE)



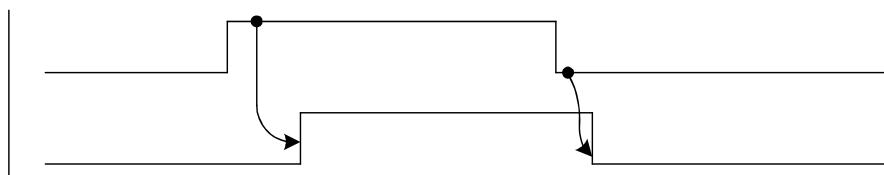
(15) MELOCK (Machine lock input/output during machine lock)

<Input>

Machine lock input (MELOCK)

<Output>

Output during machine lock
(MELOCK)



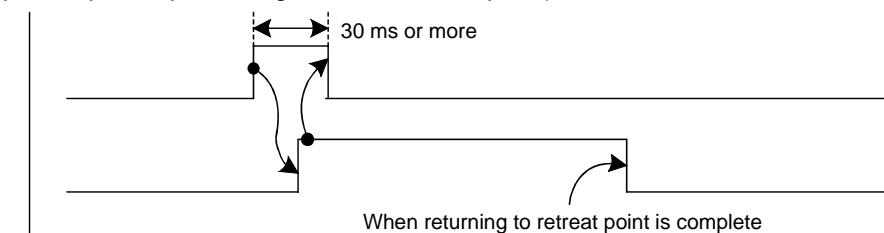
(16) SAFEPOS (Return to retreat point input/output during return to retreat point)

<Input>

Return to retreat point input
(SAFEPOS)

<Output>

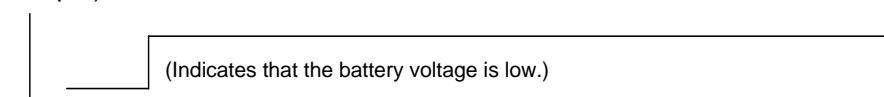
Output during return to retreat point
(SAFEPOS)



(17) BATERR (Low battery voltage output)

<Output>

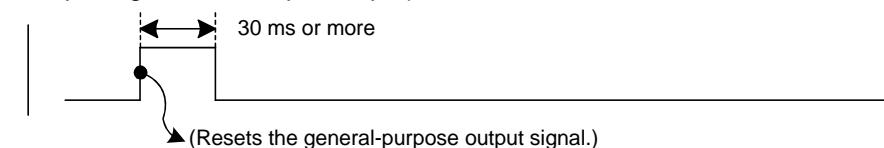
Low battery voltage (BATERR)



(18) OUTRESET (General-purpose output signal reset request input)

<Input>

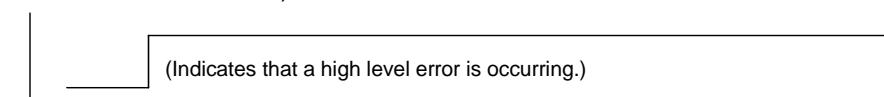
General-purpose output signal reset
(OUTRESET)



(19) HVLERR (Output during high level error occurrence)

<Output>

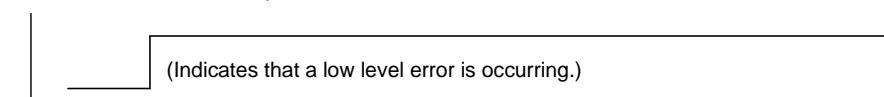
High level error output (HVLERR)



(20) LLVLER (Output during low level error occurrence)

<Output>

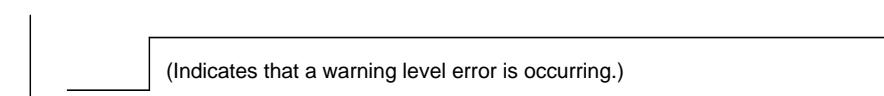
Low level error output (LLVLER)



(21) CLVLER (Output during warning level error occurrence)

<Output>

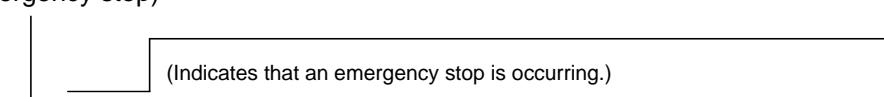
Warning level error output
(CLVLER)



(22) EMGERR (Output during emergency stop)

<Output>

Emergency stop output (EMGERR)



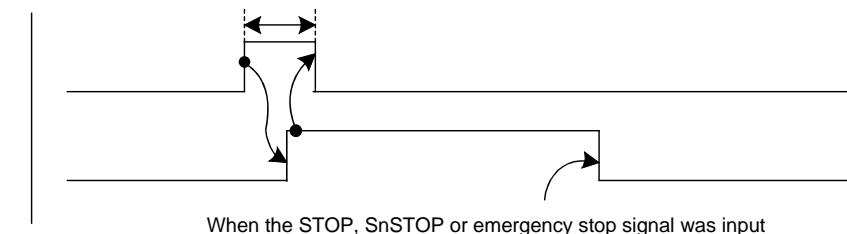
(23) SnSTART (Slot n start input/output during slot n operation)

<Input>

Slot n start input (SnSTART)

<Output>

Output during slot n operation
(SnSTART)

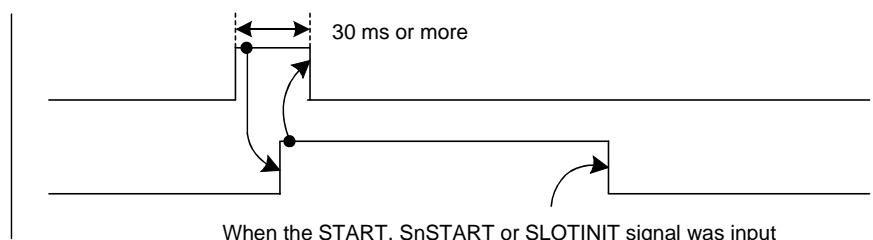


(24) SnSTOP (Slot n stop input/output during slot n aborting)

<Input>

Slot n stop input (SnSTOP)

<Output>

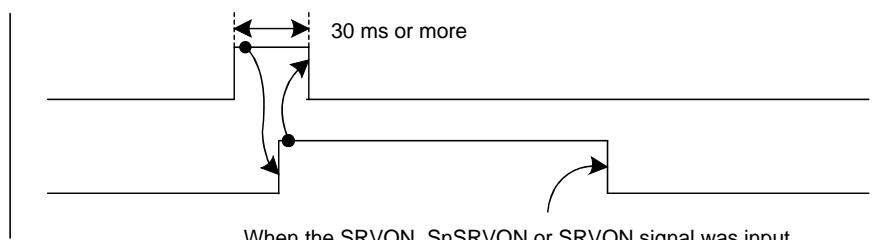
Output during slot n aborting
(SnSTOP)

(25) MnSRVOFF (Mechanical n servo OFF input/mechanical n servo ON disable output)

<Input>

Mechanical n servo OFF input
(MnSRVOFF)

<Output>

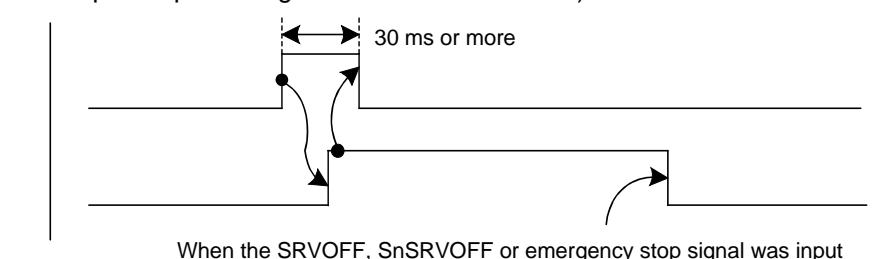
Mechanical n servo ON disable output
(MnSRVOFF)

(26) MnSRVON (Mechanical n servo ON input/output during mechanical n servo ON)

<Input>

Mechanical n servo ON input
(MnSRVON)

<Output>

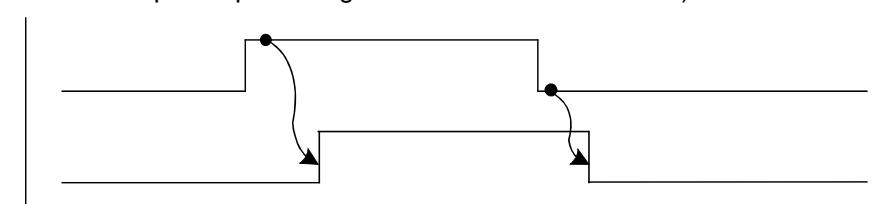
Output during mechanical n servo ON
(MnSRVON)

(27) MnMELOCK (Mechanical n machine lock input/output during mechanical n machine lock)

<Input>

Mechanical n machine lock input
(MnMELOCK)

<Output>

Output during mechanical n machine
lock (MnMELOCK)

(28) PRGSEL (Program selection input)

* This is used together with the numeric value input (IODATA).

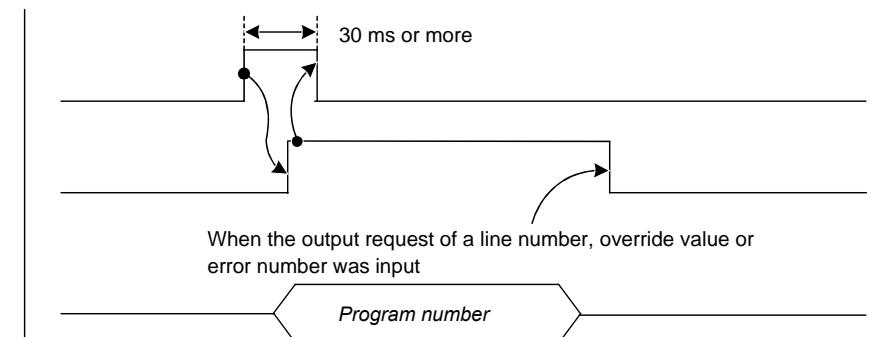
<Input>

Program number output request
(PRGOUT)

<Output>

Outputting program number (PRGOUT)

Numeric value output (IODATA)



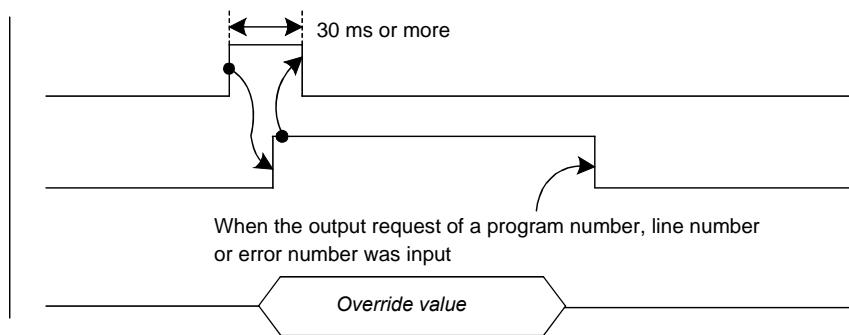
(29) OvrdSEL (Override selection input)

* This is used together with the numeric value input (IODATA).

<Input>
Override value output request
(OVRDOUT)

<Output>
Override value output request
(OVRDOUT)

Numeric value output (IODATA)



(30) IODATA (Numeric value input/numeric value output)

* This is used together with PRGSEL, OvrdSEL, PRGOUT, LINEOUT, OvrdOUT or ERROUT.

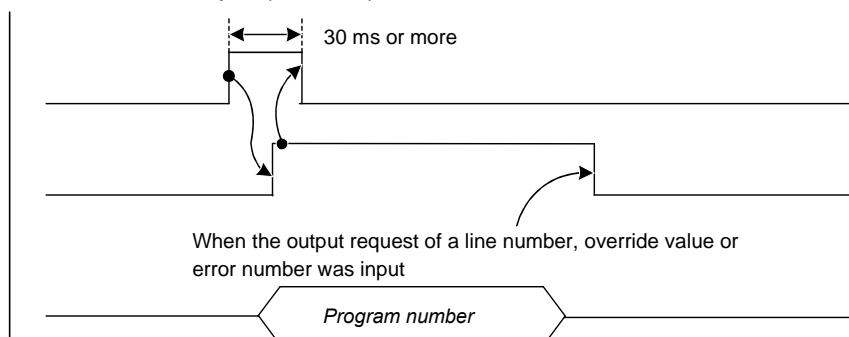
(31) PRGOUT (Program number output request input/outputting program number)

* This is used together with the numeric value output (IODATA).

<Input>
Program number output request
(PRGOUT)

<Output>
Outputting program number (PRGOUT)

Numeric value output (IODATA)



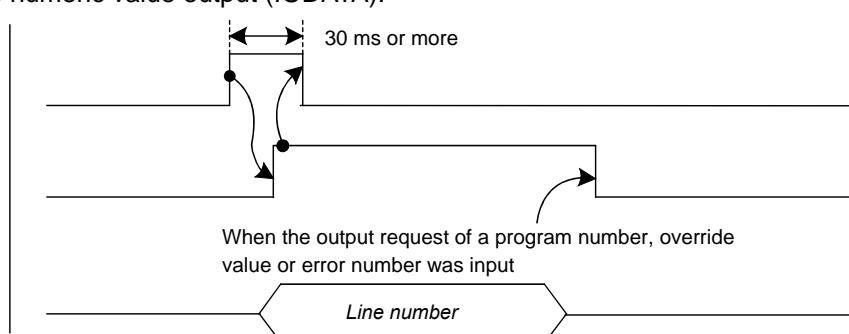
(32) LINEOUT (Line number output request input/outputting line number)

* This is used together with the numeric value output (IODATA).

<Input>
Line number output request (LINEOUT)

<Output>
Outputting line number (LINEOUT)

Numeric value output (IODATA)



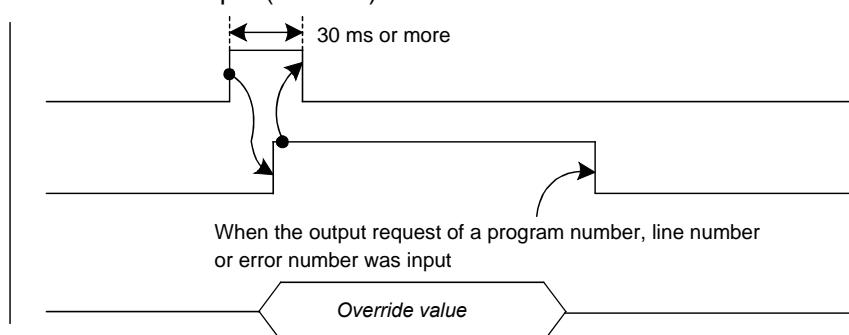
(33) OvrdOUT (Override value output request/outputting override value)

* This is used together with the numeric value output (IODATA).

