MITSUBISHI ELECTRIC

Mitsubishi Industrial Robot

CR800 Series Controller

INSTRUCTION MANUAL

Detailed explanations of functions and operations

This instruction manual apply to both the CR800-R/CR800-Q controller corresponding to iQ Platform, and the CR800-D controller of standalone type.

MELFA
BFP-A3478-M

⚠ 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

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

⚠DANGER When automatic operation of the robot is performed using multiple control devices (GOT, programmable controller, push-button switch), the interlocking of operation rights of the devices, etc. must be designed by the customer.

⚠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.

⚠DANGER Do not connect the Handy GOT when using the GOT direct connection function of this product. Failure to observe this may result in property damage or bodily injury because the Handy GOT can automatically operate the robot regardless of whether the operation rights are enabled or not.

⚠DANGER Do not connect the Handy GOT to a programmable controller when using an iQ Platform compatible product with the CR800-R/CR800-Q controller. Failure to observe this may result in property damage or bodily injury because the Handy GOT can automatically operate the robot regardless of whether the operation rights are enabled or not.

⚠DANGER Do not remove the SSCNET III cable while power is supplied to the multiple CPU system or the servo amplifier. Do not look directly at light emitted from the tip of SSCNET III connectors or SSCNET III cables of the Motion CPU or the servo amplifier. Eye discomfort may be felt if exposed to the light. (Reference: SSCNET III employs a Class 1 or equivalent light source as specified in JIS C 6802 and IEC60825-1 (domestic standards in Japan).)

⚠DANGER Do not remove the SSCNET III cable while power is supplied to the controller. Do not look directly at light emitted from the tip of SSCNET III connectors or SSCNET III cables. Eye discomfort may be felt if exposed to the light. (Reference: SSCNET III employs a Class 1 or equivalent light source as specified in JIS C 6802 and IEC60825-1 (domestic standards in Japan).)

⚠DANGER Attach the cap to the SSCNET III connector after disconnecting the SSCNET III cable. If the cap is not attached, dirt or dust may adhere to the connector pins, resulting in deterioration connector properties, and leading to malfunction.

⚠CAUTION

Make sure there are no mistakes in the wiring. Connecting differently to the way specified in the manual can result in errors, such as the emergency stop not being released. In order to prevent errors occurring, please be sure to check that all functions (such as the teaching box emergency stop, customer emergency stop, and door switch) are working properly after the wiring setup is completed.

⚠CAUTION

Use the network equipments (personal computer, USB hub, LAN hub, etc) confirmed by manufacturer. The thing unsuitable for the FA environment (related with conformity, temperature or noise) exists in the equipments connected to USB. When using network equipment, measures against the noise, such as measures against EMI and the addition of the ferrite core, may be necessary. Please fully confirm the operation by customer. Guarantee and maintenance of the equipment on the market (usual office automation equipment) cannot be performed.

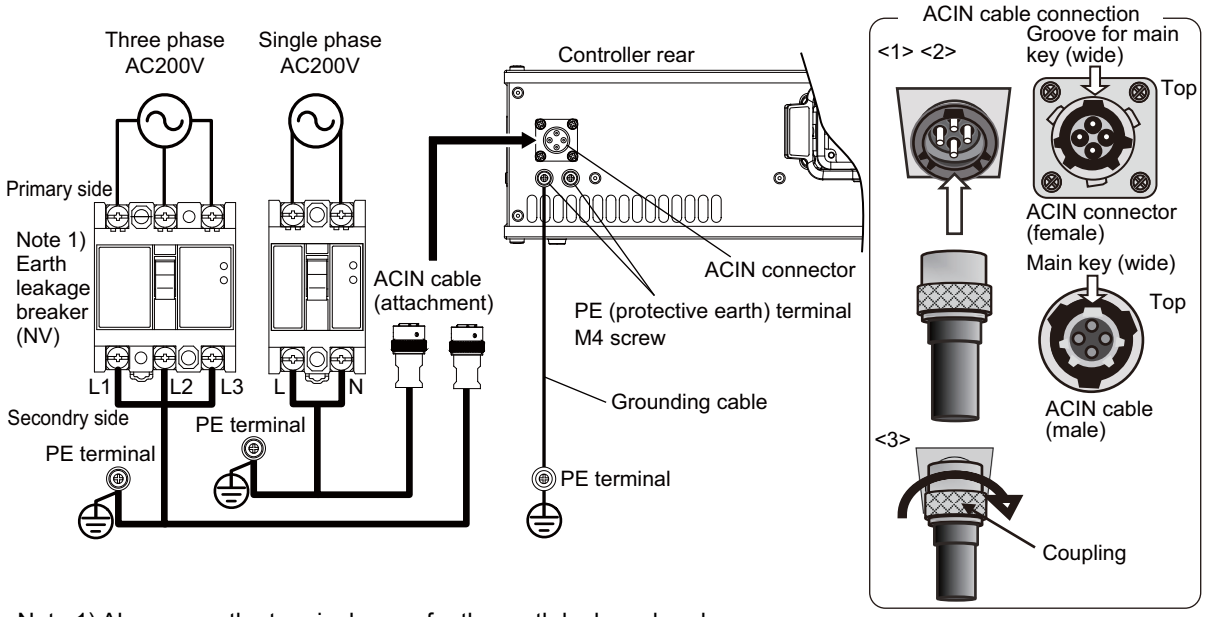
⚠CAUTION

To maintain the security (confidentiality, integrity, and availability) of the robot and the system against unauthorized access, DoS[*1] attacks, computer viruses, and other cyberattacks from unreliable networks and devices via network, take appropriate measures such as firewalls, virtual private networks (VPNs), and antivirus solutions.
Mitsubishi Electric shall have no responsibility or liability for any problems involving robot trouble and system trouble by unauthorized access, DoS attacks, computer viruses, and other cyberattacks.
*1 DoS: A denial-of-service (DoS) attack disrupts services by overloading systems or exploiting vulnerabilities, resulting in a denial-of-service (DoS) state.

*CR800 controller

Notes of the basic component are shown.

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



Note 1) Always use the terminal cover for the earth leakage breaker.

1) Prepare the following items.

Part name	Specifications	Remarks
Earth leakage breaker	The following is recommended product. Single phase: NV30FAU-2P-10A-AC100-240V-30 mA (Terminal cover: TCS-05FA2) Three phase: NV30FAU-3P-10A-AC100-240V-30 mA (Terminal cover: TCS-05FA3)	Prepared by customer.
Cable for primary power supply	AWG14 (2 mm^2) or above	Prepared by customer. Tightening torque for terminal fixing screw is 2 to 3 Nm.
Grounding cable	AWG14 (2 mm^2) or above	Prepared by customer. Tightening torque for terminal fixing screw is 2 to 3 Nm.
ACIN cable	Terminal: M5, cable length: 3 m	Supplied with the product.

2) Confirm that the primary power matches the specifications.
3) Confirm that the primary power is OFF and that the earth leakage breaker power switch is OFF.
4) Connect the ACIN cable to the breaker.
Connect the power terminals of the ACIN cable to the secondary side terminals of the earth leakage breaker. Also, ground the FG terminal of the cable.
5) Connect the ACIN cable to the ACIN connector on the rear of the controller.
<1> Face the main key on the ACIN cable plug upwards. (Refer to the "ACIN cable connection" illustration.)
<2> Align the main key of the ACIN cable plug with the grooves on the ACIN connector. Push the plug into the connector as far as it will go.
The plug may be damaged if it is not correctly aligned with the connector.
<3> Tighten the coupling on the ACIN cable, turning it to the right until it locks.
6) Connect one end of the grounding cable to the PE (protective earth) terminal on the controller and ground the other end (2-point grounding) in order to comply with the requirements of EN 61800-5-1 for the touch current of 3.5 mA AC or more.
7) Connect the primary power cable to the primary side terminal of the earth leakage breaker.

Revision history

Date	Specifications No.	Details of revisions
2017-05-30	BFP-A3478	• First print
2017-06-28	BFP-A3478-A	• “(1) PLC CPU parameter setting” in “6.2.1 Parameter setting” was modified. • Factory default setting of the hand I/O type parameter (HIOTYPE) was corrected.
2017-09-25	BFP-A3478-B	• Reference program of PrmWrite command was modified. • Supplementary explanation for commands were added. (If…Then…ElseIf…Else…EndIf, Select Case)
2017-11-01	BFP-A3478-C	• Description of countermeasures against unauthorized access was added. • Descriptions of continuity function were deleted. • “7.6 Log function” was added.
2018-03-01	BFP-A3478-D	• The CR800-Q controller was added.
2018-09-01	BFP-A3478-E	• Supplementary explanation for the compliance mode was added. (Cmp Jnt, Cmp Pos, Cmp Tool, Cmp Off)
2018-10-15	BFP-A3478-F	• The high speed spline interpolation command was added.
2018-12-25	BFP-A3478-G	• Added further explanation of the ACIN cable.
2019-04-19	BFP-A3478-H	• Revised the descriptions of device assignment. • Corrected error. (Incorrect: MELFA-BASIC V)
2020-01-24	BFP-A3478-J	• Added the following parameters: TOOLSPEC, RCDUVER • Corrected the explanation of Base, Tool, Def Work, and M_Tool. • Added “6.6 How to select and run a program using external signals”.
2020-10-30	BFP-A3478-K	• Amended the precautions regarding the prevention of unauthorized access. • Amended the example #Include statement string. (Fig. 4-6) • Revised section 3.11 "Operation to Temporarily Reset an Error that Cannot Be Canceled". • Added and corrected CR series values for the simulated component setting parameter. • Corrected the example Cmp Off command string. • Added the section "2.4 Robot CPU status indicator LEDs". • Corrected other mistakes and changed some sections.
2021-02-19	BFP-A3478-M	• Added command explanations. (Def Act, Function...FEnd) • Added FUNCSPEC parameter.

*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 (option)), and the functions and specifications of the MELFA-BASIC VI programming language.
Apply to both the CR800-R/CR800-Q series controller corresponding to iQ Platform, and the CR800-D series controller. Especially the function added individually is indicated to be "CR800-R" and "CR800-D".

Also in this instruction manual, operation of robot programs such as start-up and shutdown is explained based on key operations on the teaching pendant and operation in the operation screen of the teaching pendant. For the operation using external signals (exclusive input/output signals), refer to the following page, which summarizes the exclusive input/output signals corresponding to the T/B operation. Using the parameter settings, please assign exclusive input/output signals to general purpose input/output signals.

Always read through this manual before starting use to ensure correct usage of the robot.
As much as possible, we have tried to include all special operations in this instruction manual.
Please assume that operations not included in this manual are "not possible".
Note that this document is prepared for the following software versions.

Controller : Version
CR800-R: A5p or later
CR800-D: A5p or later
CR800-Q: A5p or later
T/B : Version 1.7 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." or "alarm may occur".
  Please contact your nearest dealer if you find any doubtful, wrong or skipped point.
- This is the original document.
- The ETHERNET® is a registered trademark of the Xerox Corp.
- AutoCAD® is a registered trademark of Autodesk, Inc. in the U.S. and other countries.
- All other company names and production names in this document are the trademarks or registered trademarks of their respective owners.
- Illustrations in this Instruction Manual may differ from the actual products.

Copyright(C) 2017-2021 MITSUBISHI ELECTRIC CORPORATION

For users operating robots that have not been mounted with an operation panel:

Operation of robot programs such as start-up and shutdown are carried out using external signals (exclusive input/output signals). This instruction manual is based on key operations on the teaching pendant and operation in the operation screen of the teaching pendant. Using the parameter settings, please assign exclusive input/output signals that correspond with each operation to general purpose input/output signals, and operate the robot using signal operations.
The following table details exclusive input/output signals that correspond with the operations explained in this manual. Please use this as a reference to assign signals and operate the robot.
For further details regarding parameters please see Page 637, "6.3 Dedicated input/output", for the time chart of each signal please see Page 650, "6.5 External signal timing chart", and for instructions on how to set parameters please see Page 84, "3.15 Operation of parameter screen".

Table: Conversion table of the buttons and dedicated I/O signals

T/B operation	Parameter name	Class	Function	Default setting
Start In operation	START	Input	Starts a program.	3,0
		Output	Indicates that a program is being executed.	
[STOP] key Standby	STOP	Input	Stops a program.	0,-1
		Output	Indicates that the program is paused.	
[RESET] key ERROR lamp Program reset	ERRRESET	Input	Releases the error state.	2,2
		Output	Indicates that an error has occurred.	
	SLOTINIT	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.	-1,-1
		Output	Outputs that in the program selection enabled state.	
Program selection [OVRD↑] key [OVRD↓] key Step display	PRGSEL	Input	Selects the value inputted into the signal assigned to the numerical input as a program number.	-1,
		Output	-	
	PRGOUT	Input	Outputs the program number selected to the signal assigned to the numerical output.	-1,-1
		Output	Indicates outputting the program number to the numerical output.	
	OVRDSEL	Input	Sets the value inputted into the signal assigned to the numerical input as a override.	-1,
		Output	-	
	OVRDOUT	Input	Outputs the override value to the signal assigned to the numerical output.	-1,-1
		Output	Indicates outputting the override value to the numerical output.	
	LINEOUT	Input	Outputs the current line number to the signal assigned to the numerical output.	-1,-1
		Output	Indicates outputting the current line number to the numerical output.	
	ERROUT	Input	Outputs the error number to the signal assigned to the numerical output.	-1,-1
		Output	Indicates outputting the error number to the numerical output.	
	IODATA	Input	Reads the program number and the override value as a binary value.	-1,-1, -1,-1
		Output	Outputs the program number, line number and override value as a binary value.	
Consecutive cycle	CYCLE	Input	Starts the cycle stop.	-1,-1
		Output	Outputs that the cycle stop is operating.	
[SERVO] key SERVO lamp	SRVON	Input	Turns ON the servo power supply.	4,1
		Output	Indicates the servo power supply is ON.	
[SERVO] key SERVO lamp	SRVOFF	Input	Turns OFF the servo power supply.	1,-1
		Output	This output indicates a status where the servo power supply cannot be turned ON. (Echo back)	

Contents

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 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 2-7
2.1 Teaching pendant (T/B) functions 2-7
2.1.1 Operation rights 2-8
2.1.2 Handling the T/B 2-9
(1) Installing the T/B 2-9
(2) Removing the T/B 2-9
2.2 Functions Related to Movement and Control 2-10
2.3 Robot type resetting 2-12
2.4 Robot CPU status indicator LEDs 2-13
2.4.1 R-type CPU unit (R16RTCPU) status indicators 2-13
2.4.2 Q-type CPU unit (Q172DSRCPU) status indicators 2-14

3 Explanation of operation methods 3-15
3.1 Operation of the teaching pendant menu screens 3-15
(1) Screen tree 3-15
(2) Input of the number/character 3-19
(3) Selecting a menu 3-20
3.2 Jog Feed (Overview) 3-22
3.2.1 Types of jog feed 3-22
3.2.2 Speed of jog feed 3-23
3.2.3 JOINT jog 3-24
3.2.4 XYZ jog 3-24
3.2.5 TOOL jog 3-25
3.2.6 3-axis XYZ jog 3-25
3.2.7 CYLNDER jog 3-26
3.2.8 WORK jog 3-26
3.2.9 Switching Tool Conversion Data 3-27
3.2.10 Changing the world coordinate (specifies the base coordinate number) 3-28
3.2.11 Impact Detection during Jog Operation 3-30
(1) Impact Detection Level Adjustment during Jog Operation 3-31
3.3 Opening/Closing the Hands 3-32
3.4 Returning to the Safe Point 3-33
3.5 Aligning the Hand 3-34
3.6 Programming 3-36
3.6.1 Creating a program 3-36
(1) Opening the program edit screen 3-36
(2) Creating a program 3-37
(3) Completion of program creation and saving programs 3-39
(4) Correcting a program 3-40
(5) Registering the current position data 3-42
(6) Deletion of the position variable 3-45
(7) Confirming the position data (Position jump) 3-46
(8) Correcting the MDI (Manual Data Input) 3-47
(9) Executing a Command Directly 3-48
3.7 Debugging 3-49
(1) Step feed 3-49
(2) Step return 3-50
(3) Step feed in another slot 3-51
(4) Step jump 3-53

Contents

Page

3.8 Automatic operation 3-54
3.8.1 Setting the operation speed 3-54
3.8.2 Starting automatic operation 3-54
3.8.3 Stopping 3-59
3.8.4 Resuming automatic operation from stopped state 3-59
3.8.5 Resetting the program 3-59
3.9 Turning the servo ON/OFF 3-60
3.10 Error reset operation 3-60
3.11 Operation to Temporarily Reset an Error that Cannot Be Canceled 3-61
3.12 Operating the program control screen 3-63
(1) Program list display 3-63
(2) Copying programs 3-64
(3) Name change of the program (Rename) 3-65
(4) Deleting a program (Delete) 3-66
(5) Protection of the program (Protect) 3-67
(6) Select the program 3-68
3.13 Operation of operating screen 3-69
3.13.1 Display of the execution line 3-69
(1) Select the confirmation menu 3-69
(2) Step feed 3-69
(3) Step jump 3-71
(4) Step feed in another slot 3-71
(5) Finishing of the confirmation screen. 3-72
3.13.2 Test operation 3-72
(1) Select the test operation 3-72
3.13.3 Operating the OPERATION screen 3-73
3.14 Operating the monitor screen 3-74
(1) Input signal monitor 3-74
(2) Output signal monitor 3-76
(3) Input register monitor 3-78
(4) Output register monitor 3-79
(5) Variable monitor 3-81
(6) Error history 3-83
3.15 Operation of parameter screen 3-84
3.16 Operation of the origin and the brake screen 3-86
(1) Origin 3-86
(2) Brake 3-86
3.17 Operation of setup / initialization screen 3-88
(1) Initialize the program 3-88
(2) Initialize the parameter 3-89
(3) Initialize the battery 3-90
(4) Operation 3-90
(5) Time setup 3-91
(6) Version 3-91
3.18 ENHANCED 3-92
(1) SQ DIRECT 3-92
(2) WORK COORD 3-92
3.19 Operation of the initial-setting screen 3-93
(1) Set the display language 3-93
(2) Adjustment of contrast 3-95

4 MELFA-BASIC VI 4-97
4.1 MELFA-BASIC VI functions 4-97
4.1.1 Robot operation control 4-98
(1) Joint interpolation movement 4-98
(2) Linear interpolation movement 4-99

Contents

Page

(3) Circular interpolation movement 4-100
(4) Continuous movement 4-102
(5) Acceleration/deceleration time and speed control 4-103
(6) Confirming that the target position is reached 4-105
(7) High path accuracy control 4-106
(8) Hand and tool control 4-107
4.1.2 Pallet operation 4-108
4.1.3 Program control 4-113
(1) Unconditional branching, conditional branching, waiting 4-113
(2) Repetition 4-114
(3) Interrupt 4-115
(4) Subroutine 4-116
(5) Timer 4-117
(6) Stopping 4-117
4.1.4 Inputting and outputting external signals 4-118
(1) Input signals 4-118
(2) Output signals 4-118
4.1.5 Communication 4-119
4.1.6 Expressions and operations 4-120
(1) List of operator 4-120
(2) Relative calculation of position data (multiplication) 4-122
(3) Relative calculation of position data (Addition) 4-122
4.1.7 Appended statement 4-123
4.2 Multitask function 4-124
4.2.1 What is multitasking? 4-124
4.2.2 Executing a multitask 4-125
4.2.3 Operation state of each slot 4-125
4.2.4 Precautions for creating multitask program 4-127
(1) Relationship between number of tasks and processing time 4-127
(2) Specification of the maximum number of programs executed concurrently 4-127
(3) How to pass data between programs via external variables 4-127
(4) Confirmation of operating status of programs via robot status variables 4-127
(5) The program that operates the robot is basically executed in slot 1. 4-127
(6) How to perform the initialization processing via constantly executed programs 4-128
4.2.5 Precautions for using a multitask program 4-128
(1) Starting the multitask 4-128
(2) Display of operation status 4-128
4.2.6 Example of using multitask 4-129
(1) Robot work details. 4-129
(2) Procedures to multitask execution 4-130
4.2.7 Program capacity 4-131
(1) Program save area 4-131
(2) Program edit area 4-131
(3) Program execution area 4-131
4.3 Detailed specifications of MELFA-BASIC VI 4-132
(1) Program name 4-132
(2) Command statement 4-132
(3) Variable 4-133
4.3.1 Statement 4-134
4.3.2 Appended statement 4-134
4.3.3 Step 4-134
4.3.4 Step No. 4-134
4.3.5 Label 4-134
4.3.6 Types of characters that can be used in program 4-135
4.3.7 Characters having special meanings 4-136
(1) Uppercase and lowercase identification 4-136
(2) Underscore (_) 4-136
(3) Apostrophe (') 4-136

Contents

Page

(4) Asterisk (*) 4-136
(5) Comma (,) 4-136
(6) Period (.) 4-136
(7) Space 4-136
4.3.8 Data type 4-137
4.3.9 Constants 4-137
4.3.10 Numeric value constants 4-137
(1) Decimal number 4-137
(2) Hexadecimal number 4-137
(3) Binary number 4-137
(4) Types of constant 4-137
4.3.11 Character string constants 4-137
4.3.12 Position constants (XYZ/work coordinate constants) 4-138
(1) Coordinate, posture and additional axis data types and meanings 4-138
(2) Meaning of structure flag data type and meanings 4-138
4.3.13 Joint constants 4-139
(1) Axis data format and meanings 4-139
4.3.14 Angle value 4-140
4.3.15 Variables 4-140
4.3.16 Numeric value variables 4-141
4.3.17 Character string variables 4-141
4.3.18 Position variables (XYZ/work coordinate variables) 4-142
4.3.19 Joint variables 4-142
4.3.20 Input/output variables 4-143
4.3.21 Array variables 4-143
4.3.22 External variables 4-144
4.3.23 Program external variables 4-144
4.3.24 User-defined external variables 4-145
4.3.25 Creating User Base Programs 4-146
4.3.26 Scope 4-147
4.3.27 Function procedure 4-147
4.3.28 #Include statement 4-148
4.4 Coordinate system description of the robot 4-149
4.4.1 About the robot's coordinate system 4-149
4.4.2 About base conversion 4-150
4.4.3 About position data 4-151
4.4.4 About tool coordinate system (mechanical interface coordinate system) 4-152
(1) Mechanical interface coordinate system 4-152
(2) Tool coordinate system 4-153
(3) Effects of use of tool coordinate system 4-154
4.5 Robot status variables 4-157
4.5.1 Logic numbers 4-161
4.6 Functions 4-162
(1) User-defined functions 4-162
(2) Built-in functions 4-162
4.7 List of Command 4-164
(1) Command related to movement control 4-164
(2) Command related to program control 4-165
(3) Definition commands 4-166
(4) Multi-task related 4-167
(5) Communications 4-167
(6) Others 4-167
4.8 Operators 4-168
4.9 Priority level of operations 4-169
4.10 Depth of program's control structure 4-169
4.11 Reserved words 4-169
4.12 Detailed explanation of command words 4-170

Contents

Page

4.12.1 How to read the described items 4-170
4.12.2 Explanation of each command word 4-170
4.13 Detailed explanation of Robot Status Variable 4-346
4.13.1 How to Read Described items 4-346
4.13.2 Explanation of Each Robot Status Variable 4-346
4.14 Detailed Explanation of Functions 4-442
4.14.1 How to Read Described items 4-442
4.14.2 Explanation of Each Function 4-442

5 Functions set with parameters 5-484
5.1 Movement parameter 5-484
5.2 Signal parameter 5-499
5.2.1 About multi CPU input offsets (CR800-R/Q controller only) 5-504
(1) Case (A) 5-504
(2) Case (B) 5-505
5.3 Operation parameter 5-507
5.4 Command parameter 5-510
5.5 Communication parameter 5-514
5.6 Standard Tool Coordinates 5-517
5.7 About Standard Base Coordinates 5-519
5.8 About user-defined area 5-520
5.8.1 Selecting a coordinate system 5-521
5.8.2 Setting Areas 5-522
(1) Position Area 5-522
(2) Posture Area 5-523
(3) Additional Axis Area 5-523
5.8.3 Selecting mechanism to be checked 5-524
5.8.4 Specifying behavior within user-defined area 5-524
5.8.5 Example of settings 5-525
5.9 Free plane limit 5-526
5.9.1 The definition of a free plane limit 5-526
5.9.2 Selection of a coordinates system for a free plane limit 5-527
5.10 Automatic return setting after jog feed at pause 5-528
5.11 Automatic execution of program at power up 5-529
5.12 About the hand type 5-530
(1) Solenoid valve types and signal numbers 5-530
5.13 About default hand status 5-531
5.14 About the output signal reset pattern 5-532
5.15 About the communication setting (Ethernet) 5-534
5.15.1 Details of parameters 5-534
(1) NETIP (IP address of robot controller) 5-534
(2) NETMSK (sub-net-mask) 5-534
(3) NETPORT (port No.) 5-534
(4) CRRCE11 to 19 (protocol) 5-535
(5) COMDEV (Definition of devices corresponding to COM1: to 8) 5-535
(6) NETMODE (server specification). 5-535
(7) NETHSTIP (The IP address of the server of the data communication point). 5-535
(8) MXTTOUT (Timeout setting for executing real-time external control command) 5-535
5.15.2 Example of setting of parameter 1 (When the Support Software is used) 5-536
5.15.3 Example of setting of parameter 2-1 5-537
5.15.4 Example of setting parameters 2-2 5-538
5.15.5 Example of setting parameters 3 5-539
5.15.6 Connection confirmation 5-540
5.15.7 Checking the connection with the Windows ping command 5-540
5.16 Hand and Workpiece Conditions (optimum acceleration/deceleration settings) 5-541

Contents

Page

5.17 About the singular point adjacent alarm 5-543
5.18 High-speed RAM operation function 5-544
5.19 Warm-Up Operation Mode 5-545
5.20 About singular point passage function 5-552
5.21 About the collision detection function 5-557
(1) Overview of the function 5-557
(2) Related parameters 5-558
(3) How to use the collision detection function 5-559
5.22 Optimizing the overload level 5-565
5.23 Multi-rotational restrictions for the pallet definition instruction 5-566
5.24 Interference avoidance function 5-567
5.24.1 Operation procedures 5-568
5.24.2 Preparing and connecting the devices 5-569
5.24.3 Registering the simulated components for interference check 5-571
5.24.4 Registering a free plane limit 5-580
5.24.5 Support of additional axes 5-581
5.24.6 Setting the CPU buffer memory expanded function (Checking for interference between robots) 5-582
5.24.7 Calibration between robots (Checking for interference between robots) 5-585
5.24.8 Enabling and disabling the interference avoidance function 5-587
5.24.9 Using the interference avoidance function 5-587
5.24.10 Sample programs 5-589
5.25 Direct control of the PLC input/output module 5-591
5.25.1 CR800-R series 5-591
(1) Operation procedure 5-591
(2) Configuration example of the hardware 5-591
(3) Setting the parameters for the system configuration example 1 5-592
(4) Setting the parameters for the system configuration example 2 5-598
(5) Controlling with status variables 5-603
5.25.2 CR800-Q series 5-605
(1) Operation procedure 5-605
(2) Configuration example of the hardware 5-605
(3) Setting the parameters for the system configuration example 1 5-606
(4) Setting the parameters for the system configuration example 2 5-612
(5) Details of robot parameters 5-619
(6) Controlling with status variables 5-620
5.26 Direct communication with robot CPUs 5-622
(1) Specification 5-622
(2) Description of the status variable 5-622
5.27 Parameter for behavior selection at the error occurrence on dual system 5-623

6 External input/output functions 6-624
6.1 Types 6-624
6.2 PLC link I/O function 6-625
6.2.1 Parameter setting 6-625
(1) PLC CPU parameter setting 6-625
(2) Robot CPU parameter setting 6-629
6.2.2 CPU buffer memory and robot I/O signal compatibility 6-632
6.2.3 Sequence ladder example 6-632
6.2.4 Assignment of the dedicated I/O signal. (at factory shipping) 6-634
6.3 Dedicated input/output 6-637
6.4 Enable/disable status of signals 6-649
6.5 External signal timing chart 6-650
6.5.1 Individual timing chart of each signal 6-650
6.5.2 Timing chart example 6-658
(1) External signal operation timing chart (Part 1) 6-658

Contents

Page

(2) External signal operation timing chart (Part 2) 6-659
(3) Example of external operation timing chart (Part 3) 6-660
(4) Example of external operation timing chart (Part 4) 6-661
(5) Example of external operation timing chart (Part 5) 6-662
6.6 How to select and run a program using external signals 6-663
6.6.1 Methods 6-663
6.6.2 Selecting a method to run a program 6-663
6.6.3 Related I/O parameters 6-663
6.6.4 Operation procedure 6-664
6.7 Emergency stop input 6-666
6.7.1 Robot Behavior upon Emergency Stop Input 6-666
6.8 Devices 6-667
6.8.1 Device list 6-667
6.8.2 Device assignment 6-669
6.8.3 PLC device assignment function 6-672

7 Appendix Appendix-677
7.1 Configuration flag Appendix-677
7.2 Spline interpolation Appendix-680
7.2.1 Outline Appendix-680
(1) Outline Appendix-680
(2) Features Appendix-680
(3) Required devices and software version Appendix-681
(4) Terminology Appendix-681
7.2.2 Specifications Appendix-682
(1) Basic specifications Appendix-682
(2) Restrictions Appendix-683
(3) Robot behavior during spline interpolation Appendix-684
(4) Check related to path points Appendix-685
7.2.3 Explanation of functions Appendix-687
(1) Path adjustment Appendix-687
(2) Operation mode Appendix-690
(3) Signal output Appendix-691
(4) Numerical setting Appendix-692
(5) Frame transformation Appendix-693
7.2.4 Work procedures Appendix-695
7.2.5 Creating the spline file Appendix-697
(1) New file Appendix-697
(2) Creating a file by DXF File Import function Appendix-698
(3) Creating new spline files with robot language Appendix-705
(4) Opening an existing spline file Appendix-705
(5) Explanation of Spline File Edit screen Appendix-706
(6) Spline file editing ribbon Appendix-708
(7) Details of path point data Appendix-711
(8) Editing the spline file Appendix-712
(9) Saving the spline file Appendix-719
(10) Deleting a spline file Appendix-721
(11) Changing the spline No. Appendix-721
(12) Copying the spline file Appendix-722
(13) Spline file manager Appendix-722
(14) Import/export function Appendix-724
(15) Auxiliary editing functions Appendix-732
(16) Displays the spline curve Appendix-734
(17) Checking edited data Appendix-734
(18) Robot program Appendix-736
7.2.6 Creating the robot program Appendix-738
7.2.7 Confirming the movement Appendix-740

Contents

Page

7.2.8 Saving in the robot controller Appendix-741
7.2.9 Adjustment work Appendix-741
(1) Position adjustment function Appendix-741
(2) Frame transformation function Appendix-743
(3) Position jump Appendix-745
(4) Parameter SPLOPTGC Appendix-746
7.2.10 High speed spline interpolation command Appendix-747
(1) Outline Appendix-747
(2) Required software versions Appendix-747
(3) Specifications Appendix-747
(4) RT ToolBox3 settings Appendix-748
(5) Precautions and limitations Appendix-749
7.3 Ex-T control Appendix-750
7.3.1 Outline Appendix-750
(1) Features Appendix-750
(2) Specifications Appendix-751
(3) Equipment and software version requirements Appendix-751
7.3.2 Ex-T coordinates setting Appendix-751
(1) Setting Appendix-751
7.3.3 Ex-T jog Appendix-753
(1) Movement of the posture element in the WORK jog Appendix-754
(2) Movement of the posture element in the Ex-T jog Appendix-754
(3) Ex-T jog operation Appendix-755
(4) WORK jog operation of the RV 6-axis type Appendix-756
(5) WORK jog operation of the RH 4-axis type Appendix-759
(6) WORK jog operation of the RH 4-axis hanging type Appendix-761
7.3.4 Creation of robot program Appendix-763
(1) List of commands/variables related to the Ex-T control Appendix-763
(2) Programming example Appendix-763
7.4 Get-position-quick function Appendix-765
7.4.1 Outline Appendix-765
7.4.2 Specifications of the GPS Function Appendix-765
7.4.3 Specifications of Digital Input Signal Appendix-766
7.4.4 Electric Specification of Digital Input Appendix-767
7.4.5 How to use the GPS function Appendix-768
(1) Position data acquisition at the sensor input timing Appendix-768
(2) Workpiece presence recognition in a cassette Appendix-769
7.5 Upgrade of the servo software Appendix-771
7.6 Log function Appendix-772
7.6.1 Outline Appendix-772
7.6.2 Specifications Appendix-773
(1) Basic specifications Appendix-773
(2) Log file specifications Appendix-773
7.6.3 Error numbers to be saved Appendix-774
7.6.4 Parameter setting Appendix-774
7.6.5 Checking the log file details Appendix-775
(1) Error history Appendix-775
(2) Event history Appendix-776
(3) Program execution history Appendix-778
7.7 Special devices Appendix-779
7.7.1 CR800-R series Appendix-779
7.7.2 CR800-D series Appendix-780
7.7.3 CR800-Q series Appendix-782

1 Before starting use

This chapter explains the details and usage methods of the instruction manuals, the basic terminology and the safety precautions. Moreover, handling and operation of a teaching pendant (T/B) are described based on R32TB in instruction manuals. If using other T/B, such as R56TB, refer to a supplied instruction manual of the T/B.

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.

Manual name	Description
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 explanations 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.
Additional axis function	Explains the specifications, functions and operations of the additional axis control.
Tracking Function	Explains the control function and specifications of conveyor tracking.
GOT Direct Connection Extended Function	Explains the detailed description of data configuration of shared memory, monitoring, and operating procedures about the GOT (standalone type robot).
iQ Platform Supporting Extended Function	Explains the detailed description of data configuration of shared memory, monitoring, and operating procedures about the PLC (iQ Platform compatible type robot).
Ethernet Function	Explains the measures to perform communication with personal computers on Ethernet with the TCP/IP protocol.

1.1.2 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

Terminology	Item/Symbol	Meaning
Item	iQ Platform compatible type	
	Controller	Indicates the controller which controls the robot arm.
	The robot CPU unit or robot CPU	Indicates the CPU unit for the robots which installed to the PLC base unit of MELSEC iQ-R/MELSEC-Q series. It is connected with the controller by the dedicated cable.
	The robot CPU system	Multi-CPU system. It consists of MELSEC units, such as the PLC base unit, the PLC CPU unit, and the robot CPU unit, etc.
	Standalone type	
	Controller	Indicates the controller which controls the robot arm.
Symbol	⚠ 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. Descriptions in this manual are based on R32TB.

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.

⚠DANGER	When automatic operation of the robot is performed using multiple control devices (GOT, programmable controller, push-button switch), the interlocking of operation rights of the devices, etc. must be designed by the customer.
⚠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.

⚠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 Do not connect the Handy GOT when using the GOT direct connection function of this product. Failure to observe this may result in property damage or bodily injury because the Handy GOT can automatically operate the robot regardless of whether the operation rights are enabled or not.

⚠DANGER Do not connect the Handy GOT to a programmable controller when using an iQ Platform compatible product with the CR800-R/CR800-Q controller. Failure to observe this may result in property damage or bodily injury because the Handy GOT can automatically operate the robot regardless of whether the operation rights are enabled or not.

⚠DANGER Do not remove the SSCNET III cable while power is supplied to the multiple CPU system or the servo amplifier. Do not look directly at light emitted from the tip of SSCNET III connectors or SSCNET III cables of the Motion CPU or the servo amplifier. Eye discomfort may be felt if exposed to the light. (Reference: SSCNET III employs a Class 1 or equivalent light source as specified in JIS C 6802 and IEC60825-1 (domestic standards in Japan).)

⚠DANGER Do not remove the SSCNET III cable while power is supplied to the controller. Do not look directly at light emitted from the tip of SSCNET III connectors or SSCNET III cables. Eye discomfort may be felt if exposed to the light. (Reference: SSCNET III employs a Class 1 or equivalent light source as specified in JIS C 6802 and IEC60825-1 (domestic standards in Japan).)

⚠DANGER Attach the cap to the SSCNET III connector after disconnecting the SSCNET III cable. If the cap is not attached, dirt or dust may adhere to the connector pins, resulting in deterioration connector properties, and leading to malfunction.

⚠CAUTION Make sure there are no mistakes in the wiring. Connecting differently to the way specified in the manual can result in failures, such as the emergency stop not being released. In order to prevent from occurring, please be sure to check that all functions (such as the teaching box emergency stop, customer emergency stop, and door switch) are working properly after the wiring setup is completed

⚠CAUTION Use the network equipments (personal computer, USB hub, LAN hub, etc) confirmed by manufacturer. The thing unsuitable for the FA environment (related with conformity, temperature or noise) exists in the equipments connected to USB. When using network equipment, measures against the noise, such as measures against EMI and the addition of the ferrite core, may be necessary. Please fully confirm the operation by customer. Guarantee and maintenance of the equipment on the market (usual office automation equipment) cannot be performed.

⚠CAUTION To maintain the security (confidentiality, integrity, and availability) of the robot and the system against unauthorized access, DoS[*1] attacks, computer viruses, and other cyberattacks from unreliable networks and devices via network, take appropriate measures such as firewalls, virtual private networks (VPNs), and antivirus solutions.
Mitsubishi Electric shall have no responsibility or liability for any problems involving robot trouble and system trouble by unauthorized access, DoS attacks, computer viruses, and other cyberattacks.
*1 DoS: A denial-of-service (DoS) attack disrupts services by overloading systems or exploiting vulnerabilities, resulting in a denial-of-service (DoS) state.

2 Explanation of functions

2.1 Teaching pendant (T/B) functions

This chapter explains the functions of R32TB (optional).



1) [EMG. 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.2-1:General-view

2.1.1 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 Note1)		ENABLE	
	Controller mode (mode selector switch)	AUTOMATIC	MANUAL	AUTOMATIC	MANUAL
Operation rights	T / B	X	X	X Note2)	○
	Personal computer	○ Note3)	X	X Note2)	X
	External signal	○ Note3)	X	X Note2)	X

Note1) In "AUTOMATIC" mode, the T/B has the operation rights while displaying the <OPERATION> screen. (The T/B's status display lamp [ENABLE] blinks)

Note2) If the controller mode is set to "AUTOMATIC" when the T/B is set to "ENABLE", the error 5000 will occur.

Note3) 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.

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

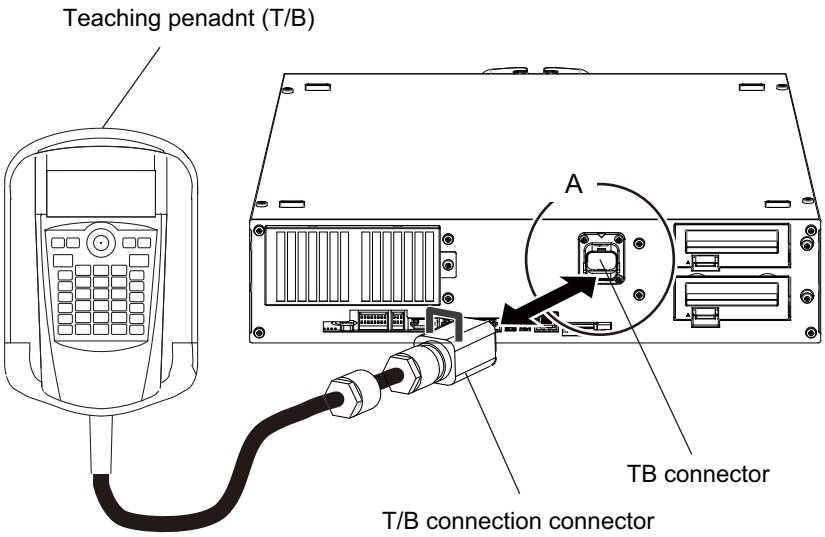
Class	Operation rights	Operation
Operation Note1)	○	Servo ON
	X	Servo OFF
	○	Program start. Starting also by operation of T/B other than the controller operation panel is possible.
	X	Program stop/cycle stop
	○	Slot initialization (program reset)
	X	Error reset
	○	Override change. Note this is always possible from the T/B.
	X	Override read
	○	Program No. change
	X	Program No./line No. read
Input/output signal Note2)	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 Note3)	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) To operate the robot on the <operation panel> of the T/B, the operating rights depend on the controller mode as shown below.
• "MANUAL": Enable the operation rights by pressing [TB ENABLE] of the T/B.
• "AUTOMATIC": The operating rights are automatically transferred to the T/B without pressing [TB ENABLE] of the T/B when the <operation panel> screen is displayed. (The T/B's status display lamp [ENABLE] blinks.)
Note2) While the <operation panel> screen is displayed on the T/B, operation using the [MONITOR] key is not possible.
Note3) When one device is being used for editing on-line, editing from other devices is not possible.

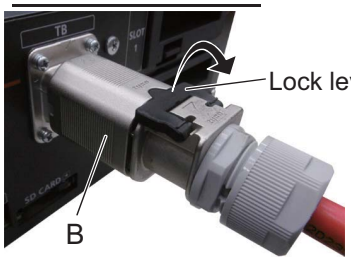
2.1.2 Handling the T/B

Installing and removing the T/B, with turning off the controller power. If T/B is installed and removed in the state of control source ON, emergency stop alarm will be occurred.
However, it allows you to remove the T/B from the controller without occurrence of alarm by pulling out the T/B connector within five seconds since the 3-position enable switch of the T/B being pulled up lightly (in the position 2) was released.

⚠ CAUTION Please do not pull the cable of T/B strongly or do not bend it too much.
It becomes the breaking of a wire of the cable and the cause of breakage of the connector. Please installing and removing so that stress does not start the cable with the connector itself.



Details of the A section



When removing the connector for T/B connection, use lock release (state which raised the lock lever to the up side), make the case of the B section slide to the front, and remove and pull up out the latch.

Fig.2-2:Installing and removing the T/B

(1) Installing the T/B

Explain the installation method of T/B below.