<Input>
Override value output request
(OVRDOUT)

<Output>
Override value output request
(OVRDOUT)

Numeric value output (IODATA)

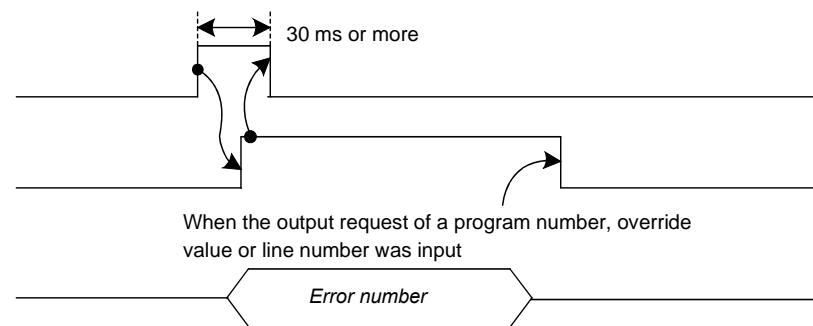


(34) ERROUT (Error number output request/outputting error number)

* This is used together with the numeric value input (IODATA).

<Input>
Error number output request (ERROUT)

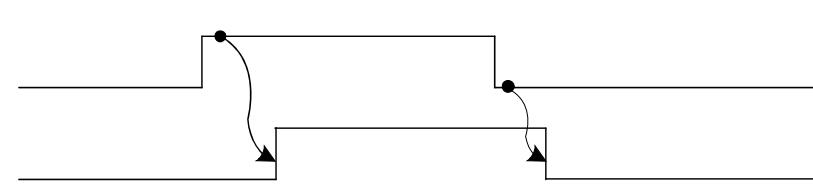
<Output>
Outputting error number (ERROUT)
Numeric value output (IODATA)



(35) JOGENA (Jog enable input/output during jog enabled)

<Input>
Jog enable input (JOGENA)

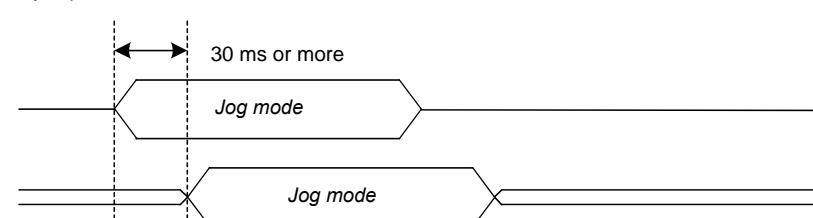
<Output>
Output during jog enabled (JOGENA)



(36) JOGM (Jog mode input/jog mode output)

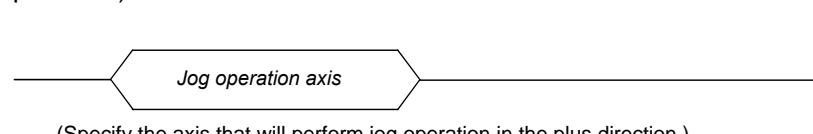
<Input>
Jog mode input (JOGM)

<Output>
Jog mode output (JOGM)



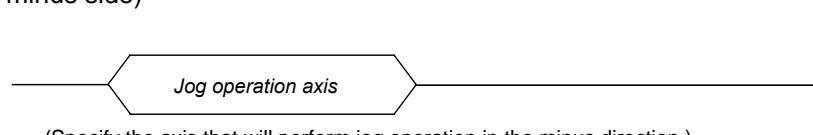
(37) JOG+ (Input for 8 axes on jog feed plus side)

<Input>
8 axes on jog feed plus side (JOG+)



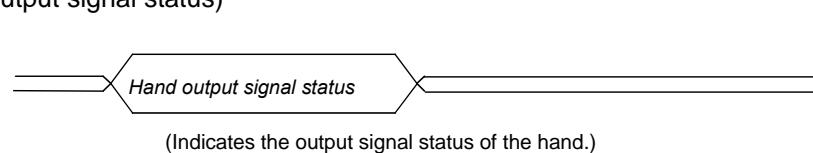
(38) JOG- (Input for 8 axes on jog feed minus side)

<Input>
8 axes on jog feed minus side (JOG-)



(39) HNDCTNLn (Mechanical n hand output signal status)

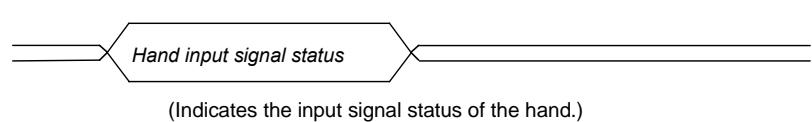
<Output>
Mechanical n hand output signal status
(HNDCTNLn)



(40) HNDstSn (Mechanical n hand input signal status)

<Output>

Mechanical n hand input signal status
(HNDSTS_n)



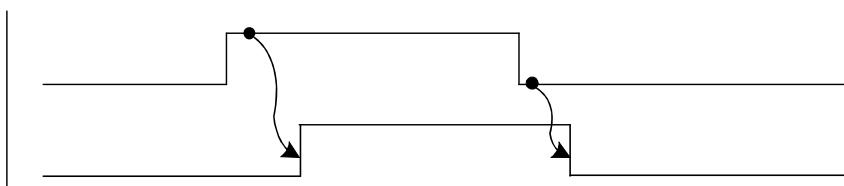
(41) HNDERRn (Mechanical n hand error input signal/output during mechanical n hand error occurrence)

<Input>

Mechanical n hand error input
(HNDERR_n)

<Output>

Output during mechanical n hand error
occurrence (HNDERR_n)



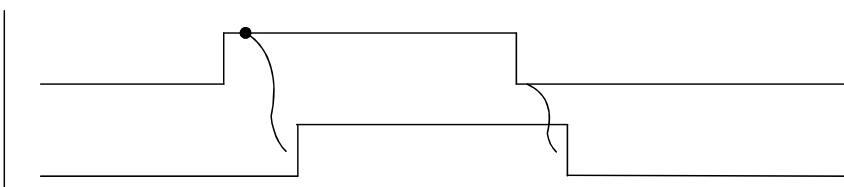
(42) AIRERRn (Mechanical n pneumatic error input signal/outputting mechanical n pneumatic error)

<Input>

Mechanical n pneumatic error input
(AIRERR_n)

<Output>

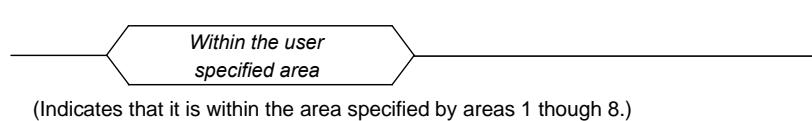
Outputting mechanical n pneumatic
error (AIRERR_n)



(43) USRAREA (User-specified area 8 points output)

<Output>

User-specified area 8 points
(USRAREA)



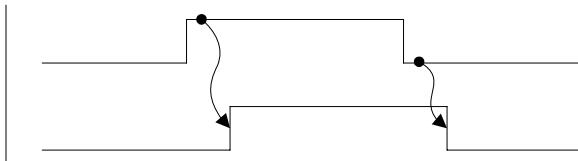
(44) MnWUPENA (Mechanism n warm-up operation mode enable input signal/ Mechanism n warm-up operation mode output signal)

<Input>

Mechanism n warm-up operation mode enable
input signal (MnWUPENA)

<Output>

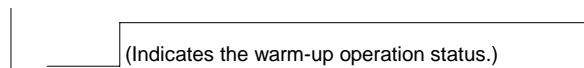
Mechanism n warm-up operation mode
output signal (MnWUPENA)



(45) MnWUPMD (Mechanism n warm-up operation status output signal)

<Output>

Mechanism n warm-up operation status output
signal (MnWUPMD)



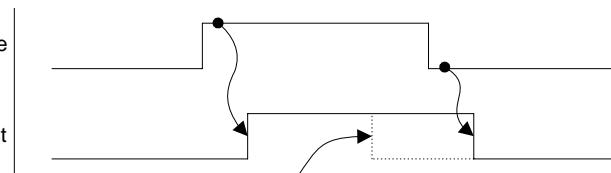
* If the mechanism n warm-up operation status output (MnWUPMD) is assigned together with the mechanism n warm-up operation mode enable input (MnWUPENA), the timing chart is as shown below.

<Input>

Mechanism n warm-up operation mode enable
input signal (MnWUPENA)

<Output>

Mechanism n warm-up operation status output
signal (MnWUPMD)



When the warm-up operation status is canceled while
the warm-up operation mode is enabled

6.5.2 Timing chart example

(1) External signal operation timing chart (Part 1)

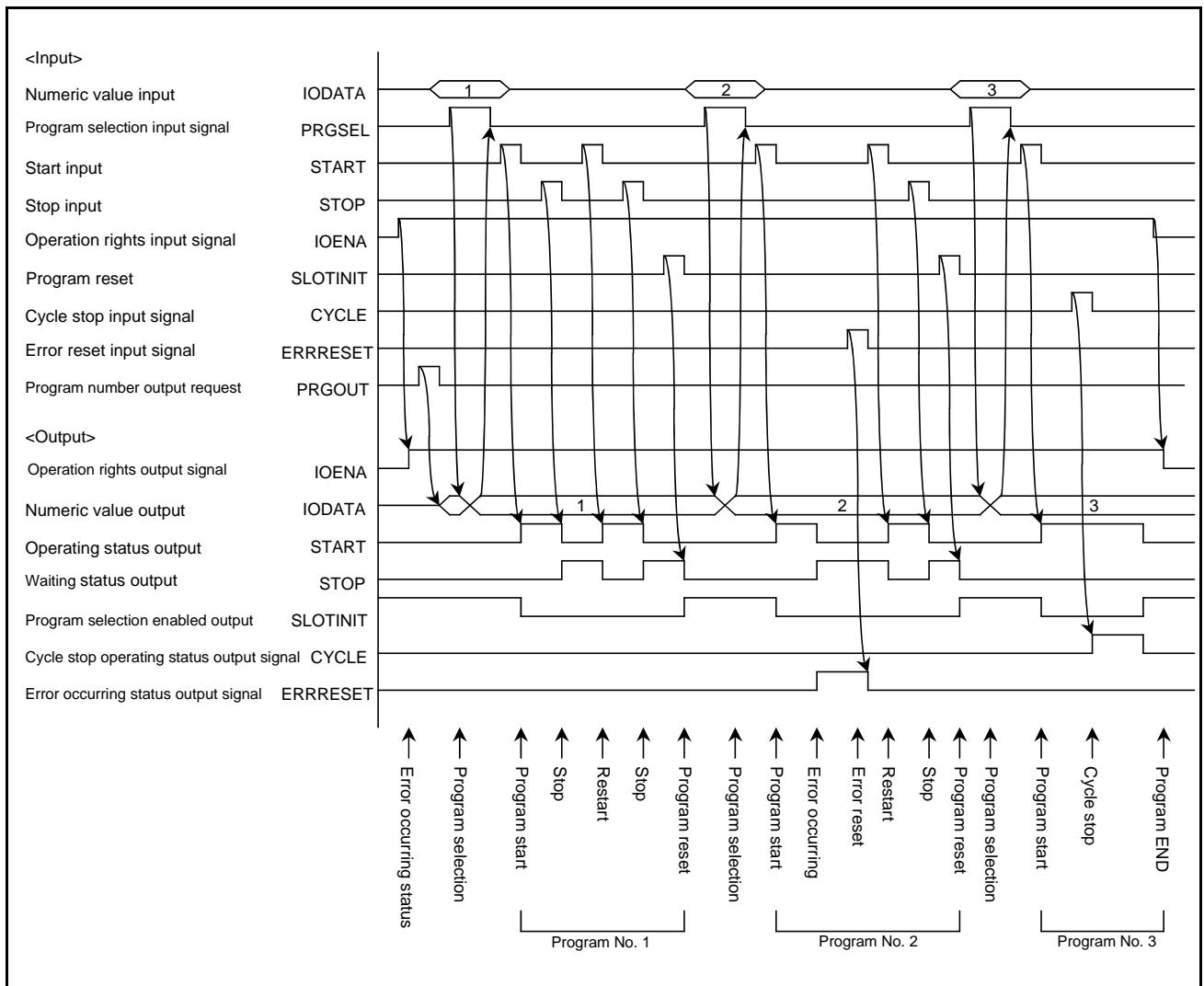


Fig.6-2:Example of external operation timing chart (Part 1)

(2) External signal operation timing chart (Part 2)

An example of timing chart the servo ON/OFF, selecting the program, selecting the override, starting and outputting the line No., etc., with external signals is shown in [Fig. 6-3](#).

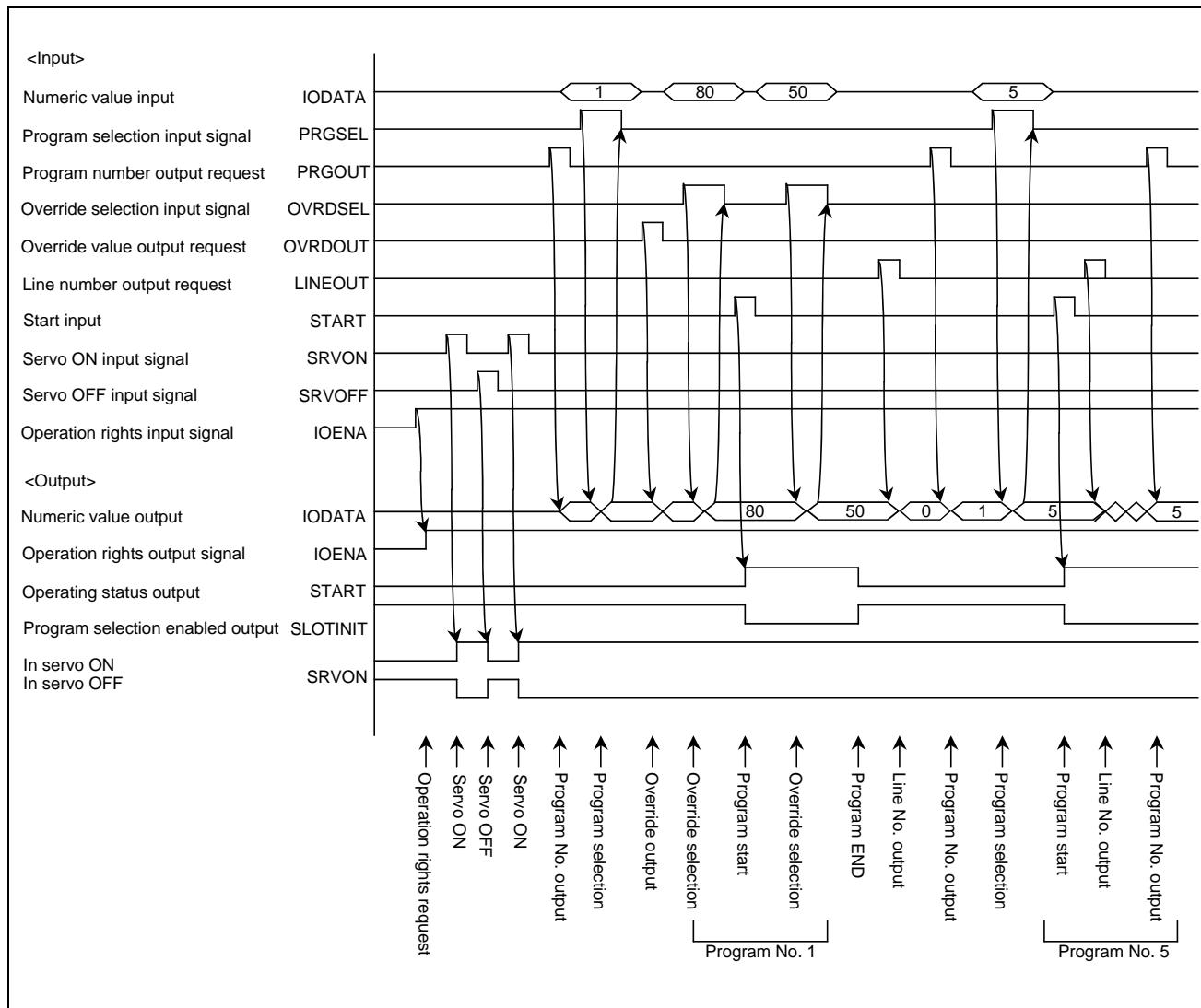


Fig.6-3:Example of external operation timing chart (Part 2)

(3) Example of external operation timing chart (Part 3)

An example of the timing chart for error reset, general-purpose output reset and program reset, etc., with external signals is shown output in [Fig. 6-4](#).

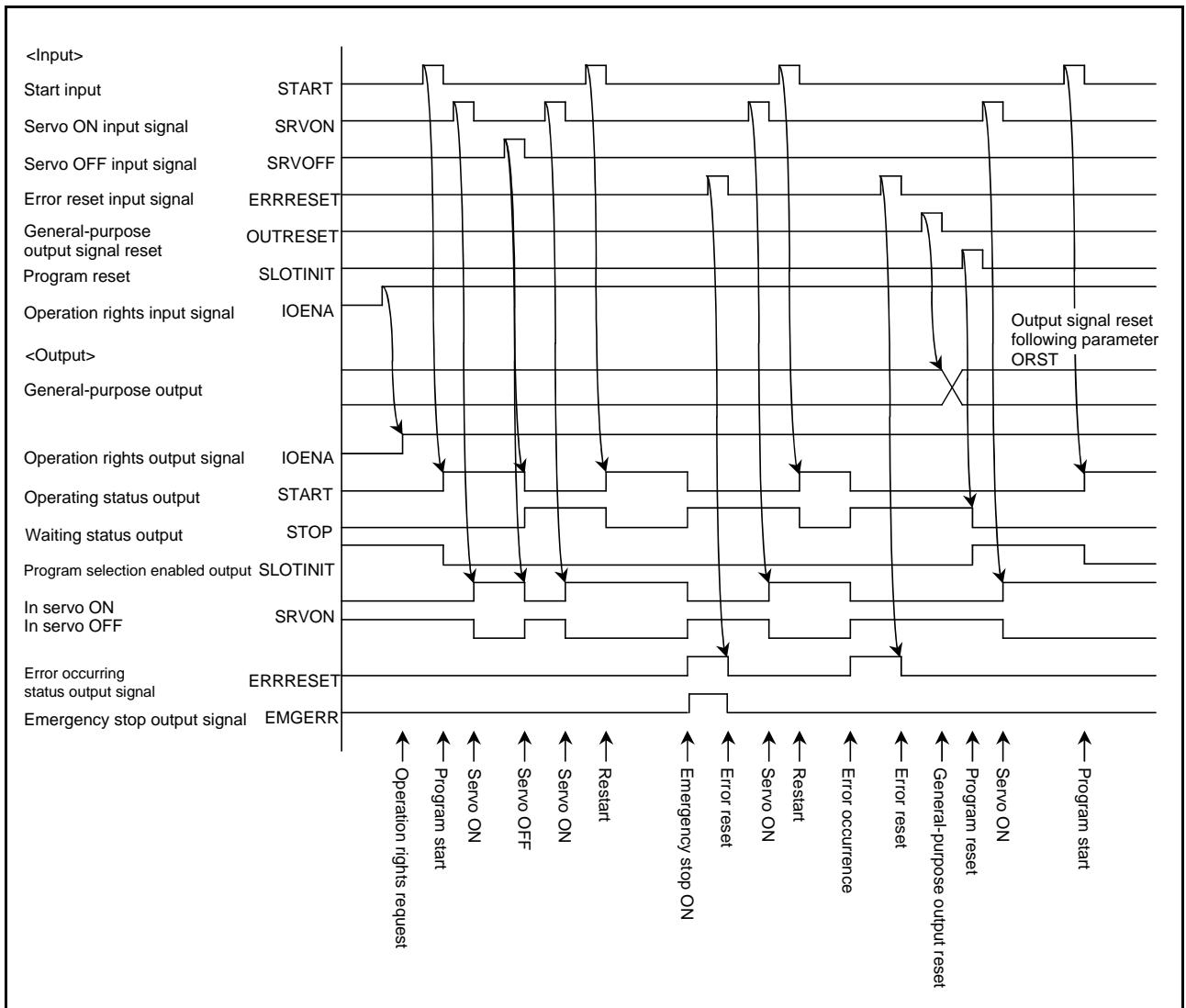


Fig.6-4:Example of external operation timing chart (Part 3)

(4) Example of external operation timing chart (Part 4)

An example of the timing chart for jog operation, safe point return and program reset, etc., with external signals is shown in [Fig. 6-5](#).

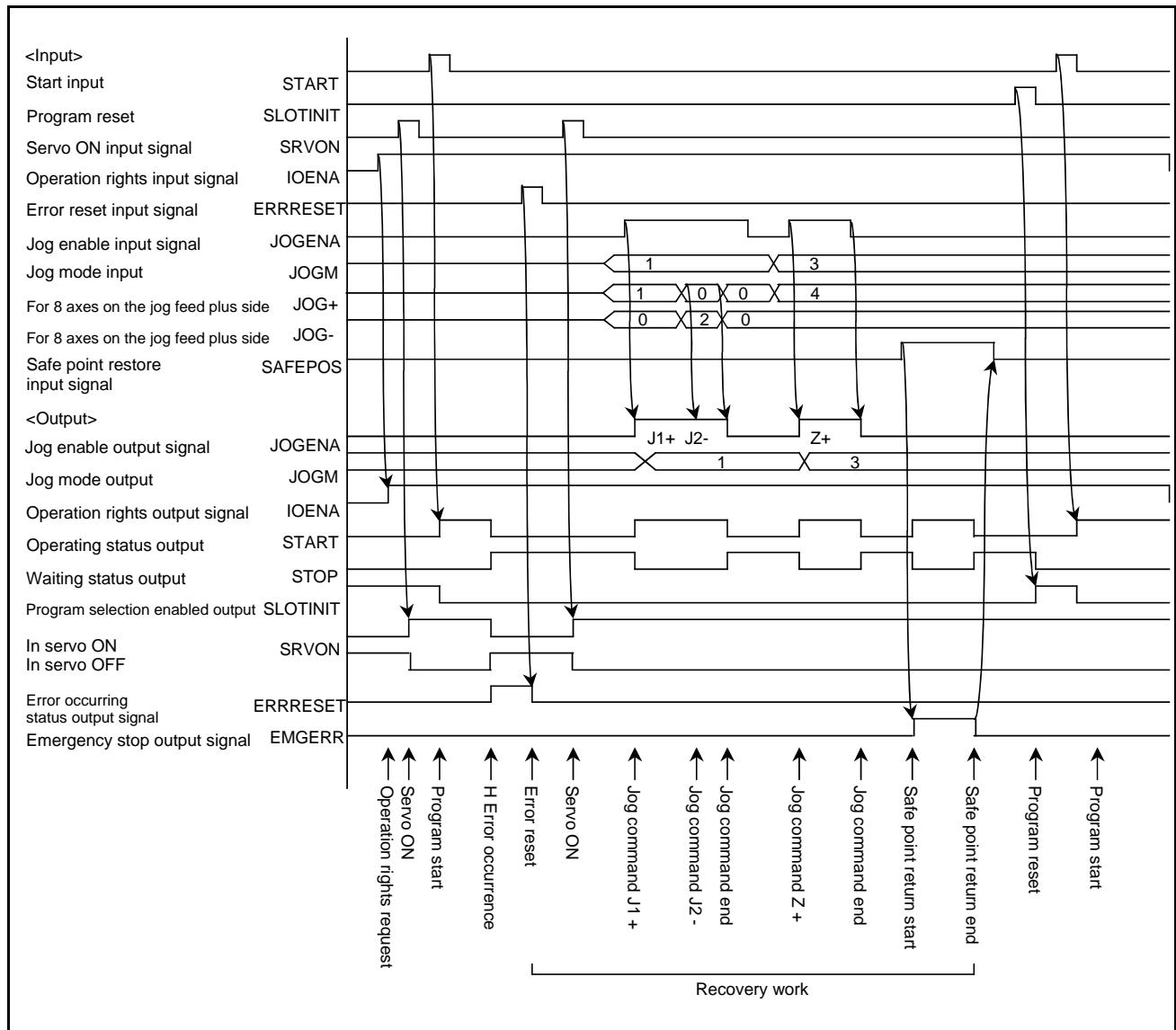


Fig.6-5:Example of external operation timing chart (Part 4)

6.6 Emergency stop input

For wiring and other aspects of the emergency stop input, refer to the separate document entitled "Controller setup, basic operation, and maintenance."

6.6.1 Robot Behavior upon Emergency Stop Input

When an emergency stop signal is input while the robot is operating, the servo power supply is cut off by means of hardware control. The robot's tip path and stopping position after the input of an emergency stop signal cannot be specified. An overrun may occur depending on the robot speed or load condition of the tool.

6.7 Display unit (GOT1000 Series) connection (reference)

By directly connecting the GOT1000 Series (GT15) display unit and CRnD-700 controller with an Ethernet cable, I/O control (256 inputs, 256 outputs) from the GOT to the robot controller is possible.

Refer to the respective operation manuals for details on how to use the GOT1000 Series and GT Designer2 image creation software.

Please note that the CRnQ-700 controller is not supported. Control via the sequencer CPU and multi CPU shared memory.

(1) Usage example

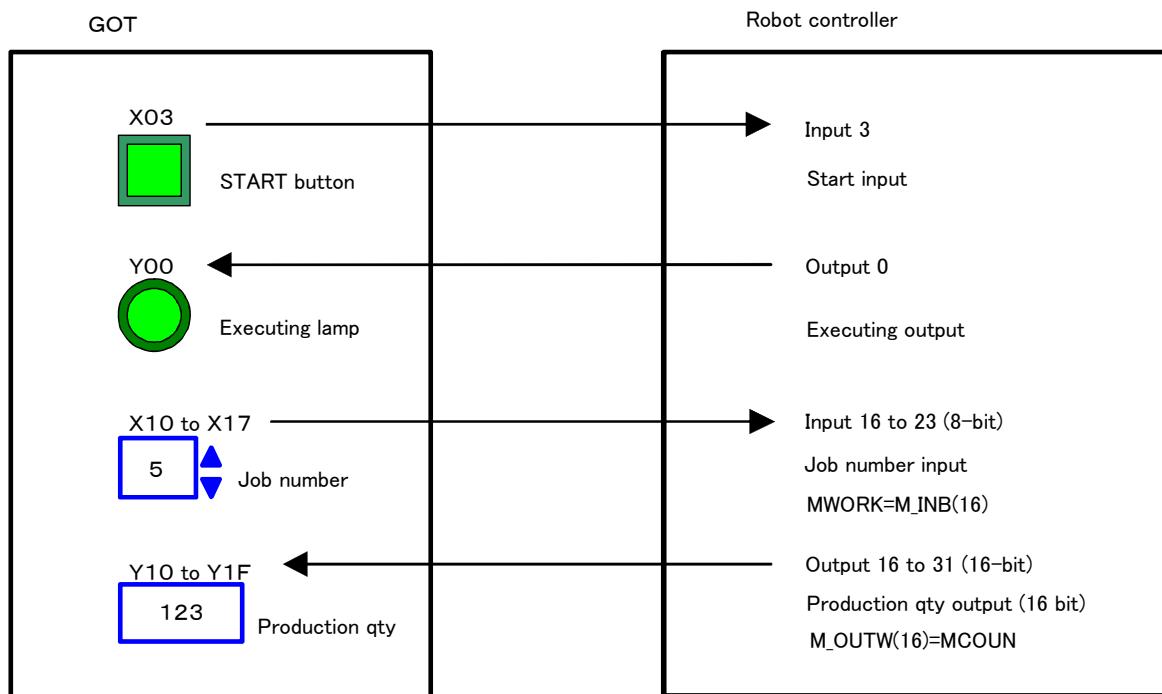


Fig.6-6:GOT usage example

(2) Specifications

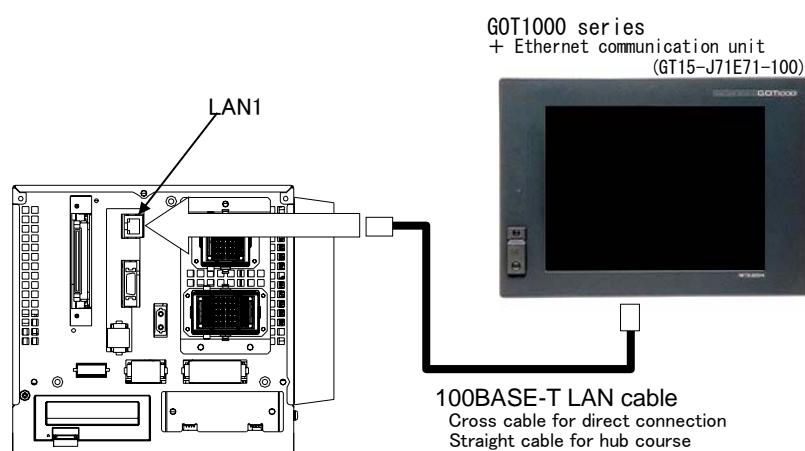
Table 6-9:GOT usage example specifications