1) Check that the power supply of the robot controller is OFF.
2) Refer to Fig. 2-2, connects T/B connector to the robot controller. Use as the upper surface the lock lever, and push in until there is sound.

The installation of T/B is finished.

(2) Removing the T/B

Explain the removing method of T/B below.

*In MANUAL mode

1) Check that the power supply of the robot controller is OFF.
2) Refer to Fig. 2-2, raise up the lock lever in the connector upper part, and pull up the connector. Please install the dummy connector, if you use the robot, without connecting T/B.

The removing of T/B is finished.

*In AUTOMATIC mode

1) Pull up lightly (in the position 2) the 3-position enable switch of the T/B.
2) Pull out the T/B connector within five seconds while 1). Refer to Fig. 2-2, raise up the lock lever in the connector upper part, and pull up the connector.
   Please install the dummy connector, if you use the robot, without connecting T/B.

The removing of T/B is finished.

2.2 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 325, "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 297, "Oadl (Optimal Acceleration)", Page 266, "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 188, "Cmp Tool (Compliance 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 195, "ColChk (Col Check)" Refer to "COL" parameter in Page 484, "5 Functions set with parameters".
Interference avoidance function	This function is used with the CR800-R/Q series controller. The robot is moved while checking for interference between two or three robots using direct communication between the robot CPUs. Robot damage can be reduced by predicting interference between robots and stopping the movement during jog operation or automatic operation. When interference is predicted, the robot movement will stop. The robot can be programmed to generate an alarm or to restore operation.	Refer to Page 567, "5.24 Interference avoidance function".
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.
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 102, "(4) Continuous movement", Page 192, "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 124, "4.2.1 What is multitasking?", Page 342, "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 484, "5 Functions set with parameters".

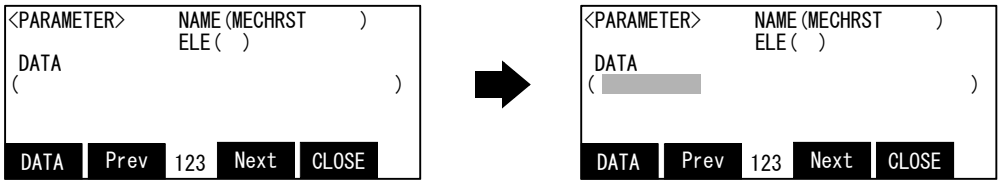
Function	Explanation	Explanation page
Additional axis control	With this function, it is possible to control up to two axes as additional axes of the robot. Since the positions of these additional axes are stored in the robot's teaching data as well, it is possible to perform completely synchronous control. In addition, arc interpolation while moving additional axes (travelling axes) is also possible.	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".
External device communication function	The following methods are available for communicating with the external devices For controlling the controller and for interlock within a program 1) Via input/output signals (CR800-R/Q: PLC link input/output: 8192/8192 max.) (CR800-D: Parallel input/output: 256/256 max.) 2)As a data link with an external device (*CR800-D only) 3)Communication via Ethernet ［Reference］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).	Refer to Page 380, "M_In/M_Inb/M_In8/M_Inw/M_In16", Page 393, "M_Out/M_Outb/M_Out8/M_Outw/M_Out16".
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 202, " Def Act (Define act)", Page 173, " 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 178, " 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 108, "4.1.2 Pallet operation", Page 216, " Def Plt (Define pallet)",Page 305, " 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.	Page 520, "5.8 About user-defined area", Page 409, "M_Uar", Page 410, "M_Uar32". Page 481, "Zone", Page 482, "Zone 2" Page 483, "Zone3"
JOINT movement range XYZ operation range Free plane limit	It is possible to restrict the robot movement range in the following three ways JOINT movement range: It is possible to restrict the movement range of each axis. XYZ operation range: It is possible to restrict the movement range using the robot's XYZ coordinate system. 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.	Refer to "MEJAR" and "MEPAR" parameter in Page 484, "5 Functions set with parameters" Refer to Page 526, "5.9 Free plane limit"

2.3 Robot type resetting

If the type information is lost by the ablation of the battery etc., the errors (H1600: Mechanism unsetting. etc.) occur, and the operation becomes impossible.
In this case, it can return to the status at factory shipping by the following type resetting operations.

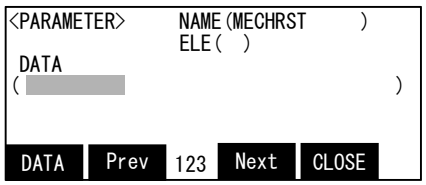
The machine reset operation can be performed from the teaching box or RT ToolBox3.
The method for resetting the robot type by teaching pendant is shown below.

1) Set "CLEAR" to the parameter MECHRST.

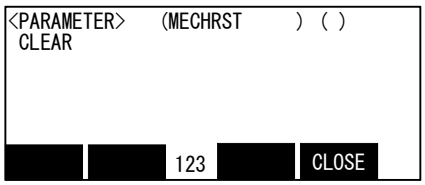


Enter the parameter name [M], [E], [C], [H], [R], [S], [T]
Input [EXE]

Current setting is displayed.



Setting the data [F1]



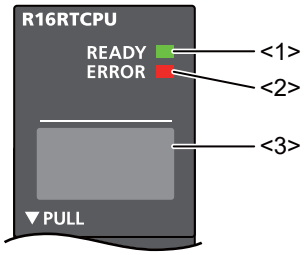
Delete the current data [CLEAR]
Enter the data [C], [L], [E], [A], [R]
Input [EXE]

2) Turn the controller power ON again.

⚠ CAUTION Because it returns to the status at factory shipping by this operation, so the parameter, the program, and various log data are eliminated.
Therefore, when the error (H1600) has not occurred, don't carry out this operation.
The deleted data can be restored using the restore function if batch backup data of RT ToolBox3 (option) is saved.

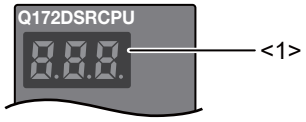
2.4 Robot CPU status indicator LEDs

2.4.1 R-type CPU unit (R16RTCPU) status indicators



No.	Name	Status	Description
<1>	READY LED (green)	OFF	While the power is off, during initialization immediately after power-on (approx. 30s), and when hardware error occurs
		Flashing	During initialization (while system software is starting up)
		ON	During normal operation (after system software has started up)
<2>	ERROR LED (red)	OFF	No error/hardware error
		ON/flashing	Error occurring
<3>	Dot-matrix display	H-1	When the power is on or immediately after reset If this is displayed for more than a few seconds, the product may be faulty.
		00 to 99	During hardware boot/OS startup
		i19 to i00	During system software startup
		E00 to E60	System error If the error does not clear even after a power reset, the product may be faulty.
		Rdy	Ready for operation
		Run	Program running
		Stp	Program paused
		AL H** ** L** ** C** **	Error occurring "AL" will flash three times first, then a 4-digit error number will flash. The error number will be split into two parts. The error levels "H", "L", or "C" and the upper two digits of the error number will flash first, followed by the lower two digits of the error number.

2.4.2 Q-type CPU unit (Q172DSRCPU) status indicators



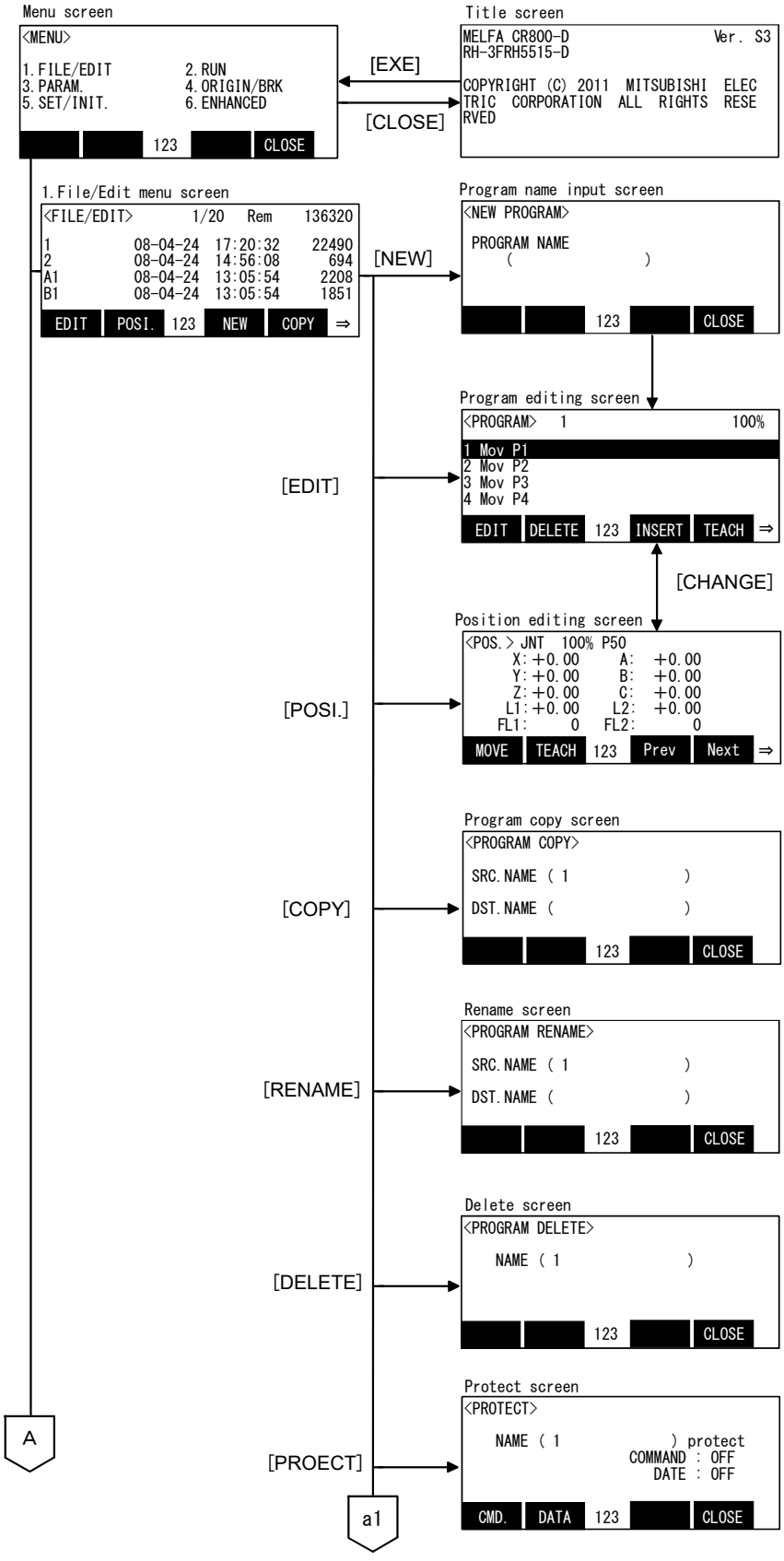
No.	Name	Status	Description
<1>	7-segment LED display	b00 to o99	Immediately after power-on or reset and during hardware boot/ OS initialization (approx. 10s). If this is displayed for one minute or more, the product may be faulty.
		qrw, qin, rin	Initializing PLC CPU and robot CPU If this is displayed for one minute or more, the product may be faulty.
		i24 to i00 8.8.8.	During system software startup If this is displayed for one minute or more, the product may be faulty.
		Rdy	Ready for operation
		Run	Program running
		Stp	Program paused
		AL H** ** L** ** C** **	Error occurring "AL" will flash three times first, then a 4-digit error number will flash. The error number will be split into two parts. The error levels "H", "L", or "C" and the upper two digits of the error number will flash first, followed by the lower two digits of the error number.
			The last dot flashing indicates normal operation
			Three solid dots indicate that the watchdog error has occurred. Hardware/software error If the error still occurs even after a power reset, the product may be faulty.

3 Explanation of operation methods

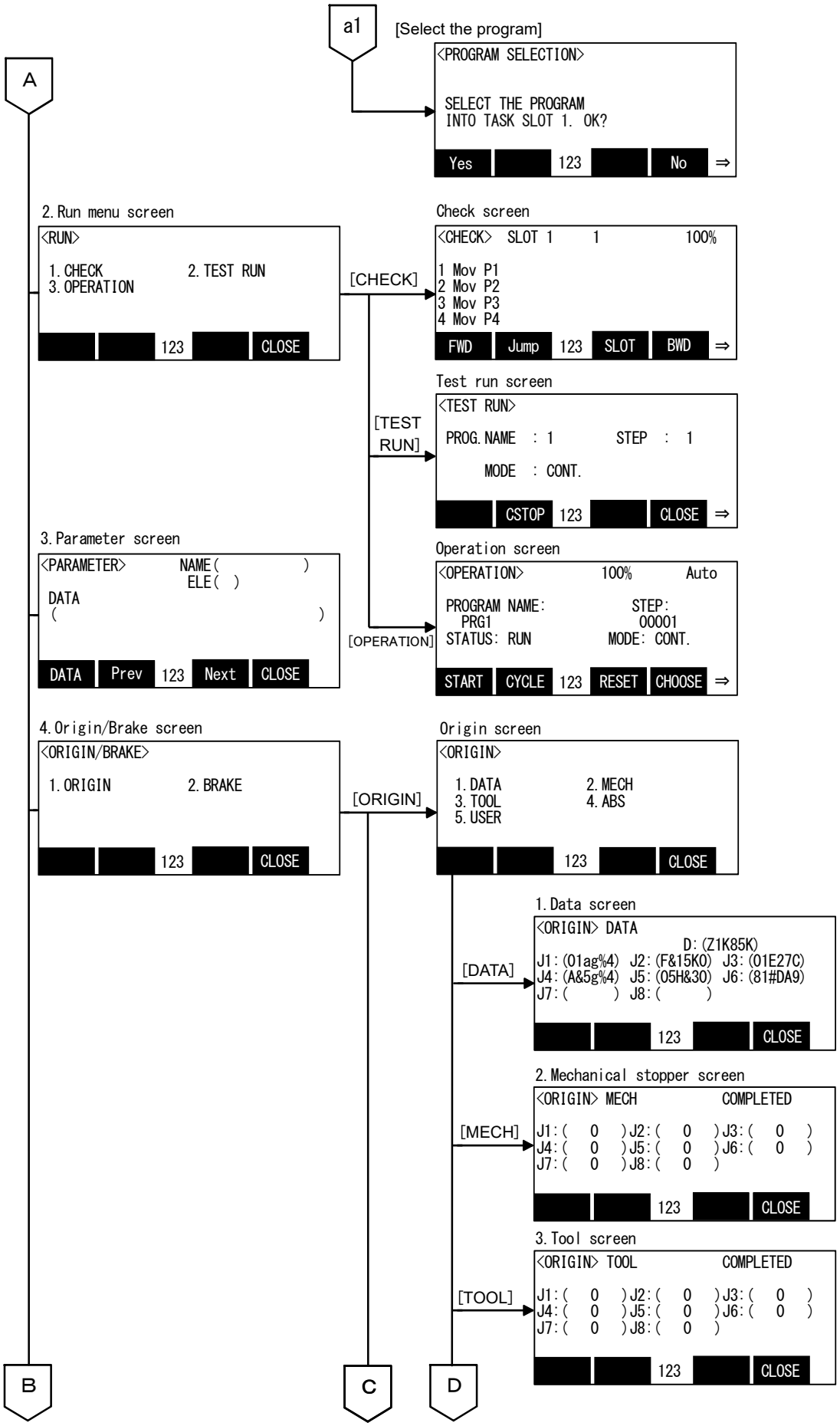
This chapter describes how to operate R32TB (optional).

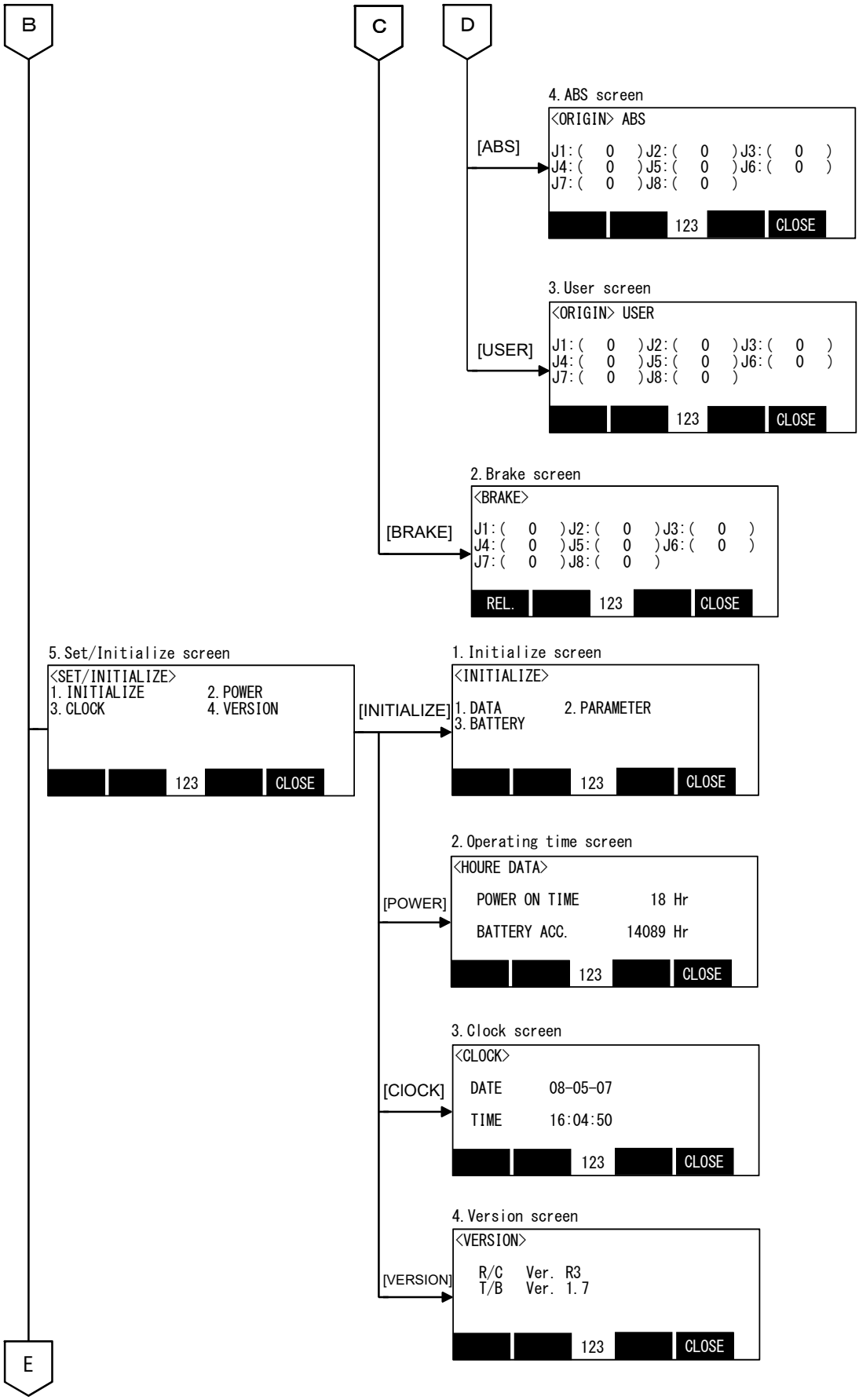
3.1 Operation of the teaching pendant menu screens

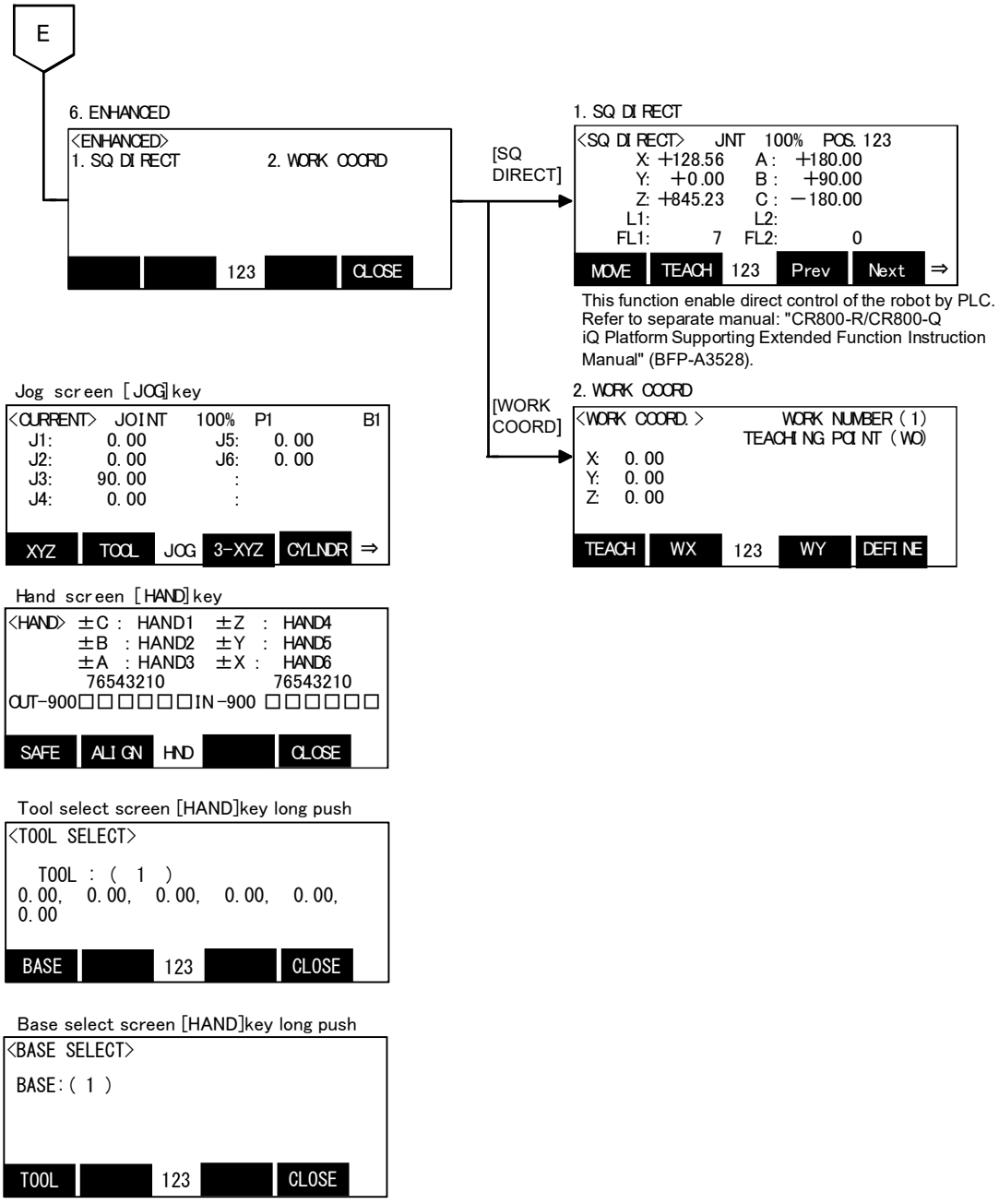
(1) Screen tree



Note 1) Refer to "separate instruction manual: CR800-R/CR800-Q, iQ Platform Supporting Extended Function Instruction Manual (BFP-A3528) for "1.SQ DIRECT" in "6.ENHANCED"


a1
[Select the program]
<PROGRAM SELECTION>
SELECT THE PROGRAM
INTO TASK SLOT 1. OK?
Yes
123
No
A
2. Run menu screen
<RUN>
1. CHECK
2. TEST RUN
3. OPERATION
123
CLOSE
[CHECK]
Check screen
<CHECK> SLOT 1 1 100%
1 Mov P1
2 Mov P2
3 Mov P3
4 Mov P4
FWD
Jump
123
SLOT
BWD
[TEST RUN]
Test run screen
<TEST RUN>
PROG. NAME : 1
STEP : 1
MODE : CONT.
CSTOP
123
CLOSE
3. Parameter screen
<PARAMETER>
NAME( )
ELE( )
DATA
( )
DATA
Prev
123
Next
CLOSE
[OPERATION]
Operation screen
<OPERATION> 100% Auto
PROGRAM NAME:
PRG1
STEP:
00001
STATUS: RUN
MODE: CONT.
START
CYCLE
123
RESET
CHOOSE
4. Origin/Brake screen
<ORIGIN/BRAKE>
1. ORIGIN
2. BRAKE
123
CLOSE
[ORIGIN]
Origin screen
<ORIGIN>
1. DATA
2. MECH
3. TOOL
4. ABS
5. USER
123
CLOSE
1. Data screen
[DATA]
<ORIGIN> DATA
D:(Z1K85K)
J1:(01ag%4) J2:(F&15K0) J3:(01E27C)
J4:(A&5g%4) J5:(05H&30) J6:(81#DA9)
J7:( ) J8:( )
123
CLOSE
2. Mechanical stopper screen
[MECH]
<ORIGIN> MECH COMPLETED
J1:(0)J2:(0)J3:(0)
J4:(0)J5:(0)J6:(0)
J7:(0)J8:(0)
123
CLOSE
3. Tool screen
[TOOL]
<ORIGIN> TOOL COMPLETED
J1:(0)J2:(0)J3:(0)
J4:(0)J5:(0)J6:(0)
J7:(0)J8:(0)
123
CLOSE
B
C
D


B
C
D
4. ABS screen
<ORIGIN> ABS
[ABS]
J1:(0)J2:(0)J3:(0)
J4:(0)J5:(0)J6:(0)
J7:(0)J8:(0)
123
CLOSE
3. User screen
<ORIGIN> USER
[USER]
J1:(0)J2:(0)J3:(0)
J4:(0)J5:(0)J6:(0)
J7:(0)J8:(0)
123
CLOSE
2. Brake screen
<BRAKE>
[BRAKE]
J1:(0)J2:(0)J3:(0)
J4:(0)J5:(0)J6:(0)
J7:(0)J8:(0)
REL.
123
CLOSE
5. Set/Initialize screen
<SET/INITIALIZE>
1. INITIALIZE 2. POWER
3. CLOCK 4. VERSION
123
CLOSE
[INITIALIZE]
1. Initialize screen
<INITIALIZE>
1. DATA 2. PARAMETER
3. BATTERY
123
CLOSE
2. Operating time screen
<HOURE DATA>
[POWER]
POWER ON TIME 18 Hr
BATTERY ACC. 14089 Hr
123
CLOSE
3. Clock screen
<CLOCK>
[CIOCK]
DATE 08-05-07
TIME 16:04:50
123
CLOSE
4. Version screen
<VERSION>
[VERSION]
R/C Ver. R3
T/B Ver. 1.7
123
CLOSE
E


E
6. ENHANCED
<ENHANCED>
1. SQ DIRECT
2. WORK COORD
123
CLOSE
[SQ DIRECT]
1. SQ DIRECT
<SQ DIRECT> JNT 100% POS. 123
X: +128.56 A : +180.00
Y: +0.00 B : +90.00
Z: +845.23 C : −180.00
L1: L2:
FL1: 7 FL2: 0
MOVE TEACH 123 Prev Next ⇒
This function enable direct control of the robot by PLC.
Refer to separate manual: "CR800-R/CR800-Q
iQ Platform Supporting Extended Function Instruction
Manual" (BFP-A3528).
Jog screen [JOG]key
<CURRENT> JOINT 100% P1 B1
J1: 0.00 J5: 0.00
J2: 0.00 J6: 0.00
J3: 90.00 :
J4: 0.00 :
XYZ TOOL JOG 3-XYZ CYLNDR ⇒
[WORK COORD]
2. WORK COORD
<WORK COORD.> WORK NUMBER (1)
TEACHING POINT (WO)
X: 0.00
Y: 0.00
Z: 0.00
TEACH WX 123 WY DEFINE
Hand screen [HAND]key
<HAND> ±C : HAND1 ±Z : HAND4
±B : HAND2 ±Y : HAND5
±A : HAND3 ±X : HAND6
76543210 76543210
OUT-900□□□□□□□IN-900 □□□□□□□
SAFE ALIGN HND CLOSE
Tool select screen [HAND]key long push
<TOOL SELECT>
TOOL : (1)
0.00, 0.00, 0.00, 0.00, 0.00,
0.00
BASE 123 CLOSE
Base select screen [HAND]key long push
<BASE SELECT>
BASE:(1)
TOOL 123 CLOSE

(2) Input of the number/character

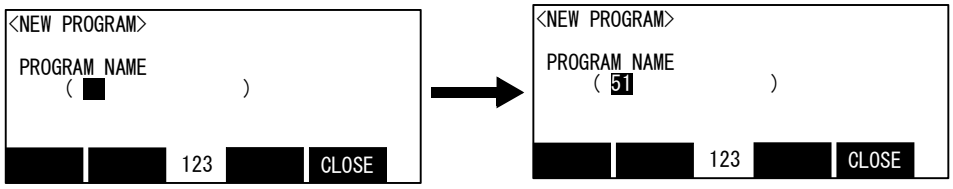
Each time the [CHARACTER] key is pressed, the number input mode and the character input mode change. The current input mode is displayed in the center under the screen, and the display of "123" shows that the number input mode and "ABC" is the character input mode.

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]

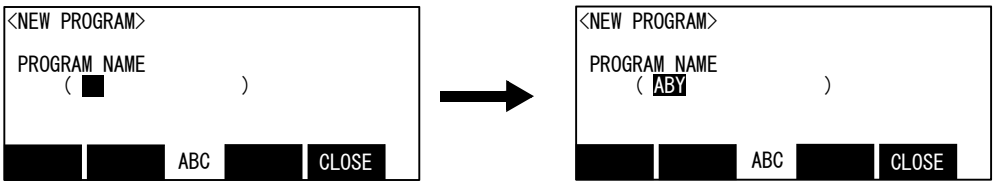
2) Input the character

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..... @→ = → + → - → * → / → < → >

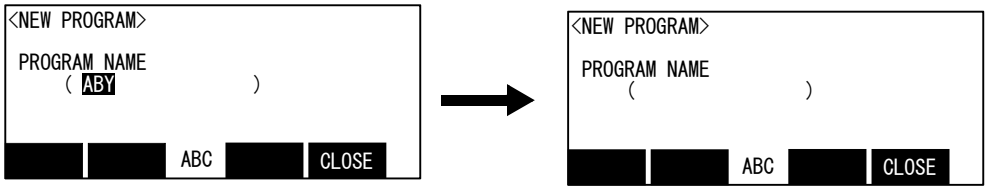
c) [, %] key..... , → % → # → $ → ! → & → _ → .

3) Delet 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

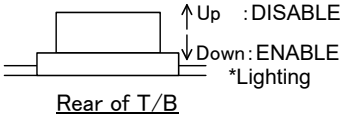
A menu can be selected with either of the following two methods.

*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. FILE/EDIT") from the <menu> screen with each method is shown below.

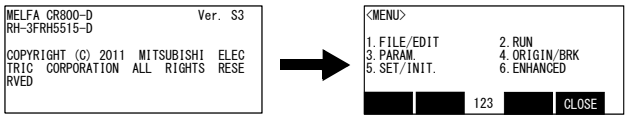
T/B



Rear of T/B

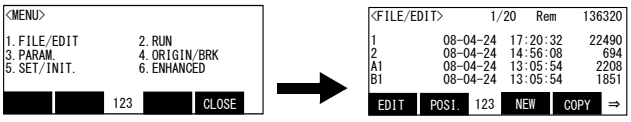
1) Set the controller mode to "MANUAL". Press the [ENABLE] switch of the T/B to enable the T/B. (The switch and the ENABLE LED light up.)

Display the MENU screen from the title screen.



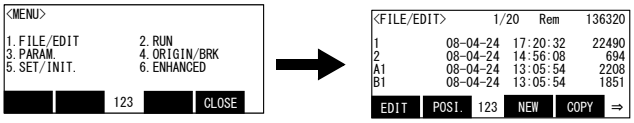
2) Press one of the keys (example, [EXE] key) while the <TITLE> screen is displayed. The <MENU> screen will appear.

*Press the number key method



Press the [1] key. The <FILE/EDIT> screen will appear.

*Use the arrow key method



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

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

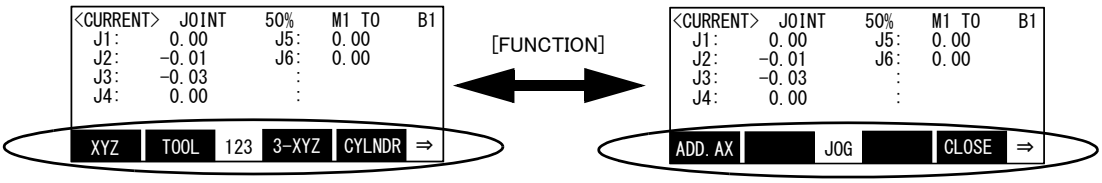
The same operations can be used on the other menu screens.

Using the T/B

Set the controller mode to "MANUAL". When the controller is set to a mode other than MANUAL mode, only specific operations (current position display on the JOG screen, changing of override, monitoring) can be performed.

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.



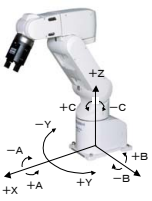
3.2 Jog Feed (Overview)

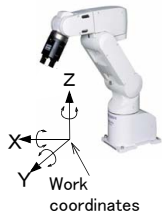
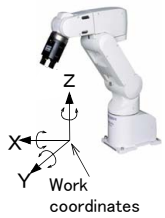
Jog feed refers to a mode of operation in which the position of the robot is adjusted manually. Here, an overview of this operation is given, using the vertical multi-joint type robot as an example. The axes are configured differently depending on the type of robot. For each individual type of robot, 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 	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.	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. The additional axis keys [J1] and [J2] correspond to axes J7 and J8, respectively.
TOOL jog 	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.	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. 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. In the case of a horizontal multi-joint type robot, the upward direction from the mechanical interface plane is +Z.
XYZ jog 	Perform steps 1) to 3) above. 4) Press the function key to change to the XYZ jog mode. 5) Press the key corresponding to each of the axes from X,Y,Z,A,B,C.	The axes are adjusted linearly with respect to the robot coordinate system. 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.
3-axis XYZ jog 	Perform steps 1) to 3) above. 4) Press the function key twice to switch to the 3-axis XYZ jog mode. 5) Press the key corresponding to each of the axes from X,Y,Z,J4,J5,J6.	The axes are adjusted linearly with respect to the robot coordinate system. 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.
CYLNDER jog 	Perform steps 1) to 3) above. 4) Press the function key twice to switch to the CYLNDER jog mode. 5) Press the key corresponding to each of the axes from X,Y,Z,A,B,C.	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. 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.

Type	Operation	Explanation
WORK jog (WORK jog mode)  	Perform steps 1) to 3) above. 4) Press the function key to change to the WORK jog mode. 5) Press the key corresponding to each of the axes from X,Y,Z,A,B,C.	It is necessary to set "0 (Work jog mode)" in the parameter WKnJOGMD (n = 1 to 8) in advance to perform this jog operation. The axes are adjusted linearly with respect to the work coordinate system. The posture rotates around the X, Y, and Z axes of the work 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. Notes) Work coordinate system: Set up beforehand.(eight kinds setting is available) If the work coordinate system is not set up, will move by the XYZ jog. Refer to separate manual: "ROBOT ARM SETUP & MAINTENANCE"
WORK jog (Ex-T jog mode)  	Perform steps 1) to 3) above. 4) Press the function key to change to the WORK jog mode. 5) Press the key corresponding to each of the axes from X,Y,Z,A,B,C.	It is necessary to set "1 (Ex-T jog mode)" in the parameter WKnJOGMD (n = 1 to 8) in advance to perform this jog operation. The axes are adjusted linearly with respect to the work coordinate system. The posture rotates around the X, Y, and Z axes of the work coordinate system by pressing the A, B, and C keys, without changing the actual position of the hand tip. By pressing the A, B, and C keys, the control point can be rotated around the X, Y, and Z axes of the same work coordinates system while changing the actual position of the hand tip. It is necessary to specify the tool length in advance using the MEXTL parameter. Notes) Work coordinate system: Set up beforehand.(eight kinds setting is available) If the work coordinate system is not set up, will move by the XYZ jog. Refer to separate manual: "ROBOT ARM SETUP & MAINTENANCE" When the work coordinates system is not specified, the XYZ jog operation is performed by pressing the X, Y, and Z keys. The rotating operation around the robot origin coordinates system is performed by pressing the A, B, and C keys.

If the robot's control point comes near a singular point during the operation of TOOL jog, XYZ jog, CYLINDER jog or WORK 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 484, "5 Functions set with parameters".) Please refer to Page 543, "5.17 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↑] / [OVRD↓] key. The following types of jog feed speed are available.

[OVRD↑] key ---------- [OVRD↓] key

LOW	HIGH	3%	5%	10%	30%	50%	70%	100%

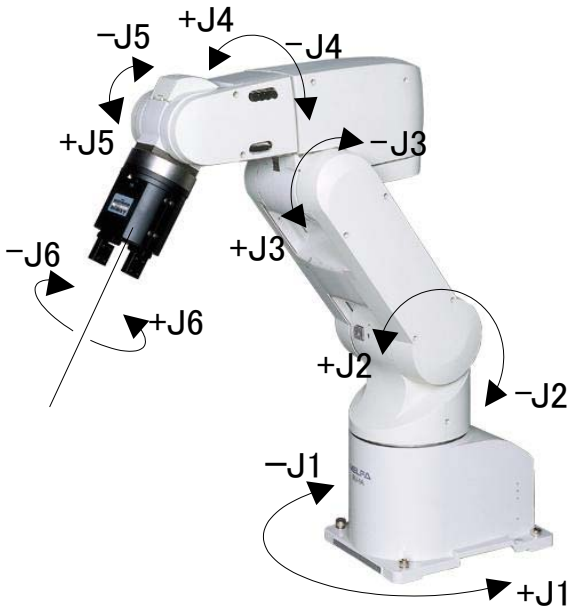
LOW and HIGH are fixed-dimension feed. In fixed-dimension feed, the robot moves a fixed amount every time the key is pressed. The amount of movement depends on the individual robot.

Table 3-2: An example of fixed-dimension

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

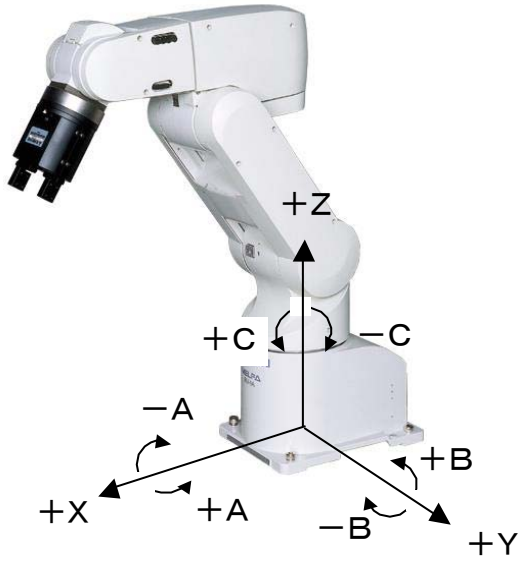
3.2.3 JOINT jog

Adjusts the coordinates of each axis independently in angle units.



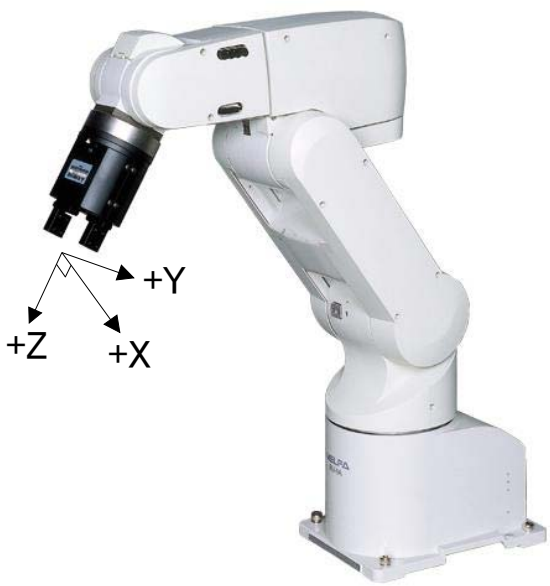
3.2.4 XYZ jog

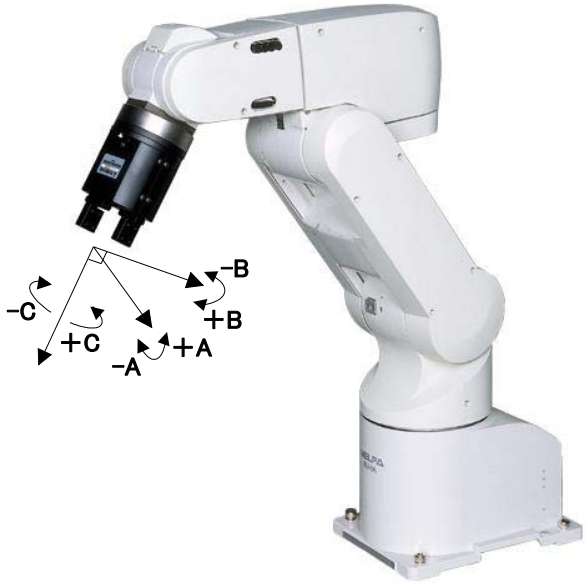
Adjusts the axis coordinates along the direction of the robot coordinate system.
The X, Y, and Z axis coordinates are adjusted in mm units. The A, B, and C axis coordinates are adjusted in angle units.



3.2.5 TOOL jog

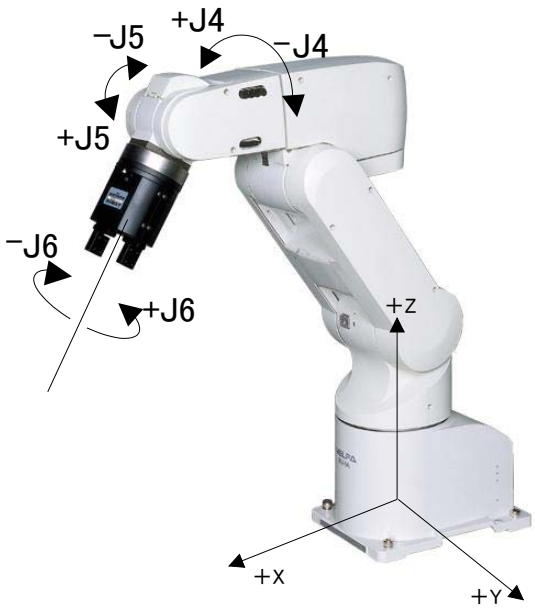
Adjusts the coordinates of each axes along the direction of the hand tip.
The X, Y, and Z axis coordinates are adjusted in mm units. The A, B, and C axis coordinates are adjusted in angle units.