Item	Item	Item
No. of I/Os	256 inputs, 256 outputs	The device names as viewed from GOT are X00 to XFF and Y00 to YFF. (Refer to "Robot I/O and GOT assignment" described later.)
GOT connection qty	1	Multiple GOT units cannot be connected to a single robot controller, however, multiple robot controllers can be connected to a single GOT unit.
Connection method	Ethernet only	UPD communication GOT optional Ethernet communication unit (GT15-J71E71-100) is required.
Applicable GOT	GOT1000 Series GT15 (full spec model)	Models other than GT15 (full spec model) do not support Ethernet and thus cannot be connected. The Transparent function using computer support software (RT ToolBox) is not supported. (Current as of December, 2007)
Image creation software	GT Designer2 Version 2.72A or later	Model: SW2D5C-GTD2-J (Japanese version) Support for the English version is scheduled from April, 2008. Upgrade to the latest version from the MELFANS Web site.

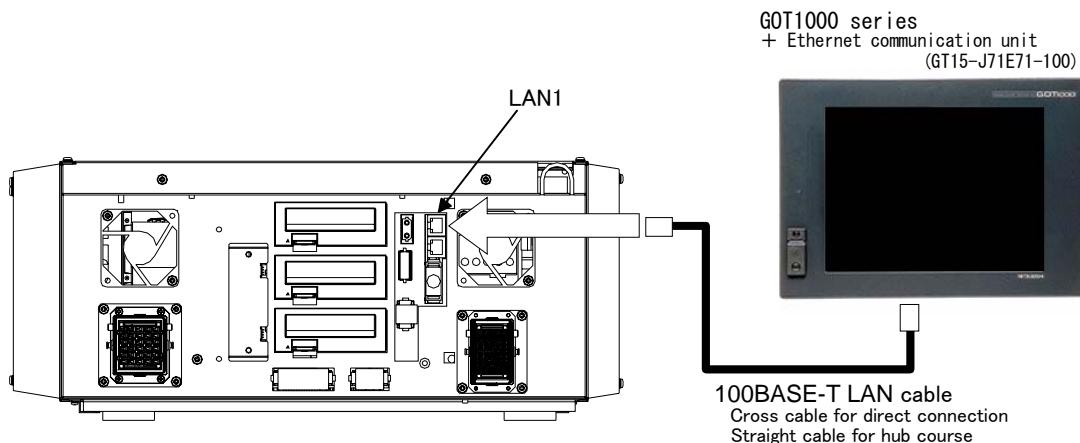
(3) Connection

Connect GOT by the Ethernet cable

CR1n-700 series



CR2n-700 series



CR3n-700 series

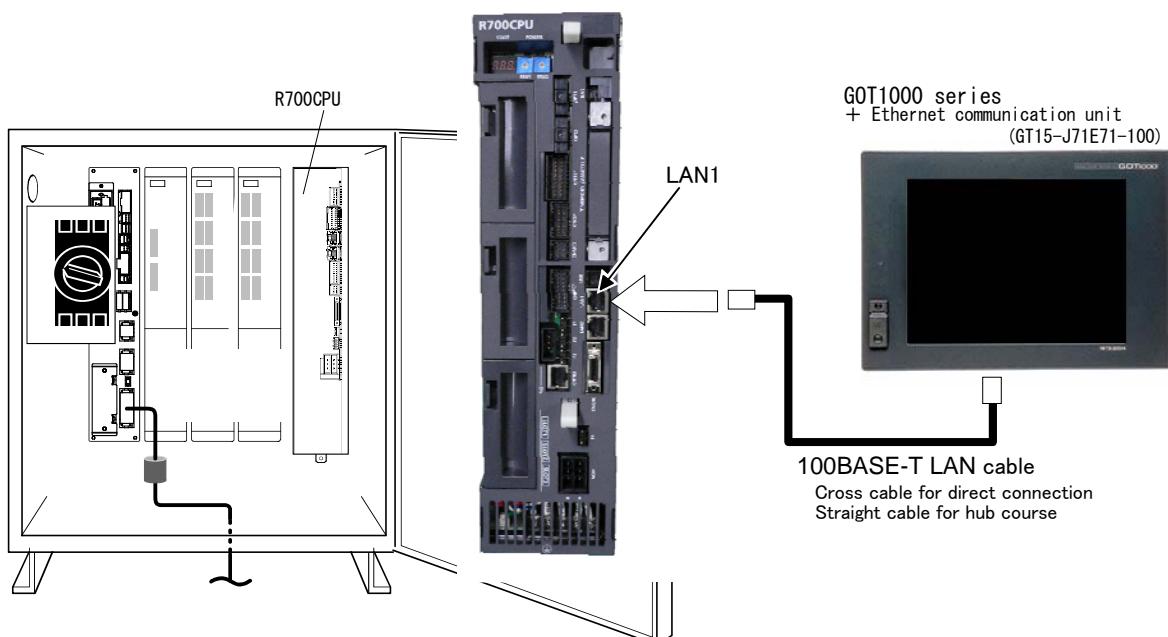


Fig.6-7:Connection of GOT

(4) Settings

1) Creating new projects with image creation software

Start up the GT Designer2 image creation software at the computer. Select [Project] ? [New] from the menu and perform settings in accordance with the messages displayed at the new project creation wizard.

Set “MELSEC-QnU, Q17nD/NC/DR, CRnD-700” for “Connection device setting”, “Expansion I/F-1 (first level)” for “Connection I/F”, and “QJ71E71/AJ71(Q)E71, Q172nNC, CRnD-700” for “Communication driver”.

2) Performing Ethernet settings with the image creation software

Perform Ethernet settings at [Common settings] ? [Ethernet] in the GT Designer2 workspace window.

Set “?” for “Own PC”, “1” for “N/W No.”, “2” for “PC No.”, “CRnD-700” for “Model”, and “192.168.0.20” for “IP address”.

If the robot controller parameters (IP address and GOT port no.) have been changed from the factory settings, check the NETIP and GOTPORT robot controller parameters, and ensure that these settings match.

3) Creating screens with the image creation software

Create screens with GT Designer2.

◆◆◆ Network Settings ◆◆◆

Set the settings made at “[2\)Performing Ethernet settings with the image creation software](#)” for the device network settings.

If connecting a single robot controller to a single GOT, set the network setting for the device setting to “Own PC”.

If connecting multiple robot controllers to a single GOT, set the network setting for the device setting to “Other PC”, and set the network number (N/W No.) and computer number (PC No.). Please note that the CPU number is not used and may be left as 0.

4) Checking importing of image creation data to GOT

Connect the computer and GOT with a communication cable (USB), set “Communication setting” to “USB” at [Communication] ? [Communication with GOT] in the GT Designer2 menu, select [Project download -> GOT], check the import data, import to GOT, and then check the operation.

5) Assignment of I/O signal numbers

I/O numbers 0 to 255 are used at GOT.

I/O numbers 0 to 255 are common for the parallel I/O interface, parallel I/O unit, and GOT.

If the customer’s system configuration uses a parallel I/O interface or parallel I/O unit, please be aware that importing from GOT to input signals for the part being used is ignored.

For example, if a parallel I/O interface is inserted in option slot 1, even if X00 to X1F from GOT is rewritten, this will not be reflected to robot controller input numbers 0 to 31, and the parallel I/O interface input numbers are reflected to robot controller input numbers 0 to 31.

Robot I/O and GOT assignment

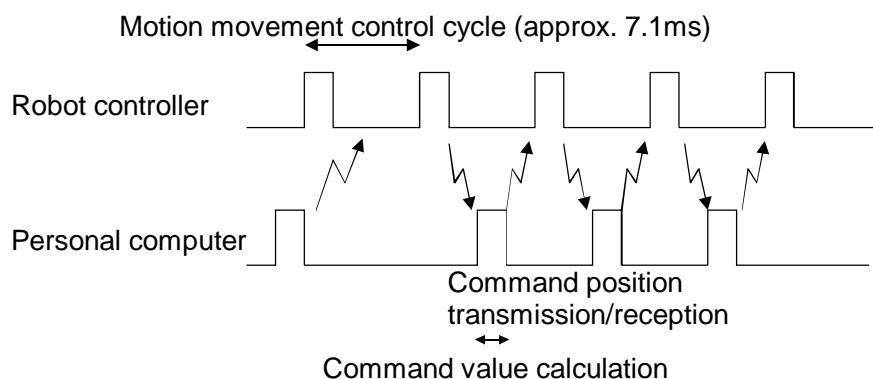
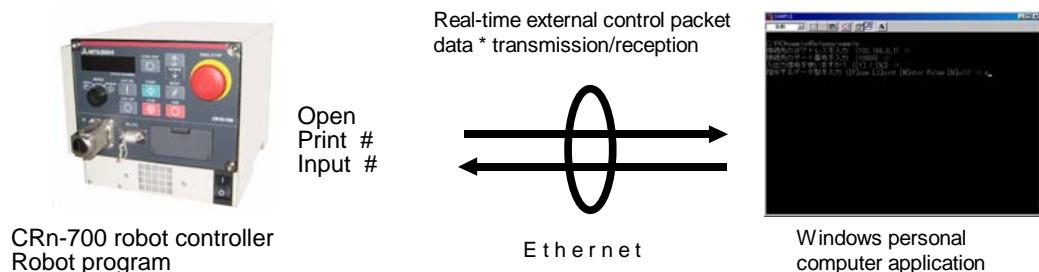
Station	Robot controller I/O no.	Installing slot of Parallel I/O interface	Station no. setting (rotary switch) of parallel I/O unit	GOT1000 XY devices as viewed from GOT
0	0 to 31	Option slot 1	0	X00-X1F / Y00-Y1F
1	32 to 63	Option slot2	1	X20-X3F / Y20-Y3F
2	64 to 95	Option slot3	2	X04-X5F / Y40-Y5F
3	96 to 127	—	3	X60-X7F / Y60-Y7F
4	128 to 159	—	4	X80-X9F / Y80-Y9F
5	160 to 191	—	5	XA0-XBF / YA0-YBF
6	192 to 223	—	6	XC0-XDF / YC0-YDF
7	224 to 255	—	7	XE0-XFF / YE0-YFF

7 Appendix

7.1 Real-time external control function

The robot motion movement control can retrieve the position command at real-time in cycle units, and move to the commanded position. It is also possible to monitor the input/output signals or output the signals simultaneously.

Using the robot language Mxt command, real-time communication (command/monitor) is carried out with communication.

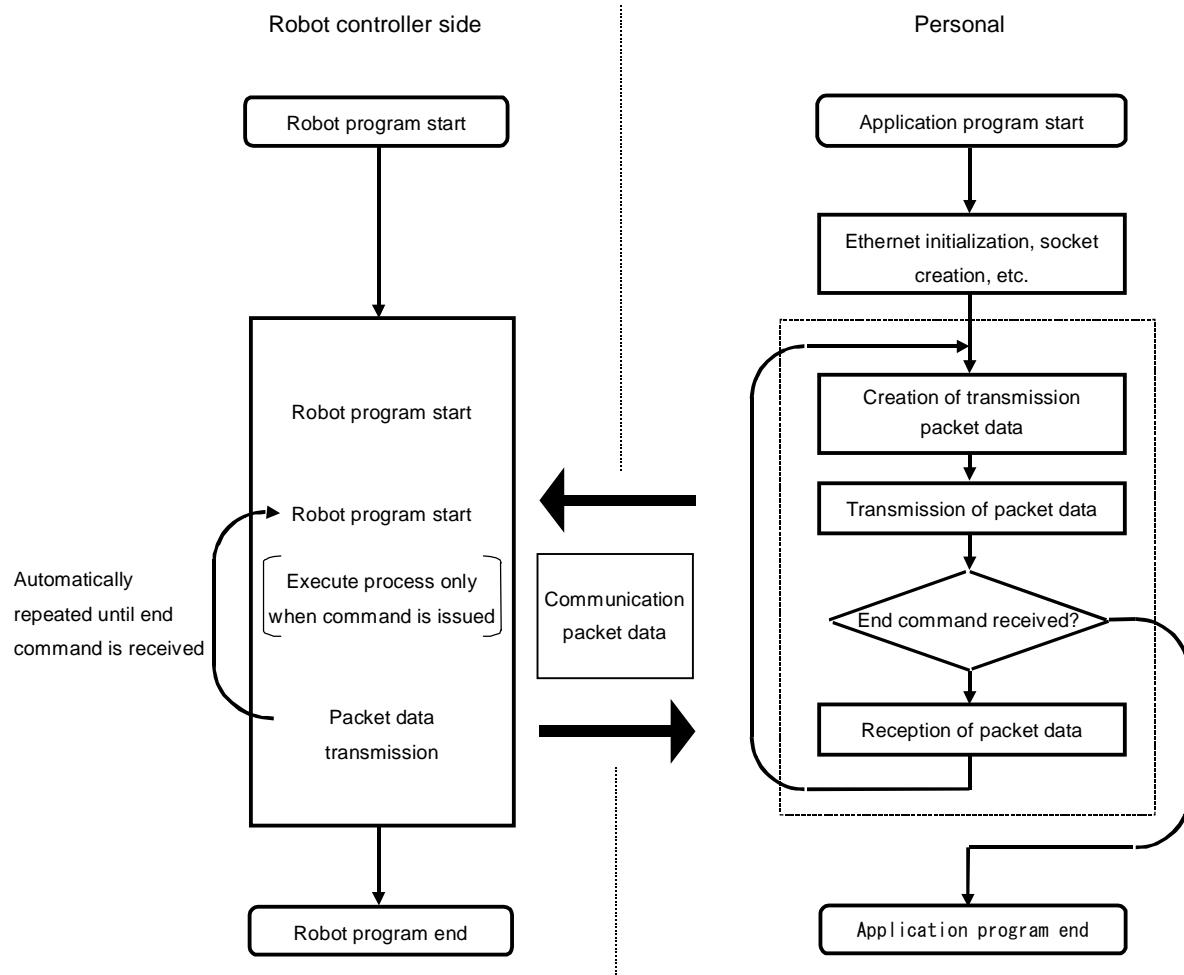


The following table lists the position command data for giving the target move position from the personal computer to the robot for each hour of the motion operation control cycle, and the monitor data types from the robot.

For more information about communication data, see the [Page 230, "Mxt \(Move External\)"](#) and [Page 454, "7.1.1 Explanation of communication data packet"](#) in this document.

Position command data type	Monitor data type
<ul style="list-style-type: none"> [1] Rectangular coordinate data [2] Joint coordinate data [3] Motor pulse coordinate data 	<ul style="list-style-type: none"> [1] Rectangular coordinate data [2] Joint coordinate data [3] Motor pulse coordinate data [4] Rectangular coordinate data (command value after filter processing) [5] Joint coordinate data (command value after filter processing) [6] Motor pulse coordinate data (after filter processing) [7] Rectangular coordinate data (encoder feedback value) [8] Joint coordinate data (encoder feedback value) [9] Motor pulse coordinate data (encoder feedback value) [10] Current command (%) [11] Current feedback (%)

* Flow of real-time external control



7.1.1 Explanation of communication data packet

The structure of the communication data packet used with the real-time external control function is explained in this section.

The same communication data packet for real-time external control is used for commanding the position and for monitoring.

The contents differ when transmitting (commanding) from the personal computer to the controller and when receiving (monitoring) from the controller to the personal computer.

Refer to the following communication data packet structure and section "5.2.2 Sample program for real-time external control function", and create the application. The C language data type is used in the following table.

Communication data packet

Command	unsigned short (2-byte)	Designate the validity of the real-time external command, and the end. 0 // Real-time external command invalid 1 // Real-time external command valid 255// Real-time external command end
Transmission data type designation SendType	unsigned short (2-byte)	<p>1) When transmitting (commanding) from the personal computer to the controller, designate the type of position data transmitted from the personal computer.</p> <p>There is no data at the first transmission.</p> <p>0 // No data 1 // XYZ data 2 // Joint data 3 // Motor pulse data</p> <p>2) When receiving (monitoring) from the controller to the personal computer, indicate the type of position data replied from the controller.</p> <p>0 // No data 1 // XYZ data 2 // Joint data 3 // Motor pulse data 4 // XYZ data (Position after filter process) 5 // Joint data (Position after filter process) 6 // Motor pulse data (Position after filter process) 7 // XYZ data (Encoder feedback value) 8 // Joint data (Encoder feedback value) 9 // Motor pulse data (Encoder feedback value) 10 // Current command [%] 11 // Current feedback [%]</p> <p>* It is the same as RecvType. You may use whichever.</p>

Reply data type designation RecvType	unsigned short (2-byte)	<p>1) When transmitting (commanding) from the personal computer to the controller, designate the type of data replied from the controller.</p> <p>0 // No data 1 // XYZ data 2 // Joint data 3 // pulse data 4 // XYZ data (Position after filter process) 5 // Joint data (Position after filter process) 6 // Motor pulse data (Position after filter process) 7 // XYZ data (Encoder feedback value) 8 // Joint data (Encoder feedback value) 9 // Motor pulse data (Encoder feedback value) 10 // Current command [%] 11 // Current feedback [%]</p> <p>2) When receiving (monitoring) from the controller to the personal computer, indicate the type of position data replied from the controller.</p> <p>0 // No data 1 // XYZ data 2 // Joint data 3 // Motor pulse data 4 // XYZ data (Position after filter process) 5 // Joint data (Position after filter process) 6 // Motor pulse data (Position after filter process) 7 // XYZ data (Encoder feedback value) 8 // Joint data (Encoder feedback value) 9 // Motor pulse data (Encoder feedback value) 10 // Current command [%] 11 // Current feedback [%]</p> <p>* It is the same as RecvType. You may use whichever.</p>
Reservation reserve	unsigned short (2-byte)	Not used.
Position data Pos / jnt / pls	POSE, JOINT or PULSE (40-byte) * Refer to strdef.h in the sample program for details on each data structure.	<p>1) When transmitting (commanding) from the personal computer to the controller, designate the command position data transmitted from the personal computer.</p> <p>Set this to the same data type as that designated for the transmission data type designation.</p> <p>2) When receiving (monitoring) from the controller to the personal computer, this indicates the position data replied from the controller.</p> <p>The data type is shown in SendType (= RecvType) .</p> <p>The contents of data are common to command/monitor.</p> <p>POSE // XYZ type [mm/rad] JOINT // Joint type [rad] PULSE // Motor pulse type [the pulse] or Current type [%].</p>
Transmission input/output signal data designation SendIOType	unsigned short (2-byte)	<p>1) When transmitting (commanding) from the personal computer to the controller, designate the data type of the input/output signal transmitted from the personal computer.</p> <p>Designate "No data" when not using this function.</p> <p>2) When receiving (monitoring) from the controller to the personal computer, this indicates the data type of the input/output signal replied from the controller.</p> <p>The contents of the data are common.</p> <p>0 // No data 1 // Output signal 2 // Input signal</p>

Reply input/output signal data designation RecvIOType	unsigned short (2-byte)	<p>1) When transmitting (commanding) from the personal computer to the controller, designate the data type of the input/output signal replied from the controller. Designate "No data" when not using this function.</p> <p>0 // No data 1 // Output signal 2 // Input signal</p> <p>2) When receiving (monitoring) from the controller to the personal computer, Not used.</p>
Input/output signal data BitTop BitMask IoData	unsigned short unsigned short unsigned short (2-byte x 3)	<p>1) When transmitting (commanding) from the personal computer to the controller, designate the output signal data transmitted from the personal computer.</p> <p>2) When receiving (monitoring) from the controller to the personal computer, this indicates the input/output signal data replied from the controller.</p> <p>The contents of the data are common.</p> <p>BitTop: // Head bit No. of input or output signal BitMask: // Bit mask pattern designation (valid only for commanding) IoData: // Input or output signal data value (for monitoring) Output signal data value (for commanding) * Data is 16-bit data</p>
Timeout time counter value Tcount	unsigned short (2-byte)	<p>1) When transmitting (commanding) from the personal computer to the controller, Not used.</p> <p>2) When receiving (monitoring) from controller to personal computer, if the timeout time parameter MXTTOUT is a value other than -1, this indicates the No. of times communication with the controller was not possible. When the No. of times is counted and reaches the maximum value, the value will return to the minimum value 0, and the count will be repeated. This is set to 0 when the MXT command is started.</p>
Counter value for communication data Ccount	unsigned long (4-byte)	<p>1) When transmitting (commanding) from the personal computer to the controller, Not used.</p> <p>2) When receiving (monitoring) from controller to personal computer, this indicates the No. of communication times. When the No. of times is counted and reaches the maximum value, the value will return to the minimum value 0, and the count will be repeated. This is set to 0 when the MXT command is started.</p>
Reply data-type specification addition 1 RecvType1	unsigned short (2-byte)	It is the same as reply data-type specification (RecvType). Don't use it for instructions.
Reservation 1 reserve1	unsigned short (2-byte)	Not used.
Data addition 1 pos / jnt / pls	Any of POSE/ JOINT/PULSE. (40-byte)	It is the same as data of pos/jnt/pls. Don't use it for instructions.
Reply data-type specification addition 2 RecvType2	unsigned short (2-byte)	It is the same as reply data-type specification (RecvType). Don't use it for instructions.
Reservation 2 Reserve2	unsigned short (2-byte)	Not used
Data addition 2 pos / jnt / pls	Any of POSE/ JOINT/PULSE. (40-byte)	It is the same as data of pos/jnt/pls. Don't use it for instructions.

Reply data-type specification addition 3 RecvType3	unsigned short (2-byte)	It is the same as reply data-type specification (RecvType). Don't use it for instructions.
Reservation 3 Reserve3	unsigned short (2-byte)	Not used.
Data addition 3 pos / jnt / pls	Any of POSE/ JOINT/PULSE. (40-byte)	It is the same as data of pos/jnt/pls. Don't use it for instructions.

7.1.2 Sample program

This is the sample program of the Ethernet interface.

(1) Sample program of data link

The sample program to do the data link with Microsoft Visual Basic 5.0/6.0 (hereafter written as VB) is herein described.

The program creation is briefly introduced with the following procedure.

For details of VB operation and application producing method, refer to the instruction manual of this software.

1) Preparation of Winsock control

2) Production of form screen

3) Program (Form1.frm)

There is the program following 2 passages. Use either according to the customer's system.

a) Program for the clients (when using the personal computer as the client and using the controller as the server).

b) Program for the server (when using the personal computer as the server and using the controller as the client).

* About the work of 1) 2), the client and the server are the same.

Here, VB requires either Professional Edition or Enterprise Edition. Learning Edition can not be used since Winsock

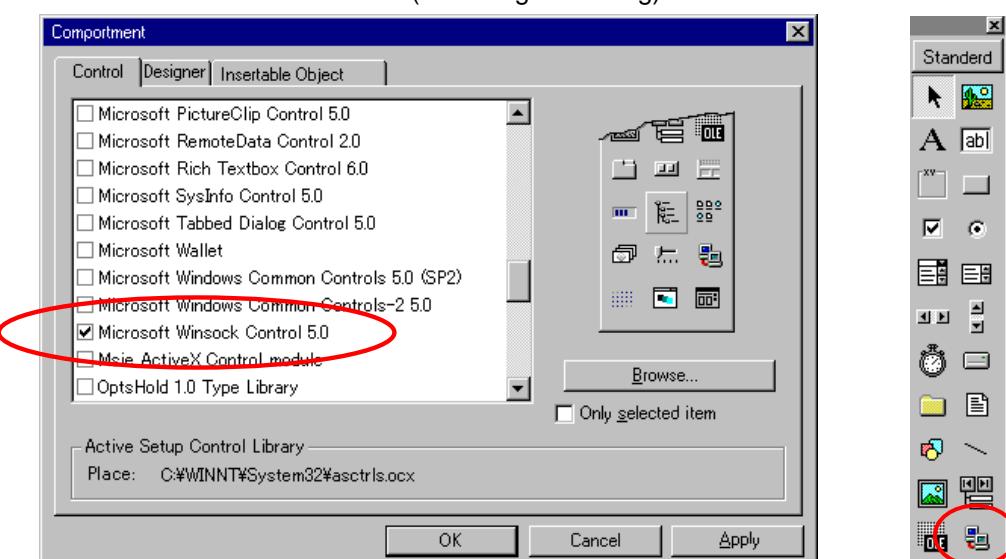
(Windows Socket) control is not appended.

1) Preparation of Winsock control

Winsock control is added to the project.

Start-up VB, newly open standard EXE and click "component" of "project" menu, and the window will be displayed as follows. And, check "Microsoft Winsock Control **". (Lower left drawing ** represents the version)

"Winsock" is added to the tool box. (Lower right drawing)

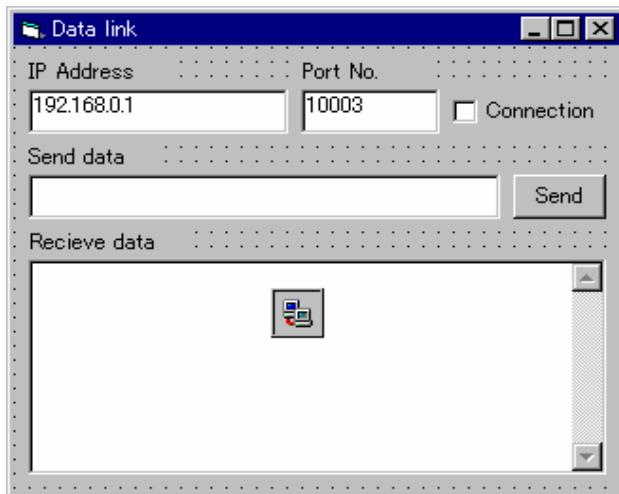


2) Production of form screen

On the form, 4 test boxes, 1 command button, 1 check box and 1 Winsock control are arranged.

The major change points of the properties are shown below.