3.2.6 3-axis XYZ jog

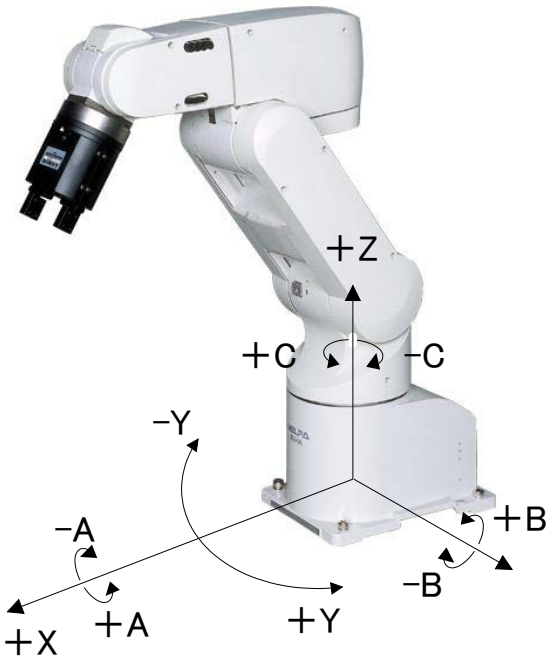
Adjusts the X, Y, and Z axis coordinates along the direction of the robot coordinate system in the same way as in XYZ jog feed. The J4, J5 and J6 axes perform the same operation as in JOINT jog feed, but the posture changes in order to maintain the position of the control point (X, Y and Z values).
The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

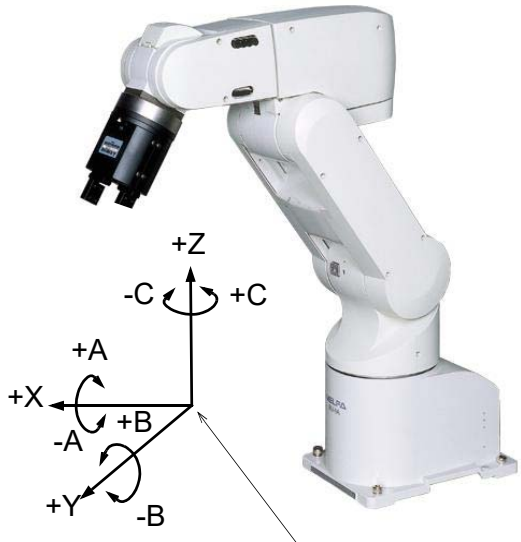
The X and Z axis coordinates are adjusted in mm units. The Y, A, B, and C axis coordinates are adjusted in angle units.



3.2.8 WORK jog

Adjusts the axis coordinates along the direction of the work 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.

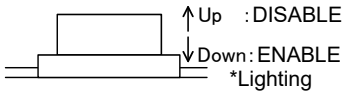


Work coordinate system:
Coordinate system squared with the work, the working table, etc.

3.2.9 Switching Tool Conversion Data

Set the tool conversion data you want to use in the MEXTL1 to 16 parameters, and select the number of the tool you want to use according to the following operation.

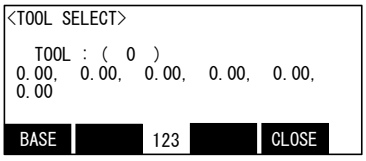
1) Set the controller mode to "MANUAL". Press the [ENABLE] switch of the T/B to enable the T/B. (The switch and the ENABLE LED light up.)

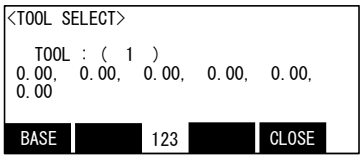


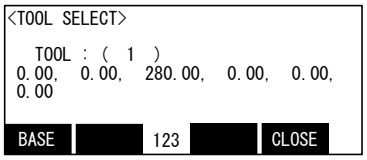
Rear of T/B

2) Long press the [HAND] key, and display the <TOOL SELECT> screen.
3) If the number key to wish is pressed and the [EXE] key is pressed, tool conversion data will change. MEXTL1-16 of the parameter corresponds to 1-16 of the number.

Display of the tool change screen [HAND]

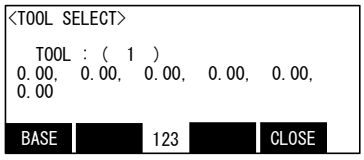






The change of tool conversion data [1] - [16] [EXE]

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



Completed [F4]

5) The current tool number (T1-T16) 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 conversion data (MEXTL1 to 16 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 conversion data. Exercise caution as the robot moves to an unexpected direction if the tool conversion data during teaching does not match the tool number during operation.

⚠CAUTION To move the robot while switching tool conversion data during the step operation of the program, exercise caution as the robot moves to an unexpected direction if the tool conversion 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 SELECT> screen, <JOG> screen, or with the M_Tool variable.

Related Information

MEXTL, MEXTL1, MEXTL2, MEXTL3, MEXTL4 ..... MEXTL16 parameters
Tool instruction, M_Tool variable
The MEXTL parameter holds tool conversion data at that point. When using the MEXTL1 to 16 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.

3.2.10 Changing the world coordinate (specifies the base coordinate number)

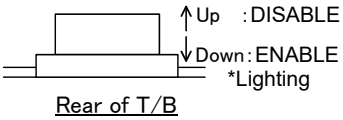
The world coordinate which is the standard of position control of the robot can be changed easily by T/B operation
In use of the base conversion function (Base instruction), this function is convenient for teaching operations. Set the base coordinate system to specify as parameter WK1 CORD-WK8CORD previously. (Refer to "Work jog operation" of "ROBOT ARM SETUP & MAINTENANCE" of separate volume. Refer to Page 484, "5.1 Movement parameter" of this volume for details of the parameter MK1CODE - MK8CODE also.)

⚠CAUTION

When the world coordinate is changed by this function, although the robot does not move, the current coordinate value will change.
Confirm that the relation of the position in the program to teach corresponding to the Base instruction and the base coordinate number which you are using now is right.
Failure to confirm this could lead to personal injuries or damage if you teach by the wrong base coordinate number, because the robot does the unexpected motion at program execution.
Make related the name of the position variable corresponding to the base coordinate number, and please manage rightly.

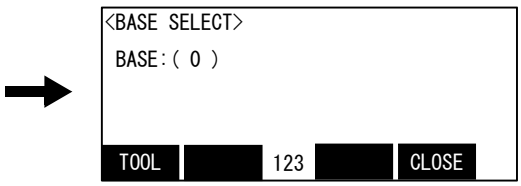
Operating procedure is shown below.

1) Set the controller mode to "MANUAL". Press the [ENABLE] switch of the T/B to enable the T/B. (The switch and the ENABLE LED light up.)

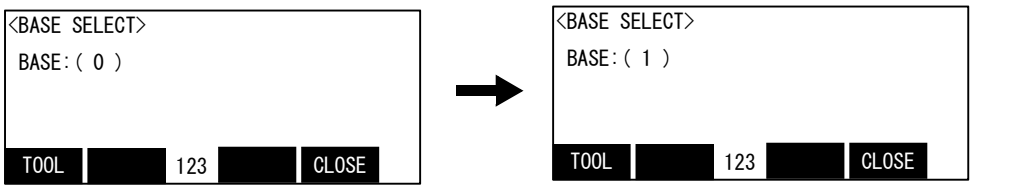


2) Long press the [HAND] key, and display the <BASE SELECT> screen.
If the <TOOL SELECT> screen is displayed, press the function key [F1] corresponding to the "BASE" under the screen.

Display of the <BASE SELECT> screen :
Long press the [HAND] key



3) If the base coordinate number to wish is inputted and the [EXE] key is pressed, the world coordinate will change.
1 to 8 : Base coordinate number (correspond to parameter WK1CORD - WK8CORD)
0: Return to condition at shipment. (Condition without base conversion)



(Example : Change to the base coordinate number 1.)

The setup of the world coordinate by the base coordinate number 「0」,「1」to「8」,「EXE」

4) Press the function key corresponding to "CLOSE" and finish.



Completed 「F4」

5) The current tool number (B1-B8) is displayed on the upper right of the jog screen.

3.2.11 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	Define whether the impact detection function can/cannot be used, and whether it is enabled/disabled immediately after power ON. Element 1: The impact detection function can (1)/cannot (0) be used. Element 2: It is enabled (1)/disabled (0) as the initial state during operation. Element 3: Enable (1)/disable (0)/NOERR mode (2) during jog operation 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. The specification depends on the settings for jog operation (element 3) in cases other than program operation (including position jump and step feed).	RH-3FRH/ 6FRH/12FRH/ 20FRH series: 1,0,1 RV-FR series: 0,0,1 RH-3FRHR series: 1,1,1
Detection level during jog operation	COLLVLJG	Integer 8	Set the detection level (sensitivity) during jog operation (including pause status) for each joint axis. Unit: % 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. Setting range: 1 to 500 (%)	The setting varies depending on the model.
Hand condition	HNDDAT0	Real value 7	Set the initial condition of the hand. (Specify with the tool coordinate system.) 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) Unit: Kg, mm	The setting varies depending on the model.
Workpiece condition	WRKDAT0	Real value 7	Set the initial condition of the workpiece. (Specify with the tool coordinate system.) Immediately after power ON, this setting value is used during jog operation. (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

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 COLLVLJG 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 61, "3.11 Operation to Temporarily Reset an Error that Cannot Be Canceled".

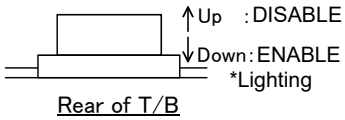
Relationship with impact detection for automatic operation

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

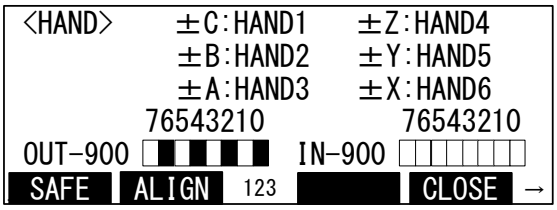
The open/close operation of the hands attached to on the robot is explained below.
The a maximum of six hands are controllable. The hand 6, 5, 4, 3, 2, and 1 are assigned to each key of X, Y, Z, A, B, and C axis. To open the hand press each the key of "+", to close the hand press each the key of "-".

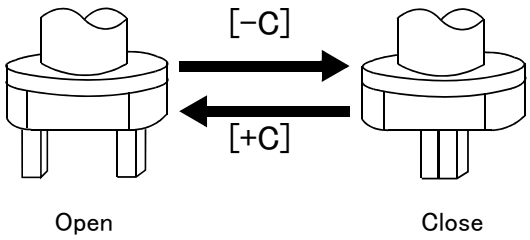
1) Set the controller mode to "MANUAL". Press the [ENABLE] switch of the T/B to enable the T/B. (The switch and the ENABLE LED light up.)



2) Press the [HAND] key and display the <HAND> screen. The opening and closing condition of the hand is shown in OUT-900, and the ON/OFF condition of the hand check input signal is shown in IN-900. To open the hand1 press the key of [+C], to close the hand1 press the key of [-C].
The other hands can be operated in the same way by the key of X, Y, Z, A, and B axis.

Display the <HAND> screen: [HAND] key





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 to 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	

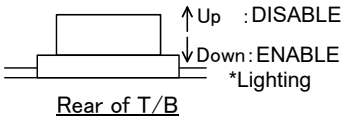
IN-900 to IN-907	7	6	5	4	3	2	1	0
Input signal	907	906	905	904	903	902	901	900

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 530, "5.12 About the hand type" and Page 531, "5.13 About default hand status".

3.4 Returning to the Safe Point

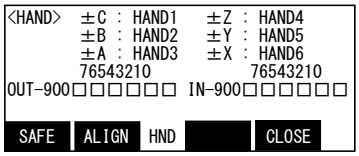
The robot can be moved to the safe point specified by the JSAFE parameter.
The safe point return procedure is as follows:

1) Set the controller mode to "MANUAL". Press the [ENABLE] switch of the T/B to enable the T/B. (The switch and the ENABLE LED light up.)

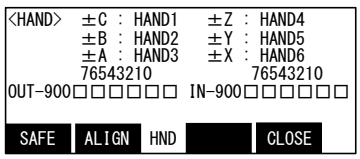


2) Press down the enabling switch (3 position switch), press the [SERVO] key and carry out servo-on.
3) Press the "HAND" key and display the <hand> screen.

Hand screen [HAND]



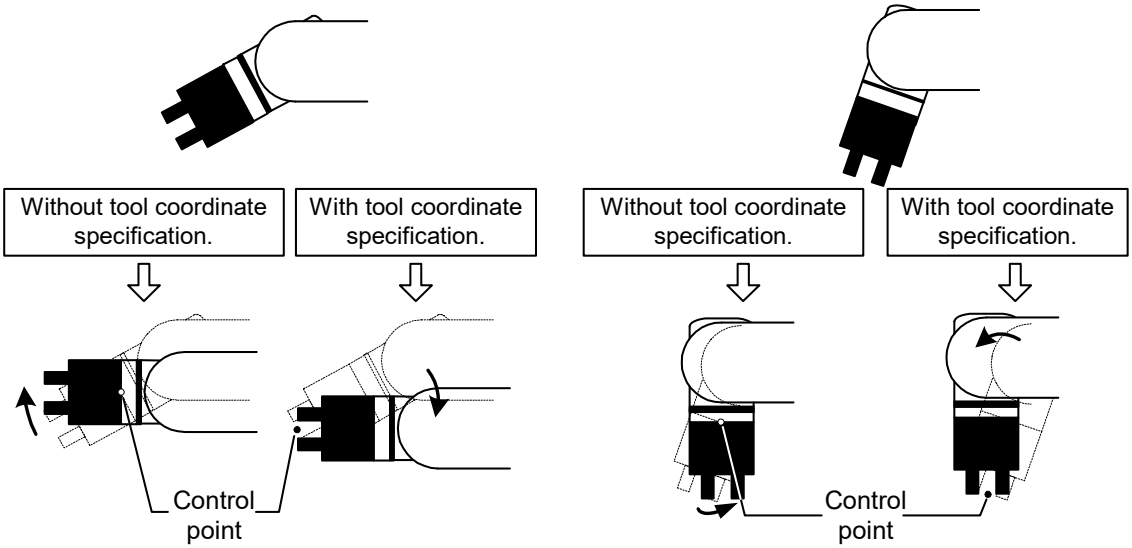
4) Pressing the function key currently assigned to "SAFE" is kept with the enabling switch (3 position switch) pressed down. While keeping pushing, the robot does safe point return movement and [START] LED of the controller unit turns on during movement.
If either is detached in the middle of movement, the robot will stop.



Execution of safe point return "SAFE"

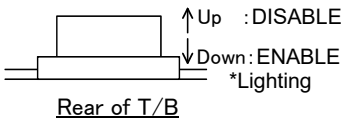
3.5 Aligning the Hand

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.



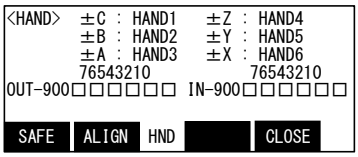
If the tool coordinates are specified by the Tool command or parameters, the hand is aligned at the specified tool coordinates. If the tool coordinates are not specified, the hand is aligned at the center of the mechanical interface. The above illustration shows an example of a small vertical robot. [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 517, "5.6 Standard Tool Coordinates".
The hand alignment procedure is as follows:

1) Set the controller mode to "MANUAL". Press the [ENABLE] switch of the T/B to enable the T/B. (The switch and the ENABLE LED light up.)

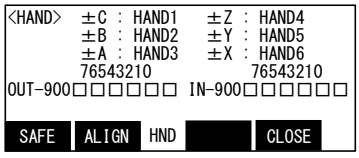


2) Press down the enabling switch (3 position switch), press the [SERVO] key and carry out servo-on.
3) Press the "HAND" key and display the <hand> screen.

Hand screen [HAND]



4) Pressing the function key currently assigned to "ALIGN" 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.

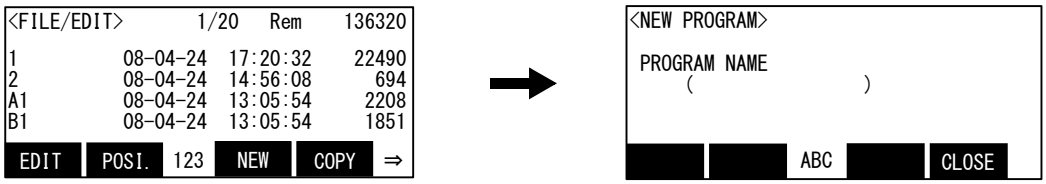
3.6 Programming

MELFA-BASIC VI 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. Refer to Page 170, "4.12 Detailed explanation of command words" in this manual for details on the MELFA-BASIC VI commands and description methods.

3.6.1 Creating a program

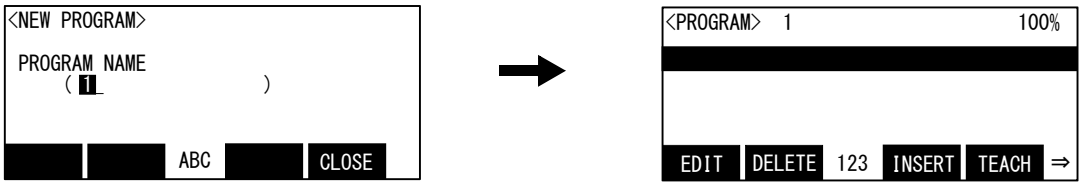
(1) Opening the program edit screen

1) Select "1. FILE/EDIT" screen on the <MENU> screen.
2) Press the function key corresponding to "NEW." Display the program name input screen.



Select the function [F3]

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



Inputs "1" of program name [1] [EXE]

Edit of the constantly-executed program (ALWAYS attribute)

The program set as the constantly-executed (ALWAYS) attribute with the SLTn parameter once cancels the attribute of the constantly executed, and edits it. Since the program of the constantly-executed attribute is always executed, edit of it is impossible. Change ALWAYS into START with the SLTn parameter, re-turn on the power supply of the controller, and stop the constantly executed.

Open the existing program.

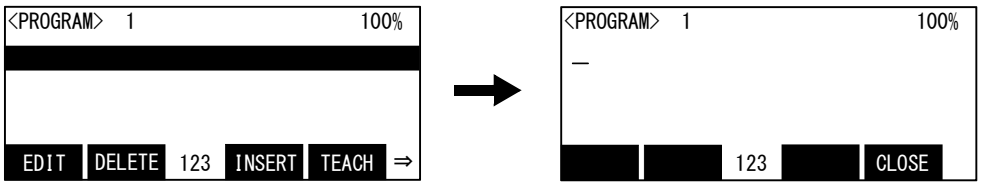
Except how to input the existing program name as mentioned above, it is also possible to press the arrow key on <FILE/EDIT> screen, and to press the function key currently assigned to "EDIT" in the condition that the program is chosen.

(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

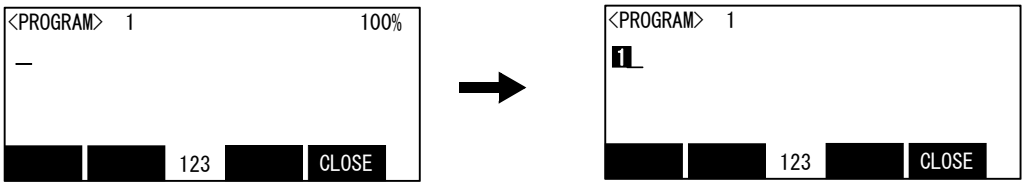
1) Press the function key ([F3]) corresponding to "INSERT" in the command edit screen.



Step insertion [F3]

2) Input of step number "1."

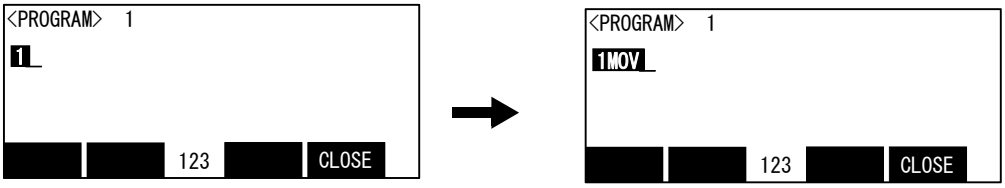
Press the [CHARACTER] key, set it in the number input mode and press the [1] key.
The space between step number and command is omissible.



Step number input [1]

3) Input "Mov."

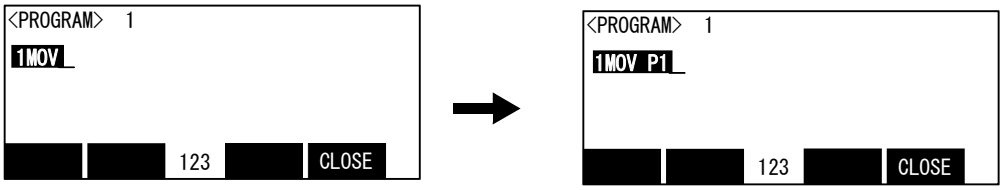
Press the [CHARACTER] key, set it in the character input mode
Press the [MNO] ("M"),[→], [MNO]("O") 3 times, and [TUV] ("V") 3 times in order.



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

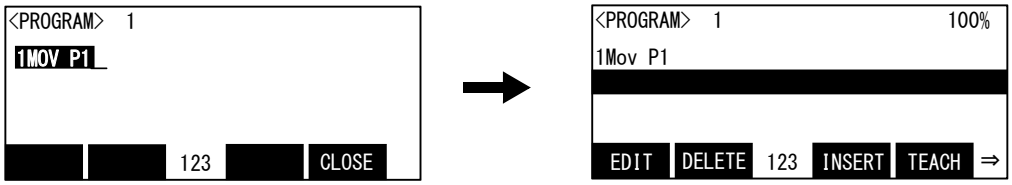
4) Input "P1."

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.



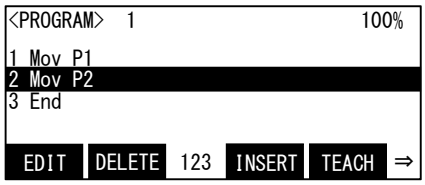
Input "P1" [SP] [PQRS] [CHARACTER] [1]

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 JUNP 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.

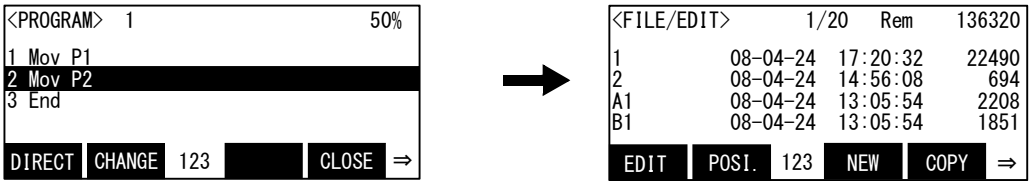
The capital letter and the small letter are changed automatically.

Display the reserved word and the variable name in MELFA BASIC VI 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 "CLOSE" 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.



Save & exit of the program [F4]

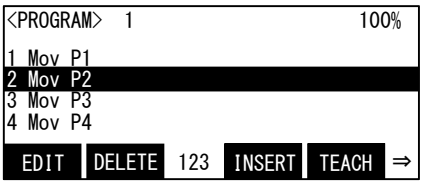
Precautions when saving programs

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 36, "3.6.1 Creating a program" in "(1)Opening the program edit screen", and open the program edit screen.

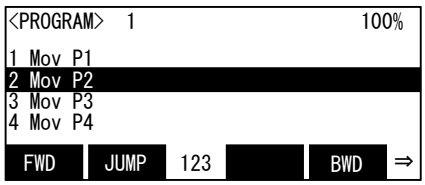
An example, change˝5 Mov P5" to "5 Mvs P5".



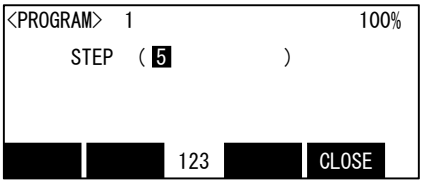
1) Display the step 5

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.



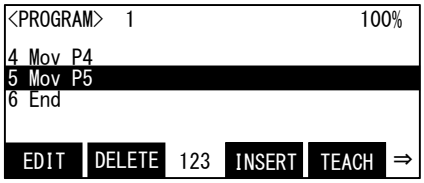
Call the step 5 [F2]



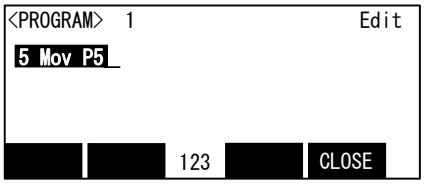
Call the step 5 [5] [EXE]

2) Correction of the instruction word.

Press the function key corresponding to "EDIT".

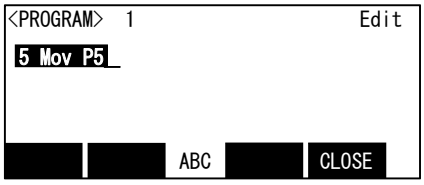


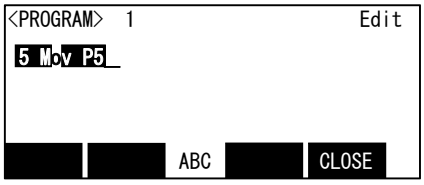
Correct the command [F1]



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

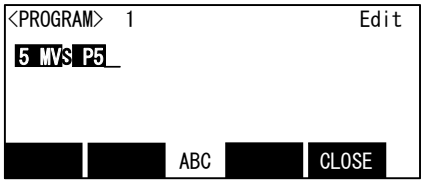
Press the [CLEAR] key twice and delete "ov". Leave "M". Press the [TUV] key 3 times (input "v"), the [→] key, the [PQRS] key 4 times (input "s"). Then, 5 step is "Mvs P5". Press the [EXE] key, and register step 5.

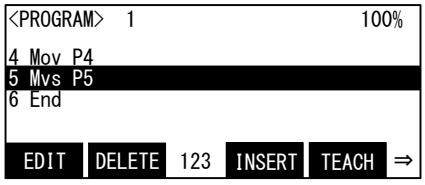




[TUV] [TUV] [TUV] [→]

Correct the command [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 "CLOSE" 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 "CLOSE" 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

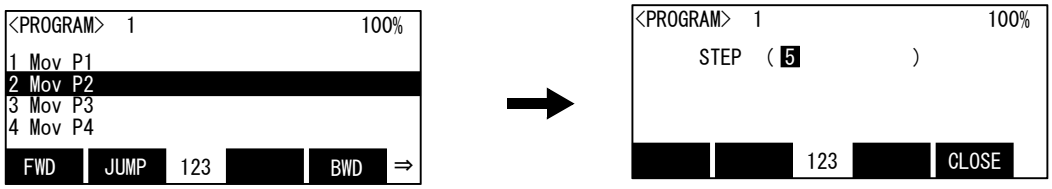
Call the step which is using the position variable to teach.

The operating procedure in the case of teaching the current position to the below to the position variable P5 of step 5 "Mvs 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.

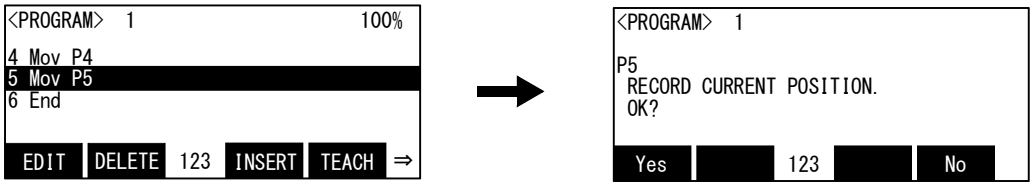


Call the step 5 [F2]

Call the step 5 [5] [EXE]

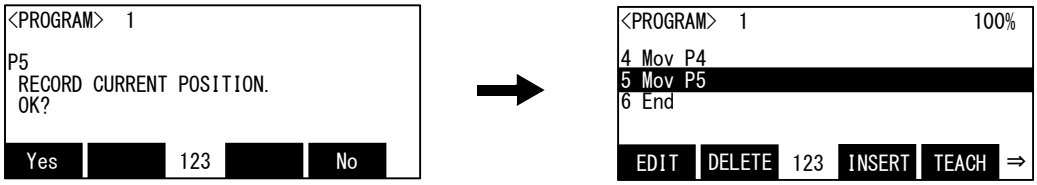
2) Teaching of the current position

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.

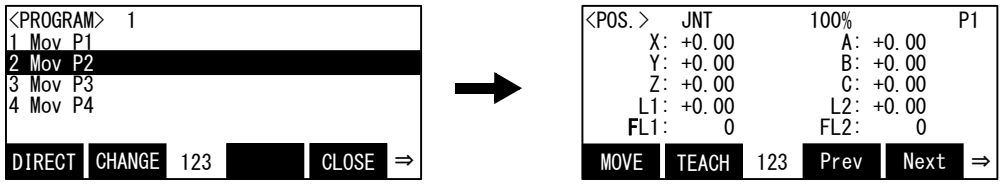
And, as shown in "Mov p1+P2", the position variable of the capital letter and the small letter is intermingled, the position variable of the capital letter is target. (The software version of T/B is 1.3 or later) It is the following page if it teaches other variables. Refer to "(b) Teaching in the position edit screen" as follows.

(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.

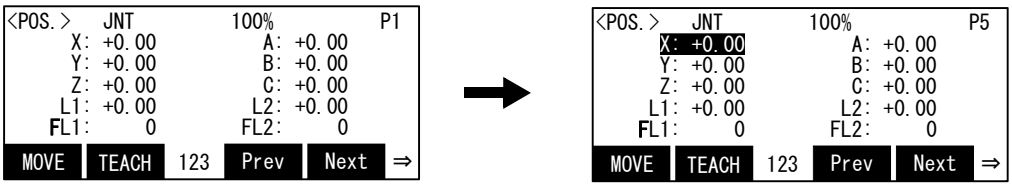
1) Teaching in the position edit screen

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



Display the current position [F2]

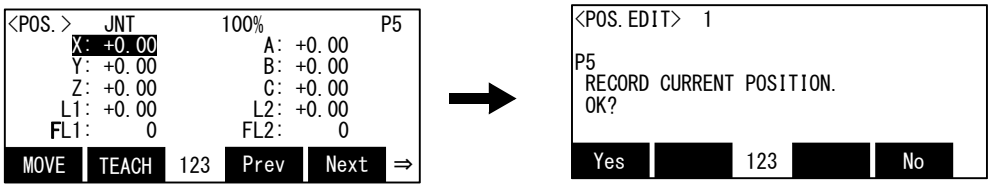
2) Press the function key corresponding to "Prev" and "Next", and call "P5".



Call the position 5 [F3] [F4]

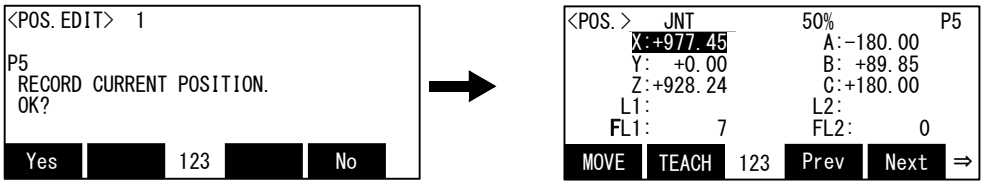
3) Teaching of the current position

Press the function key corresponding to "TEACH" ([F2]), 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.



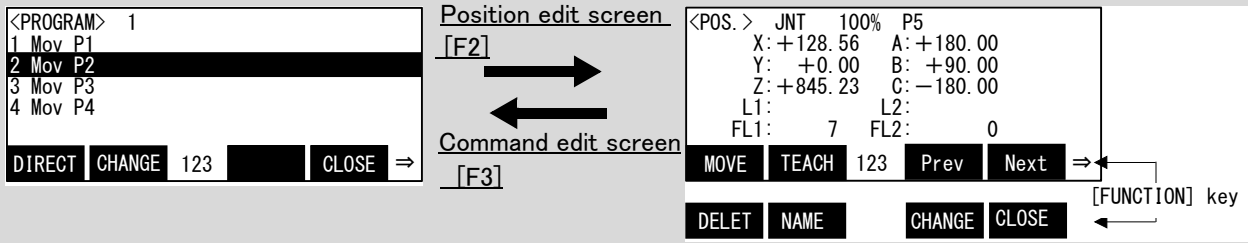
Register the current position [F1]

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 "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.

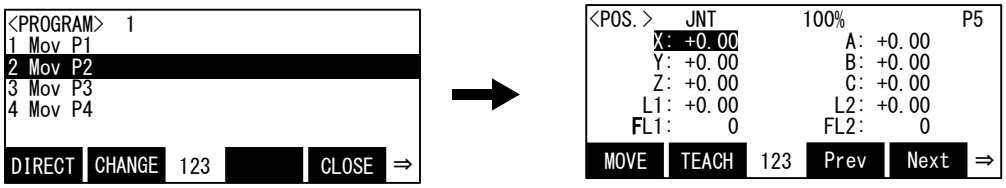
(6) Deletion of the position variable

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.

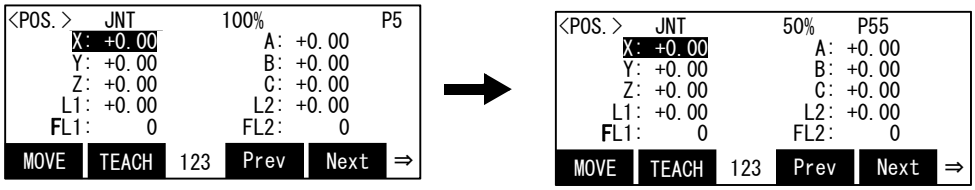
Press the function key corresponding to "CHANGE", and display the position edit screen.



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.

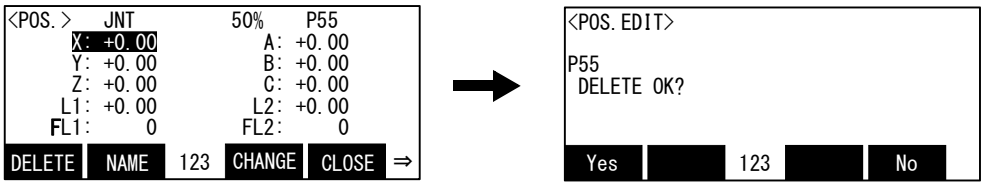


Call the position 5 [F3] [F4]

3) Deletion of the position variable

Press the function key corresponding to "DELETE", then the confirmation screen is displayed.

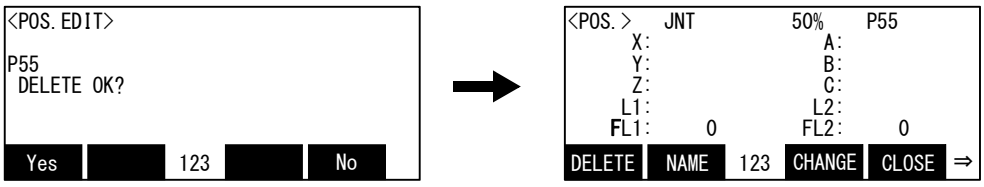
(When "DELETE" is not displayed, it is displayed that the [FUNCTION] key is pressed).



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)

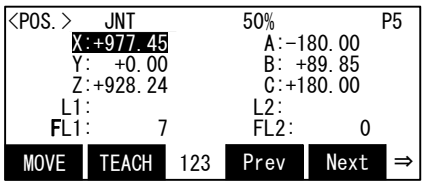
Move the robot to the registered position data place.
The robot can be moved with the "joint mode" or "XYZ mode" method.
Perform a servo ON operation while lightly holding the deadman switch before moving positions.

Table 3-4: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, CYLNDER 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.



Display the position edit screen [F1]

⚠ 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.

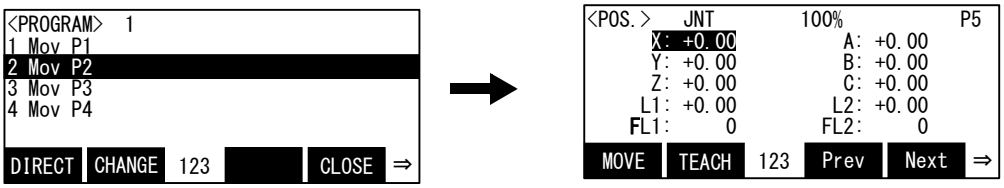
Reference) Position data as an amount of relative displacement

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.

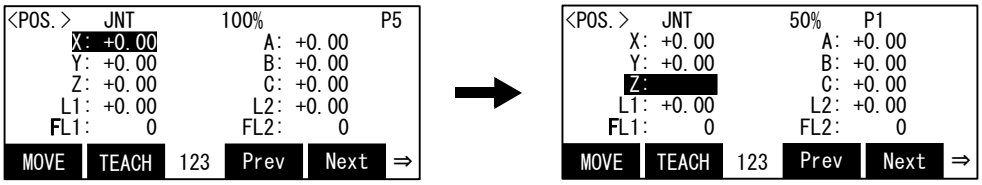
Press the function key corresponding to "CHANGE", and display the position edit screen.



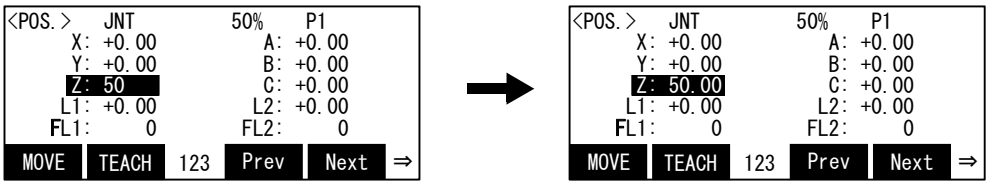
Display the position edit screen [F2]

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]

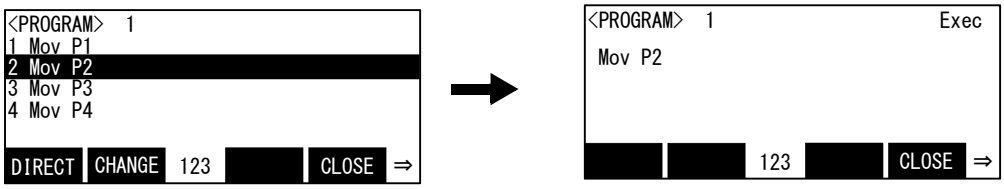


Input 50 [5] [0] [EXE]

(9) Executing a Command Directly

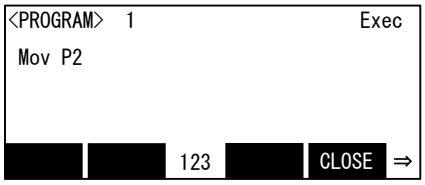
Direct execution is the method of executing the input statements instantaneously.
The robot operation can be checked during the progress of editing programs.
The operational procedure for direct execution of the Mov command is shown below.

1) Display the directly execution screen.
Press the function key corresponding to "DIRECT", and display the direct execution screen.



Display the direct execution screen [F1]

2) Execute the command (after changing any statement if necessary).
To edit the command, press the [→] key to move the cursor to the target point and change it.
To perform the command as is, press the [EXE] key. The command is executed while the [EXE] key is held pressed.
The execution of the command stops when the [EXE] key is released.



3.7 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.

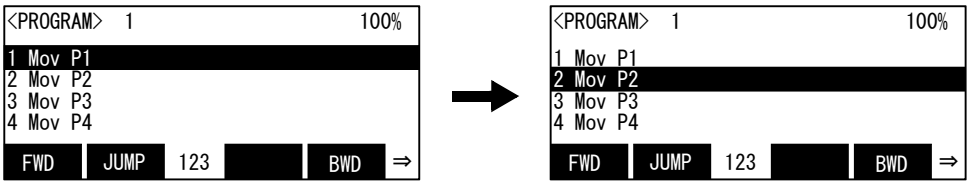
(1) Step feed

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.

⚠ CAUTION The robot's locus of movement may change with specified speed.
Especially as for the corner section, short cut distance may change. Therefore, when beginning automatic operation, moves at low speed at first, and you should gather speed slowly with being careful of interference with peripheral equipment.

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").
Refer to Page 53, "(4) Step jump".

Immediately stopping the robot during operation

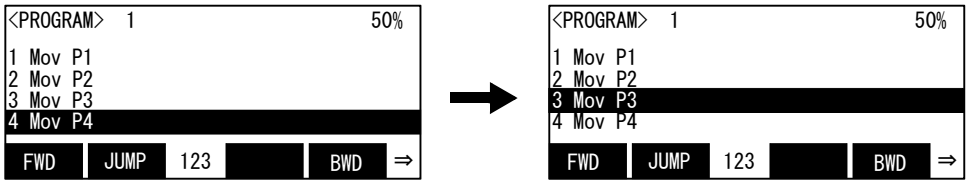
- Press the [EMG.STOP] (emergency stop) switch.
  The servo will turn OFF, and the moving robot will immediately stop.
  To resume operation, reset the alarm, turn the servo ON, and start step operation.
- Release or for cibly press the "enable" switch.
  The servo will turn OFF, and the moving robot 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.

(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.