Major changed points of properties		
Object name	Property	Setting value
Form1	Caption	Data link
Command1	Caption	Send
	Enabled	False
Text1	Text	192.168.0.1
Text2	Text	10003
Text3	Text	
Text4	MultiLine	True
	ScrollBars	2-Vertical
Check1	Caption	Connection



3) Program (Form1.frm)

VERSION 5.00

Object = "{248DD890-BB45-11CF-9ABC-0080C7E7B78D}#1.0#0"; "MSWINSCK.OCX"

```

Begin VB.Form Form1      'Screen setting  From here
Caption      = "Data link"
ClientHeight = 3795
ClientLeft   = 60
ClientTop    = 345
ClientWidth  = 4800
LinkTopic    = "Form1"
ScaleHeight  = 3795
ScaleWidth   = 4800
StartUpPosition= 3 Predefined value of Windows
Begin MSWinsockLib.Winsock Winsock1
    Left        = 2040
    Top         = 2040
    _ExtentX   = 741
    _ExtentY   = 741
End
Begin VB.CommandButton Command1
    Caption    = "Send"
    Enabled    = 0      'False
    Height    = 375
    Left       = 3960
    TabIndex   = 6
    Top        = 1080
    Width      = 735
End
Begin VB.CheckBox Check1
    Caption    = "Connection"
    Height    = 375
    Left       = 3960
    TabIndex   = 4
    Top        = 360

```

```

Width      = 735
End
Begin VB.TextBox Text4
    Height     = 1815
    Left       = 120
    MultiLine  = -1      'True
    ScrollBars = 2        'Vertical
    TabIndex   = 7
    Top        = 1800
    Width      = 4575
End
Begin VB.TextBox Text3
    Height     = 375
    Left       = 120
    TabIndex   = 5
    Top        = 1080
    Width      = 3735
End
Begin VB.TextBox Text2
    Height     = 375
    Left       = 2280
    TabIndex   = 3
    Text       = "10003"
    Top        = 360
    Width      = 1575
End
Begin VB.TextBox Text1
    Height     = 375
    Left       = 120
    TabIndex   = 2
    Text       = "192.168.0.1"
    Top        = 360
    Width      = 2055
End
Begin VB.Label Label4
    Caption    = "Receive data"
    Height    = 195
    Left      = 120
    TabIndex  = 9
    Top       = 1560
    Width     = 975
End
Begin VB.Label Label3
    Caption    = "Send data"
    Height    = 195
    Left      = 120
    TabIndex  = 8
    Top       = 840
    Width     = 975
End
Begin VB.Label Label2
    Caption    = "Port No."
    Height    = 195
    Left      = 2280
    TabIndex  = 1
    Top       = 120

```

```

Width      = 975
End
Begin VB.Label Label1
    Caption    = "IP address"
    Height     = 255
    Left       = 120
    TabIndex   = 0
    Top        = 120
    Width      = 1095
End
End      'Screen setting To here

Attribute VB_Name      = "Form1"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable   = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed      = False

```

a) Program for the clients (when using the personal computer as the client and using the controller as the server).

Option Explicit

Dim RecvData() As Byte

Private Sub Check1_Click() ' Process when the connection check button is clicked

If Check1.Value Then

```

    Winsock1.RemoteHost= Text1.Text
    Winsock1.RemotePort= Text2.Text
    Winsock1.Connect

```

Else

```
    Winsock1.Close

```

End If

End Sub

Private Sub Winsock1_Connect() ' Process when the network can be connected

```
    Command1.Enabled = True

```

End Sub

Private Sub Winsock1_Close() ' Process when the network is closed

```
    Check1.Value = False

```

End Sub

Private Sub Command1_Click() ' Process when "Transmission" command button is clicked

```
    Winsock1.SendData (Text3.Text)

```

End Sub

Private Sub Winsock1_DataArrival(ByVal bytesTotal As Long) ' Process when the received data arrives

If bytesTotal > 0 Then

```
    ReDim RecvData(bytesTotal - 1)

```

```
    Call Winsock1.GetData(RecvData, , bytesTotal)

```

```
    Text4.SelStart = Len(Text4.Text)

```

```
    Text4.SelText = StrConv(RecvData, vbUnicode)

```

End If

End Sub

```

Private Sub Winsock1_Error(ByVal Number As Integer, _
    Description As String, ByVal Scode As Long, _
    ByVal Source As String, ByVal HelpFile As String, _
    ByVal HelpContext As Long, CancelDisplay As Boolean) ' Process when an error occurs in Window
    Socket
    Check1.Value = False
    Command1.Enabled = False
    Winsock1.Close
    MsgBox " Error:" & Number & "(" & Description & ")"
End Sub

```

b) Program for the server (when using the personal computer as the server and using the controller as the client).

```

Option Explicit
Dim RecvData() As Byte

```

```

Private Sub Form_Load()
    Text1.Enabled = False ' Make edit of the IP address impossible.
End Sub

```

```

Private Sub Check1_Click() ' Process when the connection check button is clicked
    If Check1.Value Then
        Text1.Text = Winsock1.LocalIP
        Winsock1.LocalPort = Text2.Text
        Winsock1.Listen
    Else
        Command1.Enabled = False
        Winsock1.Close
    End If
End Sub

```

```

Private Sub Winsock1_Connect()' Process when the network can be connected
    Command1.Enabled = True
End Sub

```

```

Private Sub Winsock1_Close()' Process when the network is closed
    Check1.Value = False
End Sub

```

```

Private Sub Command1_Click()' Process when "Transmission" command button is clicked
    Winsock1.SendData (Text3.Text)
End Sub

```

```

Private Sub Winsock1_ConnectionRequest(ByVal requestID As Long) ' Process when the connection
    demand comes
    If Winsock1.State <> sckClosed Then Winsock1.Close
    Winsock1.Accept requestID
    Command1.Enabled = True
End Sub

```

```

Private Sub Winsock1_DataArrival(ByVal bytesTotal As Long)' Process when the received data arrives
    If bytesTotal > 0 Then

```

```

ReDim RecvData(bytesTotal - 1)
Call Winsock1.GetData(RecvData, , bytesTotal)
Text4.SelStart = Len(Text4.Text)
Text4.SelText = StrConv(RecvData, vbUnicode)
Text4.Text = Text4.Text & vbCrLf
End If
End Sub

Private Sub Winsock1_Error(ByVal Number As Integer, _
    Description As String, ByVal Scode As Long, _
    ByVal Source As String, ByVal HelpFile As String, _
    ByVal HelpContext As Long, CancelDisplay As Boolean)' Process when an error occurs in Window
Socket
    Check1.Value = False
    Command1.Enabled = False
    Winsock1.Close
    MsgBox "Error:" & Number & "(" & Description & ")"
End Sub

```

*Relation of Open command communication line file name COMn: and parameter COMDEV
 COMDEV (1), (2), (3), (4), (5), (6), (7), (8)

Communication line file name	COMDEV
COM1:	(1)
COM2:	(2)
COM3:	(3)
COM4:	(4)
COM5:	(5)
COM6:	(6)
COM7:	(7)
COM8:	(8)

*Channel name assigned to parameter COMDEV and protocol setting parameter name
 OPT11 to OPT19 are assigned to (1) to (8).
 The protocol is set in 2 (data link).

Port No.*1	Channel name	Protocol	
	COMDEV setting value	CPRCE**	Setting value
10001	OPT11	CPRCE11	2
10002	OPT12	CPRCE12	2
10003	OPT13	CPRCE13	2
10004	OPT14	CPRCE14	2
10005	OPT15	CPRCE15	2
10006	OPT16	CPRCE16	2
10007	OPT17	CPRCE17	2
10008	OPT18	CPRCE18	2
10009	OPT19	CPRCE19	2

(2) Sample program for real-time external control function

A sample program that establishes a data link using Microsoft Visual C++5.0/6.0 (hereinafter VC) is shown below.

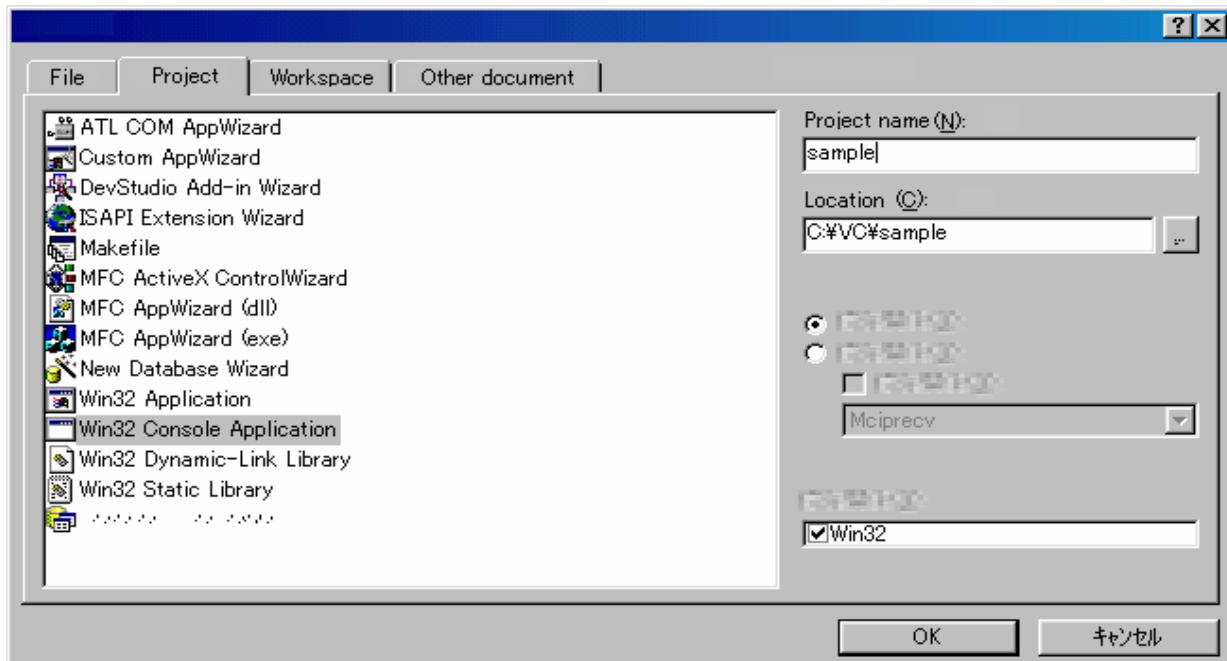
The procedures for creating the program are briefly explained below.

Refer to the software manuals for details on operating VC and creating the application.

- 1) Create new project
- 2) Create program sample.cpp/strdef.h

1) Create new project

Start VC, and create a new project. Set the name to Win32 Console Application.



Using the project setting, add ws2_32.lib to the object/library module.

2) Create program sample.cpp/strdef.h

Newly create the header file strdef.h and source file sample.cpp.

<Notes at compiling>

Use the setup of the alignment compiler option of the structure member with the 8 bytes of initial value. After new creation of the project of Visual C++, if the setup is used with initial value, there is no problem. Refer to the help of Visual C++ for details.

a)Header file strdef.h

```
/****************************************************************************
// Real-time control sample program
// Communication packet data structure definition header file
// strdef.h

/*
/*Joint coordinate system (Set unused axis to 0)*/
/*Refer to the instruction manual enclosed */
/*with each robot for details on each element. */
/*
```

```

typedef struct{
    float    j1;//J1 axis angle (radian)
    float    j2;//J2 axis angle (radian)
    float    j3;//J3 axis angle (radian)
    float    j4;//J4 axis angle (radian)
    float    j5;//J5 axis angle (radian)
    float    j6;//J6 axis angle (radian)
    float    j7;//Additional axis 1 (J7 axis angle) (radian)
    float    j8;//Additional axis 2 (J8 axis angle) (radian)
} JOINT;

/*****************************************/
/*XYZ coordinate system (Set unused axis to 0)*/
/*Refer to the instruction manual enclosed */
/*with each robot for details on each element. */
/*****************************************/

typedef struct{
    float    x;//X axis coordinate value (mm)
    float    y;// Y axis coordinate value (mm)
    float    z;// Z axis coordinate value (mm)
    float    a;// A axis coordinate value (radian)
    float    b;// B axis coordinate value (radian)
    float    c;// C axis coordinate value (radian)
    float    l1;// Additional axis 1 (mm or radian)
    float    l2;// Additional axis 2 (mm or radian)
} WORLD;

typedef struct{
    WORLD      w;
    unsigned int sflg1;//Structural flag 1
    unsigned int sflg2;//Structural flag 2
} POSE;

/*****************************************/
/*Pulse coordinate system (Set unused axis to 0)*/
/*These coordinates express each joint*/
/*with a motor pulse value. */
/*****************************************/

typedef struct{
    long     p1;//Motor 1 axis
    long     p2;// Motor 2 axis
    long     p3;// Motor 3 axis
    long     p4;// Motor 4 axis
    long     p5;// Motor 5 axis
    long     p6;// Motor 6 axis
    long     p7;//Additional axis 1 (Motor 7 axis)
    long     p8;//Additional axis 2 (Motor 8 axis)
} PULSE;

/*****************************************/
/*Real-time function communication data packet */
/*****************************************/

typedef struct enet_rtcmd_str {
    unsigned short Command;//Command

```

```

#define MXT_CMD_NULL0//Real-time external command invalid
#define MXT_CMD_MOVE1// Real-time external command valid
#define MXT_CMD_END255//Real-time external command end

    unsigned short SendType;//Command data type designation
    unsigned short RecvType;//Monitor data type designation
    /////////// Command or monitor data type /**
#define MXT_TYP_NULL0//No data
//For the command and monitor /////////////
#define MXT_TYP_POSE1//XYZ data
#define MXT_TYP_JOINT2//Joint data
#define MXT_TYP_PULSE3 //pulse data
////////// For position related monitor /**
#define MXT_TYP_FPOSE4// XYZ data (after filter process)
#define MXT_TYP_FJOINT5// Joint data (after filter process)
#define MXT_TYP_FPULSE6// Pulse data (after filter process)
#define MXT_TYP_FB_POSE7// XYZ data (Encoder feedback value)
#define MXT_TYP_FB_JOINT8// Joint data (Encoder feedback value)
#define MXT_TYP_FB_PULSE9// Pulse data (Encoder feedback value)
//For current related monitors ///////////
#define MXT_TYP_CMDCUR10//Electric current command
#define MXT_TYP_FBKCUR11//Electric current feedback

    unsigned short reserve;// Reserved
    union rtdata { //Command data
        POSE pos;//XYZ type [mm/rad]
        JOINT jnt;//Joint type [rad]
        PULSE pls;//Pulse type [pls]
    long Ing1[8];//Integer type [% / non-unit]
    } dat;

    unsigned short SendIOType;// Send input/output signal data designation
    unsigned short RecvIOType;// Return input/output signal data designation
#define MXT_IO_NULL0//No data
#define MXT_IO_OUT1//Output signal
#define MXT_IO_IN2//Input signal

    unsigned short BitTop;// Head bit No.
    unsigned short BitMask;// Transmission bit mask pattern designation (0x0001-0xffff)
    unsigned short IoData;// Input/output signal data (0x0000-0xffff)

    unsigned short TCount;// Timeout time counter value
    unsigned long CCount;// Transmission data counter value

    unsigned short RecvType1;// Reply data-type specification 1 .
    unsigned short reserve1;// Reserved 1
    union rtdata1 { // Monitor data 1 .
        POSE pos1;// XYZ type [mm/rad] .
        JOINT jnt1;// JOINT type [mm/rad] .
        PULSE pls1; // PULSE type [mm/rad] .
        long Ing1[8]; // Integer type [% / non-unit] .
    } dat1;
    unsigned short RecvType2;// Reply data-type specification 2 .
    unsigned short reserve2;// Reserved 2
    union rtdata2 { //Monitor data 2 .
        POSE pos2;// XYZ type [mm/rad] .