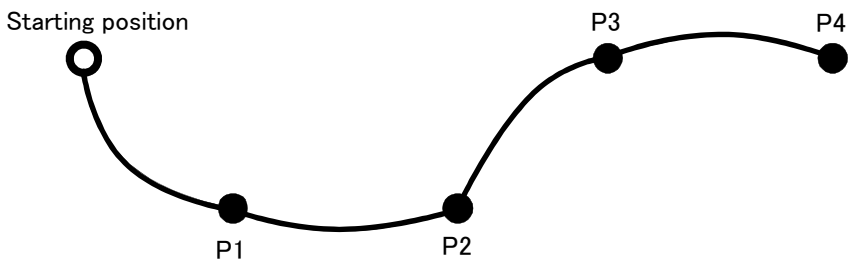


Step return [F4]

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

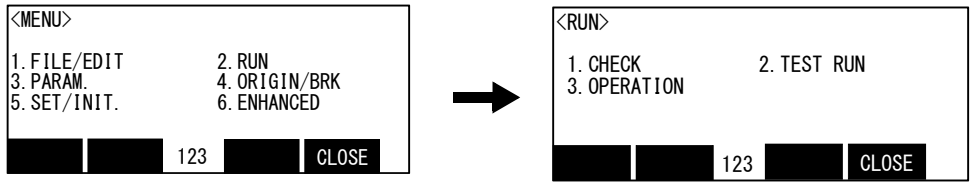
- Press the [EMG. STOP] (emergency stop) switch.
  The servo will turn OFF, and the moving robot will immediately stop.
  To resume operation, reset the alarm, turn the servo ON, and start step operation.
- Release or for cibly press the "enable" switch.
  The servo will turn OFF, and the moving robot 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 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. RUN".

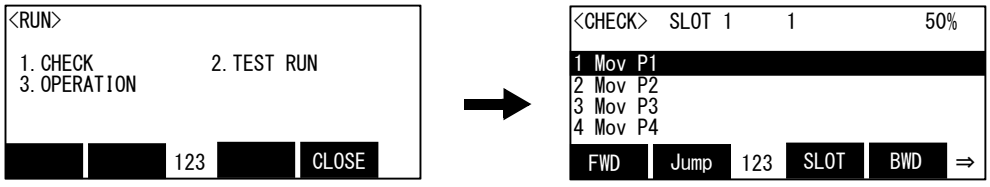


Display the run menu screen [2]

2) Selection of the confirmation screen

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

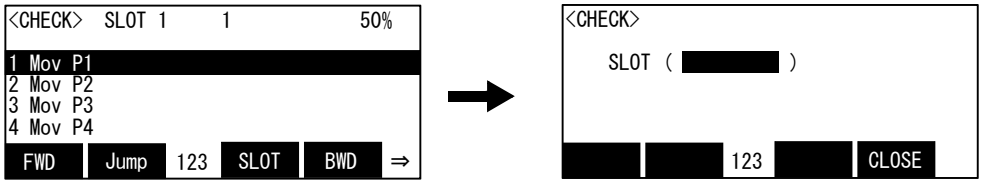
Display the program set as the slot 1. The program name is displayed following the slot number.



Display the check menu screen [2]

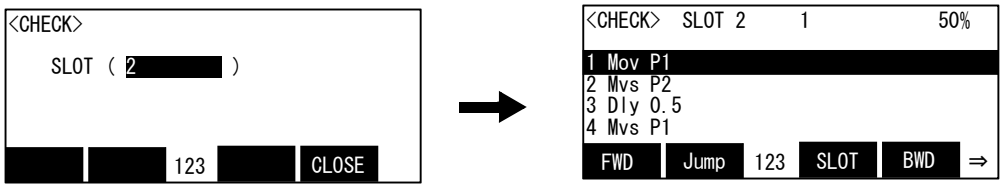
3) Change of the slot

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)



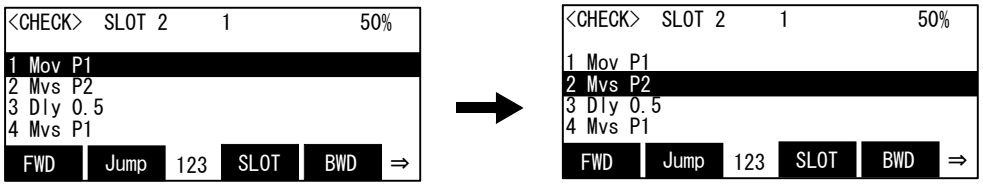
4) Execution of step operation

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

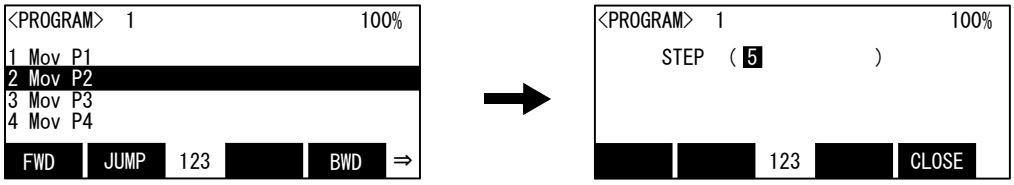
The execution step can be changed by cursor movement by the arrow key, and jump operation ("JUMP").

Refer to Page 53, "(4) Step jump".

(4) Step jump

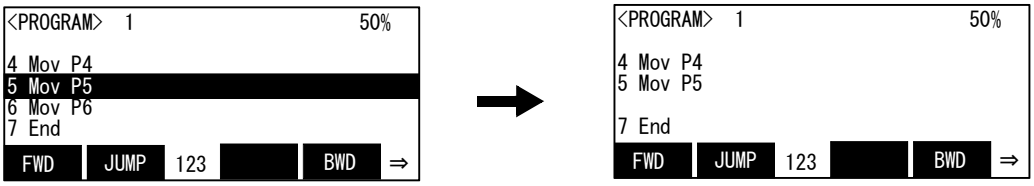
It is possible to change the currently displayed step or line.
The operation in the case of doing step operation from Step 5 as an example is shown.

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.

3.8 Automatic operation

3.8.1 Setting the operation speed

The operation speed is set with the T/B.

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

1) 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.

The following table shows the available robot operation and its operation speed depending on the status of each setting switch.

Setting switch	Controller	Controller mode (mode selector switch)	AUTOMATIC	MANUAL
	T/B	T/B ENABLE switch	OFF	ON
		Enabling switch (3-position enable switch)		
Robot operation that can be carried out from the T/B			Automatic operation (program execution)	
Operation speed			Program specified speed (max. 100%)	Safe operating speed (250 mm/s) or less

Robot operation other than the automatic operation that can be carried out from the T/B includes the jog operation, step operation, and position jump.

These operations are available when the status of setting switches is as shown below. The robot operates at the safe operating speed (250 mm/s) or less.

Setting switch	Controller	Controller mode (mode selector switch)	MANUAL
	T/B	T/B ENABLE switch	ON
		Enabling switch (3-position enable switch)	
Operation speed			Safe operating speed (250 mm/s) or less

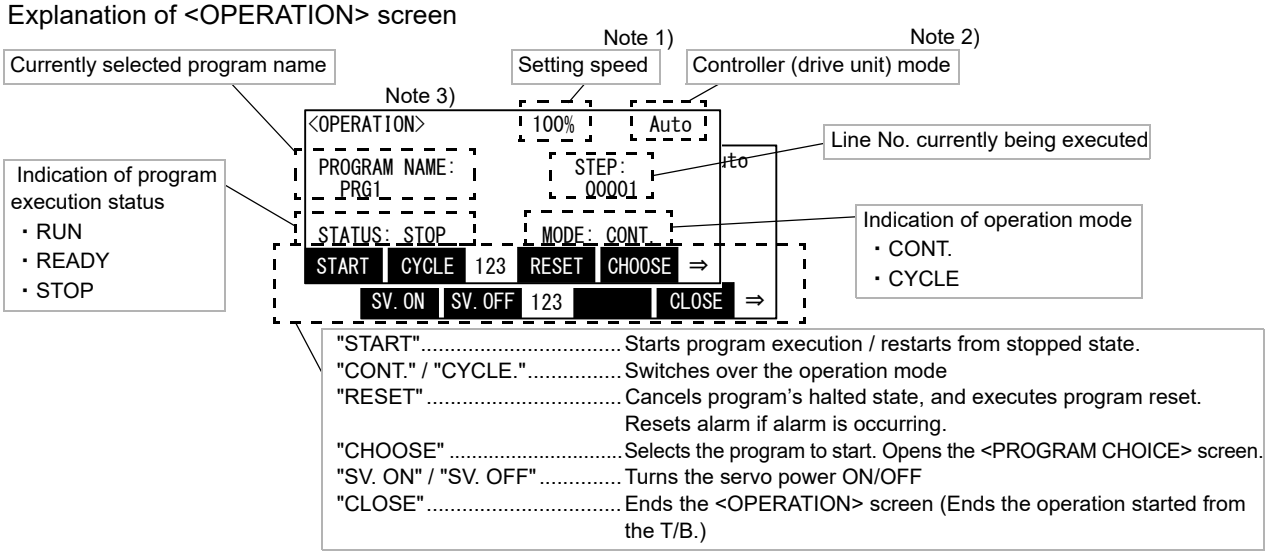
3.8.2 Starting automatic operation

Operations are carried out from the <OPERATION> screen opened by selecting <MENU> → <RUN>.

This function can be disabled by setting parameter: TBOP.

The <OPERATION> screen is as shown below.

Explanation of <OPERATION> screen



Note 1) When the controller mode is "MANUAL", the robot moves at low speed even if the set speed is 100%. (For operation check)

Note 2) When the controller mode is "AUTOMATIC", the T/B's status display lamp [ENABLE] blinks to indicate that the T/B is enabled. The T/B is disabled when the <OPERATION> screen is ended.

Note 3) The [JOG], [HAND] and [MONITOR] keys are disabled while the <OPERATION> screen is opened.

The functions which can be executed from the T/B <OPERATION> screen and the operations which can be executed while the <OPERATION> is displayed are shown below.

Operations from <OPERATION> screen (Each function key)

<1> Selecting a program....."CHOOSE"
<2> Turning servo ON/OFF ....."SV. ON" / "SV. OFF"
(When the controller mode is set to "AUTOMATIC", the ENABLE switch does not need to be turned ON.)
<3> Executing automatic operation ....."START"
<4> Changing operation mode (continuous/cycle) ..."CONT./CYCLE"
<5> Resetting a program ....."RESET"

Other key operations

<1> Changing the movement speed.....[OVRD↑] / [OVRD↓] key
Note) When the controller mode is set to "AUTOMATIC", the robot operates at the speed specified by a user. When it is set to "MANUAL", the robot operates at the safe operating speed (250 mm/s) or less (for operation check). Set the controller to the desired mode.
<2> Turning servo ON.....[SERVO] key
(When the controller mode is set to "AUTOMATIC", the ENABLE switch does not need to be turned ON.)
<3> Resetting the alarm.....[RESET] key

* The robot can be stopped at any time with the [EMG. STOP] switch or [STOP] key.
Operations using the [JOG], [HAND] or [MONITOR] key will be invalid

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

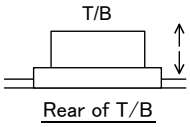
- Before initiating the automatic operation, ensure that there are no exposed persons in the safeguarded space.
- Always start automatic operation outside the safety protection area (outside the safety fence).
- Before starting automatic operation, always confirm that the work details and selected program match.
- When the modification of the program is done during the suspension, be sure to verify the program at least one time before initiating the automatic operation.
- If a held workpiece or other object may fly out and cause a hazard by the centrifugal force during automatic operation or by the impact force of an emergency stop, suitable means must be taken to prevent such hazard.
- If the robot collides with a workpiece or the peripheral equipment at high speed due to e.g. a program error, then the workpiece, the peripheral equipment, the robot arm, and the reduction system of the robot may be considerably damaged.

The methods for starting automatic operation from the T/B are explained in this section.

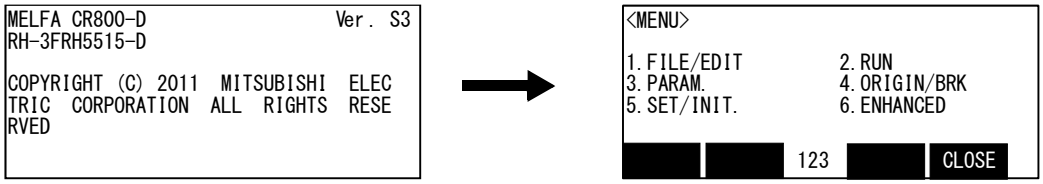
1) Press the T/B ENABLE switch on the rear side of the teaching box (T/B) to disable the T/B operation (the T/B ENABLE switch lamp turns off), and set the mode selector switch of the user prepared controller to "AUTOMATIC".
When the mode selector switch is switched to "AUTOMATIC" while the T/B ENABLE switch is enabled, the error H5000 "TB Enable key is ON" occurs.



Up :DISABLE
Down :ENABLE
*Lighting

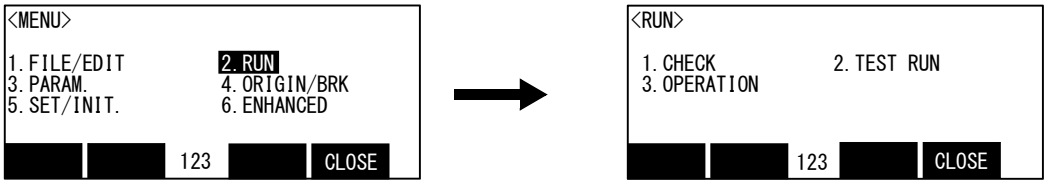


2) Press the [EXE] key while the title screen is displayed. The <MENU> screen appears.



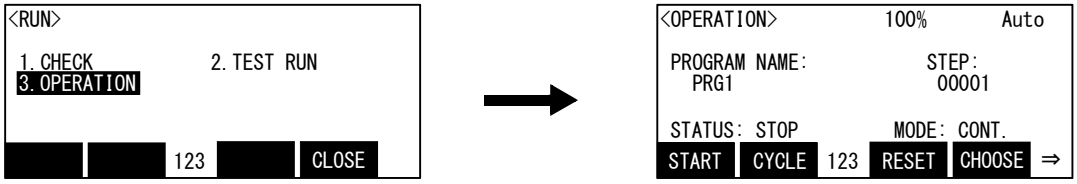
MENU screen display [EXE]

3) Press the [2] key while the <MENU> screen is displayed. The <RUN> screen appears.

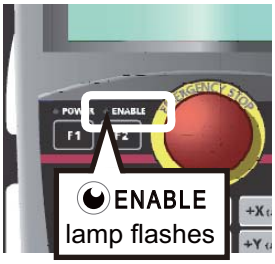


RUN menu selection [2]

4) Press [3] key while the <RUN> screen is displayed. The <OPERATION> screen appears.

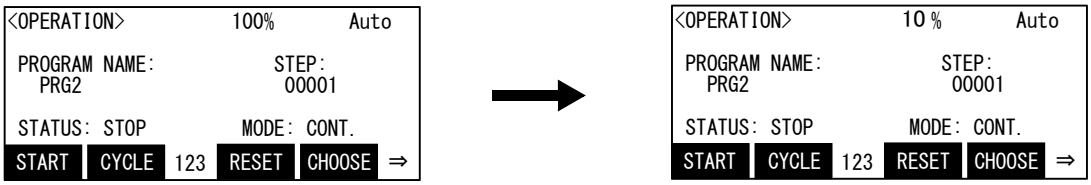


OPERATION selection [3]



The status display lamp "ENABLE" on the front of the T/B blinks.

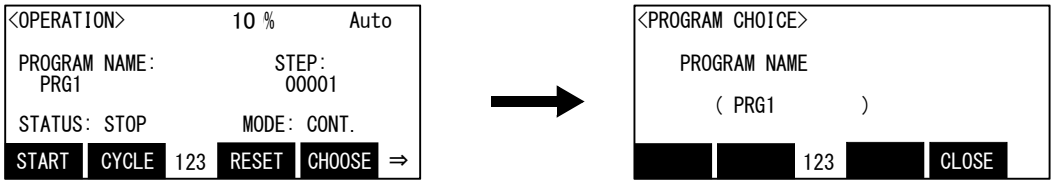
5) Set the safe speed by pressing the [OVRD↑] and [OVRD↓] keys.



Speed setting [OVRD↑], [OVRD↓]

When the speed is changed to 10%

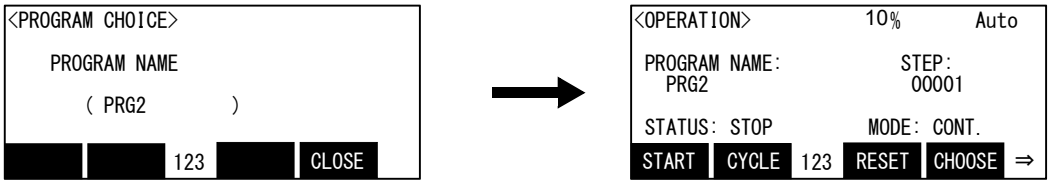
6) Press the [F4] key ("CHOOSE") while the <OPERATION> screen is displayed. The <PROGRAM CHOICE> screen appears.



Program selection [F4]

Name of the currently selected program is displayed.

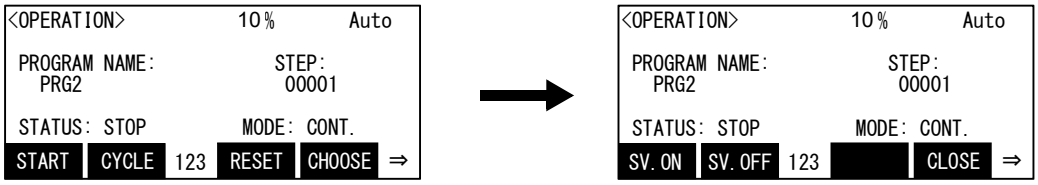
7) Enter the name of the program into the Program Name brackets, and press the [EXE] key.
A new program will be displayed and the screen will return to the <OPERATION> screen.
* Always check that the target program is selected.
* The program cannot be changed when the program execution status is "STOP". Before operation, press [RESET] (the F3 key), and set the status indication to "READY".



Enter the program name and press [EXE] key

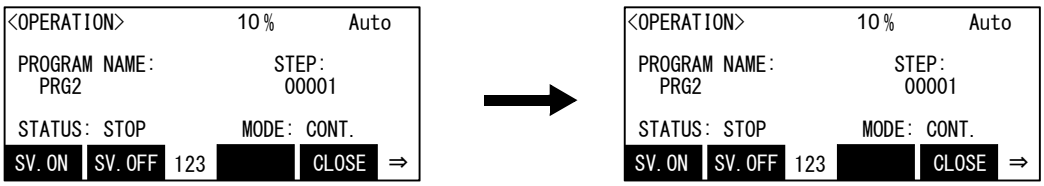
The figure shows an example of changing program name to “PRG2”.

8) When the "SV. ON" function is not displayed, press the [FUNCTION] key to switch the function menu display.



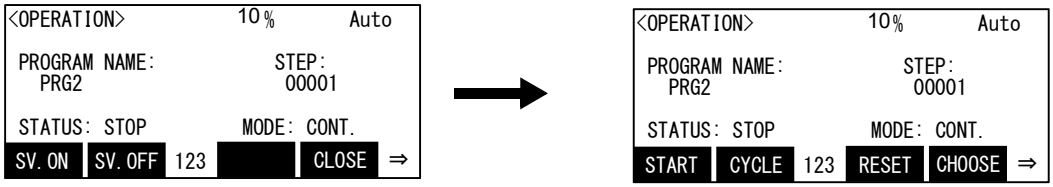
Switch the function menu [FUNCTION]

9) Check that no operators are in the robot operation area before turning on the servo, and press the [F1] key ("SV. ON"). * The servo can be turned on with the [SERVO] key on the T/B.
* During this operation, you do not need to hold the [Enable] switch.



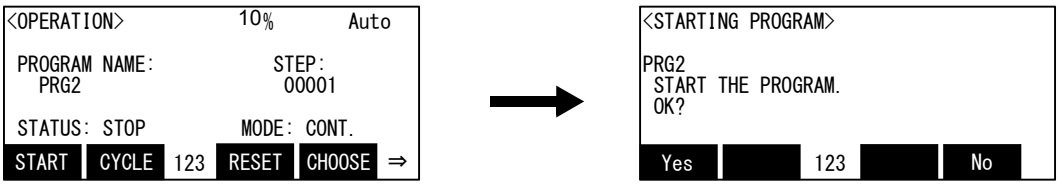
Servo ON [F1]

10) Press the [FUNCTION] key to switch the function menu display.



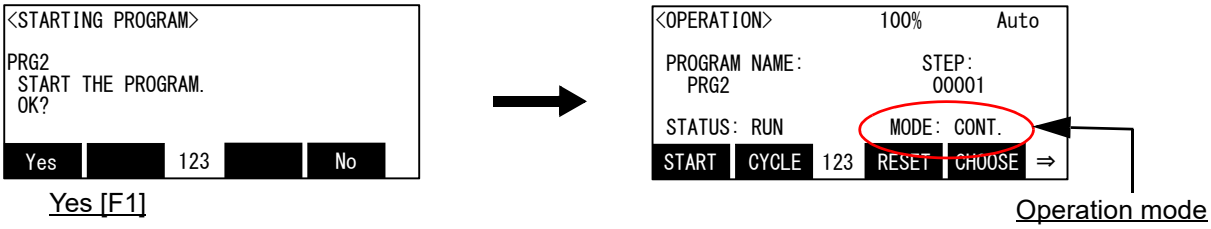
Start [F1]

11) Press the [F1] key ("START"). The check screen appears.



Start [F1]

12) Press the [F1] key ("YES"). Automatic operation of the currently selected program will start. The screen will return to the <OPERATION> screen. The operation mode follows the mode displayed on the screen.



Yes [F1]

This completes the starting of automatic operation from the T/B.

3.8.3 Stopping

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

Stop [STOP]

1) Press the [STOP] key.

Operation rights not required

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

3.8.4 Resuming automatic operation from stopped state

The operation method is the same as the start of automatic operation. Please do restart operation from T/B with reference to Page 54, "3.8.2 Starting automatic operation".

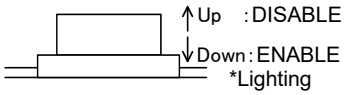
⚠CAUTION Before starting automatic operation, always confirm that the target program No. is selected.

⚠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.

⚠CAUTION Don't turn off the power supply of robot CPU during automatic operation. The memory in robot CPU may become abnormal and the program may break. Use the emergency stop, when stopping the robot immediately.

3.8.5 Resetting the program

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



T/B enable

Rear of T/B

1) Set the controller mode to "MANUAL".
2) Set the T/B [ENABLE] switch to "ENABLE".

Execute of program reset

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.

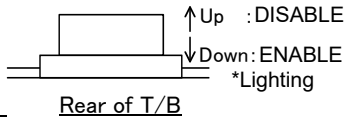
Program reset [RESET] + [EXE]

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.

3.9 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.



1) Set the controller mode to "MANUAL".
2) Set the T/B [ENABLE] switch to "ENABLE".

Execute servo ON

Servo ON operation [SERVO]

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

Execute servo OFF

Servo OFF operation [enabling]

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

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.

3.10 Error reset operation

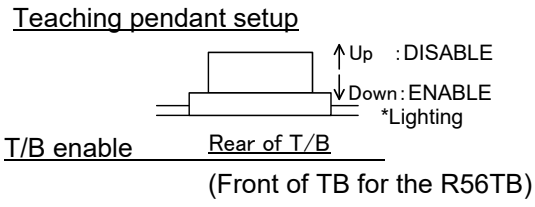
Cancel errors

1) Press the [RESET] key.

Error reset [RESET]

3.11 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.
The [RESET] key corresponding to the [CAUTION] key in R56TB. When operating with R56TB, use the [CAUTION] key and perform following operation.



1) Set the controller mode to "MANUAL".
2) Set the Enable switch of the teaching pendant to ENABLE.
3) Press the [JOG] key to display the JOG screen. When using the force sensor function, errors can be temporarily reset by accessing the Force sense function screen from the JOG screen.
4) Lightly hold the Enable switch (3-position switch).
5) For the R32TB, press and hold the [RESET] key. For the R56TB, press and hold the [CAUTION] key.
6) Press the [SERVO] key to turn the servo on.
7) Move the robot using Jog operation.

The operation above will reset errors temporarily.
Temporary error reset mode can be disabled using any of the following methods:
(1) Releasing the [RESET] key ([CAUTION] key).
(2) Setting the Enable switch of the teaching pendant to DISABLE and changing the controller mode to AUTOMATIC.
(3) Displaying the JOG screen or a screen unrelated to the force sense function. (This method will only work for the R32TB.)

Errors will still occur after Temporary error reset mode is disabled if they have not been resolved.
Press and hold the [RESET] key ([CAUTION] key) when moving the robot in Jog operation mode.

⚠CAUTION The robot will not stop at the operating range limit while using Jog operation in Temporary error reset mode.
Be careful not to move the robot outside of the operating range.

[Reference] Controller LED displays
The front of the controller contains 4 LEDs that represent controller status.

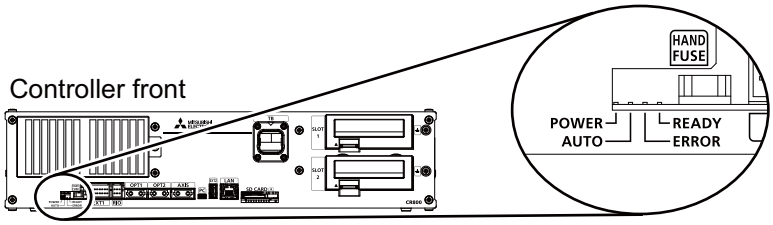
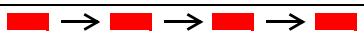
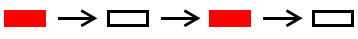
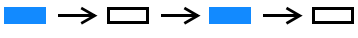
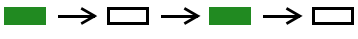
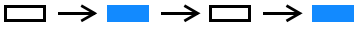
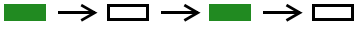
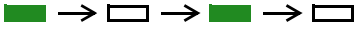
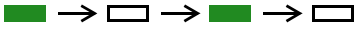


Fig.3-1:Controller status LED

Table 3-5:LED displays

LED	Details
POWER	Indicates the control power status. On: Control power ON Off: Control power OFF
AUTO	Indicates the controller mode. On: AUTOMATIC mode Off: MANUAL mode
ERROR	Indicates the error status. On: Low level error or warning occurred. Rapid flashing: High level error occurred. Off: Normal operation
READY	Indicates the operation status. On: ON (ready) Slow flashing: During operation Rapid flashing: Operation suspended.

Table 3-6:LED display status for each function

Controller status	LED display status		
Low level error Warning	ERROR LED lights up.		
High level error	ERROR LED blinks.		Blinks quickly.
System writing data to the SD card or USB memory	AUTO LED blinks.		AUTO and READY LEDs blink quickly.
	READY LED blinks.		
During startup (From the time the power is turned on until startup completes.)	AUTO LED blinks.		AUTO and READY LEDs alternately blink slowly. Both LEDs light up for 1 second when startup completes.
	READY LED blinks.		
During operation	READY LED blinks.		Blinks slowly.
While operation is suspended	READY LED blinks.		Blinks quickly.

3.12 Operating the program control screen

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

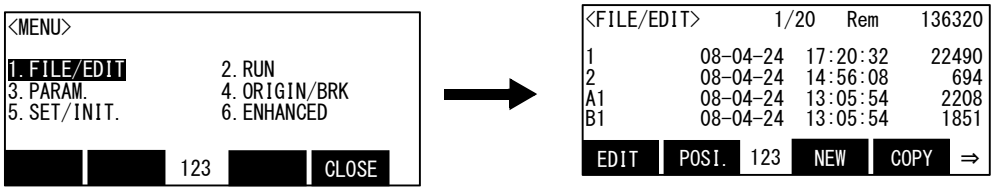
"(1)Program list display"
"(2)Copying programs"
"(3)Name change of the program (Rename)"
"(4)Deleting a program (Delete)"
"(5)Protection of the program (Protect)"

(1) Program list display

This functions 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. FILE/EDIT" are selected and display the list of the programs.



Display the <FILE/EDIT> screen [1]

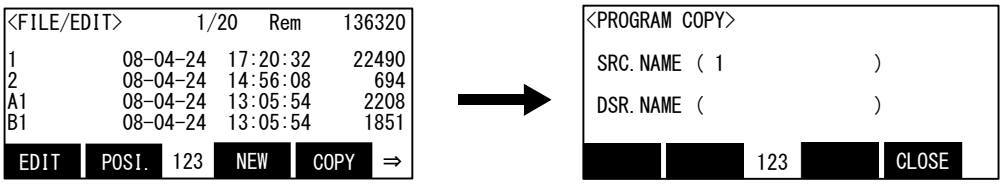
Same operation can be done, even if the cursor is moved to "1. FILE/EDIT" by the [↑] or [↓] key and it presses the [EXE] key.
And, the program which is the target of each operation can also be selected.

The menu ("EDIT", "POSI.", "NEW", "COPY") corresponding to the function key is displayed under the screen.
Press the [FUNCTION] key, then display the "RENAME", "DELETE", "PRTCT", "CLOSE".

(2) Copying programs

1) Select the copy menu

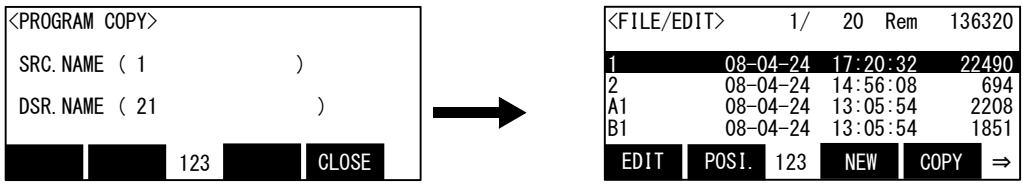
Press the function key corresponding to the "COPY" by program list display. Display the copy screen.



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

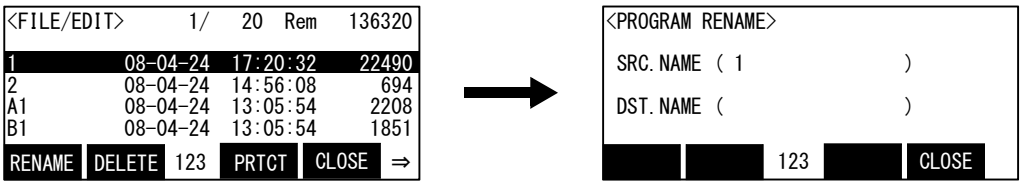
The program protection information and variable protection information is not copied with the copy operation.

Reset this information as necessary.

(3) Name change of the program (Rename)

1) Select the rename menu

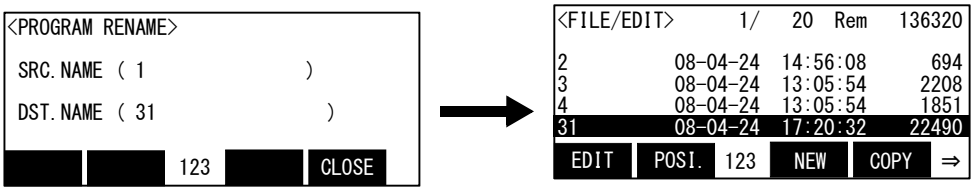
Press the function key corresponding to the "RENAME" by program list display. Display the rename screen. If the "renaming" menu is not displayed, press and display the [FUNCTION] key.



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.



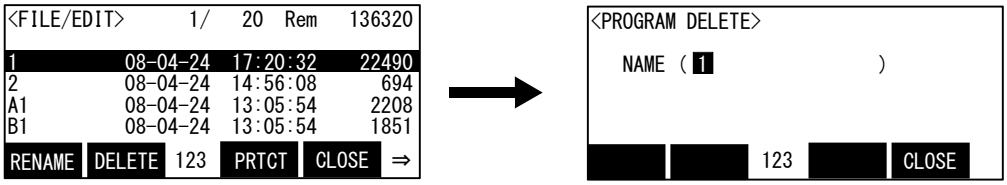
The program name protected cannot be changed.

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

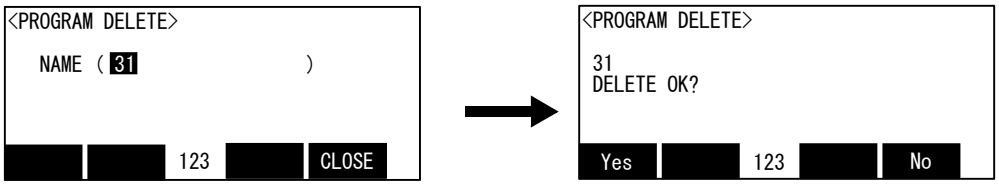
Press the function key corresponding to the "DELETE" by program list display. Display the delete screen. If the "DELETE" menu is not displayed, press and display the [FUNCTION] key



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.

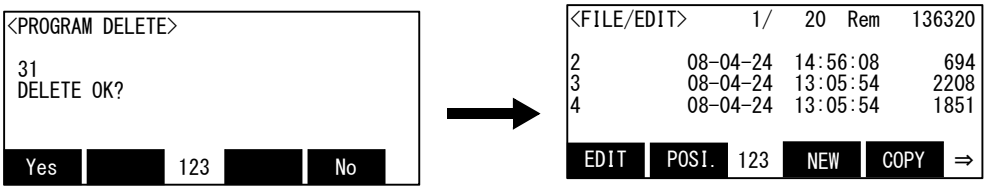
Press the [EXE] key, and display the confirmation screen.

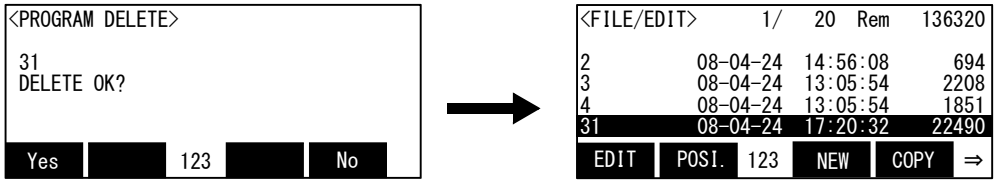


3) Delete the program

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.





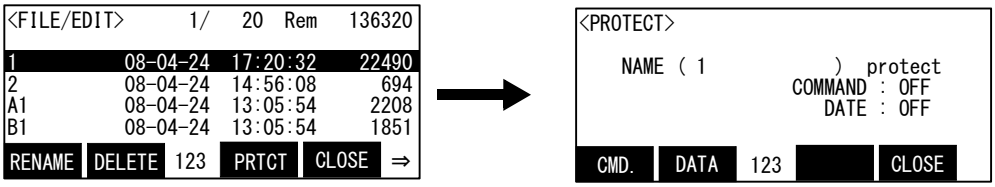
The program name protected cannot be deleted.

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)

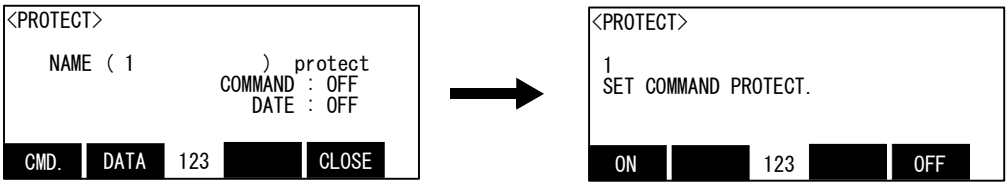
1) Select the protect menu

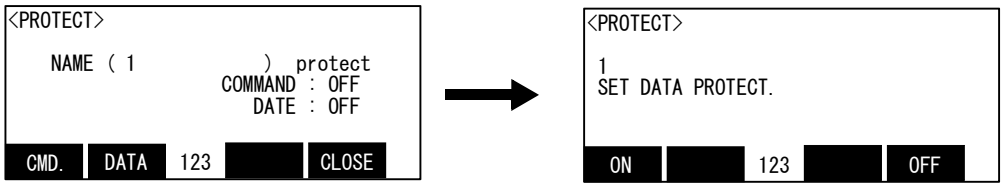
Press the function key corresponding to the "PRTCT" by program list display. Display the protect screen. If the "PRTCT" menu is not displayed, press and display the [FUNCTION] key



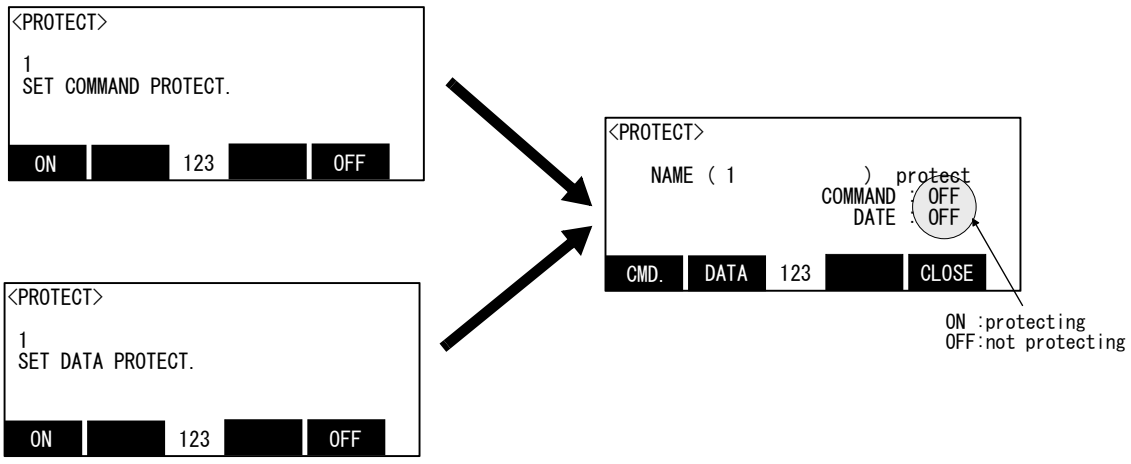
2) Setup of the protection.

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 "CMD." 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.

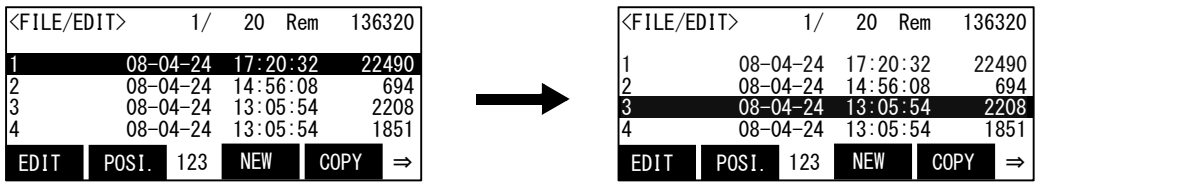
- Protection information is not copied in copy operation.
- In initialization operation, protection information is disregarded and execute initialization.

(6) Select the program

The program of step or automatic execution can be selected.
The operation method is shown in the following.

1) Select the program

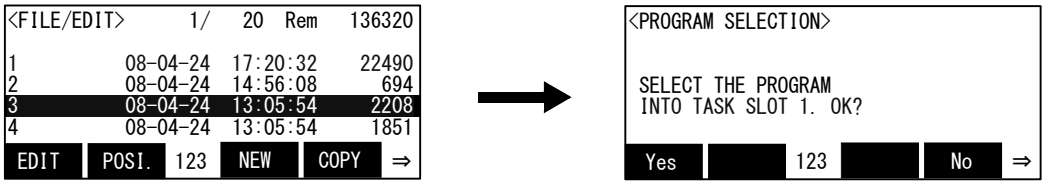
Move the cursor to the program which select by the key [↑], [↓].



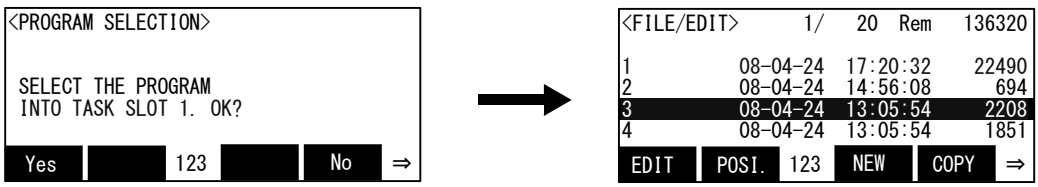
The figure is the example which selected the program 3.

2) Setting of the program name (number)

Press the [FUNCTION]+[EXE] key of T/B. The confirmation screen is displayed.



Confirm the program name (number) currently displayed. If the function key corresponding to "Yes" ([F1]) is pressed, the program name (number) is selected.
If the function key corresponding to "No" ([F4]) is pressed, the operation is canceled. Each returns to the program list display.



Selection of the program is finishing above.

3.13 Operation of operating screen

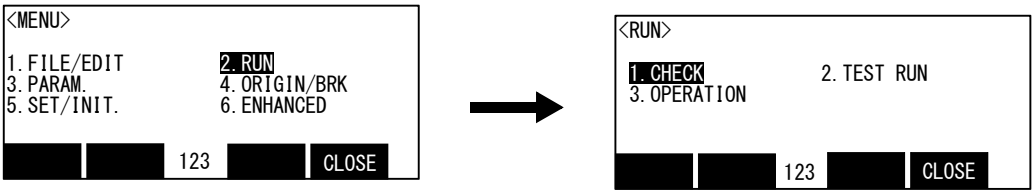
(1)Display of the execution line..... 1.Confirmation:
Display the executing program line, or execute step feed.

(2)Display of the test execution line .... 2. 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.

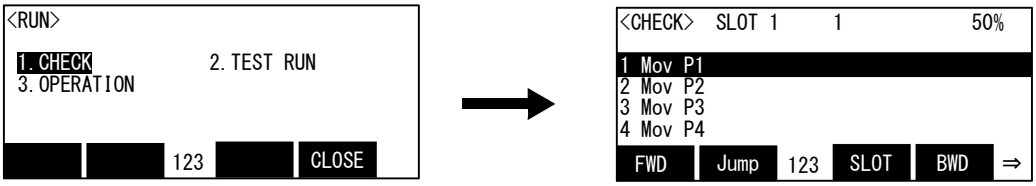
3.13.1 Display of the execution line

(1) Select the confirmation menu

1) Press the [2] key in the menu screen, and display the <RUN> 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.



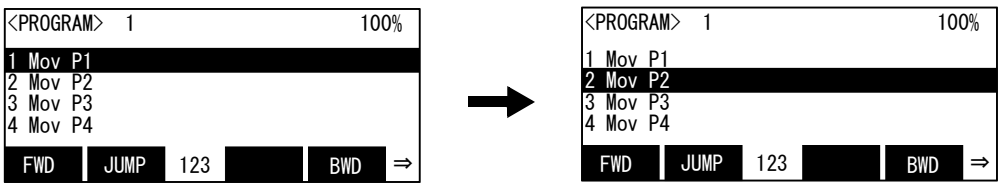
The cursor moves to the execution line during program execution.

(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.
If execution of the one step is completed, 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").

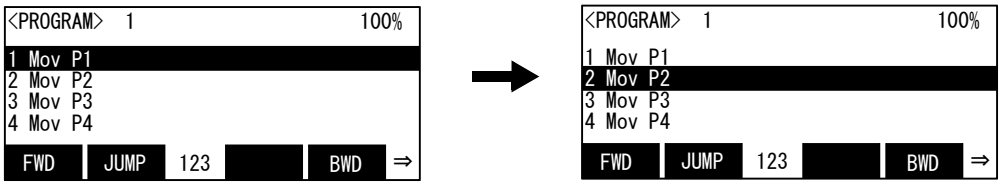
Immediately stopping the robot during operation

- 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 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.

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.
If execution of the one step is completed, 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.

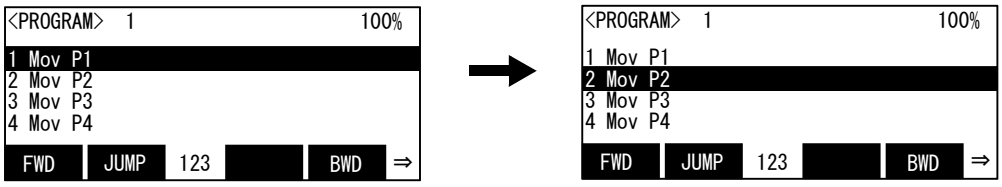
Immediately stopping the robot during operation

- 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.

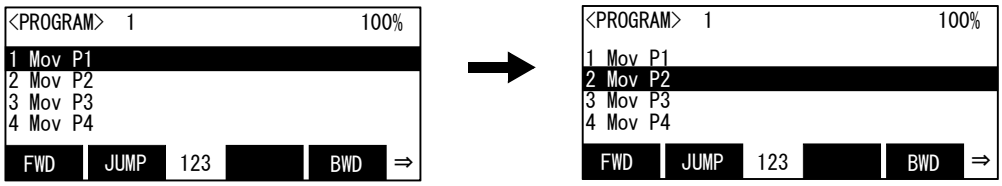
(3) Step jump

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.



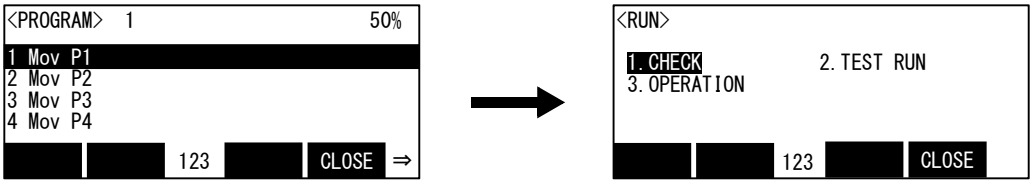
⚠ CAUTION Take special care to the robot movements during automatic operation. If any abnormality occurs, press the [EMG. STOP] switch and immediately stop the robot.

(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 51, "(3) Step feed in another slot" for operation method.

(5) Finishing of the confirmation screen.

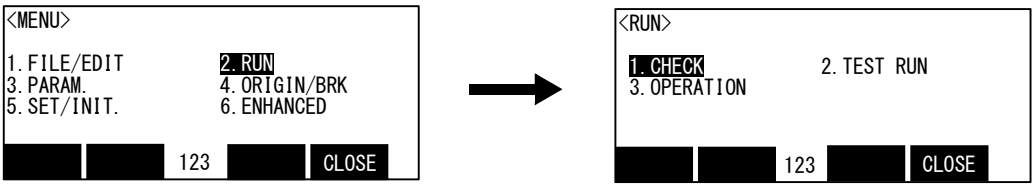
1) Press the function key corresponding to "CLOSE", and return to the <OPERATION> screen.



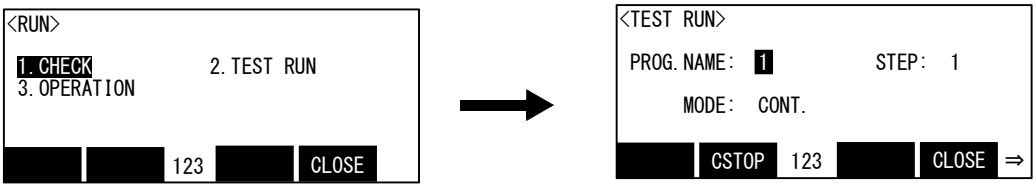
3.13.2 Test operation

(1) Select the test operation

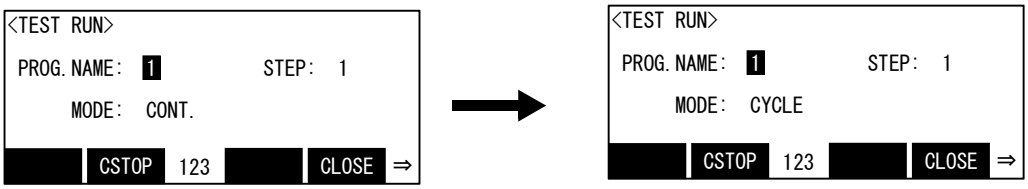
1) Press the [2] key in the menu screen, and display the operation menu screen.



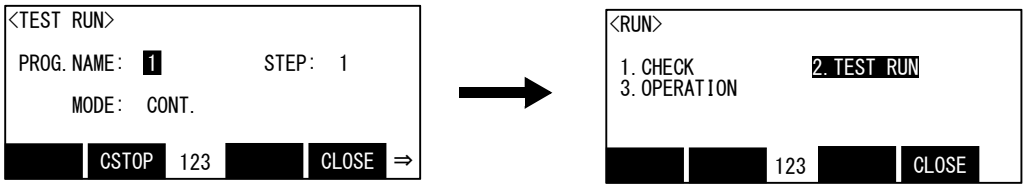
2) Press the [2] key, and display the test operation screen.
The program name, execution step number, and operating mode is displayed.



3) When the function key ([F2]) corresponding to "CSTOP" is pressed during program execution, it is change to the cycle mode of operation. "CYCLE" is displayed as the mode name. The operation finishes after the last line of the program or the END command is executed.



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.

3.13.3 Operating the OPERATION screen

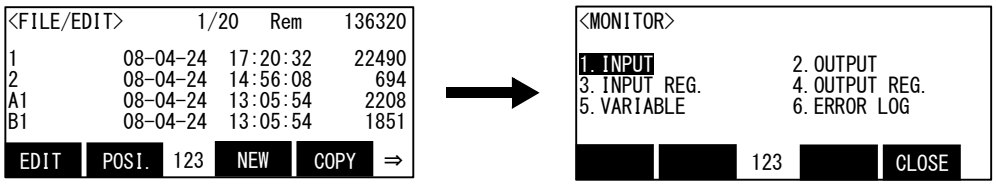
Turning on and off the servo power, selecting the program, starting of automatic operation, etc can be done.
Refer to Page 54, "3.8.2 Starting automatic operation".

3.14 Operating the monitor screen

Here, explain the operation method of the following functions.

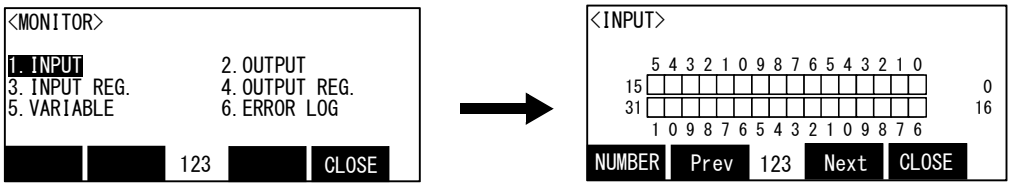
(1)Input signal monitor .....1.Input: Parallel input signal monitor
(2)Output signal monitor.....2.Output: Parallel output signal monitor. Setup of ON/OFF
(3)Input register monitor.....3.Input register: Input register of CC-Link
(4)Output register monitor.....4.Output register: Output register of CC-Link
(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.



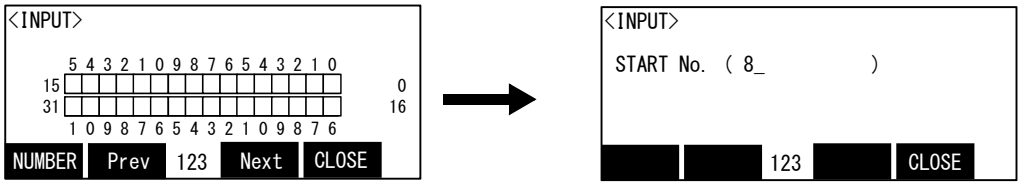
(1) Input signal monitor

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.

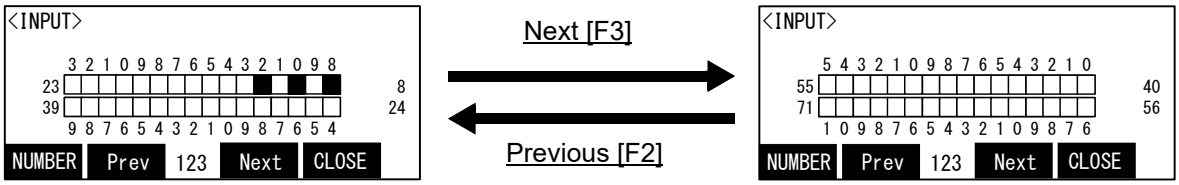


The case where the state of the input signals 8-15 is confirmed is shown in the following.

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

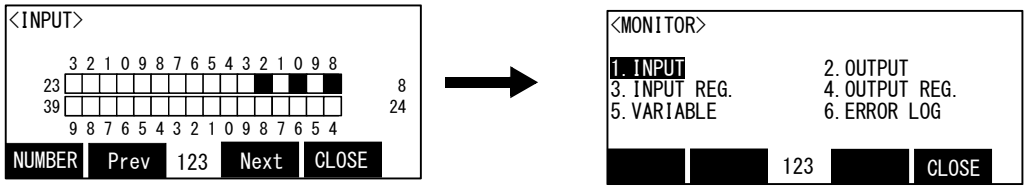


3) Display the ON/OFF state of the 32 points at the head for the input signal No. 8. Black painting indicates ON and white indicates OFF.

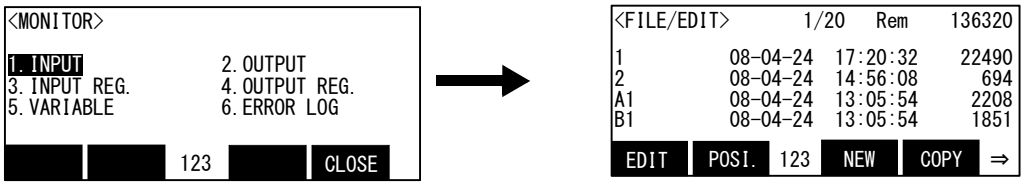


Press the function key corresponding to "Next", then display the next input signal screen. Press the function key corresponding to "Prev", then display the previous input screen.

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.



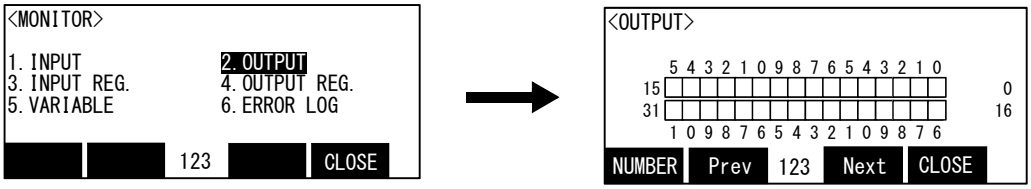
Finish the monitor

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

(2) Output signal monitor

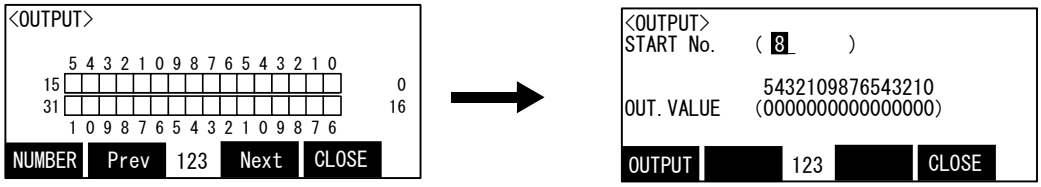
There are the function which always makes the ON/OFF state of the output signal the monitor, and the function outputted compulsorily.

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.

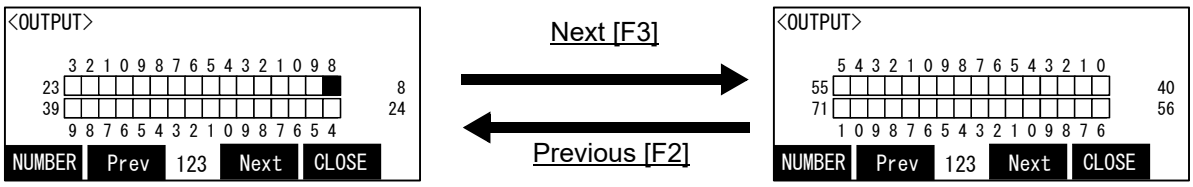


The case where the state of the output signals 8-15 is confirmed is shown in the following.

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

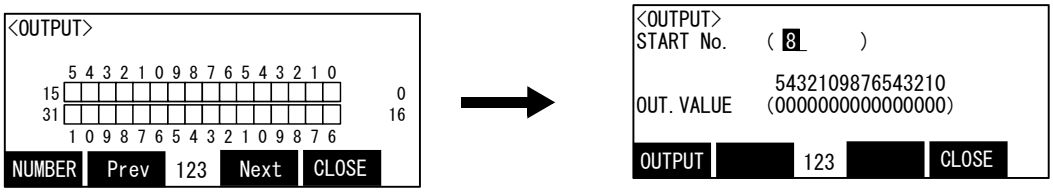


Although the state of the current output signal is displayed on the output value on the display, it is not always the display here in the section which sets up the compulsive output value of the signal.
Press the function key corresponding to "CLOSE". Display the ON/OFF state of the 32 points at the head for the output signal No. 8. Black painting indicates ON and white indicates OFF.

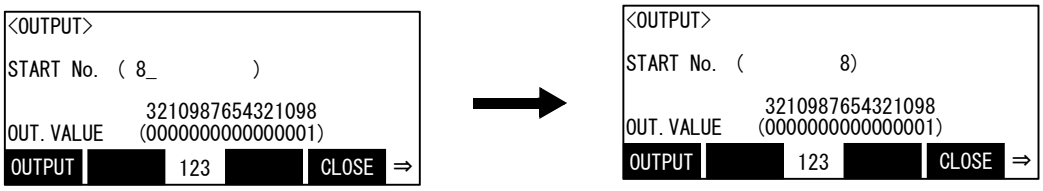


Press the function key corresponding to "Next", then display the next output signal screen. Press the function key corresponding to "Prev", then display the previous output screen.

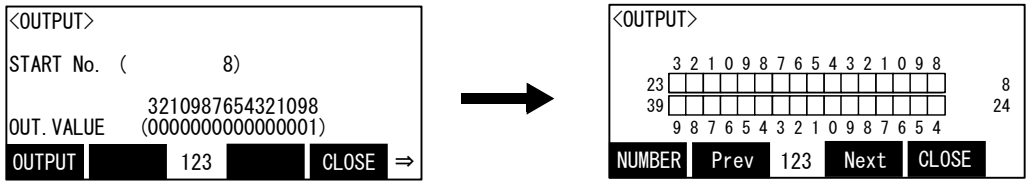
3) The compulsive output of the output signal.
In the following, the operation method in the case of turning off the output signal No. 8 compulsorily is shown.
Press the function key corresponding to "NUMBER".
Set "8" as the start number. (Press [8], and [EXE] key)



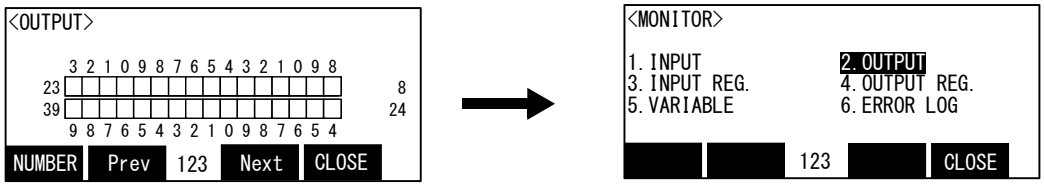
4) Move the cursor to the position of "8" of the output value by the arrow key.
Since the output signal 8 number is turned on now, value "1" is displayed.
If the value is changed into "0" which shows OFF and the function key ([F1]) corresponding to the "OUTPUT" is pressed, this output signal will actually be off.



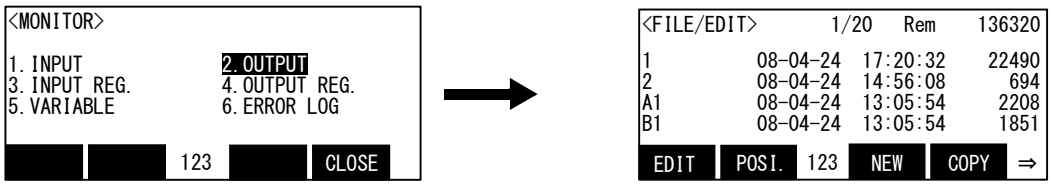
5) Press the function key corresponding to "CLOSE", and return to the output monitor screen.



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



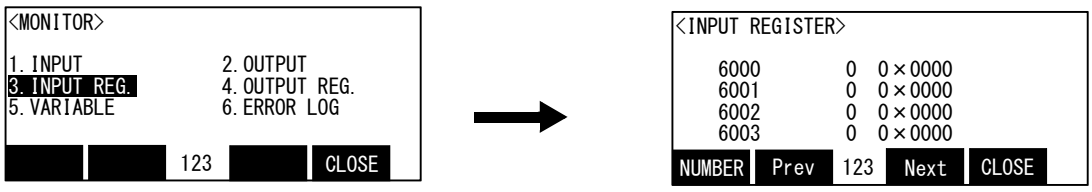
7) Press the function key corresponding to "CLOSE" in monitor menu screen is pressed, finish the monitor, and return to the original screen.



(3) Input register monitor

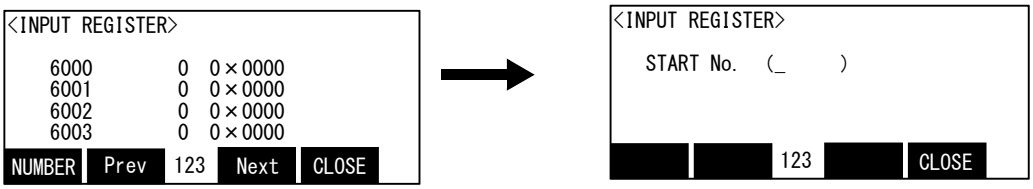
If CC-Link is used, it is the function which always monitors the value of the input register.
Note) Since there is no CC-Link option in the CR800-R/Q series, this function can not be used.

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.

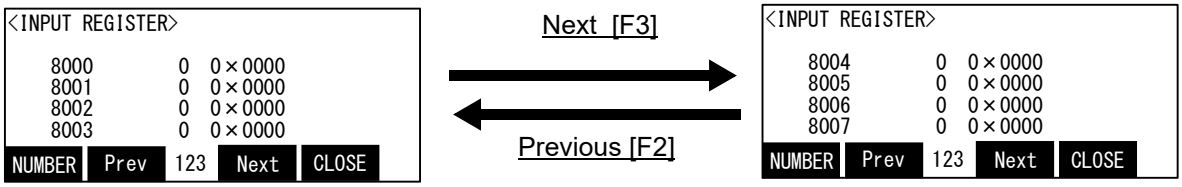


The case where the state of the input register 8000 is confirmed is shown in the following.

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

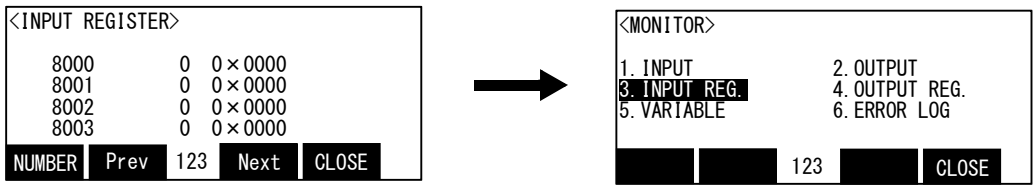


3) Display the ON/OFF state of the 4 input register at the head for the input register No. 8000.

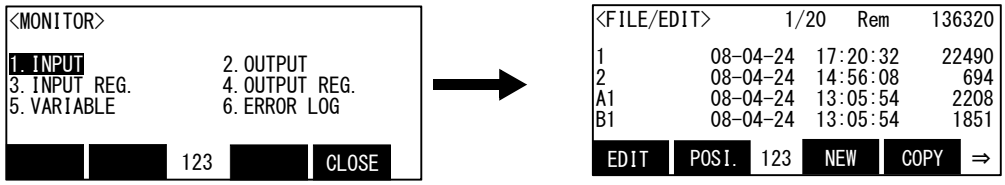


Press the function key corresponding to "Next", then display the next input register screen. Press the function key corresponding to "Prev", then display the previous input register screen.

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

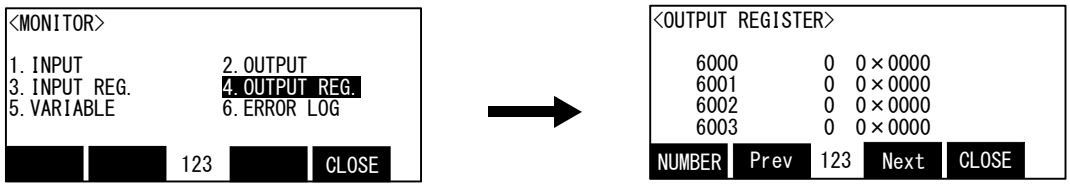
If the [MONITOR] key is pressed, the monitor will be finished always and it will return to the original screen.

(4) Output register monitor

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

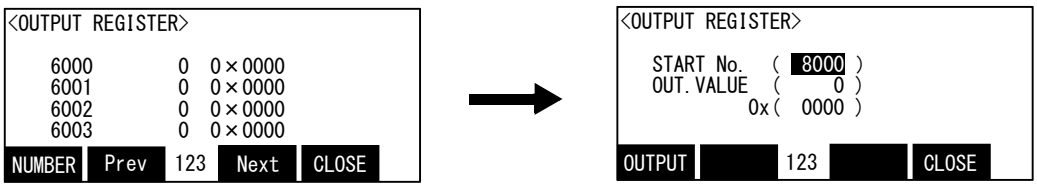
Note) Since there is no CC-Link option in the CR800-R/Q series, this function can not be used.

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.

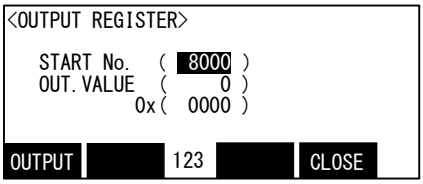


The case where the state of the output register 8000 is confirmed is shown in the following.

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

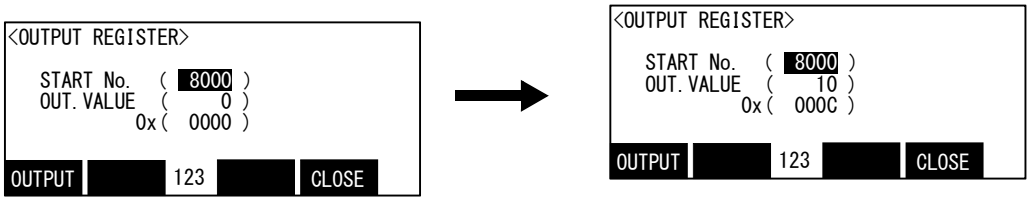


3) The current output value of No. 8000 is displayed by the decimal number in the parenthesis following the output value. The value in the parenthesis following lower 0x is the hexadecimal number.

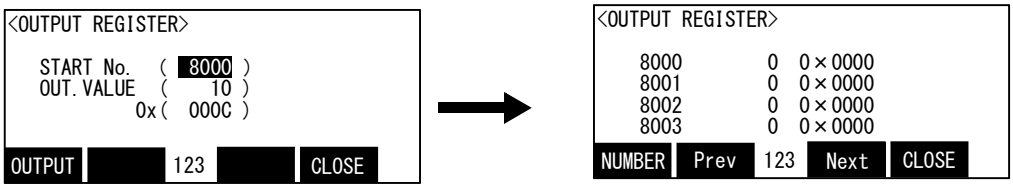


If the function key which corresponds for "CLOSE" is pressed, it can also return to the monitoring screen on the basis of No. 8000 of the output register, but the output value can be changed on the current screen.
The case where the value of the output register No. 8000 is set as 12 (decimal number)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

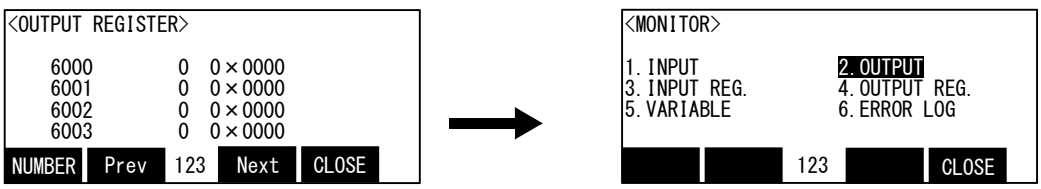
4) The setup of the value can be set up by the decimal number or the hexadecimal number.
If it sets up by the decimal number, move the cursor to the output value by the arrow key, and input "10". The unnecessary character should press and erase the [CLEAR] key.
If it sets up by the hexadecimal number, 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.



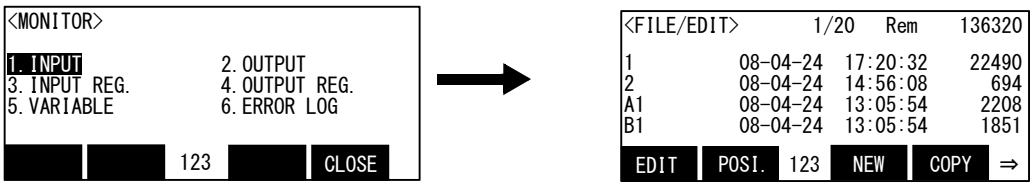
5) Press the function key corresponding to "CLOSE", and return to the output register monitor screen.



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.



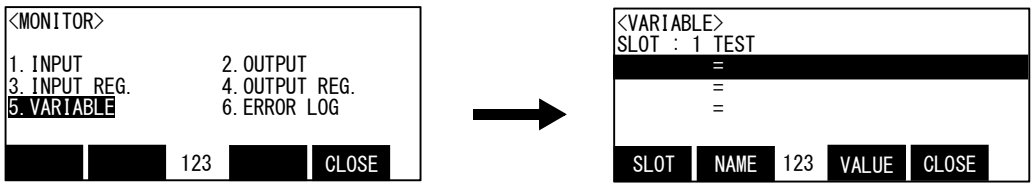
Finish the monitor

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

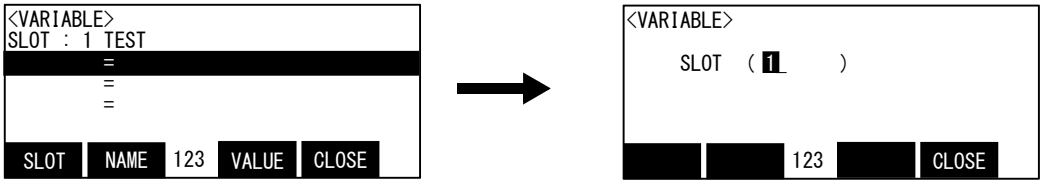
(5) Variable monitor

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

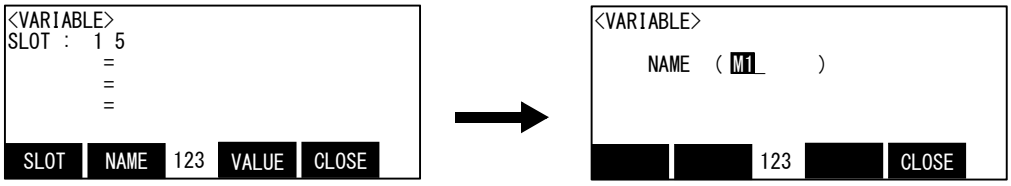
1) Press the [5] key in the monitor menu screen, and display the variable monitor screen.



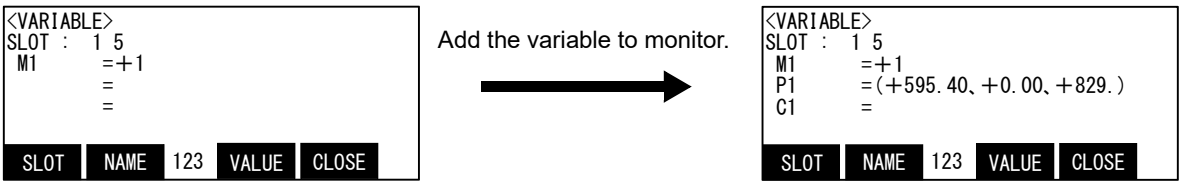
2) Specify the target program of the monitor with the slot number.
Press the function key corresponding to "SLOT", and input the slot number.
Set up "1", if the multitasking function is not being used.



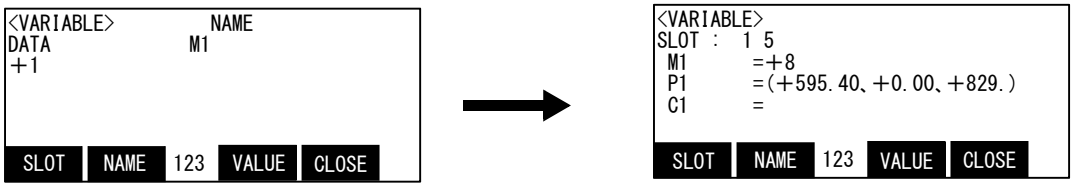
3) Display the slot number and the program name after "slot:".
Press the function key corresponding to the "NAME", and input the variable name to monitor.



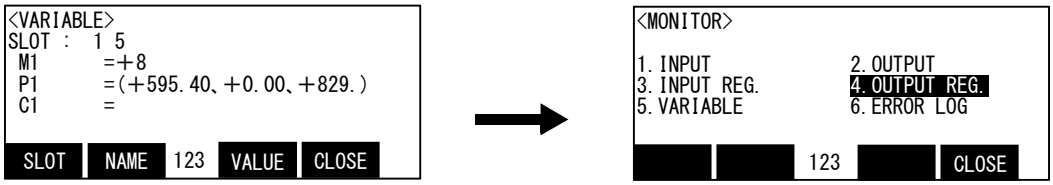
4) Display the value of the numeric variable M1 on the screen.
The variable which will be monitored if the cursor is moved to the line which is vacant in the arrow key and operation of the above "3)" is repeated can be added. The variable which can be monitored simultaneously is to the three pieces.



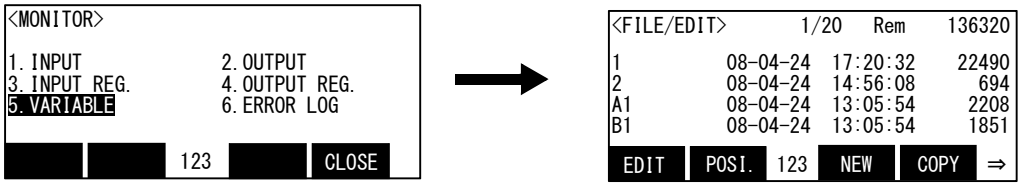
5) Change the variable value.
The value of the variable currently displayed can be changed.
Move the cursor to the variable name changed by the arrow key, and press the function key corresponding to the "VALUE"
Although the current value (data) is displayed, it can input and change.



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.



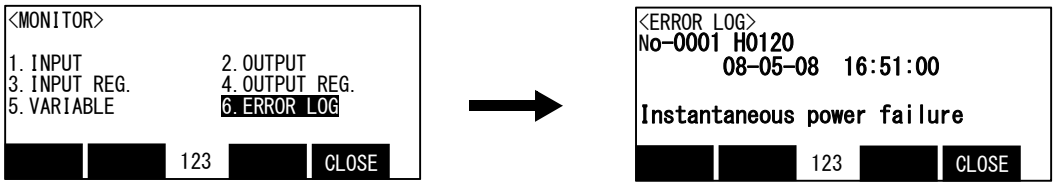
The right of operation is unnecessary.
It operates, even when T/B is invalid.
And, the value (data) of the variable can be changed also in automatic operation.

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

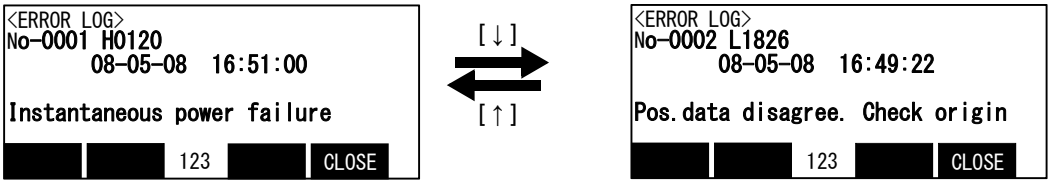
(6) Error history

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.

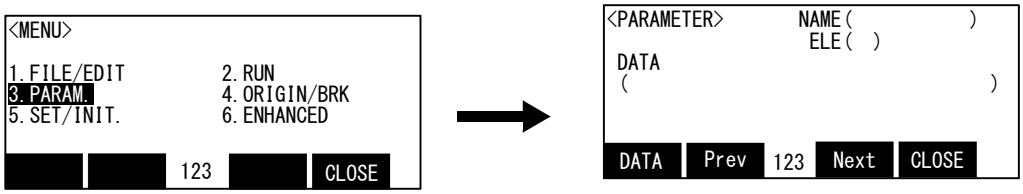
It operates, even when T/B is invalid.

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

3.15 Operation of parameter screen

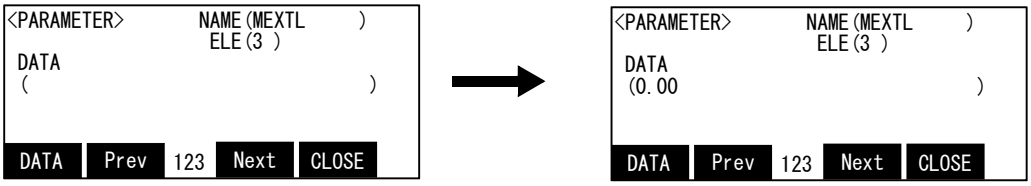
The parallel I/O designated input/output settings and settings for the tool length, etc., are registered as parameters. The robot moves based on the values set in each parameter. This function allows each parameter setting value to be displayed and registered.

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

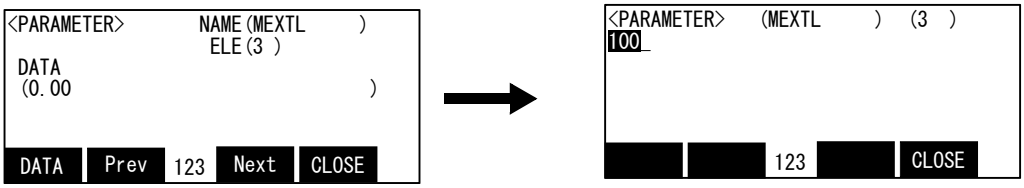


An example of changing the parameter "MEXTL (tool conversion data)" Z axis (3rd element) setting value from 0 to 100mm is shown below.

2) Input "MEXTL" into the name and input "3" into the element.
The data set up now is displayed.

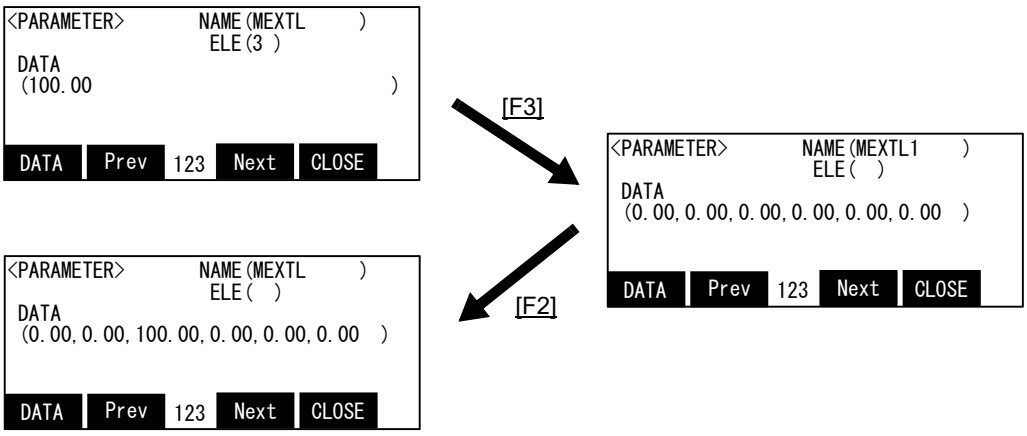


3) Press the function key corresponding to the "DATA", and input new preset value "100".
Delete the unnecessary number by the [CLEAR] key.

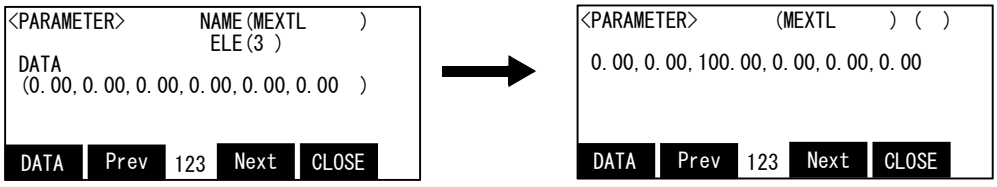


If the [EXE] key is pressed, the buzzer will sound, the value will be fixed and it will return to the screen of the parameter.
If the function key corresponding to the "CLOSE" is pressed also after inputting the new preset value, change can be canceled and it can return to the parameter screen.

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.

3.16 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.

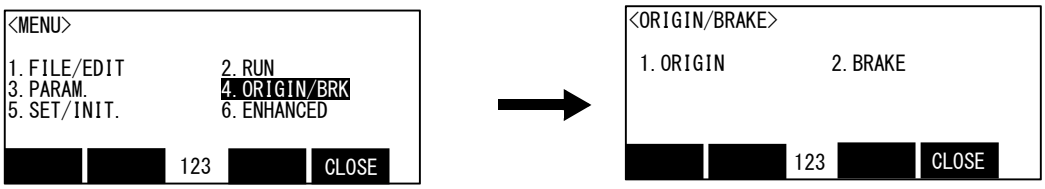
(2) Brake

In the state of servo off, it is the function to release the brake of the servo motor. Refer to the Page 60, "3.9 Turning the servo ON/OFF" for servo off operation.
Use it, if it moves the robot arm directly by hand.

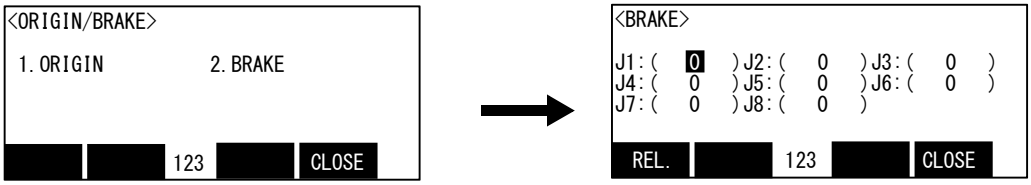
⚠CAUTION Due to the robot configuration, when the brakes are released, the robot arm will drop with its own weight depending on the released axis.
To ensure safety, take appropriate measures such as supporting the axis to avoid the free fall.

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

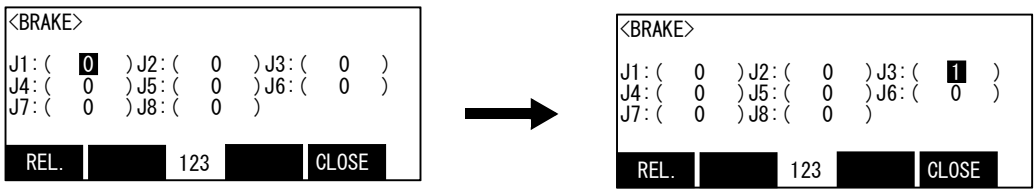
1) Press the [4] key in the <menu> screen, and display the <ORIGIN/BRAKE> screen.



2) Press the [2] key in the <ORIGIN/BRAKE> screen, and display the <BRAKE> screen.



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

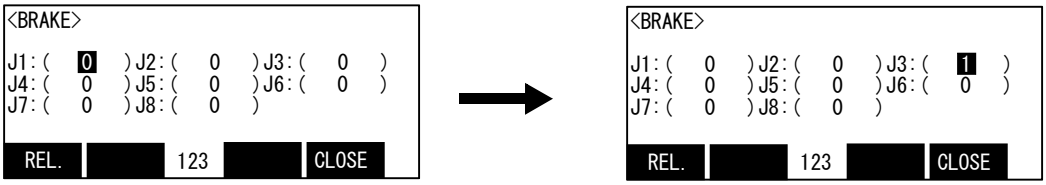


⚠CAUTION Due to the robot configuration, when the brakes are released, the robot arm will drop with its own weight depending on the released axis.
To ensure safety, take appropriate measures such as supporting the axis to avoid the free fall.

4) Press function key continuously corresponding to "REL." to release the brake of the specified axis only while the keys are pressed.
Only while the function key ([F1]) is pushing, the brake of the specified axis is released.

- RH-3FRH series, RV-2FR/4FR/7FR series, RH-3FRHR series:
  The brake is released continuously.
- RH-6FRH/12FRH/20FRH series:
  The brake is released in an off-and-on way.
  (released → locked → released → locked →...)
- RV-13FR series, RV-20FR series:
  J2 and J3 axis brakes are released in an off-and-on way.
  (released → locked → released →locked →...)
  The others are released continuously.