```

```

JOINT jnt2; // JOINT type [mm/rad] .
PULSE pls2; // PULSE type [mm/rad] or Integer type [% / non-unit].
long lng2[8];// Integer type [% / non-unit] .
} dat2;
unsigned short RecvType3;// Reply data-type specification 3 .
unsigned short reserve3; // Reserved 3
union rtdata3 { // Monitor data 3 .
    POSE pos3;// XYZ type [mm/rad] .
    JOINT jnt3;// JOINT type [mm/rad] .
    PULSE pls3;// PULSE type [mm/rad] or Integer type [% / non-unit].
    long lng3[8];// Integer type [% / non-unit] .
} dat3;
} MXTCMD;

```

b)Source file sample.cpp
// sample.cpp

```

#include <windows.h>
#include <iostream.h>
#include <winsock.h>
#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <math.h>
#include "strdef.h"
#define NO_FLAGS_SET 0
#define MAXBUFLEN 512

INT main(VOID)
{
    WSADATA Data;
    SOCKADDR_IN destSockAddr;
    SOCKET destSocket;
    unsigned long destAddr;
    int status;
    int numsnnt;
    int numrcv;
    char sendText[MAXBUFLEN];
    char recvText[MAXBUFLEN];
    char dst_ip_address[MAXBUFLEN];
    unsigned short port;
    char msg[MAXBUFLEN];
    char buf[MAXBUFLEN];
    char type, type_mon[4];
    unsigned short IOSendType;// Send input/output signal data designation
    unsigned short IORecvType;// Reply input/output signal data designation
    unsigned short IBitTop=0;
    unsigned short IBitMask=0xffff;
    unsigned short IBitData=0;

    cout << " Input connection destination IP address (192.168.0.1) ->";
    cin.getline(dst_ip_address, MAXBUFLEN);
    if(dst_ip_address[0]==0) strcpy(dst_ip_address, "192.168.0.1");

    cout << " Input connection destination port No. (10000) -> ";

```

```

cin.getline(msg, MAXBUFLEN);
if(msg[0]!=0)  port=atoi(msg);
else          port=10000;

cout << " Use input/output signal? [Y] / [N]-> ";
cin.getline(msg, MAXBUFLEN);
if(msg[0]!=0 && (msg[0]=='Y' || msg[0]=='y')) {
    cout << "What is target? Input signal/output signal [I]nput / [O]utput-> ";
    cin.getline(msg, MAXBUFLEN);
    switch(msg[0]) {
        case 'O':// Set target to output signal
        case 'o':
            IOSendType = MXT_IO_OUT;
            IORcvType = MXT_IO_OUT;
            break;
        case 'I':// Set target to input signal
        case 'i':
            default:
                IOSendType = MXT_IO_NULL;
                IORcvType = MXT_IO_IN;
                break;
    }
}

cout << " Input head bit No. 0`32767-> ";
cin.getline(msg, MAXBUFLEN);
if(msg[0]!=0)  IOBitTop = atoi(msg);
else          IOBitTop = 0;

if(IOSendType==MXT_IO_OUT) { // Only for output signal
    cout << "Input bit mask pattern for output as hexadecimal 0000`FFFF-> ";
    cin.getline(msg, MAXBUFLEN);
    if(msg[0]!=0)  sscanf(msg,"%4x",&IOBitMask);
    else          IOBitMask = 0;
    cout << "Input bit data for output as hexadecimal 0000`FFFF-> ";
    cin.getline(msg, MAXBUFLEN);
    if(msg[0]!=0)  sscanf(msg,"%4x",&IOBitData);
    else          IOBitData = 0;
}
}

cout << " --- Input the data type of command. --- \n";
cout << "[0: None / 1: XYZ / 2:JOINT / 3: PULSE]\n";
cout << " -- please input the number -- [0] - [3]->";
cin.getline(msg, MAXBUFLEN);
type = atoi(msg);

for(int k=0; k<4; k++) {
    sprintf (msg, " --- input the data type of monitor ( %d-th ) --- \n", k);
    cout << msg;
    cout << "[0: None]\n";
    cout << "[1: XYZ / 2:JOINT / 3: PULSE] Command value \n";
    cout << "[4: XYZ / 5: JOINT / 6: PULSE] Command value after the filter process \n";
    cout << "[7: XYZ/ 5:JOINT/ 6:PULSE] Feedback value. \n";
    cout << "[10: Electric current value / 11: Electric current feedback] ... Electric current value. \n";
    cout << "Input the numeral [0]`[11]->";
    cin.getline(msg, MAXBUFLEN);
    type_mon[k] = atoi(msg);
}

```

```

    }

    sprintf(msg, "IP=%s / PORT=%d / Send Type=%d / Monitor Type0/1/2/3=%d/%d/%d/%d"
    , dst_ip_address, port , type
    , type_mon[0], type_mon[1], type_mon[2], type_mon[3]);
    cout << msg << endl;

    cout << "[Enter]= End Ä^ [d]= Monitor data display";
    cout << "[z/x]= Increment/decrement first command data transmitted by the delta amount. ";

    cout << " Is it all right? [Enter] / [Ctrl+C] ";
    cin.getline(msg, MAXBUFLEN);

// Windows Socket DLL initialization
status=WSAStartup(MAKEWORD(1, 1), &Data);
if (status != 0)
cerr << "ERROR: WSAStartup unsuccessful" << endl;

// IP address, port, etc., setting
memset(&destSockAddr, 0, sizeof(destSockAddr));
destAddr=inet_addr(dst_ip_address);
memcpy(&destSockAddr.sin_addr, &destAddr, sizeof(destAddr));
destSockAddr.sin_port=htons(port);
destSockAddr.sin_family=AF_INET;

// Socket creation
destSocket=socket(AF_INET, SOCK_DGRAM, 0);
if (destSocket == INVALID_SOCKET) {
    cerr << "ERROR: socket unsuccessful" << endl;
    status=WSACleanup();
    if (status == SOCKET_ERROR)
        cerr << "ERROR: WSACleanup unsuccessful" << endl;
    return(1);
}

MXTCMD MXTsend;
MXTCMD MXTrecv;
JOINT jnt_now;
POSE pos_now;
PULSE pls_now;

unsigned long counter = 0;
int loop = 1;
int disp = 0;
int disp_data = 0;
int ch;
float delta=(float)0.0;
long ratio=1;
int retry;
fd_set SockSet;// Socket group used with select
timeval sTimeOut;// For timeout setting

memset(&MXTsend, 0, sizeof(MXTsend));
memset(&jnt_now, 0, sizeof(JOINT));
memset(&pos_now, 0, sizeof(POSE));
memset(&pls_now, 0, sizeof(PULSE));

```

```

while(loop) {

    memset(&MXTsend, 0, sizeof(MXTsend));
    memset(&MXTrecv, 0, sizeof(MXTrecv));

    // Transmission data creation
    if(loop==1) { // Only first time
        MXTsend.Command = MXT_CMD_NULL;
        MXTsend.SendType = MXT_TYP_NULL;
        MXTsend.RecvType = type;
        MXTsend.SendIOType = MXT_IO_NULL;
        MXTsend.RecvIOType = IOSendType;
        MXTsend.CCount = counter = 0;
    }
    else { // Second and following times
        MXTsend.Command = MXT_CMD_MOVE;
        MXTsend.SendType = type;
        MXTsend.RecvType = type*_mon[0];
        MXTsend.RecvType1= type_mon[1];
        MXTsend.RecvType2= type_mon[2];
        MXTsend.RecvType3= type_mon[3];

        switch(type) {
            case MXT_TYP_JOINT:
                memcpy(&MXTsend.dat.jnt, &jnt_now, sizeof(JOINT));
                MXTsend.dat.jnt.j1 += (float)(delta*ratio*3.141592/180.0);
                break;
            case MXT_TYP_POSE:
                memcpy(&MXTsend.dat.pos, &pos_now, sizeof(POSE));
                MXTsend.dat.pos.w.x += (delta*ratio);
                break;
            case MXT_TYP_PULSE:
                memcpy(&MXTsend.dat.pls, &pls_now, sizeof(PULSE));
                MXTsend.dat.pls.p1 += (long)((delta*ratio)*10);
                break;
            default:
                break;
        }
        MXTsend.SendIOType = IOSendType;
        MXTsend.RecvIOType = IORecvType;
        MXTsend.BitTop = IOBitTop;
        MXTsend.BitMask = IOBitMask;
        MXTsend.IoData = IOBitData;
        MXTsend.CCount = counter;
    }

    // Keyboard input
    // [Enter]=End / [d]= Display the monitor data, or none / [0/1/2/3]= Change of monitor data display
    // [z/x]=Increment/decrement first command data transmitted by the delta amount
    while(kbhit() != 0) {
        ch=getch();
        switch(ch) {
            case 0xd:
                MXTsend.Command = MXT_CMD_END;
                loop = 0;
                break;
        }
    }
}

```

```

        case 'Z':
        case 'z':
            delta += (float)0.1;
            break;
        case 'X':
        case 'x':
            delta -= (float)0.1;
            break;
        case 'C':
        case 'c':
            delta = (float)0.0;
            break;
        case 'd':
            disp = ~disp;
            break;
        case '0': case '1': case '2': case '3':
            disp_data = ch - '0';
            break;
    }
}

memset(sendText, 0, MAXBUFSIZE);
memcpy(sendText, &MXTsend, sizeof(MXTsend));
if(disp) {
    sprintf(buf, "Send (%ld):",counter);
    cout << buf << endl;
}
numsn=sendto(destSocket, sendText, sizeof(MXTCMD), NO_FLAGS_SET
, (LPSOCKADDR) &destSockAddr, sizeof(destSockAddr));
if (numsn != sizeof(MXTCMD)) {
    cerr << "ERROR: sendto unsuccessful" << endl;
    status=closesocket(destSocket);
    if (status == SOCKET_ERROR)
        cerr << "ERROR: closesocket unsuccessful" << endl;
    status=WSACleanup();
    if (status == SOCKET_ERROR)
        cerr << "ERROR: WSACleanup unsuccessful" << endl;
    return(1);
}

memset(recvText, 0, MAXBUFSIZE);

retry = 1;// No. of reception retries
while(retry) {
    FD_ZERO(&SockSet);// SockSet initialization
    FD_SET(destSocket, &SockSet);// Socket registration
    sTimeOut.tv_sec = 1;// Transmission timeout setting (sec)
    sTimeOut.tv_usec = 0;// (Épsec)
    status = select(0, &SockSet, (fd_set *)NULL, (fd_set *)NULL, &sTimeOut);
    if(status == SOCKET_ERROR) {
        return(1);
    }
    // If it receives by the time-out
    if((status > 0) && (FD_ISSET(destSocket, &SockSet) != 0)) {
        numrcv=recvfrom(destSocket, recvText, MAXBUFSIZE, NO_FLAGS_SET, NULL, NULL);
        if (numrcv == SOCKET_ERROR) {

```

```

        cerr << "ERROR: recvfrom unsuccessful" << endl;
        status=closesocket(destSocket);
        if (status == SOCKET_ERROR)
            cerr << "ERROR: closesocket unsuccessful" << endl;
        status=WSACleanup();
        if (status == SOCKET_ERROR)
            cerr << "ERROR: WSACleanup unsuccessful" << endl;
        return(1);
    }
    memcpy(&MXTrecv, recvText, sizeof(MXTrecv));
    char str[10];
    if(MXTrecv.SendIOType==MXT_IO_IN)
        sprintf(str,"IN%04x", MXTrecv.loData);
    else if(MXTrecv.SendIOType==MXT_IO_OUT)
        sprintf(str,"OT%04x", MXTrecv.loData);
    else    sprintf(str,"-----");

    int DispType;
    void *DispData;

    switch(disp_data) {
        case 0:
            DispType = MXTrecv.RecvType;
            DispData = &MXTrecv.dat;
            break;
        case 1:
            DispType = MXTrecv.RecvType1;
            DispData = &MXTrecv.dat1;
            break;
        case 2:
            DispType = MXTrecv.RecvType2;
            DispData = &MXTrecv.dat2;
            break;
        case 3:
            DispType = MXTrecv.RecvType3;
            DispData = &MXTrecv.dat3;
            break;
        default:
            break;
    }

    switch(DispType) {
        case MXT_TYP_JOINT:
        case MXT_TYP_FJOINT:
        case MXT_TYP_FB_JOINT:
            if(loop==1) {
                memcpy(&jnt_now, DispData, sizeof(JOINT));
                loop = 2;
            }
            if(disp) {
                JOINT *j=(JOINT*)DispData;
                sprintf(buf, "Receive (%d): TCount=%d Type(JOINT)=%d\n"
%7.2f,%7.2f,%7.2f,%7.2f,%7.2f,%7.2f,%7.2f (%s)"
,MXTrecv.CCount,MXTrecv.TCount,DispType
,j->j1, j->j2, j->j3 ,j->j4, j->j5, j->j6, j->j7, j->j8, str);
                cout << buf << endl;
            }
    }
}

```

```

    }
    break;
  case MXT_TYP_POSE:
  case MXT_TYP_FPOSE:
  case MXT_TYP_FB_POSE:
    if(loop==1) {
      memcpy(&pos_now, &MXTrecv.dat.pos, sizeof(POSE));
      loop = 2;
    }
    if(disp) {
      POSE *p=(POSE*)DispData;
      sprintf(buf, "Receive (%ld): TCount=%d Type(POSE)=%d\n"
      "%7.2f,%7.2f,%7.2f,%7.2f,%7.2f,%7.2f, %04x,%04x (%s)"
      ",MXTrecv.CCount,MXTrecv.TCount,DispType
      ,p->w.x, p->w.y, p->w.z, p->w.a, p->w.b, p->w.c
      , p->sflg1, p->sflg2, str);
      cout << buf << endl;
    }
    break;
  case MXT_TYP_PULSE:
  case MXT_TYP_FPULSE:
  case MXT_TYP_FB_PULSE:
  case MXT_TYP_CMDCUR:
  case MXT_TYP_FBKCUR:
    if(loop==1) {
      memcpy(&pls_now, &MXTrecv.dat.pls, sizeof(PULSE));
      loop = 2;
    }
    if(disp) {
      PULSE *l=(PULSE*)DispData;
      sprintf(buf, "Receive (%ld): TCount=%d Type(PULSE/OTHER)=%d\n"
      "%ld,%ld,%ld,%ld,%ld,%ld,%ld,%ld (%s)"
      ",MXTrecv.CCount,MXTrecv.TCount,DispType
      ,l->p1, l->p2, l->p3, l->p4, l->p5, l->p6, l->p7, l->p8, str);
      cout << buf << endl;
    }
    break;
  case MXT_TYP_NULL:
    if(loop==1) {
      loop = 2;
    }
    if(disp) {
      sprintf(buf, "Receive (%ld): TCount=%d Type(NULL)=%d\n (%s)"
      ",MXTrecv.CCount,MXTrecv.TCount, DispType, str);
      cout << buf << endl;
    }
    break;
  default:
    cout << "Bad data type.\n" << endl;
    break;
}
counter++;// Count up only when communication is successful
retry=0;// Leave reception loop
}
else { // Reception timeout
  cout << "... Receive Timeout! <Push [Enter] to stop the program>" << endl;
}

```

```
    retry--;// No. of retries subtraction
    if(retry==0)  loop=0; // End program if No. of retries is 0
}
} /* while(retry) */
} /* while(loop) */

// End
cout << "/// End /// ";
sprintf(buf, "counter = %ld", counter);
cout << buf << endl;

//Close socket
status=closesocket(destSocket);
if (status == SOCKET_ERROR)
    cerr << "ERROR: closesocket unsuccessful" << endl;
status=WSACleanup();
if (status == SOCKET_ERROR)
    cerr << "ERROR: WSACleanup unsuccessful" << endl;

return 0;
}
```

7.2 Configuration flag

The configuration flag indicates the robot posture.