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



3.17 Operation of setup / initialization screen

Here, explain the operation method of the following functions.

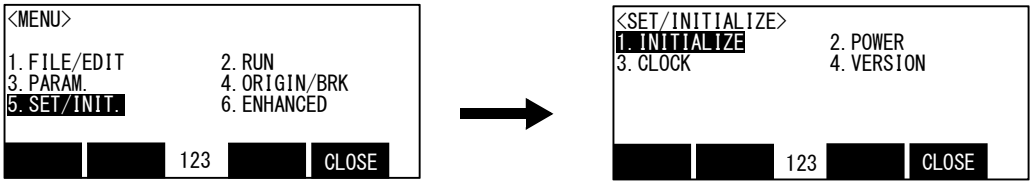
(1)Initialization .....1. Programs: Delete all the programs
2. Parameter: Return the parameter to the setup at the time of shipment.
3. Battery: Reset the expended hours of the battery.

(2)Operation .....Display 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)Version .....Display the software version of the controller and the teaching pendant.

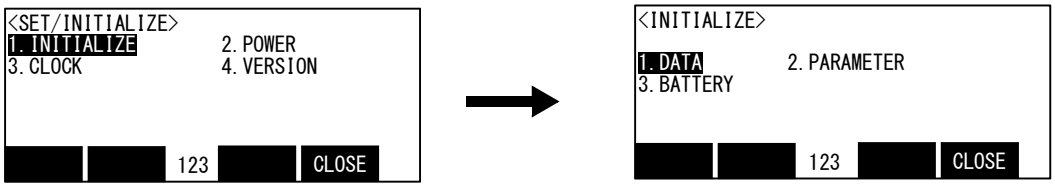
Press the [5] key in the menu screen, and display the <SET/INITIALIZE> screen.



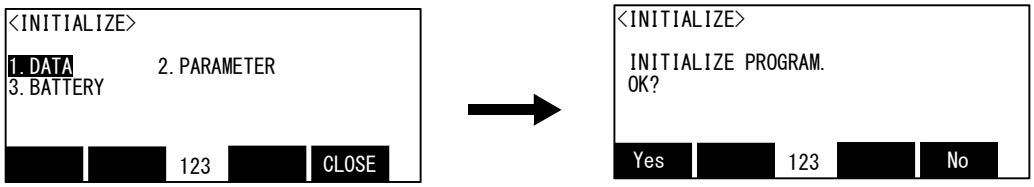
(1) Initialize the program

Delete all the programs.

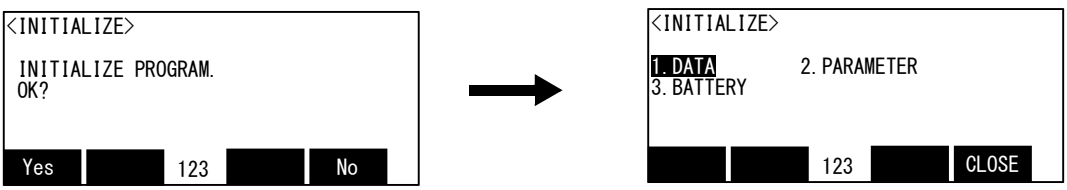
1) Press the [1] key in the <SET/INITIALIZE> screen, and display the initial menu screen.



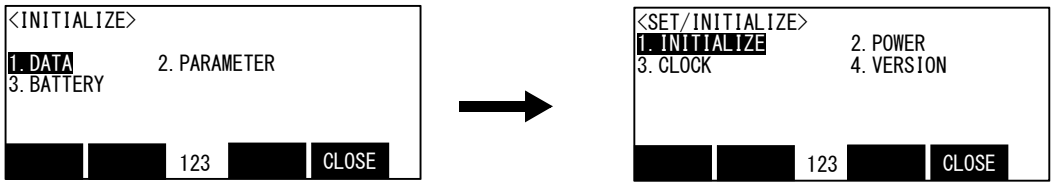
2) Press the [1] key in the initial menu screen, and select the program.
Display the screen of confirmation.



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 <INITIALIZE> screen.



4) Press the function key corresponding to "CLOSE", and return to the <SET/INITIALIZE> screen.



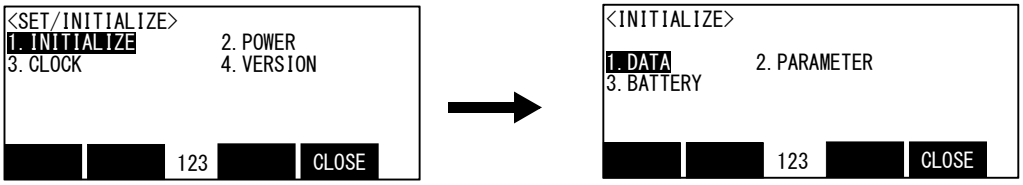
Executed even when protected

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

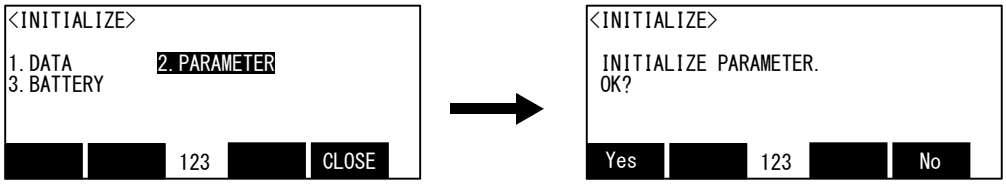
(2) Initialize the parameter

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

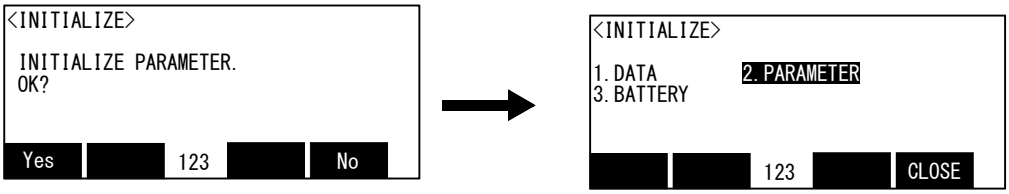
1) Press the [1] key in the <SET/INITIALIZE> screen, and display the <INITIALIZE> screen.



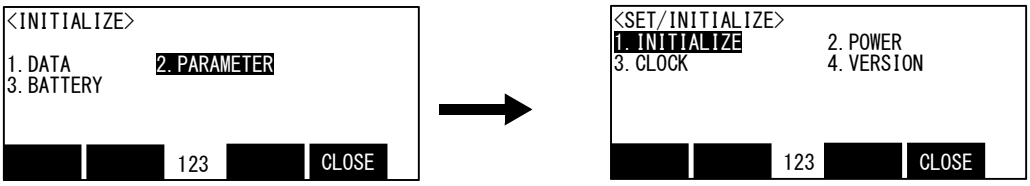
2) Press the [2] key in the initial menu screen, and select the parameter. Display the screen of confirmation.



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 <INITIALIZE> screen.



4) Press the function key corresponding to "CLOSE", and return to the <SET/INITIALIZE> screen.



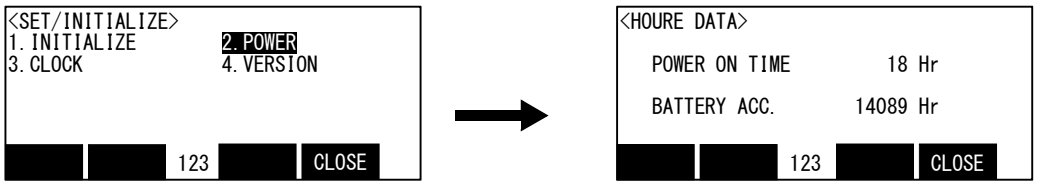
(3) Initialize the battery

In CR800 series controller, consumption of the battery is automatically checked by detection of the voltage. Thus, it is not necessary to perform the battery initialization after replacing the battery. The battery initialization will be invalid even if it is performed.

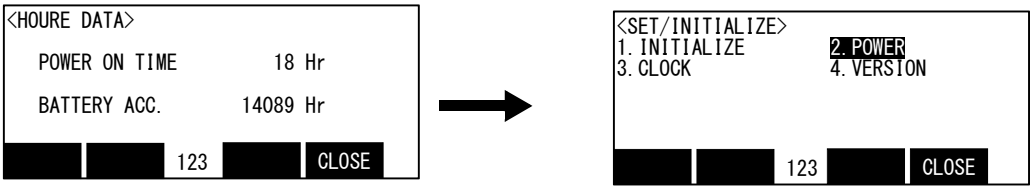
(4) Operation

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

1) Press the [2] key in the <SET/INITIALIZE> screen, and display the <HOUR DATA> screen.



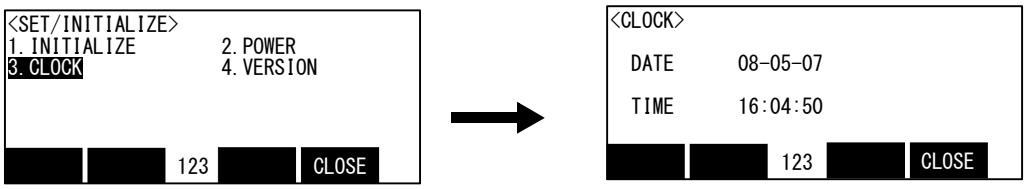
2) Press the function key corresponding to "CLOSE", and return to the <SET/INITIALIZE> screen.



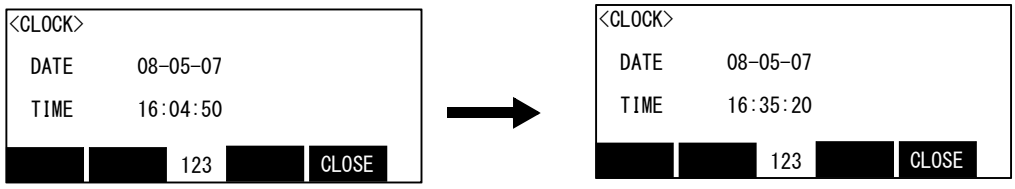
(5) Time setup

Display of the date and time, the setup

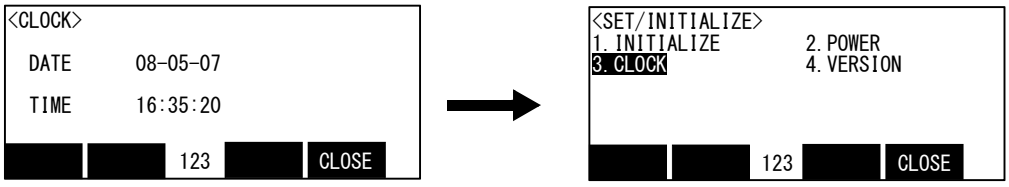
1) Press the [3] key in the <SET/INITIALIZE> screen, and display the <CLOCK> screen.



2) Date and time can be setup on the <CLOCK> screen.
Move the cursor by the arrow key and input the current date and time.



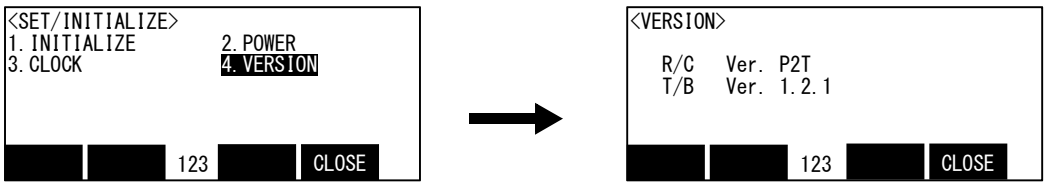
3) Press the function key corresponding to "CLOSE", and return to the <SET/INITIALIZE> screen.



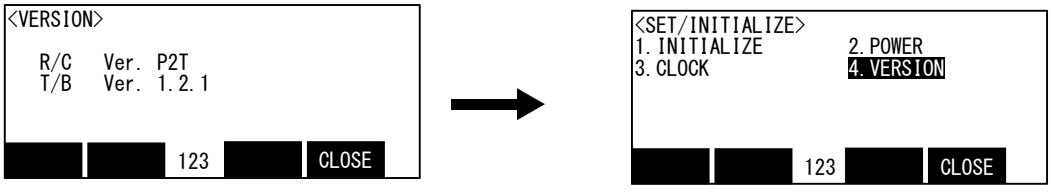
(6) Version

Display the software version of the controller and the teaching pendant

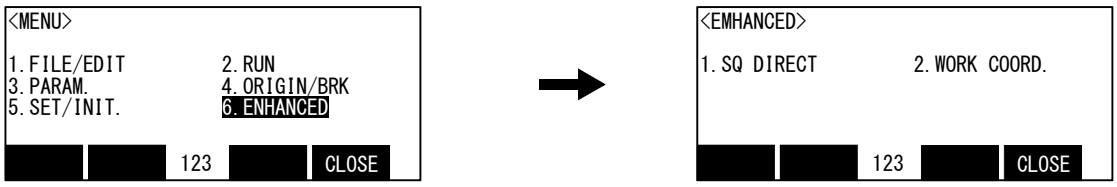
1) Press the [4] key in the <SET/INITIALIZE> screen, and display the <VERSION> screen.



2) Press the function key corresponding to "CLOSE", and return to the <SET/INITIALIZE> screen.



3.18 ENHANCED



(1) SQ DIRECT

This function controls the robot by the program of the PLC directly. (CR800-R or CR800-Q series)
Please refer to separate manual: "CR800-R/CR800-Q series, iQ Platform Supporting Extended Function Instruction Manual (BFP-A3528)".

(2) WORK COORD

This screen defines the work coordinates system necessary for work jog operation. If you use the work jog, define the target work coordinates system.
The details of the operation method are described in the jog operation. Please use with reference to them.
[Reference]

1) Setting of work coordinates, work jog operation:
   Separate manual: "ROBOT ARM SETUP & MAINTENANCE"
2) Types of jog feed: This instruction manual/ Page 22, "3.2.1 Types of jog feed".
3) Related parameter: This instruction manual/ "Work coordinates" on Page 484, "5.1 Movement parameter".

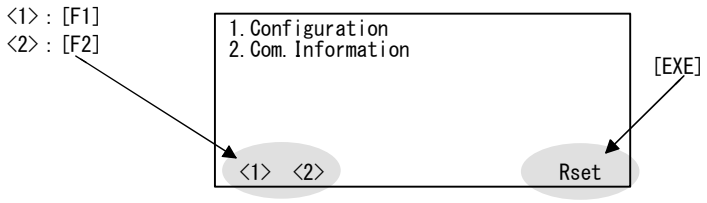
3.19 Operation of the initial-setting screen

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.

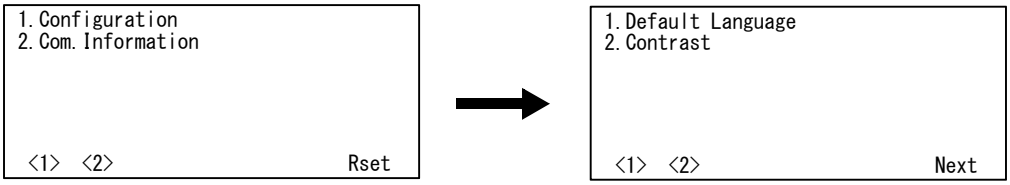
Operate this operation on the initial-setting screen displayed at turning on the control power in the condition of pushing both of [F1] key and [F3] key of T/B.



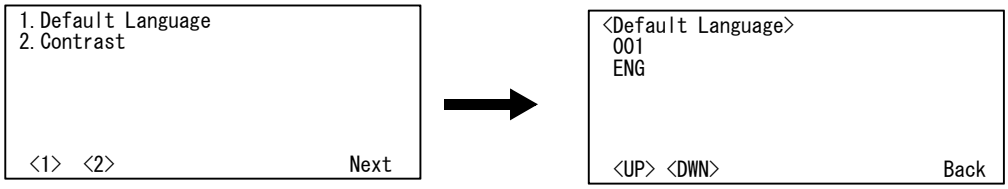
(1) Set the display language

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

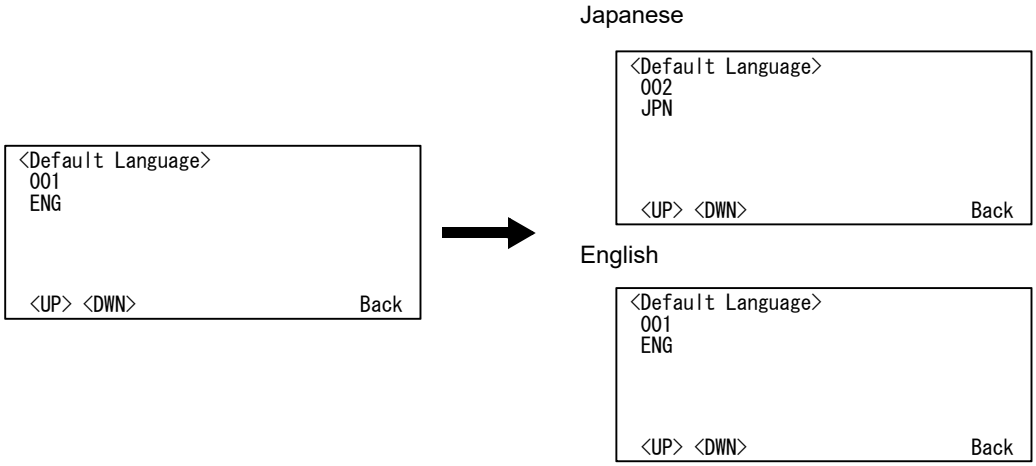
1) Press the [F1] key in the initial-setting screen, and select "1. Configuration".



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

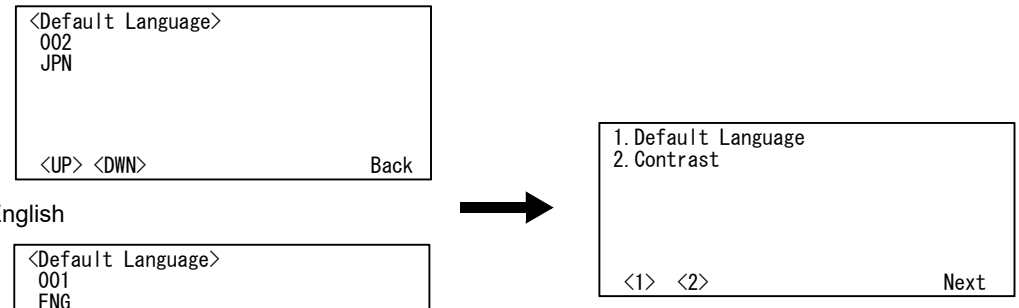


3) Display the "JPN" by [F1] or [F2] key, then language is set as Japanese. And, display the "ENG", then language is set as English.

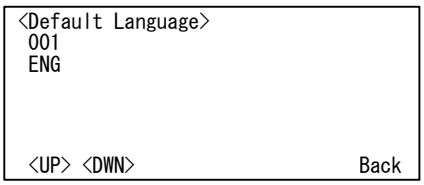


4) Press the [EXE] key, and fix it.

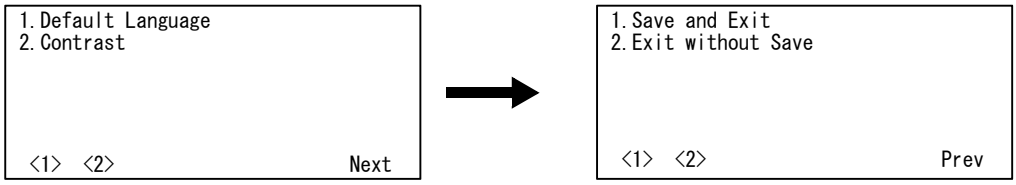
Japanese



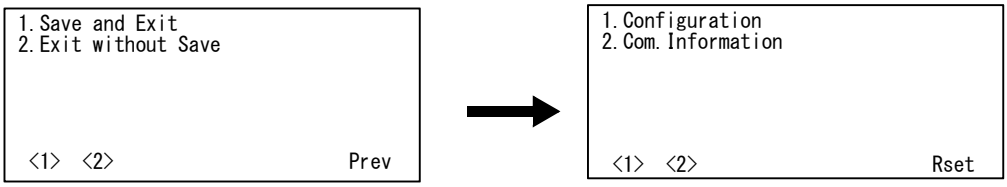
English



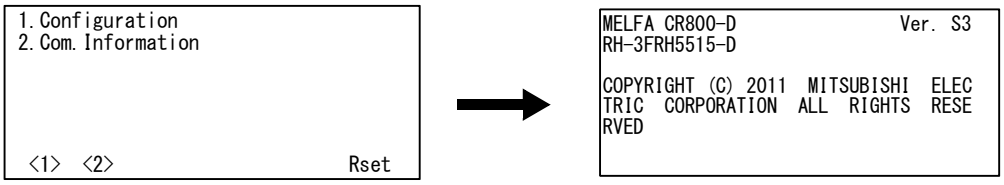
5) Press the [EXE] key, and display finish screen.



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.



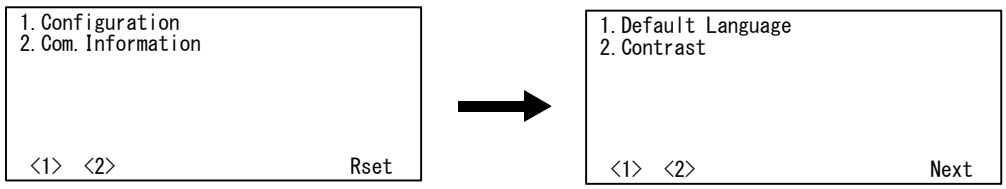
7) T/B starts in the language set up when the [EXE] key was pressed.



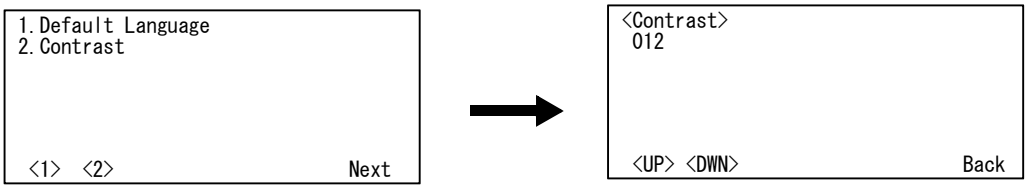
(2) Adjustment of contrast

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

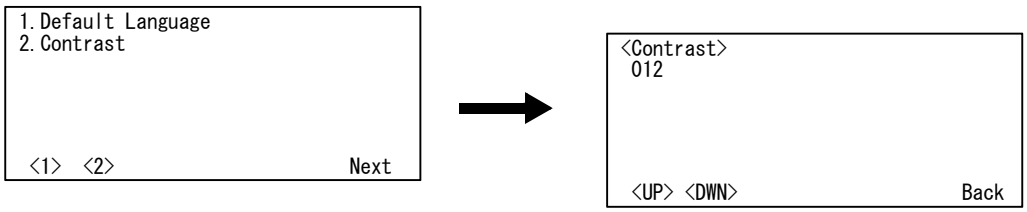
1) Press the [F1] key in the initial-setting screen, and select "1. Configuration".



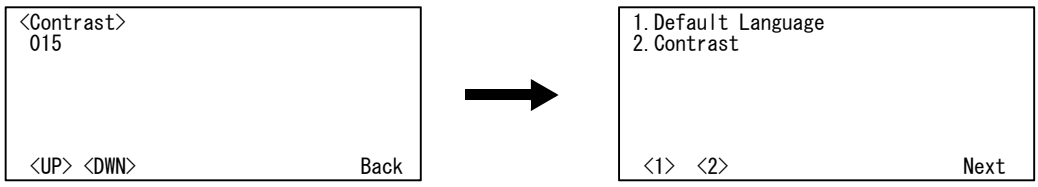
2) Press the [F2] key and select "2. Contrast."
The brightness set up now is displayed as the numerical value of 0 to 15.



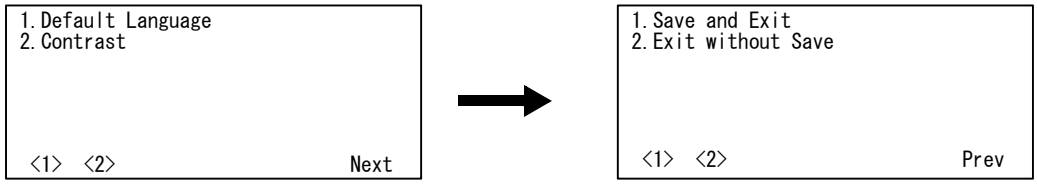
3) If it makes the screen bright, the [F1] key is pressed, if it makes it dark, press the [F2] key, and set it as the good brightness. It becomes so bright that the numerical value is large.



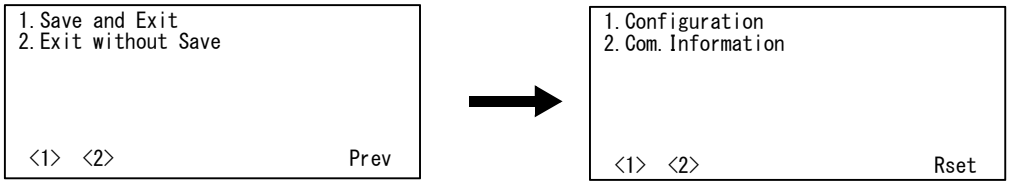
4) Press the [EXE] key and fix it.



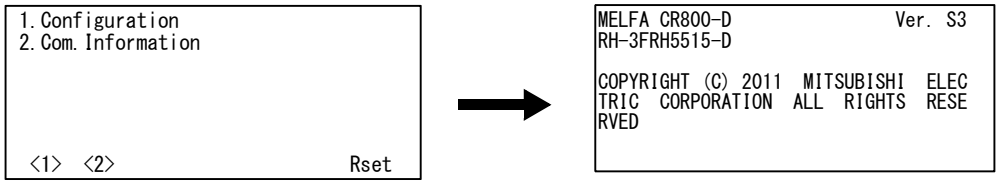
5) Press the [EXE] key, and display finish screen.



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 contrast set up when the [EXE] key was pressed.



4 MELFA-BASIC VI

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

4.1 MELFA-BASIC VI functions

The outline of the programming language "MELFA-BASIC VI" 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	----------	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

For the detailed description of each instruction, please refer to Page 170, "4.12 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

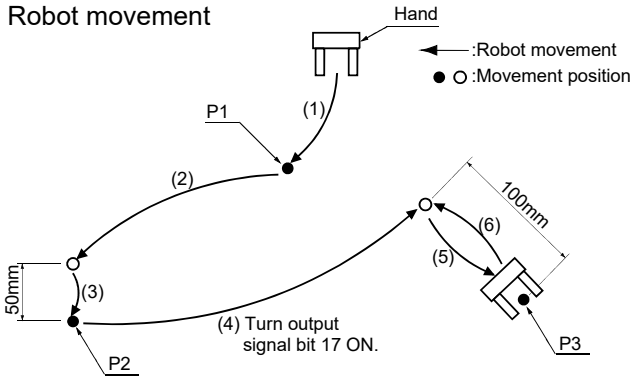
Command word	Explanation
Mov	The robot moves to the designated position with joint interpolation. It is possible to specify the interpolation form using the TYPE instruction. An appended statement Wth or WthIf can be designated

*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 122.
Mov P1*P2	' Moves to the position relatively converted from P1 to P2. Refer to Page 122.
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 WthIf 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 ..... (Default value: Long way around)	' Specify either roundabout (or shortcut) when the operation angle of each axis exceeds 180 deg..

*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.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 103, "(5) Acceleration/deceleration time and speed control"
Designate the acceleration/deceleration time.	Page 103, "(5) Acceleration/deceleration time and speed control"
Confirm that the target position is reached.	Page 105, "(6) Confirming that the target position is reached"
Continuously move to next position without stopping at target position.	Page 102, "(4) Continuous movement"
Move linearly.	Page 99, "(2) Linear interpolation movement"
Move while drawing a circle or arc.	Page 100, "(3) Circular interpolation movement"
Add a movement command to the process.	Page 337, " Wth (With)"

(2) Linear interpolation movement

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

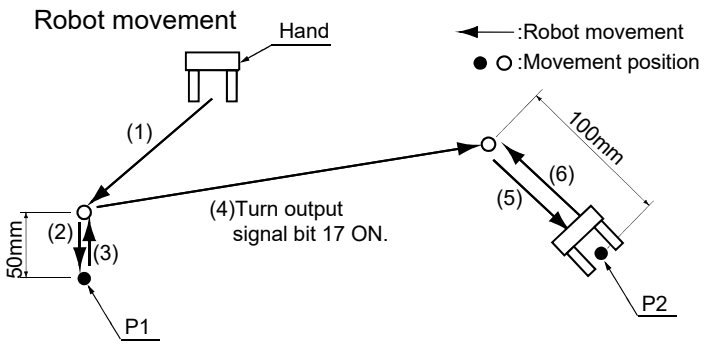
*Command word

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 WthIf 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 122.
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 WthIf 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.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 103, "(5) Acceleration/deceleration time and speed control"
Designate the acceleration/deceleration time.	Page 103, "(5) Acceleration/deceleration time and speed control"
Confirm that the target position is reached.	Page 105, "(6) Confirming that the target position is reached"
Continuously move to next position without stopping at target position.	Page 102, "(4) Continuous movement"
Move with joint interpolation.	Page 98, "(1) Joint interpolation movement"
Move while drawing a circle or arc.	Page 100, "(3) Circular interpolation movement"
Add a movement command to the process.	Page 337, " 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

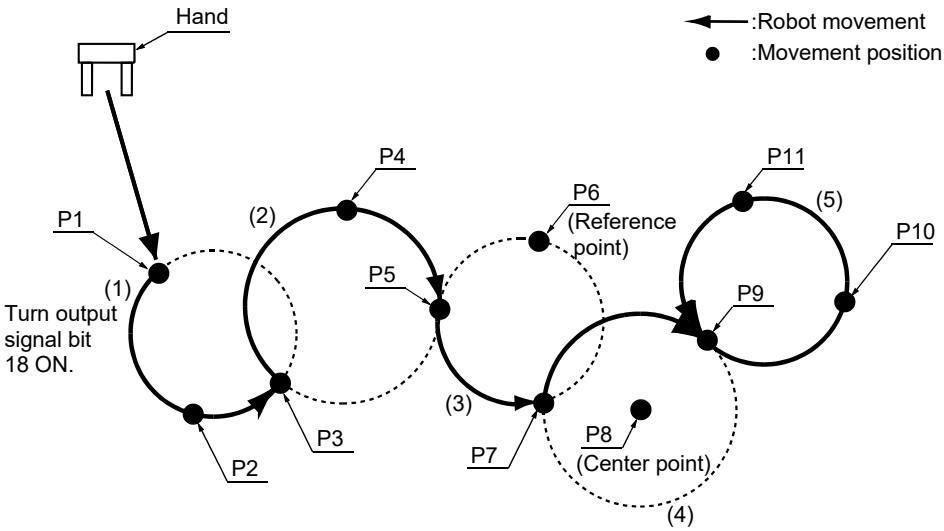
Command word	Explanation
Mvr	Designates the start point, transit point and end point, and moves the robot with circular interpolation in order of the start point - transit point - end point. It is possible to specify the interpolation form using the TYPE instruction. An appended statement Wth or WthIf 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 WthIf 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 WthIf 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 WthIf 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 WthIf 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 103, "(5) Acceleration/deceleration time and speed control"
Designate the acceleration/deceleration time.	Page 103, "(5) Acceleration/deceleration time and speed control"
Confirm that the target position is reached.	Page 105, "(6) Confirming that the target position is reached"
Continuously move to next position without stopping at target position.	Page 102, "(4) Continuous movement"
Move with joint interpolation.	Page 98, "(1) Joint interpolation movement"
Move linearly.	Page 99, "(2) Linear interpolation movement"
Add a movement command to the process.	Page 337, " 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

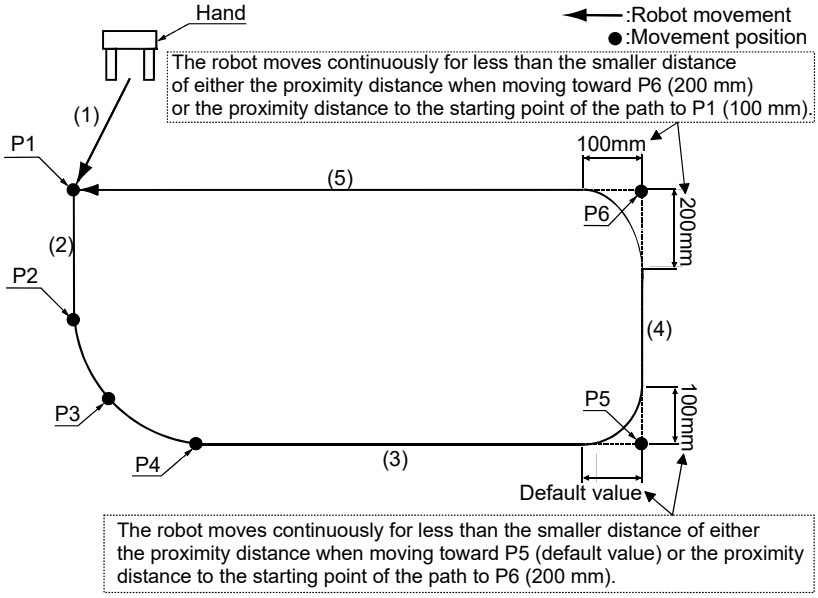
Command word	Explanation
Cnt	Designates the start and end of the continuous movement.

*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.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.

CAUTION

The robot's locus of movement may change with specified speed.
Especially as for the corner section, short cut distance may change. Therefore, when beginning automatic operation, moves at low speed at first, and you should gather speed slowly with being careful of interference with peripheral equipment.

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 103, "(5) Acceleration/deceleration time and speed control"
Designate the acceleration/deceleration time.	Page 103, "(5) Acceleration/deceleration time and speed control"
Confirm that the target position is reached.	Page 105, "(6) Confirming that the target position is reached"
Move with joint interpolation.	Page 98, "(1) Joint interpolation movement"
Move linearly.	Page 99, "(2) Linear interpolation movement"
Move while drawing a circle or arc.	Page 100, "(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

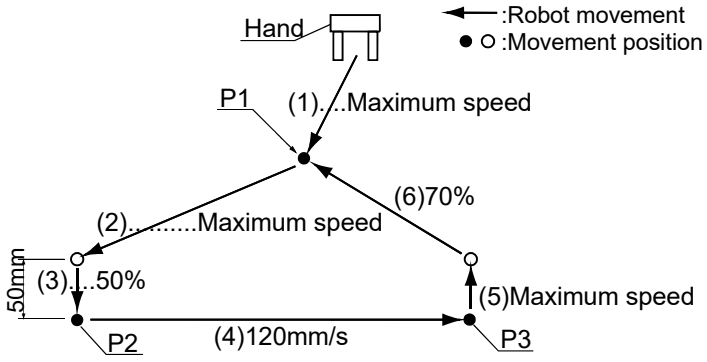
Controller (T/B) setting value x Ovrd command setting value x JOvrd command setting value.

*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.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 98, "(1) Joint interpolation movement"
Move linearly.	Page 99, "(2) Linear interpolation movement"
Move while drawing a circle or arc.	Page 100, "(3) Circular interpolation movement"
Continuously move to next position without stopping at target position.	Page 102, "(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

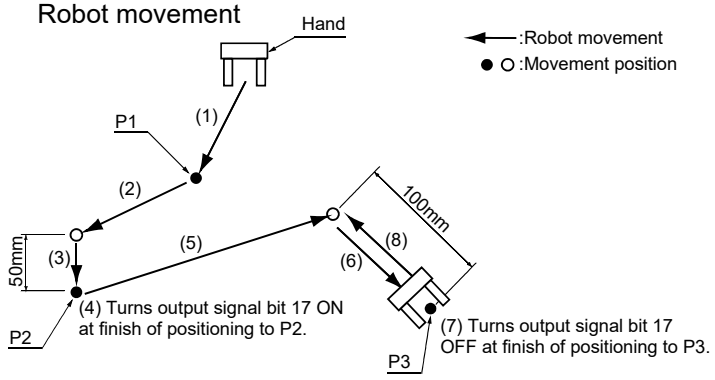
Command word	Explanation
Fine	Designates the positioning finish conditions with a No. of pulses. Specify a small number of pulses to allow more accurate positioning.
Mov and Dly	After the Mov movement command, command the Dly instruction (timer) to complete positioning .

*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.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 100, "(3) Circular interpolation movement"
Move linearly.	Page 99, "(2) Linear interpolation movement"
Continuously move to next position without stopping at target position.....	Page 102, "(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

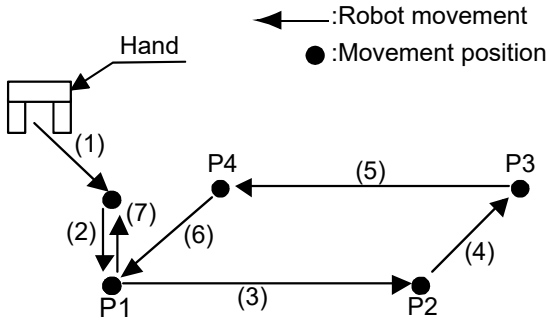
Command word	Explanation
Prec	This instruction specifies whether the high path accuracy mode should be enabled or disabled.

*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.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 ÇÑisableÇÑ.
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

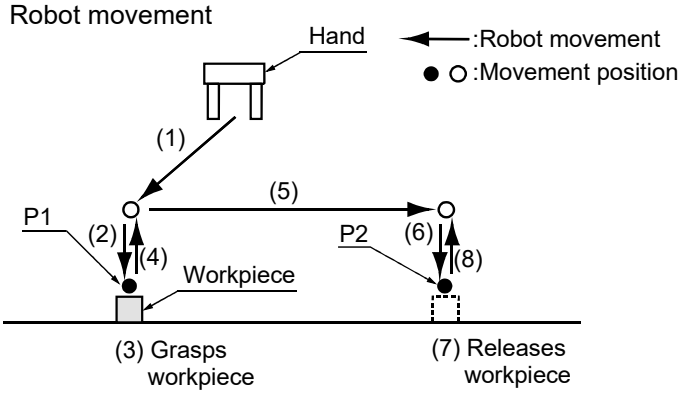
Command word	Explanation
HOpen	Opens the designated hand.
HClose	Closes the designated hand.
Tool	Sets the shape of the tool being used, and sets the control point.

*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.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 337, " 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.

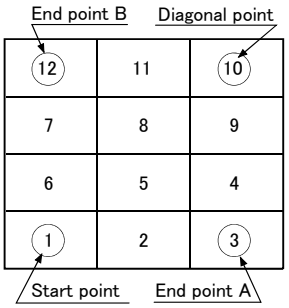
*Command word

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

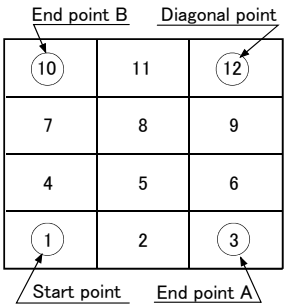
*Statement example

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.

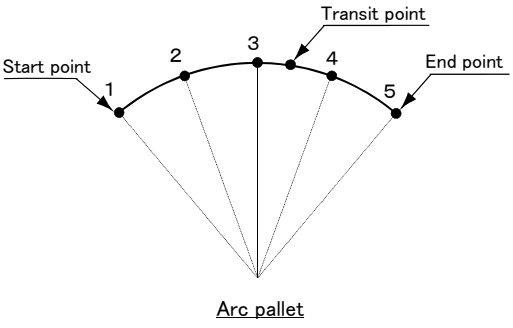
The relation of the position designation and a pallet pattern is shown below.



Pallet pattern = 1 (zigzag)



Pallet pattern = 2 (same direction)



Pallet pattern = 3 (arc pallet)

<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 gird positions are calculated by dividing the distance between -180 degrees and +180 degrees.

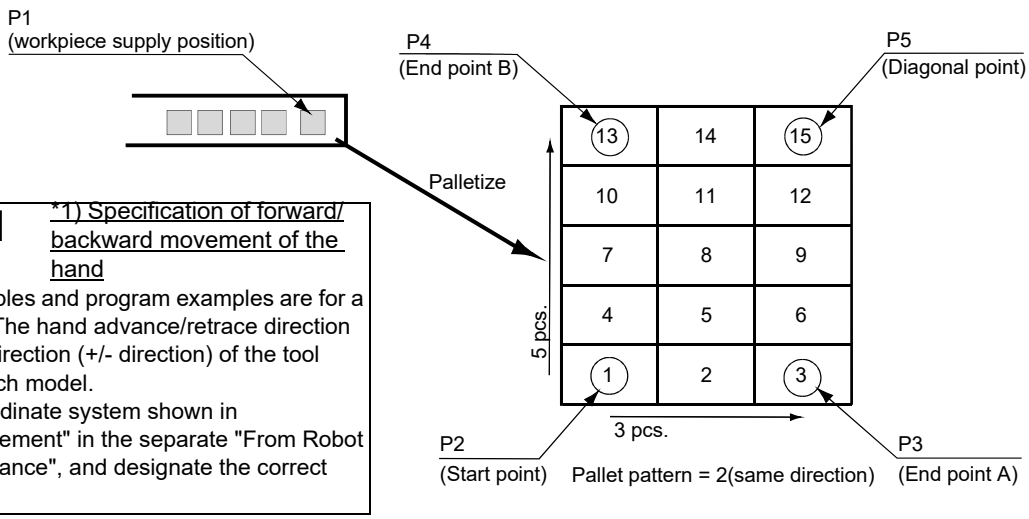
Whether a posture component is +180 degrees or -180 degrees, the posture will be the same. Use the same sign, either + or -, consistently for position data used to define a pallet.

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

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

The statement examples and program examples are for a vertical 6-axis robot.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.

⚠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 (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.

•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.