For the 6-axis type robot, the robot hand end is saved with the position data configured of X, Y, Z, A, B and C. However, even with the same position data, there are several postures that the robot can change to. The posture is expressed by this configuration flag, and the posture is saved with FL1 in the position constant (X, Y, Z, A, B, C) (FL1, FL2).

The types of configuration flags are shown below.

*For vertical multi-joint type robot

(1) Right/Left

P is center of flange in comparison with the plane through the J1 axis vertical to the ground.

Q is center of J5 axis rotation in comparison with the plane through the J1 axis vertical to the ground.

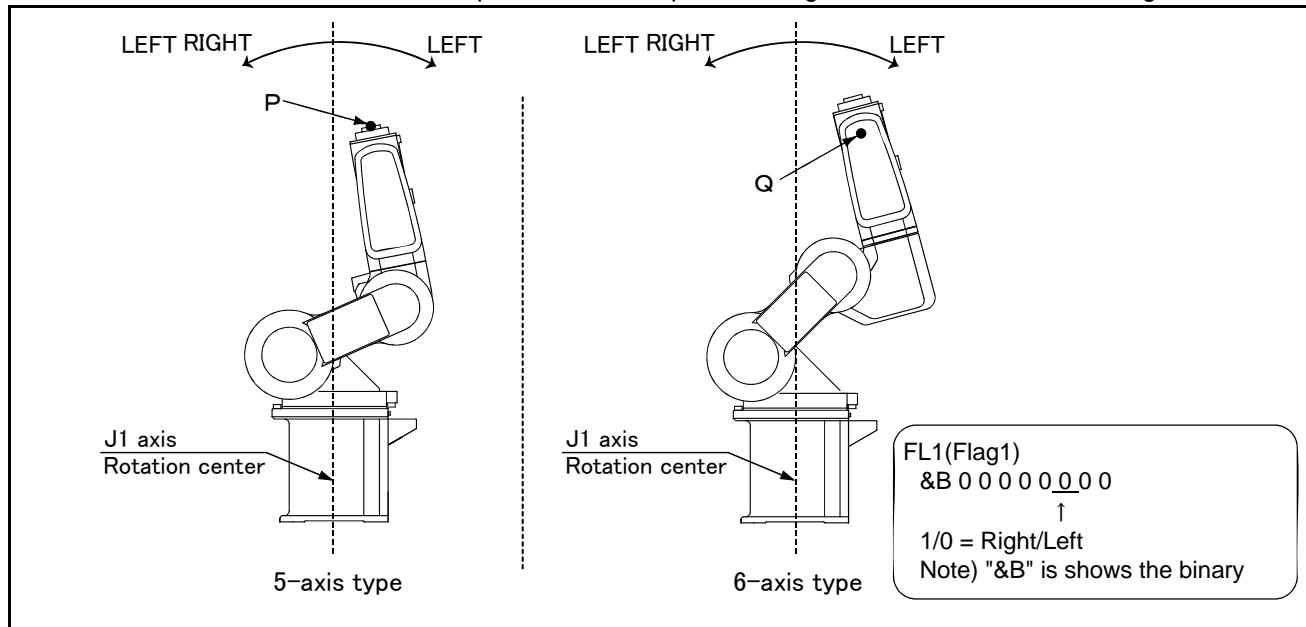


Fig.7-1:Configuration flag (Right/Left)

(2) ABOVE/BELOW

Q is center of J5 axis rotation in comparison with the plane through both the J3 and the J2 axis.

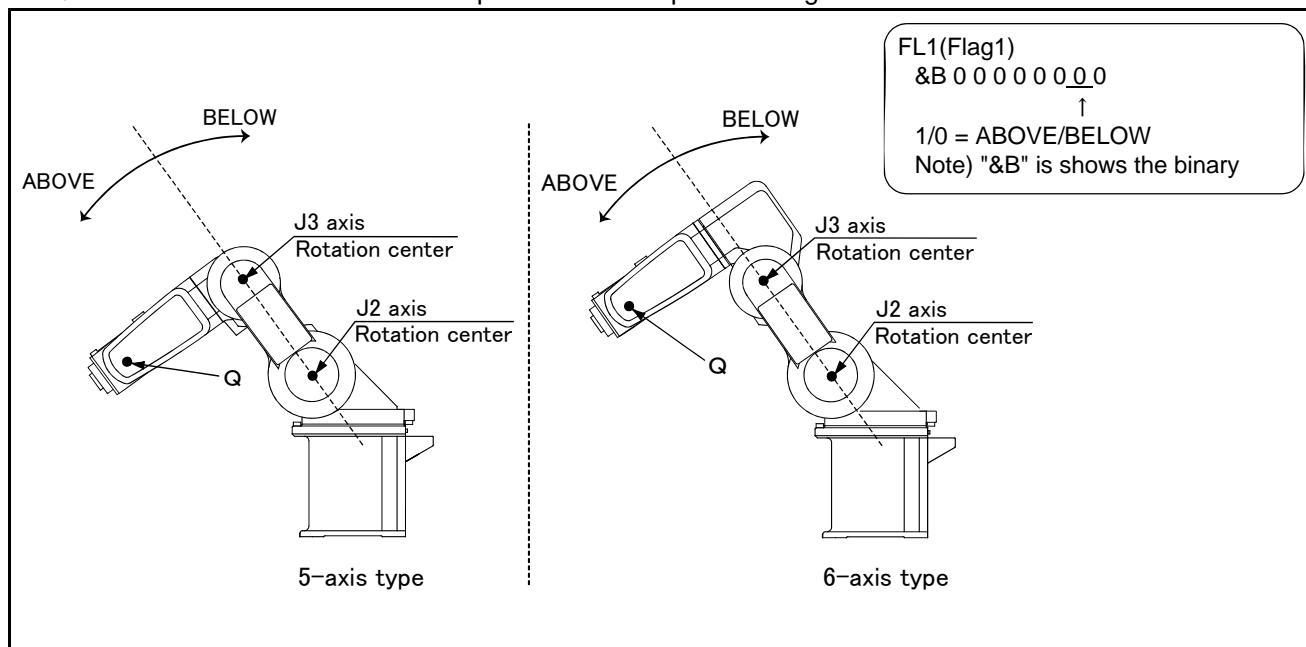


Fig.7-2:Configuration flag (ABOVE/BELOW)

(3) NONFLIP/FLIP (6-axis robot only.)

This means in which side the J6 axis is in comparison with the plane through both the J4 and the J5 axis.

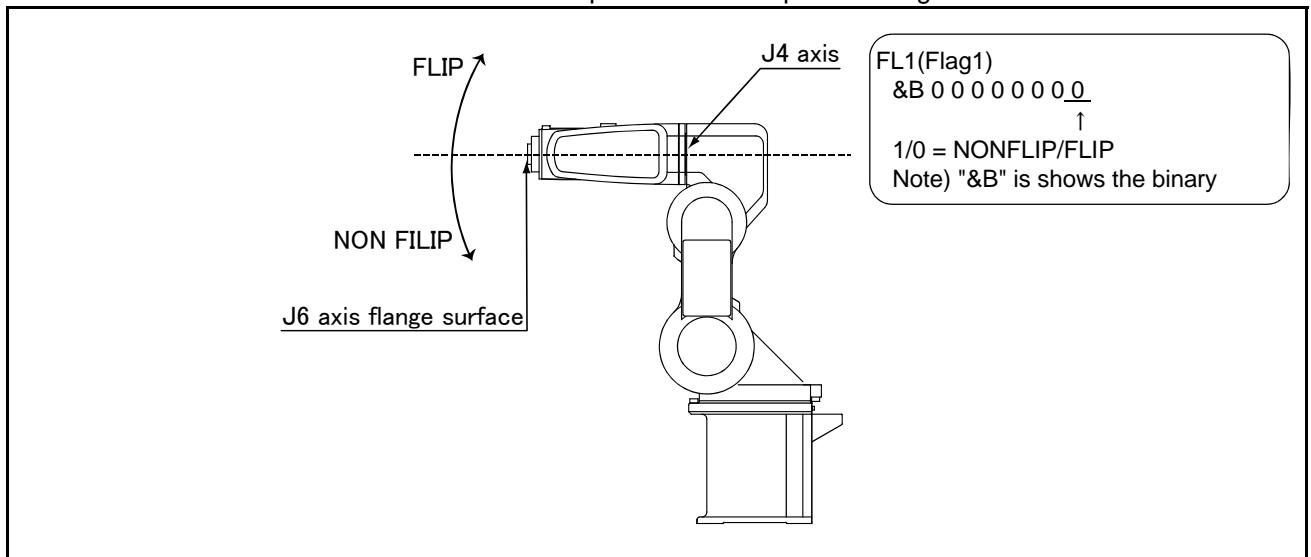


Fig.7-3 : Configuration flag (NONFLIP/FLIP)

*For horizontal multi-joint type robot

(1) Right/Left

Indicates the location of the end axis relative to the line that passes through both the rotational center of the J1 axis and the rotational center of the J2 axis.

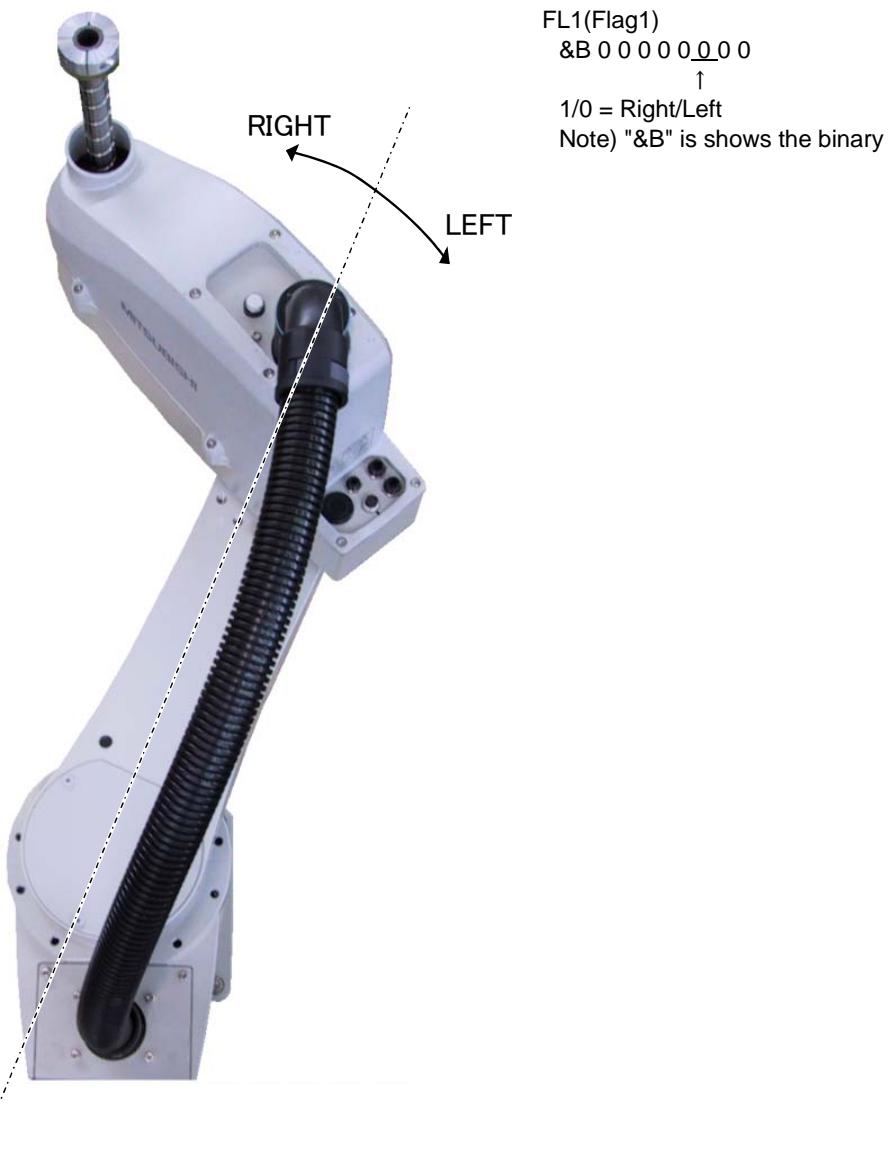


Fig.7-4:Configuration flag (Right/Left)



HEAD OFFICE: TOKYO BUILDING, 2-7-3, MARUNOUCHI, CHIYODA-KU, TOKYO 100-8310, JAPAN
NAGOYA WORKS: 5-1-14, YADA-MINAMI, HIGASHI-KU, NAGOYA 461-8670, JAPAN