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 *FN	Jumps unconditionally to the label FN 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 : Break	If the value is 10, executes only between Case 10 and the next Case 11.
Case IS 11 : Break	If the value is 11, executes only between Case 11 and the next Case IS <5.
Case IS <5 : Break	If the value is smaller than 5, executes only between Case IS <5 and next Case 6 TO 9.
Case 6 TO 9 : Break	If value is between 6 and 9, executes only between Case 6 TO 9 and next Default.
Default : Break	If value does not correspond to any of the above, executes only between Default and next End Select.
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 114, "(2) Repetition"
Interrupt	Page 115, "(3) Interrupt"
Subroutine	Page 116, "(4) Subroutine"
External signal input	Page 118, "(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 113, "(1) Unconditional branching, conditional branching, waiting"
Interrupt	Page 115, "(3) Interrupt"
Input signal wait	Page 118, "(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 113, "(1) Unconditional branching, conditional branching, waiting"
Subroutine	Page 116, "(4) Subroutine"
Communication	Page 119, "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 115, "(3) Interrupt"
Communication	Page 119, "4.1.5 Communication"
Unconditional branching.....	Page 113, "(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 118, "(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 WthIf 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 337, " 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 157, "4.5 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 118, "(2) Output signals"
Branching with input signal	Page 113, "(1) Unconditional branching, conditional branching, waiting"
Interrupting with input signal	Page 115, "(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 157, "4.5 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 118, "(1) Input signals"
Timer	Page 117, "(5) Timer"

4.1.5 Communication

Data can be exchanged with an external device, such as a personal computer.

*Command word

Command word	Explanation
Open	Opens the communication line.
Close	Closes the communication line.
Print#	Outputs the data in the AscII format. CR is output as the end code.
Input#	Inputs the data in the AscII 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 116, "(4) Subroutine"
Interrupt	Page 115, "(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

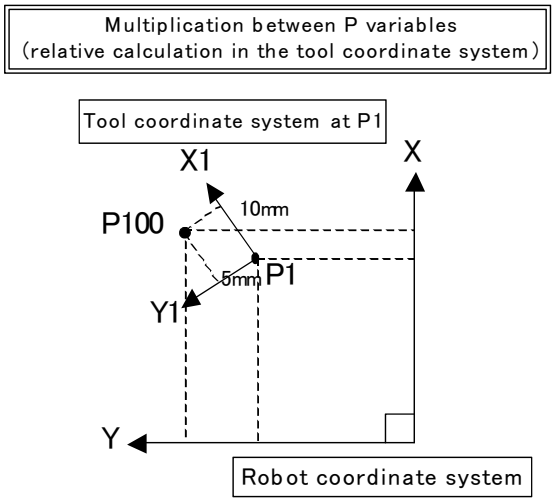
Class	Operator	Meaning	Statement example	
Substitution	=	The right side is substituted in the left side.	P1=P2 P5=P_Curr P10.Z=100.0 M1=1 STS$="OK"	'Substitute P2 in position variable P1. 'Substitute the current coordinate value in current position variable P5. 'Set the position variable P10 Z coordinate value to 100.0. 'Substitute value 1 in numeric variable M1. 'Substitute the character string OK in the character string variable STS$.
Numeric value operation	+	Add	P10=P1+P2 Mov P8+P9 M1=M1+1 STS$="ERR"+"001"	'GSubstitute the results obtained by adding the P1 and P2 coordinate elements to position variable P10. 'Move to the position obtained by adding the position variable P8 and P9 coordinate elements. 'Add 1 to the numeric variable M1. 'Add the character string 001 to the character string ERR and substitute in character string variable STS$.
	-	Subtract	P10=P1-P2 Mov P8-P9 M1=M1-1	'Substitute the results obtained by subtracting the P2 coordinate element from P1 in position variable P10. ' Move to the position obtained by subtracting the P9 coordinate element from the position variable P8. 'Subtract 1 from the numeric variable M1.
	*	Multiply	P1=P10*P3 M1=M1*5	'Substitute the relative conversion results from P10 to P3 in position variable P1. 'Multiple the numeric variable M1 value by 5.
	/	Divide	P1=P10/P3 M1=M1/2	'Substitute the reverse relative conversion results from P10 to P3 in position variable P1. 'Divide the numeric variable M1 value by 2.
	^	Exponential operation	M1=M1^2	'Square the numeric variable M1 value.
	\	Integer division	M1=M1\3	'Divide the numeric variable M1 value by 3 and make an integer (round down).
	MOD	Remainder operation	M1=M1 Mod 3	'Divide the numeric variable M1 value by 3 and leave redundant.
	-	Sign reversal	P1=-P1 M1=-M1	'Reverse the sign for each coordinate element in position variable P1. 'Reverse the sign for the numeric variable M1 value.
Comparison operation	=	Compare whether equal	If M1=1 Then *L1 If STS$="OK" Then *L2	'Branch to step *L1 if numeric variable M1 value is 1. 'Branch to step *L2 if character string in character string variable STS$ is OK.
	<> or ><	Compare whether not equal	If M1<>2 Then *L3 If STS$<>"OK" Then *L4	'Branch to step *L3 if numeric variable M1 value is 2. 'Branch to step *L4 if character string in character string variable STS$ is not OK.
	<	Compare whether smaller	If M1< 10 Then *L3 If Len(STS$)<3 Then *L4	'Branch to step *L3 if numeric variable M1 value is less than 10. 'Branch to step *L4 if No. of characters in character string STS$ variable is less than 3.
	>	Compare whether larger	If M1>9 Then *L3 If Len(STS$)>2 Then *L4	'Branch to step *L3 if numeric variable M1 value is more than 9. 'Branch to step *L4 if No. of characters in character string variable STS$ is more than 2.
	=< or <=	Compare whether equal to or less than	If M1<=10 Then *L3 If Len(STS$)<=5 Then *L4	'Branch to step *L3 if numeric variable M1 value is equal to or less than 10. 'Branch to step *L4 if No. of characters in character string variable STS$ is equal to or less then 5.
	=> or >=	Compare whether equal to or more than	If M1=>11 Then *L3 If Len(STS$)>=6 Then *L4	'Branch to step *L3 if numeric variable M1 value is equal to or more than 11. 'Branch to step *L4 if No. of characters in character string variable STS$ is equal to or more than 6.

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 122, "Relative calculation of position data (multiplication)".
Note 2) Please refer to Page 122, "Relative calculation of position data (Addition)".

(2) Relative calculation of position data (multiplication)

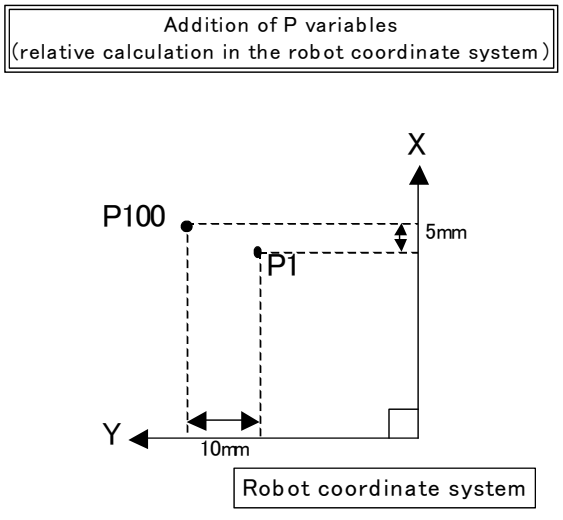
Numerical variables are calculated by the usual four arithmetic operations. The calculation of position variables involves coordinate conversions, however, not just the four basic arithmetic operations. This is explained using simple examples.



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.

CAUTION)
Relative calculation cannot be used in 5-axis robots, because correct values cannot be obtained.

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 98, "(1) Joint interpolation movement"
Linear interpolation movement	Page 99, "(2) Linear interpolation movement"
Circular interpolation movement	Page 100, "(3) Circular interpolation movement"
Stopping	Page 117, "(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.

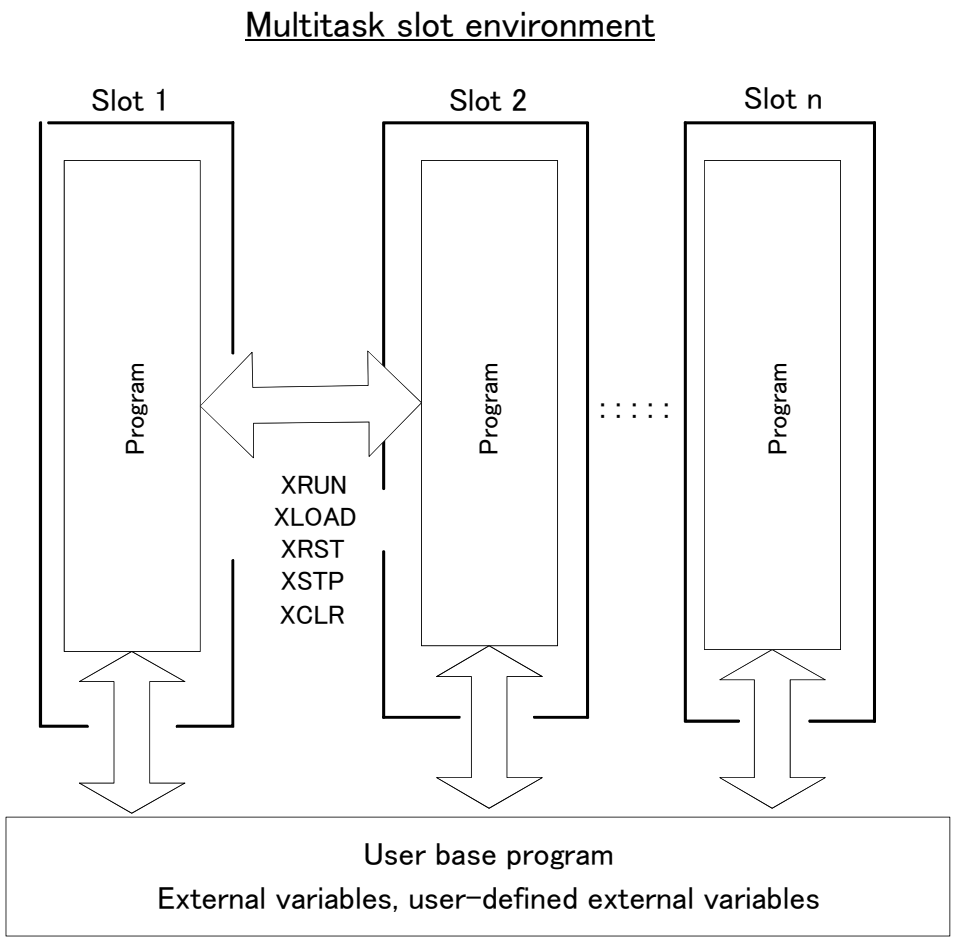


Fig.4-1:Multitask slot environment

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 T/B operation panel), the controller system unconditionally places the program selected with the operation panel in slot 1 and executes it.

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	This method starts parallel operation of the programs from a position in the program using a MELFA-BASIC VI command. The programs to be run in parallel can be designated, and a program running in parallel can be stopped. This method is effective when selecting the programs to be run in parallel according to the program flow. The related commands include the "XLoad (X Load)", "XRun (X Run)", "XStp (X Stop)", "XRst (X Reset)" and "XClr (X Clear)" commands. Refer to Page 170, "4.12 Detailed explanation of command words" in this manual for details.
2	Execution from T/B operation panel or external input/output signal	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. 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.
3	Executing automatically when the power is turned on	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. This eliminates the trouble of starting the programs in task slots used for monitoring input/output signals from the PLC side. 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 1 in order to execute X** instructions such as XRun and XLoad, the Servo instruction, and the Reset instruction.

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.

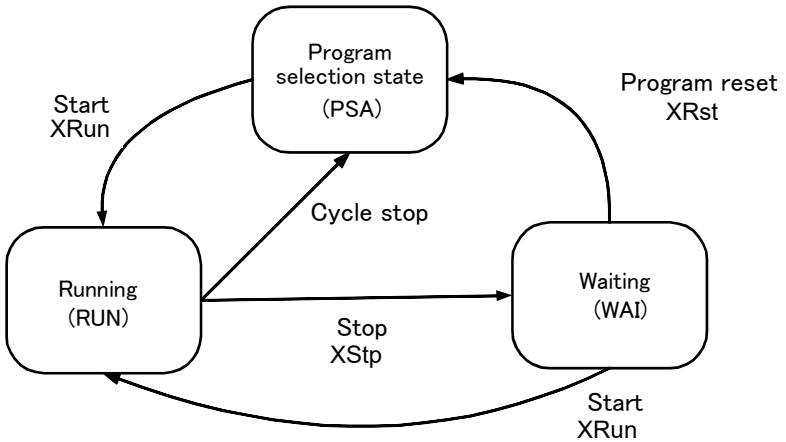


Fig.4-2:Operation state of each slot

<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 484, "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 484, "5 Functions set with parameters" for details.
3. Starting conditions	START	START : Execution of a program using the I/O START signal on the <operation panel> of the T/B	Select START when it is desired to start normally. Note1)
		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 XStp 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> of the T/B or by sending the dedicated input signal START will simultaneously start the execution of all the task slot programs for which "START" is set as the start condition.
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 625, "6.2 PLC link I/O function" 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: 5
SLT2=5, REP, ALWAYS, 5

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 Page 484, "5 Functions set with parameters"). 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 144, "4.3.22 External variables") and the user-defined external variables (refer to Page 145, "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>
   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.
```

```
<Slot 2> (Program of signals and variables)
   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.
            :
```

(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 157, "4.5 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 170, "4.12 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> of the T/B or with the dedicated input signal START, all the task slot programs for which "START" is set as the start condition of the task slot parameter 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 637, "6.3 Dedicated input/output" for details on the assignment of the dedicated input/output.

(2) Display of operation status

RUN and READY on the <operation panel> of the T/B and the dedicated input/output signals START and STOP indicate the operation conditions of the task slot programs for which "START" is set as the start condition of the "SLT*" parameter. If at least one program is operating, the operation status of the <operation panel> of the T/B changes to RUN and the dedicated output signal START turns on. If all the programs stop, the operation status of the <operation panel> of the T/B changes to READY 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 637, "6.3 Dedicated input/output" of this manual.

The status of the programs for which ALWAYS or ERROR is set as the start condition does not affect the LEDs that indicate the run and ready state on the <operation panel> of the T/B. 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

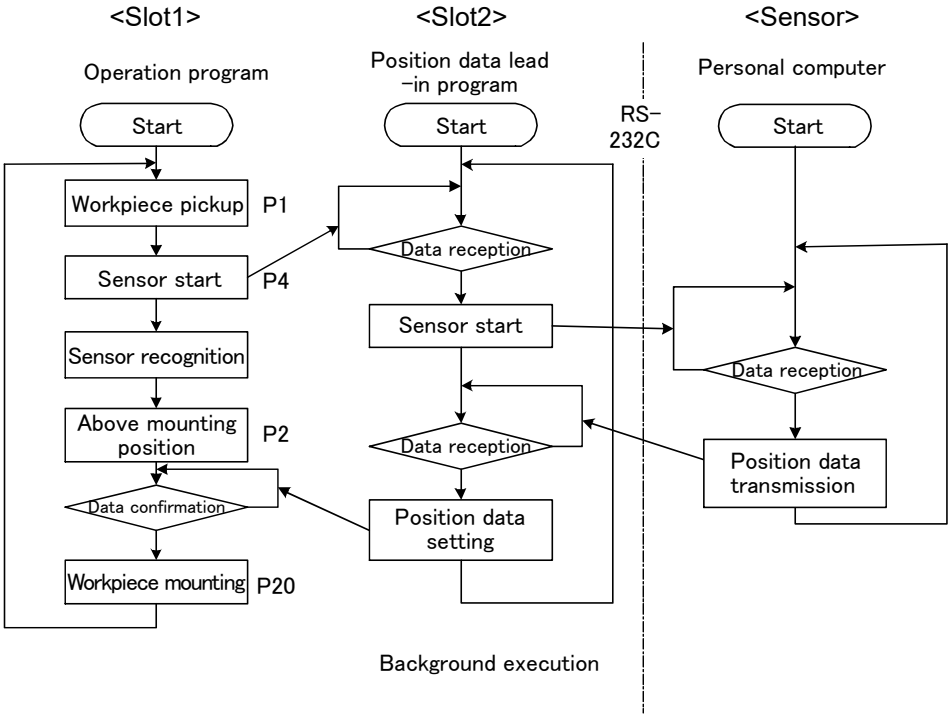
An example of the multitask execution is given in this section.

(1) Robot work details.

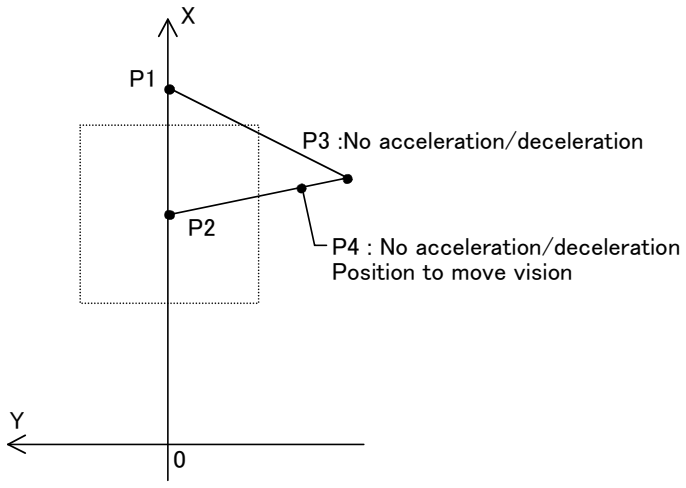
The robot programs are the "movement program" and "position data lead-in program".

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
21 If M_02#=0 Then GoTo *L       'Wait for interlock variable M_02 to reach 1
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
2 If M_01#=0 Then GoTo *R        'Wait for interlock variable M_01 to reach 1
3 Open "COM1:" AS #1             'Open Ethernet line
4 Dly M_03#                      'Hypothetical process timer (0.05 second)
5 Print #1,"SENS"                'Transmit character string "SENS" to Ethernet (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 Ethernet 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	Order of priority
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 920 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 380 Kbytes, which is the maximum size of one program. The capacity of the program edit area cannot be increased by mounting expansion memory.

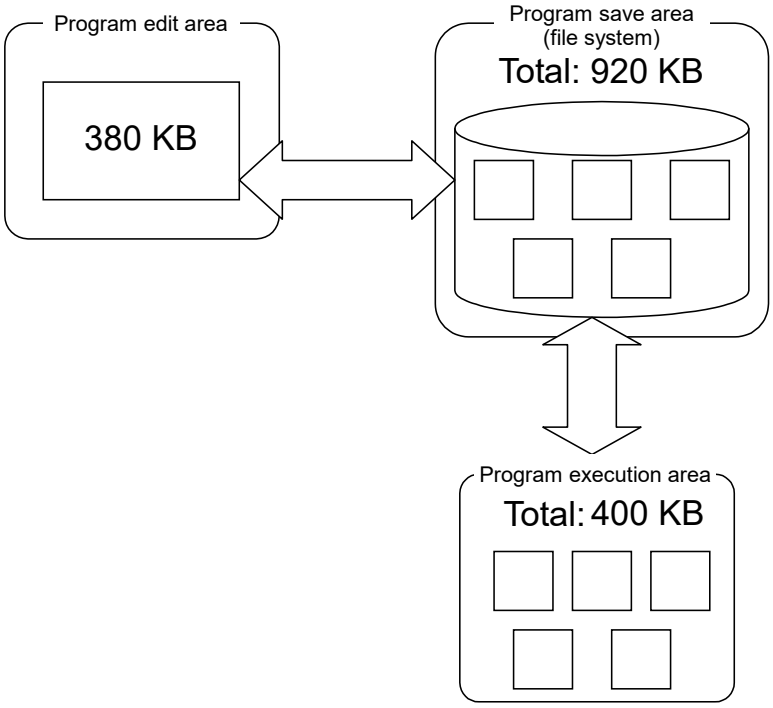
(3) Program execution area

The program execution area is used when operating a program automatically. The capacity of the program execution area is 400 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 400 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	
	Standard memory	With expansion memory
(1) Program save area	920 Kbytes	2 Mbytes
(2) Program edit area	380 Kbytes	
(3) Program execution area	400 Kbytes	

The capacity of each program can be checked with the teaching pendant and the Program Manager window of the Personal Computer Support Software (RT ToolBox3).



4.3 Detailed specifications of MELFA-BASIC VI

In this section, detailed explanations of the MELFA-BASIC VI 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. 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 the program is selected from the external input signal, specify the program name using numbers.

(2) Command statement

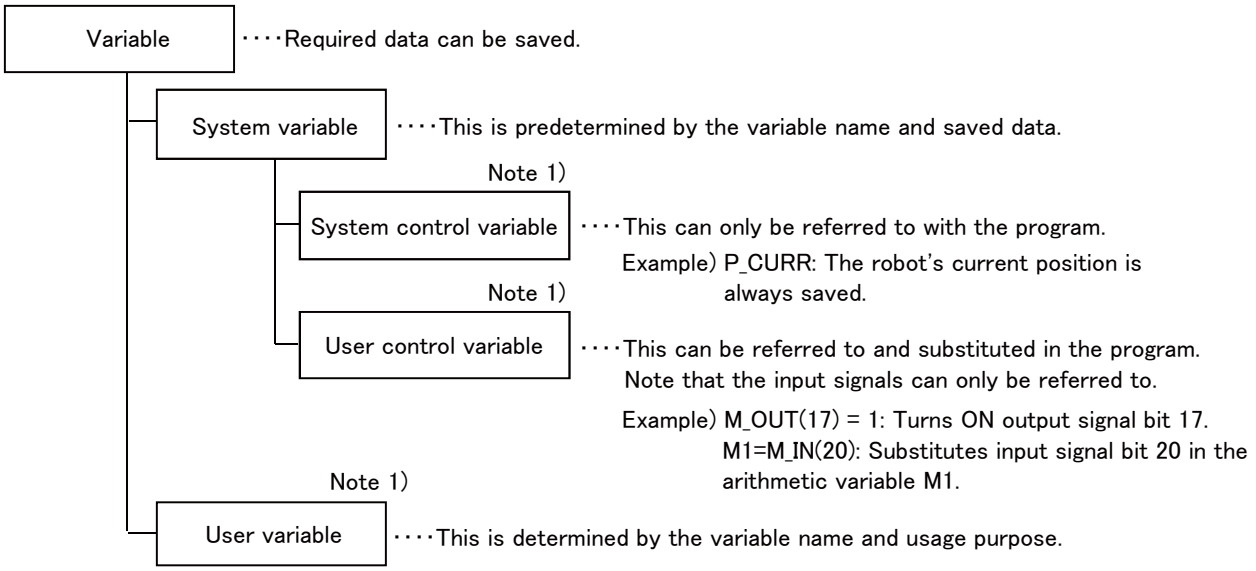
Example of constructing a statement

1 Mov P1 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.

Position type variable ····The robot's XYZ/work coordinate type data is saved.
The name of an XYZ variable/work coordinate variable starts with 'P' (or 'p')/'W' (or 'w').
Example) MOV P1: The robot moves to the position saved in variable name P1.

Joint type variable ····The robot's joint angle is saved. The variable name starts with "J".
Example) MOV J1: The robot moves to the position saved in variable name J1.

Numeric value type variable ····A numeric value (integer, real value, etc.) is saved. The variable name starts with "M".
Example) M1 = 1: The value 1 is substituted in variable name M1.

Character type variable ····A character string is saved. A "$" is added to the end of the variable name.
Example) C1$ = "ERROR": the character string "ERROR" is substituted in variable name C1$.

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 337, "Wth (With)" or Page 338, "WthIf (With If)", as well as each of the movement instructions (Mov (Move), Mva (Move Arch), Mvs (Move S), Mvr (Move R), Mvr2 (Move R 2), Mvr3 (Move R 3), Mvc (Move C)) 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.

For MELFA-BASIC VI, a step number (line number) cannot be specified as the destination of GoTo, GoSub, or other statements.

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 sixteen 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
  - "_" (underscore) cannot be used as 2nd character of the label 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	O	O	O
	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	X	O	O
Numerals	0 1 2 3 4 5 6 7 8 9	O	@Note1)	@Note2)
Symbols	" ' & ( ) * + - . , / : ; = < > ? @ ` [\ ] ^ { } ~ |	X	X	X
	! # $ % &	X	Available for type specification	X
	_(Underscore)	X	@Note3)	@Note4)
Spaces	Space character	X	X	X

Note1) Only alphabetical characters can be used as the first character of the variable name. Numerals can be used as the second and succeeding characters.

Note2) The head of the label name in the program can use only the English character. The numeral can be used in 2nd character or later.

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) "_" (underscore) can be used in 3rd character or later of the label name.

Refer to Page 132, "(1) Program name" for detail of program names, refer to Page 140, "4.3.15 Variables" for detail of variable names, and refer to Page 134, "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 144, "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.

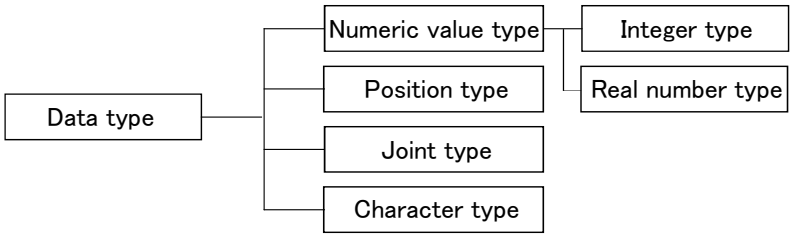
Example) M1 = P2.X ; Substitute the position variable P2.X coordinate element in numeric variable M1.

(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 170, "4.12 Detailed explanation of command words", the space is indicated with a "[]" where required.

4.3.8 Data type

In MELFA BASIC VI 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.

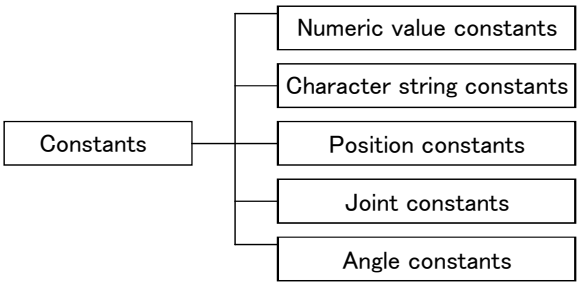


Example)

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) (0,0) [Position constants]
Joint type	J1 [Joint variables], (0,0,0,0,0,0) [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), 10.000000003# (Double-precision real number)

4.3.11 Character string constants

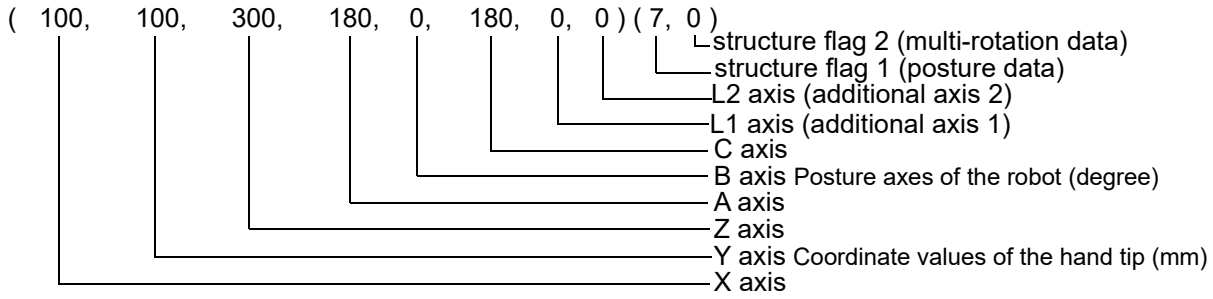
String constants are strings of characters enclosed by double quotation marks (").
Example) "ABCDEFGHIJKLMN" "123"

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 (XYZ/work coordinate 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 and the position of the work in the work coordinates. (The unit is mm.)

A, B, C: Posture data. This is the angle of the posture. The posture of the tip of the robot's hand in the XYZ coordinates and the posture of the work in the work coordinates. (The unit is deg.) Note1)

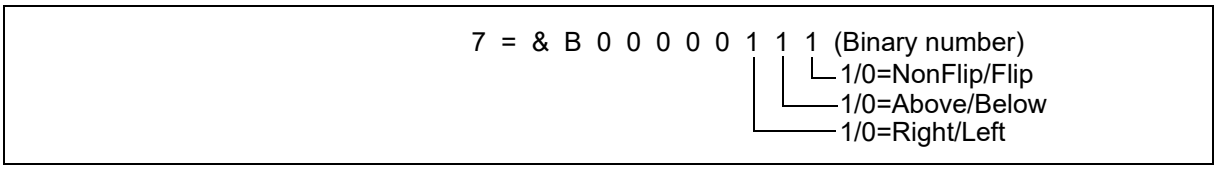
L1, L2: Additional axis data. These are the coordinates for additional axis 1 and additional axis 2, respectively. (The unit is mm or deg.)

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.

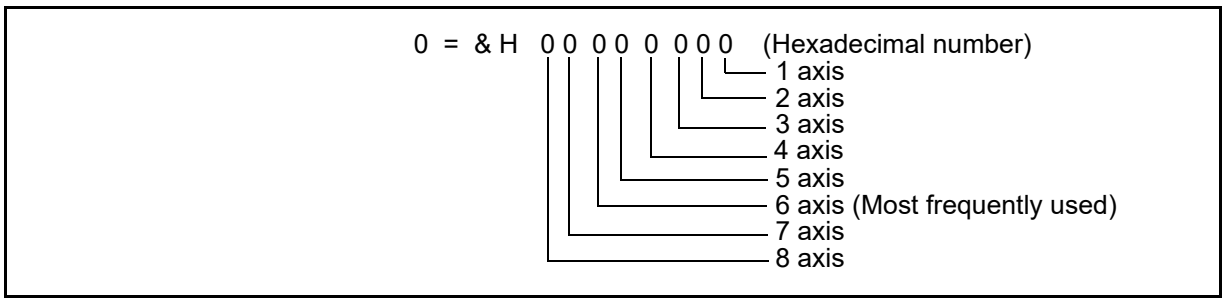
(2) Meaning of structure flag data type and meanings

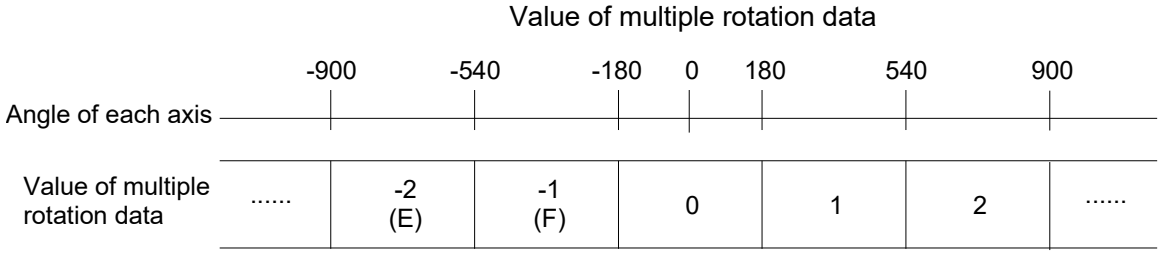
[Format] FL1, FL2

[Meaning] FL1: Posture data. It indicates the robot arm posture in the XYZ coordinates.



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.





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, posture, and additional axis 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

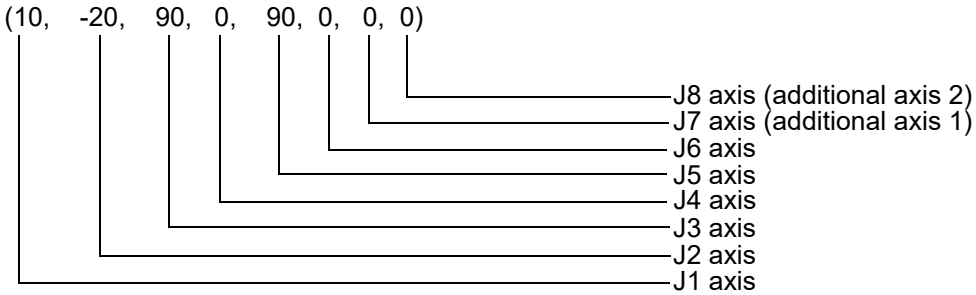
The coordinate, position, additional axis data and structure flag data are called the position element data. A variable cannot be contained in the position element data that configures the position constant.

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".

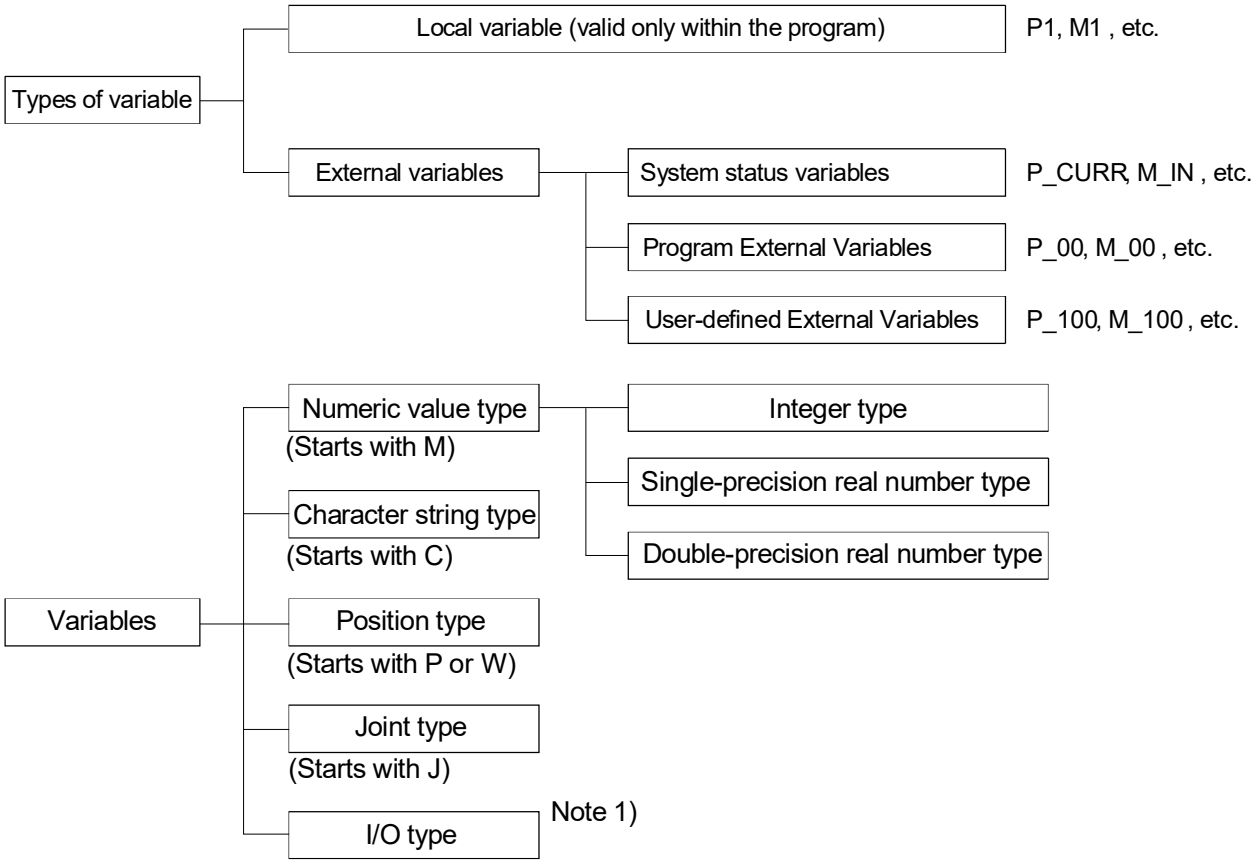
If written as 100Deg, this value becomes an angle and can be used as an argument of trigonometric functions.

Example) Sin(90Deg).....A 90 degree sine is indicated.

4.3.15 Variables

A variable name should be specified using up to 32 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 144, "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, C, or W are considered numeric value variables. In MELFA-BASIC VI, 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
&	Long-precsion real number type
!	Single-precsion real number type
#	Double-precsion 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 data of long-precision real number type is substituted into a numeric value variable of integer type and the value is exceeded, an error will occur at execution (L3110).
If data of double-precision real number type is substituted into a numeric value variable of single-precision real number type and the value is exceeded, an infinite value (signed "Inf") will be substituted.

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 (XYZ/work coordinate variables)

Variables whose names begin with character 'P' (or 'p') or 'W' (or 'w') are considered as XYZ/work coordinate variables. If it is defined by the Def Pos/Def Work instruction, it is possible to specify a name beginning with a character other than 'P' (or 'p') or 'W' (or 'w'). 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)    (Unit: degree)
         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_DIn	For referencing input signal double words (32-bit signals: Compatible with CC_Link)
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_DOut	For referencing/assigning output signal double words (32-bit signals: Compatible with CC_Link)

Please refer to the robot status variables Page 380, " M_In/M_Inb/M_In8/M_Inw/M_In16", Page 393, " M_Out/M_Outb/M_Out8/M_Outw/M_Out16", and Page 365, " 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-precsion 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 variables (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 137, "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/DOUBLE 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_PI, 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 Note1)	Qty.	Remarks
Position	P_00 to P_19 P_20 to P_39	20 20	
Position array (No. of elements 10)	P_100( ) to P_104( ) P_105( ) to P_109( )	5 5	Use the array element in the first dimensions.
Joint	J_00 to J_19 J_20 to J_39	20 20	
Joint array (No. of elements 10)	J_100( ) to J_104( ) J_105( ) to J_109( )	5 5	Use the array element in the first dimensions.
Numeric value	M_00 to M_19 M_20 to M_39	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( )	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	20 20	
Character string array (No. of elements 10)	C_100( ) to C_104( ) C_105( ) to C_109( )	5 5	Use the array element in the first dimensions.

Note1) 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 Scope

The scope of variables (accessible range) is as follows from the narrowest range: In a procedure → Outside a procedure (In a program → Outside a program) → User-defined external variables/Program external variables/System status variables. (Refer to Fig. 4-3.)

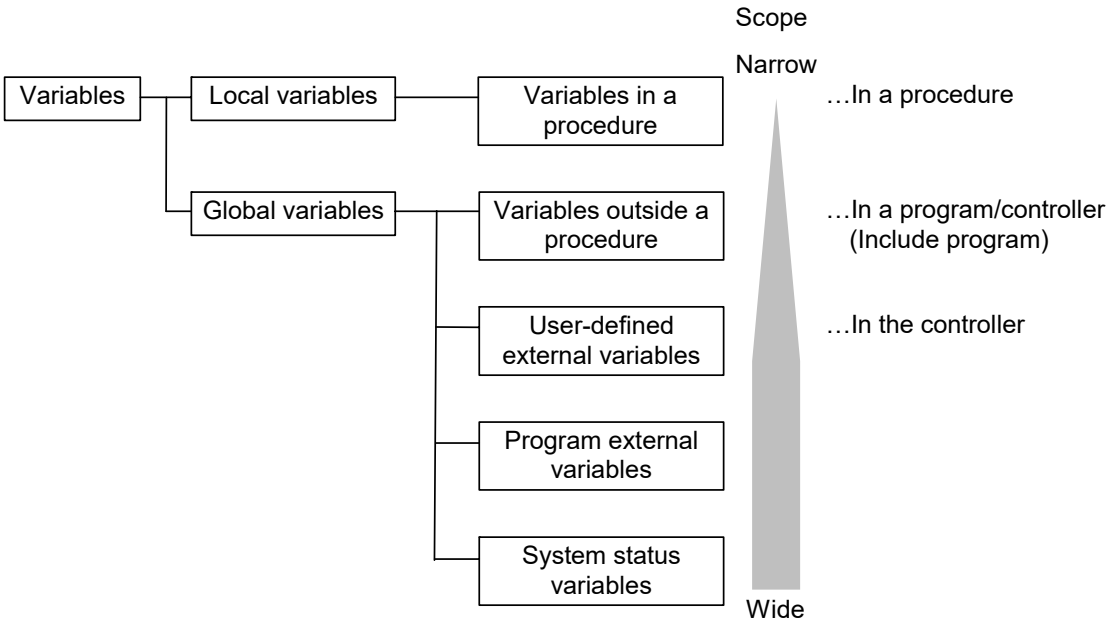


Fig.4-3:Scope of variables

4.3.27 Function procedure

The Function procedure is a set of instructions used as an independent function (processing). In the Function procedure, a processing can be used as a program part because variables can be used independently from the outside of the procedure and return values of functions can be defined. This allows a simple program structure.

When a Function procedure is defined, the entry point Note 1) of the program is at the Function Main procedure.

* If a procedure needs to be defined, a Function Main procedure must be defined.

* When a procedure is defined, the preprocessor Note 2) is executed before the entry point Note 1).

Note 1) When a program is started by specifying the program name, the program is executed from the first line or the Function Main procedure (without dummy argument) in the program. A point where a program starts its execution is called "Entry point".

Note 2) The preprocessor executes the instruction statements described outside the Function procedure.

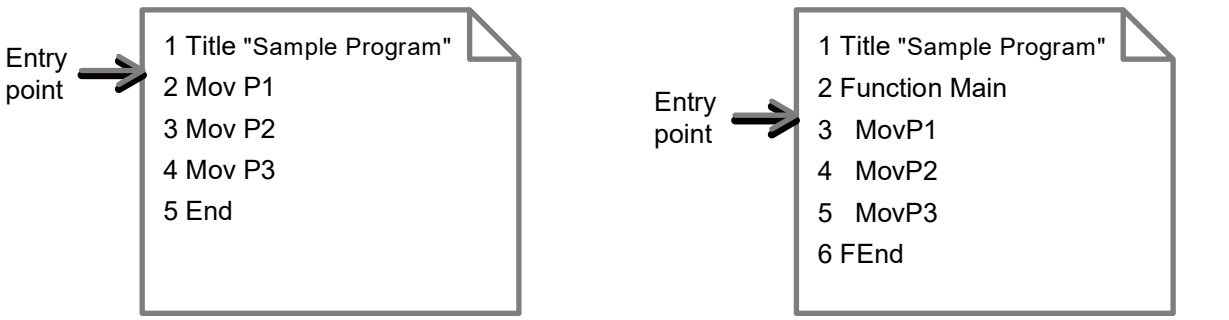


Fig.4-4:Entry point

The flow of processing with the Function procedure is shown in Fig. 4-5.

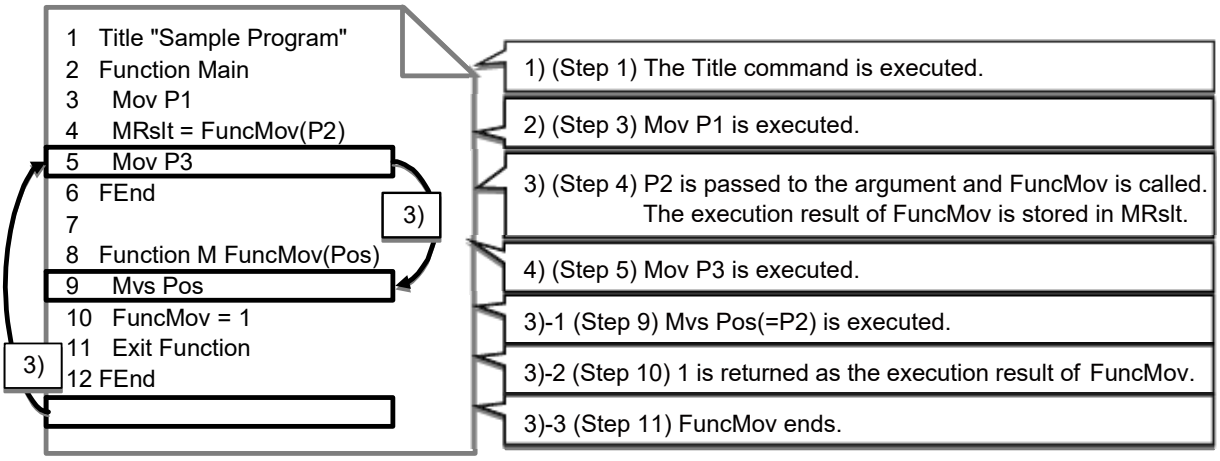


Fig.4-5:Flow of processing with the Function procedure

For the format of the Function procedure, refer to the section of the Function procedure in "Detailed explanation of command words".

4.3.28 #Include statement

A specified program can be read from programs. Frequently used programs (functions) and common programs (functions) can be managed in a library to use these programs as needed.

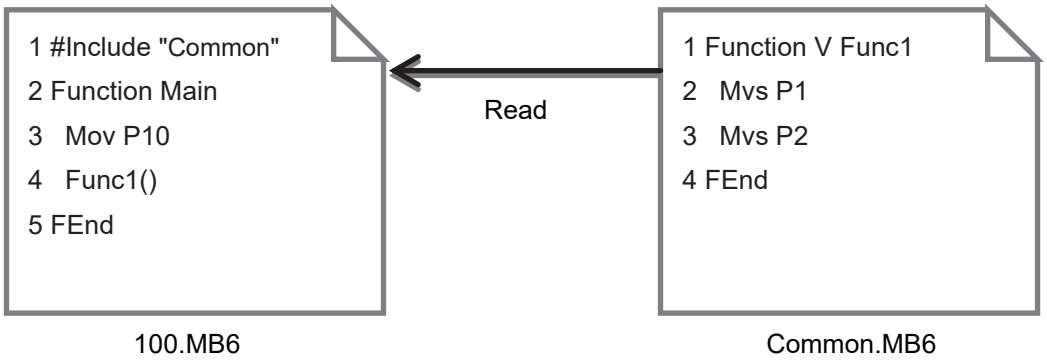


Fig.4-6:Reading a program (#Include statement)

* A program can be simultaneously specified and read from multiple programs.
For the details, refer to the section of Include statement in "Detailed explanation of command words".

4.4 Coordinate system description of the robot

4.4.1 About the robot's coordinate system

The robot's coordinate system has following four.

① World coordinate system: Origin is *1
The coordinate system as the standard for displaying the current position of robot. note 1)

② Base coordinate system: Origin is *2 (J1 axis rotation center on the bottom of the robot.)
A coordinate system established with reference to the robot mounting face. It is set by specifying parameter MEXBS with data on a center position for robot installation (base conversion data) as viewed from the world coordinate system or by executing a base command.
By default, because the base conversion data is set to zero (0), the world coordinate system is in agreement with the base coordinate system.

③ Mechanical interface coordinate system: Origin is *3 (J6 axis rotation center on the tool installation surface.)
A coordinate system established with reference to the robot's mechanical interface.

④ Tool coordinate system: Origin is *4
A coordinate system established with reference to the robot's mechanical interface. Its relation to the interface coordinate system is determined by the tool conversion data (i.e., by specified settings for parameter MEXTL or by the execution of a tool command.)

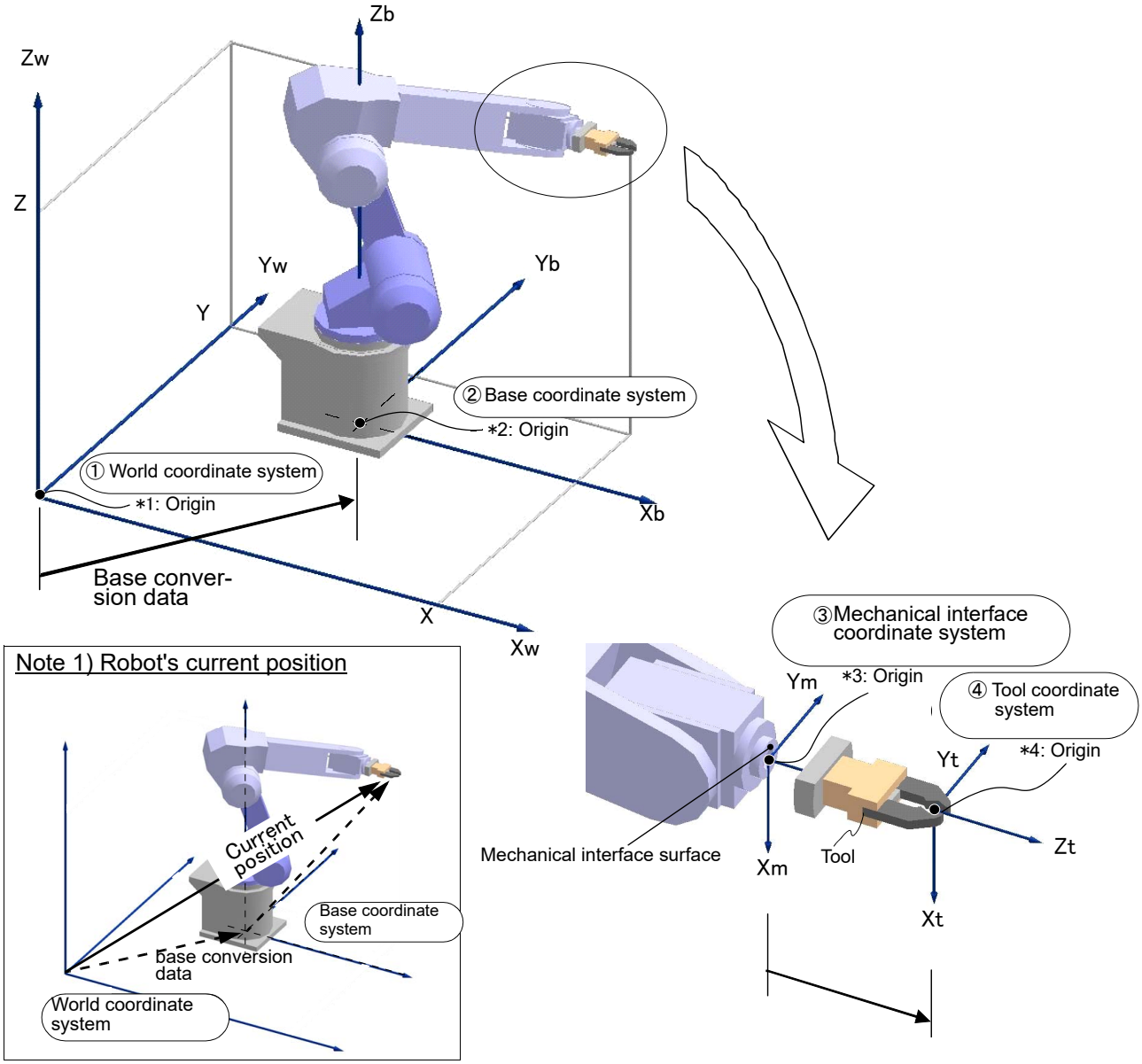


Fig.4-7:Robot's coordinate system

4.4.2 About base conversion

The base conversion permits the world coordinate system to be moved, when required, to the reference position of the work table or the work.
Under the control of this function, the robot's current position is treated as the one relative to the work table or the work. Therefore, where there are a plurality of work groups involved on which the robot shares an identical motional/positional relation, the robot can perform the same operations (sequence of motions) just with a change being made to the world coordinate system, i.e., without the need to be taught the operations (sequence of motions) for each work group. Change to the world coordinate system stated here are called base conversion, which is accomplished by specifying parameter MEXBS with base conversion data (coordinate values) or by executing a Base command.

Base conversion data to be specified should be data on the position of the origin point of the base coordinate system as viewed from a world coordinate system which is newly established. Thus, when you specify the data by using the robot's current position (using a Fram function, etc.), do so by inversely converting the position data [for example, Base Inv(P1)].
When you specify work coordinate system parameters WK1CRD - WK3CRD by executing a Base command, however, you do not have to make the inverse conversion yourself as it is done in internal processing. (Example: Base1 to Base 8)

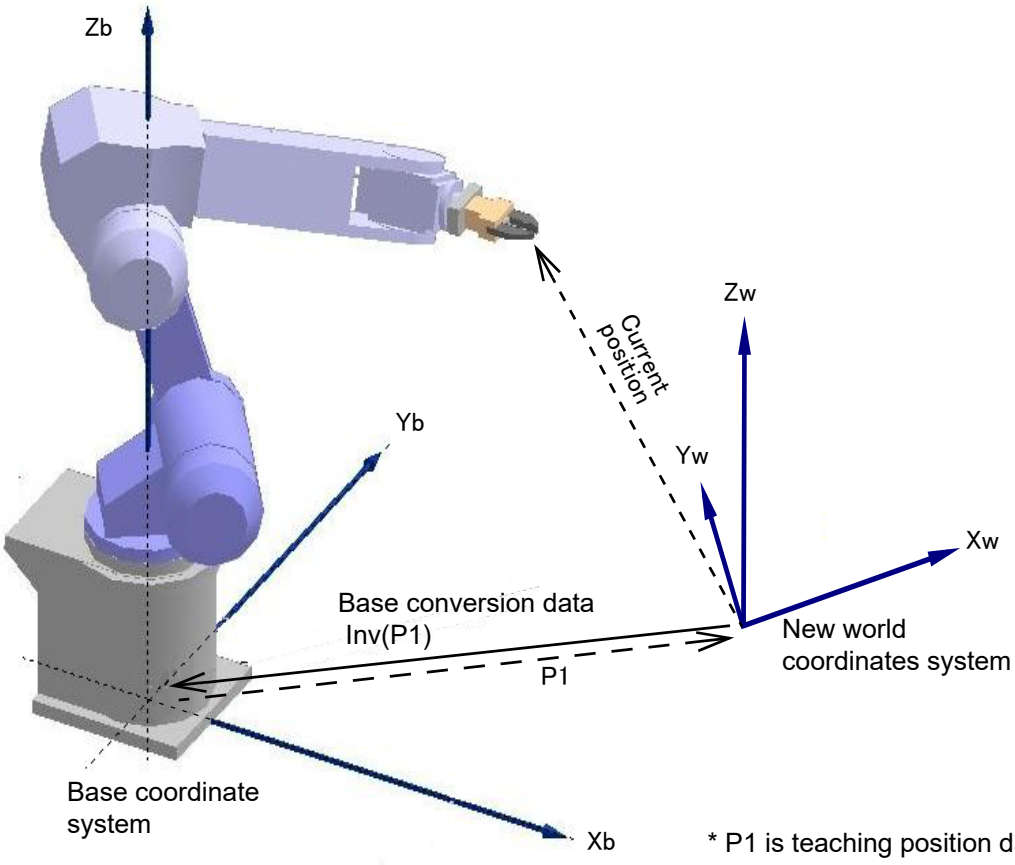


Fig.4-8:Base conversion

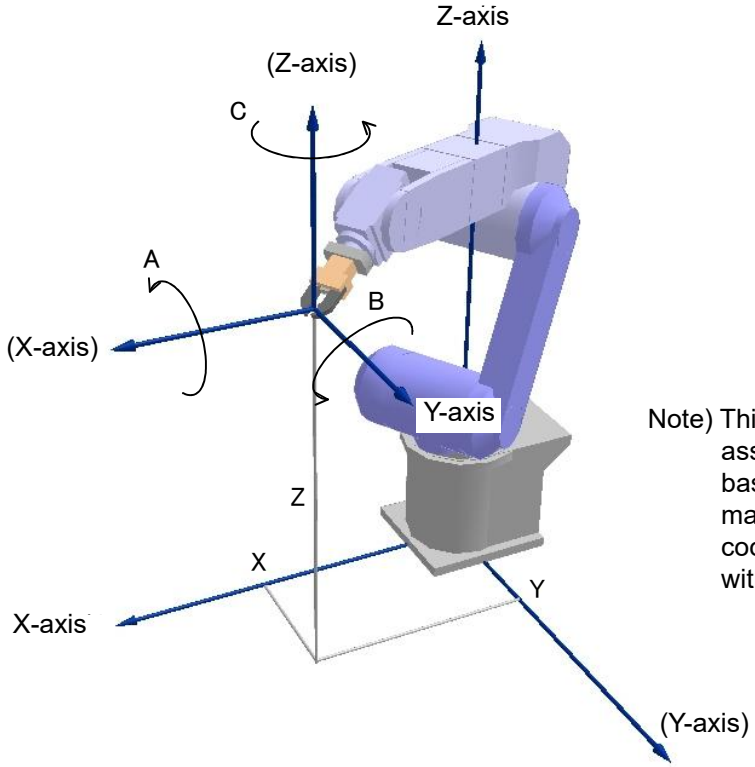
⚠CAUTION

Performing a base conversion changes the robot's current position to values that refer to the newly established world coordinate system. The target position specified by the movement command will also be treated as a position in the newly established world coordinate system.
As a result, data taught till then may become unusable as it is. The coordinate system when taught needs to be the same as the newly established coordinate system.
If they do not match, the robot can stray to unexpected positions, possibly resulting in property damage or personal injury.
When using the base conversion function, be sure to maintain positive control over relation between the base coordinate system subject to conversion and the position which the robot is taught to take so that a proper robot operation and an effective use of the base conversion function are insured.

4.4.3 About position data

Positional data for the robot is comprised of six elements which indicate the position of the hand's leading end (mechanical interface center where no tool setting is made) (X, Y, and Z) and the robot's posture (A, B, and C), plus a structure flag.
Each element constitutes reference data for the robot's world coordinate system.

[Meaning] X, Y, Z: Coordinate data. Position of the robot hand's leading end (in mm).
A, B, C: Posture data. Angle that defines the robot's posture (in degrees)
A → Angle of rotation on X axis
B → Angle of rotation on Y axis
C → Angle of rotation on Z axis



Note) This diagram is produced by assuming a situation in which no base conversion data setting is made, i.e., the robot's world coordinate system is in agreement with its base coordinate system.

Fig.4-9:Reference for posture angles

A, B, and C represent the robot's posture in the coordinate system of its hand's leading end (or flange center where no tool setting is made), each indicating a angle of rotation on the X axis, Y axis, and Z axis of the world coordinate system. Rotation corresponding to the direction of a right-handed screw when you look at the + side of each coordinate axis is "+" rotation. Also, rotation is set to take place in a predetermined sequence, and the amount of rotation is calculated (controlled) first for a rotation on the Z axis, followed by one on the Y axis and one on the Z axis in the order shown.

4.4.4 About tool coordinate system (mechanical interface coordinate system)

To set the robot's control point at the leading end of the hand attached thereto, it is necessary to make tool conversion data settings. Tool conversion data defines the position of the tool's leading end with reference to a mechanical interface coordinate system that is established for the flange. Therefore, our explanation deals with the mechanical interface coordinate system in the first place.

In helping you to understand the tool coordinate system, explanation here uses a vertical 6-axis robot by way of example. For details about other models (vertical 5-axis robot, horizontal articulated arm robot, and others), refer to Page 517, "5.6 Standard Tool Coordinates".

(1) Mechanical interface coordinate system

As shown in Fig. 4-10 , a coordinate system having its origin point chosen at the center of the flange is called a mechanical interface coordinate system. X axis, Y axis and Z axis of the mechanical interface coordinate system are denoted as Xm, Ym and Zm, respectively.

Zm is an axis which passes through the flange center and is perpendicular to the flange face. The direction which goes outside from the flange face is + (plus). Xm and Ym are coplanar with the flange face. A line joining the flange center with the positioning pin hole is represented by Xm axis. "+" direction of the Xm axis is opposite to the pin hole as seen from the center.

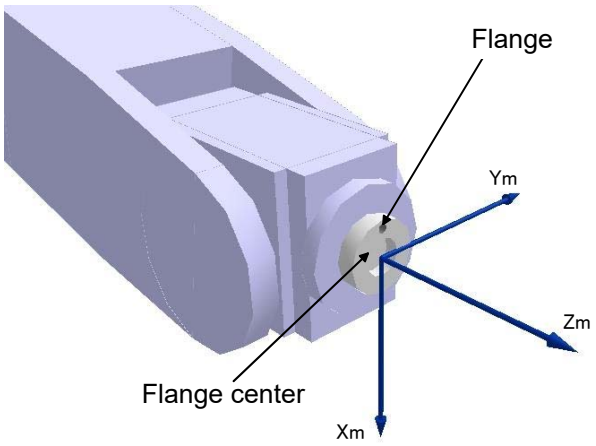


Fig.4-10:Mechanical interface coordinate system

When the flange rotates, the mechanical interface coordinate system rotates, as well. (Fig. 4-11)

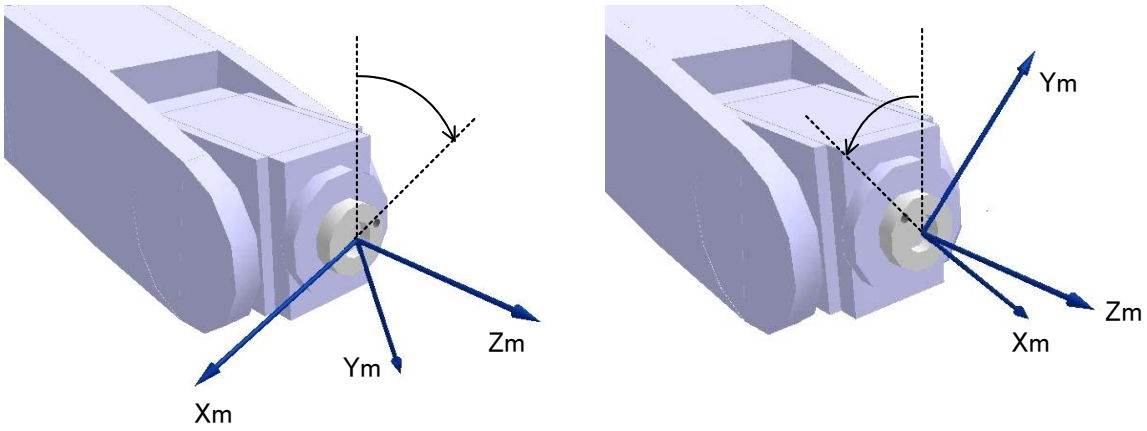


Fig.4-11:Rotation of flange and mechanical interface coordinate system

(2) Tool coordinate system

A tool coordinate system is one that is defined for the leading end of the robot hand (control point for the robot hand).
It is obtained by shifting the origin point of a mechanical interface coordinate system to the leading end of the robot hand (control point hand) and adding given rotational elements.
X axis, Y axis and Z axis of the tool coordinate system are denoted as Xt, Yt and Zt, respectively.

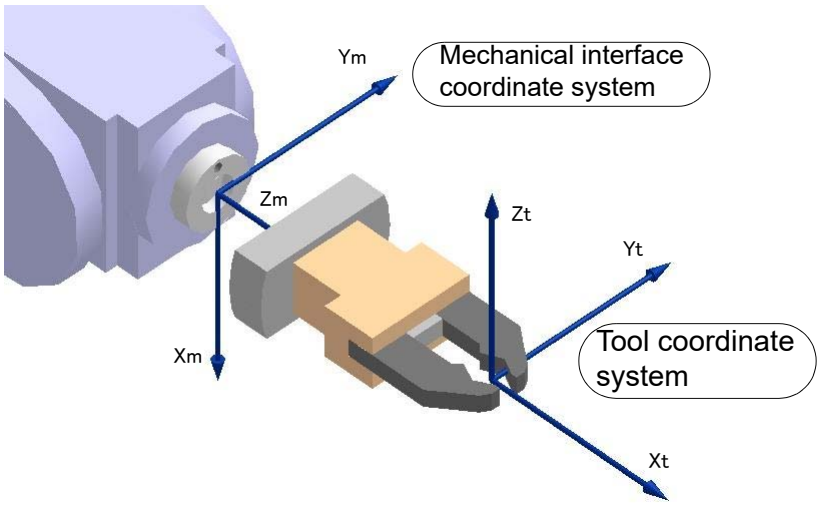


Fig.4-12:Mechanical interface coordinate system and tool coordinate system

Tool conversion data consists of the same elements as position data.

X, Y, Z: Amount of shift. Amount by which the origin point of the mechanical interface coordinate system is shifted to agree with that of the tool coordinate system (in mm).

A, B, C: Angle of rotation of each coordinate axis (in degrees)

A → Angle of rotation on X axis

B → Angle of rotation on Y axis

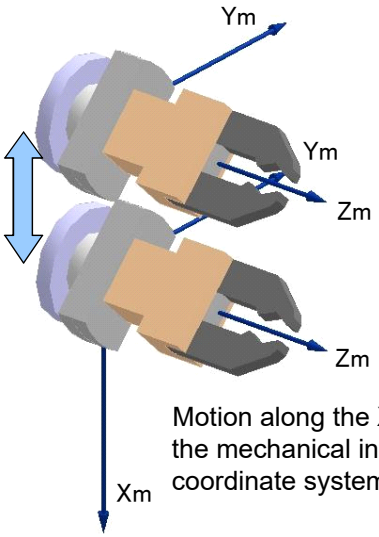
C → Angle of rotation on Z axis

(3) Effects of use of tool coordinate system

1) Jogging and teaching operations
When placing the robot into tool-jog mode, you can let it operate in the direction of the face of the robot hand. This makes it easier to adjust the posture of the robot hand toward the work concerned or the posture of the work being held by the robot hand.

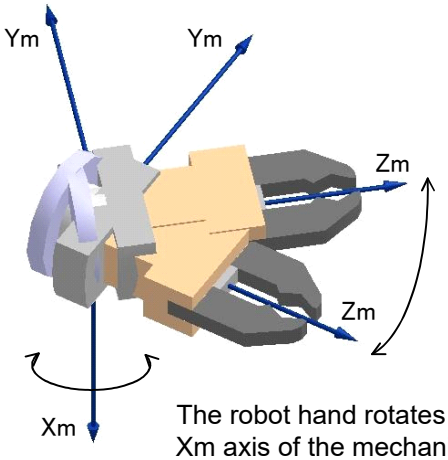
In the case of tool conversion data setting being not made

Travel in the direction of X axis



Motion along the Xm axis of the mechanical interface coordinate system

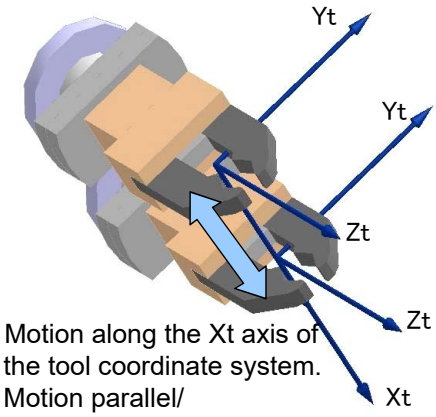
Travel in the direction of A axis



The robot hand rotates on the Xm axis of the mechanical interface coordinate system, thus having a wide range of motion at its leading end.

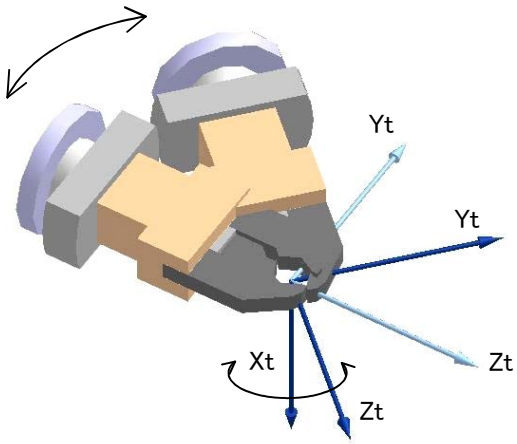
In the case of tool conversion data setting being made

Travel in the direction of X axis



Motion along the Xt axis of the tool coordinate system. Motion parallel/ perpendicular to the face of the robot hand assures a register with the orientation of the work.

Travel in the direction of A axis



The robot hand rotates on the Xt axis of the tool coordinate system. Rotational motion on the leading end of the robot hand permits a change of posture without the need to displace the work from its original position.

Fig.4-13:Tool jogging operation with/without tool conversion data

2) Automatic operation

Travel command permits you to set robot motion during the removal or transfer of processed work by specifying approach/pullout distance settings. Approach or pullout takes place in the direction of the Z axis of the robot's tool coordinate system.

To move the robot hand to a point 50mm over the work transfer position as shown in Fig. 4-14, the following indication is used:

```
Mov  P1,50
```

This means that the robot hand should move +50mm in the direction of the Z axis at P1 (tool coordinate system).

Setting the direction of the Z axis of the tool coordinate system to suit the orientation of work being process and/or the operating condition of the robot leads to an improved workability.

In the example shown in Fig. 4-14, because the robot hand is oriented laterally to insert or remove the work, the direction of the Z axis of the tool coordinate system is chosen to agree with the orientation of the work.

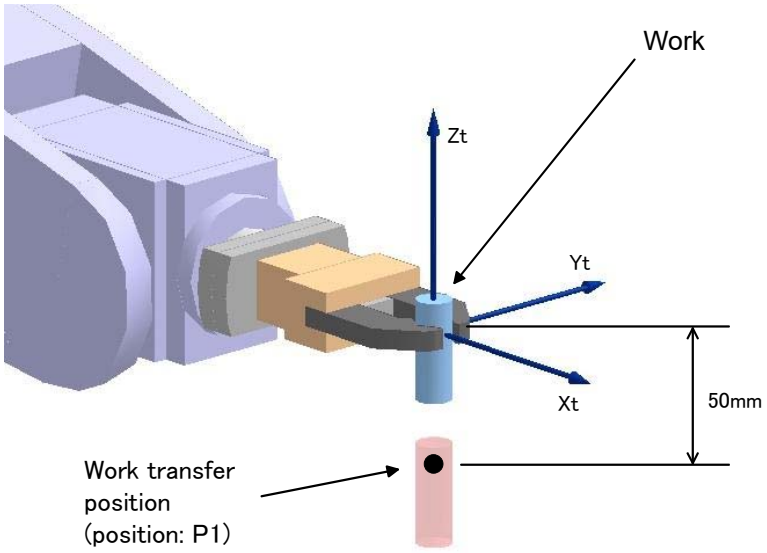


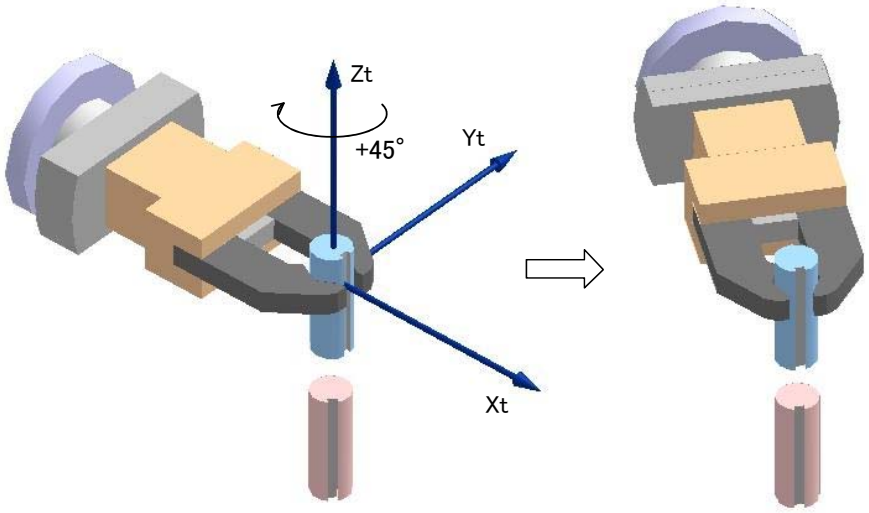
Fig.4-14:Approach/pullout motion

Making tool conversion data settings will come in useful when you have to make changes to the posture of your work as in work phasing, as well.

To achieve work phasing by turning the work on its center axis as shown in Fig. 4-15, the following indication is used:

```
Mov  P1*(0,0,0,0,0,45)
```

"*(0, 0, 0, 0, 0, 45)" means that a position calculation should be carried out at "*" and that C out of (X, Y, Z, A, B, C) should be rotated 45 degree. As C represents a rotation on the Z axis, the robot comes to rotate 45 degree on the Z axis (Zt axis of tool coordinate system) at P1.



(a) Position of P1 (b) Position of Mov P1* (0, 0, 0, 0, 0, 45)

Fig.4-15:Rotational motion in tool coordinate system

4.5 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	427
2	J_Curr	Mechanism No.(1 to 3)	Current position (joint)	R	Joint type	351
3	J_ECurr	Mechanism No.(1 to 3)	Current encoder pulse position	R	Joint type	355
4	J_Fbc	Mechanism No.(1 to 3)	Joint position generated based on the feedback value from the servo	R	Joint type	356
5	J_AmpFbc	Mechanism No.(1 to 3)	Current feedback value	R	Joint type	356
6	P_Fbc	Mechanism No.(1 to 3)	XYZ position generated based on the feedback value from the servo	R	Position type	430
7	M_Fbd	Mechanism No.(1 to 3)	Distance between commanded position and feedback position	R	Position type	371
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	361
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	362
10	P_Tool	Mechanism No.(1 to 3)	Currently designated tool conversion data	R	Position type	437
11	P_Base	Mechanism No.(1 to 3)	Currently designated base conversion data	R	Position type	423
12	P_NTool	Mechanism No.(1 to 3)	System default value (tool conversion data)	R	Position type	437
13	P_NBase	Mechanism No.(1 to 3)	System default value (base conversion data)	R	Position type	423
14	M_Tool	Mechanism No.(1 to 3)	Tool No. (1 to 16)	RW	Integer type	408
15	J_ColMxl	Mechanism No.(1 to 3)	Difference between estimated torque and actual torque	R	Joint type, %	352
16	M_ColSts	Mechanism No.(1 to 3)	Collision detection status (1: Colliding, 0: Others)	R	Integer type	363
17	P_ColDir	Mechanism No.(1 to 3)	Movement direction at collision	R	Position type	425
18	P_CordR	Mechanism No.(1 to 3)	In interference avoidance function, The robot's base coordinate system origin point looking from common coordinate system.	R	Position type	426
19	P_CurrR	Mechanism No.(1 to 3)	In interference avoidance function, Local robot's current position looking from the common coordinate system.	R	Position type	428
20	M_Cavsts	Mechanism No.(1 to 3)	In interference avoidance function, The CPU number of interfering robot when interference is detected.	RW	Integer type	360
21	P_CavDir	Mechanism No.(1 to 3)	In interference avoidance function, The direction which the robot was moving when interference is detected.	R	Position type	424
22	M_OPOvrd	None	Speed override on the operation panel (0 to 100%)	R	Integer type, %	384
23	M_Ovrd	Slot No.(1to 32)	Override in currently designated program (0 to 100%)	R	Integer type, %	384
24	M_JOvrd	Slot No.(1to 32)	Currently designated joint override (0 to 100%)	R	Integer type, %	384
25	M_NOvrd	Slot No.(1to 32)	System default value (default value of M_Ovrd) (%)	R	Single-precision real number type, %	384
26	M_NJovrd	Slot No.(1to 32)	System default value (default value of M_JOvrd) (%)	R	Single-precision real number type, %	384
27	M_Wupov	Mechanism No.(1 to 3)	Warm-up operation override (50 to 100%)	R	Single-precision real number type, %	415

No	Variable name	Array designation Note1)	Details	Attribute Note2)	Data type, Unit	Page
28	M_Wuprt	Mechanism No.(1 to 3)	Time until the warm-up operation status is canceled (sec.)	R	Single-precision real number type, sec	416
29	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	417
30	M_Ratio	Slot No.(1to 32)	Fraction of the current movement left before reaching the target position (%)	R	Integer type, %	398
31	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	400
32	M_Spd	Slot No.(1to 32)	Current specified speed (valid only for linear/ circular interpolation)	R	Single-precision real number type, mm/s	403
33	M_NSpd	Slot No.(1to 32)	System default value (default value of M_Spd) (mm/s)	R	Single-precision real number type, mm/s	403
34	M_RSpd	Slot No.(1to 32)	Current directive speed (mm/s)	R	Single-precision real number type,mm/s	403
35	M_Acl	Slot No.(1to 32)	Current specified acceleration rate (%)	R	Single-precision real number type, %	357
36	M_DAcl	Slot No.(1to 32)	Current specified deceleration rate (%)	R	Single-precision real number type, %	357
37	M_NAcl	Slot No.(1to 32)	System default value (default value of M_Acl) (%)	R	Single-precision real number type, %	357
38	M_NDAcl	Slot No.(1to 32)	System default value (default value of M_DAcl) (%)	R	Single-precision real number type, %	357
39	M_AclSts	Slot No.(1to 32)	Current acceleration/deceleration status 0 = Stopped, 1 = Accelerating, 2 = Constant speed, 3=Decelerating	R	Integer type	357
40	M_SetAdl	Axis No.(1 to 8)	Specify the acceleration/deceleration time ratio (%) of each axis.	RW	Single-precision real number type, %	401
41	M_LdFact	Axis No.(1 to 8)	The load factor of the servo motor of each axis. (%)	R	Single-precision real number type, %	385
42	M_Run	Slot No.(1to 32)	Operation status (1: Operating, 0: Not operating)	R	Integer type	400
43	M_Wai	Slot No.(1to 32)	Pause status (1: Pausing, 0: Not pausing)	R	Integer type	414
44	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	397
45	M_Cys	Slot No.(1to 32)	Cycle operation status (1: Cycle operation, 0: Non-cycle operation)	R	Integer type	364
46	M_Cstp	None	Cycle stop operation status (1: Cycle stop, 0: Not cycle stop)	R	Integer type	364
47	C_Prg	Slot No.(1to 32)	Execution program name	R	Character string type	349
48	M_Line	Slot No.(1to 32)	Currently executed line No.	R	Integer type	387
49	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	402
50	M_BrkCq	None	Result of the BREAK instruction (1: BREAK, 0: None)	R	Integer type	359
51	M_Err	None	Error occurring (1: An error has occurred, 0: No errors have occurred)	R	Integer type	369

No	Variable name	Array designation Note1)	Details	Attribute Note2)	Data type, Unit	Page
52	M_ErrLvl	None	Reads an error level. ・ S/W version R1c or before (SQ series) / S1c or before (SD series) No error / Caution / Low / High = 0/1/2/3 ・ S/W version R1d or later(SQ series) / S1d or later(SD series) No error / Caution / Low / High / Caution1 / Low1 / High1 = 0/1/2/3/4/5/6	R	Integer type	369
53	M_Errno	None	Reads an error number.	R	Integer type	369
54	M_Svo	Mechanism No.(1 to 3)	Servo motor power on (1: Servo power on, 0: Servo power off)	R	Integer type	406
55	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	409
56	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	410
57	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	380
58	M_Inb/ M_In8	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	380
59	M_Inw/ M_In16	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	380
60	M_In32	Input No.(0 to 32767)	Use this variable when inputting external input signals (32-bit units) numerically General-purpose bit device: double word signal input The signal numbers will be 6000s for CC-Link	R	Integer type	382
61	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	393
62	M_Outb/ M_Out8	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	393
63	M_Outw/ M_Out16	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	393
64	M_Out32	Output No.(0 to 32767)	Use this variable when outputting numerical value to external output signals (32-bit units) General-purpose bit device: double word signal input The signal numbers will be 6000s for CC-Link	RW	Integer type	395
65	M_DIn	Input No.(from 6000)	CC-Link's remote register: Input register Cannot use in CR800-R/Q series.	R	Integer type	365
66	M_DOut	Output No.(from 6000)	CC-Link's remote register: output register Cannot use in CR800-R/Q series.	RW	Integer type	365
67	M_HndCq	Input No.(1 to 8)	Returns a hand check input signal.	R	Integer type	379
68	P_Safe	Mechanism No.(1 to 3)	Returns an safe point position.	RW	Position type	437

No	Variable name	Array designation Note1)	Details	Attribute Note2)	Data type, Unit	Page
69	J_Origin	Mechanism No.(1 to 3)	Returns the joint coordinate data when setting the origin.	R	Joint type	356
70	M_Open	File No.(1 to 8)	Returns the open status of the specified file or the communication line.	R	Integer type	392
71	M_NvOpen	Vision sensor No.(1 to 8)	Returns the line connection status of the vision sensor.	R	Integer type	390
72	C_Mecha	Mechanism No.(1 to 3)	Returns the type name of the robot.	R	Character string type	349
73	C_Maker	None	Shows manufacturer information (a string of up to 64 characters).	R	Character string type	348
74	C_User	None	Returns the content of the parameter "USERMSG."(a string of up to 64 characters).	R	Character string type	350
75	C_Date	None	Current date expressed as "year/month/date".	R	Character string type	348
76	C_Time	None	Current time expressed as "time/minute/second".	R	Character string type	350
77	M_BTime	None	Returns the remaining battery capacity time (hours).	R	Integer type, Time	359
78	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	407
79	P_Zero	None	A variable whose position coordinate values (X, Y, Z, A, B, C, FL1, FL2) are all 0	R	Position type	441
80	M_PI	None	Circumference rate (3.1415...)	R	Double-precision real number type	397
81	M_Exp	None	Base of natural logarithm (2.71828...)	R	Double-precision real number type	371
82	M_G	None	Specific gravity constant (9.80665)	R	Double-precision real number type	372
83	M_On	None	1 is always set	R	Integer type	391
84	M_Off	None	0 is always set	R	Integer type	391
85	M_Mode	None	Contains the status of the mode selector switch. MANUAL/AUTOMATIC (O/P)/AUTOMATIC (External)=(1/2/3)	R	Integer type	388
86	M_SplPno	Mechanism No.(1 to 3)	During the spline interpolation movement, the transit point No. passed through most recently is returned.	R	Integer type	404
87	M_SplVar	Mechanism No.(1 to 3)	During the spline interpolation movement, a random numeric value set for the transit point is returned. The random value can be held until the value is set again.	RW	Integer type	405
88	P_WkCord	Work coordinates No. (1 to 8)	By designating a work coordinates number, work coordinates values concerned are read, or work coordinate values are specified.	RW	Position type	440
89	M_ESpd	Mechanism No.(1 to 3)	During the Ex-T control/Ex-T spline interpolation movement, the transit speed at current Ex-T coordinate system is returned.	R	Single-precision real number type,mm/s	370
90	P_ECord	Mechanism No.(1 to 3)	During the Ex-T control/Ex-T spline interpolation movement, the Ex-T coordinate system origin data currently used is returned.	R	Position type	429
91	M_Mxt	Target CPU No. (2 to 4)	Used for inter-robot synchronization	RW	Integer type	389
92	P_GCurr	Target CPU No. (2 to 4)	Master robot current position	R	Position type	431

No	Variable name	Array designation Note1)	Details	Attribute Note2)	Data type, Unit	Page
93	M_Gps	Monitoring No. (1 to 8)	Returns the number of the position data stored in the P_GpsX() for the monitoring number defined in the Def Gps command, using the get-position-quick function (GPS function). ("X" indicates the same number as the target monitoring number from 1 to 8.)	R	Integer type	376
94	P_Gps1 to P_Gps8	Position number (1 to 400)	Returns XYZ coordinate data for the current position data when the condition defined in the Def Gps command is met, using the get-position-quick function (GPS function). (Up to 400 position data can be saved.)	R	Integer type	434
95	M_Map1 to M_Map8	Number of segments (1 to 130)	Returns the segment number in which a workpiece is present in the condition defined in the Def Map command, using the get-position-quick function (GPS function).	R	Integer type	387

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.

4.5.1 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 logic operation (if true) *Switch ON *Input/output signal ON *Hand open (supply current to the hand) *Settings for enable/valid such as for interrupts	*Result of comparison operation (if false) *Result of logic operation (if false) *Switch OFF *Input/output signal OFF *Hand close (do not supply current to the hand) *Settings for disable/invalid such as for interrupts

4.6 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 FNMADD(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	443	Numeric value
	Cint (<Numeric expression>)	Rounds off the decimal value and converts into an integer.	450	
	Deg (<Numeric expression:radian>)	Converts the angle unit from radian (rad) to degree (deg).	453	
	Exp (<Numeric expression>)	Calculates the value of the expression's exponential function	454	
	Fix (<Numeric expression>)	Produces an integer section	455	
	Int (<Numeric expression>)	Produces the largest integer that does not exceed the value in the expression.	457	
	Len(<Character string expression>)	Produces the length of the character string.	459	
	Ln (<Numeric expression>)	Produces the logarithm.	460	
	Log (<Numeric expression>)	Produces the common logarithm.	460	
	Max (<Numeric expression>...)	Obtains the max. value from a random number of arguments.	461	
	Min (<Numeric expression>...)	Obtains the min. value from a random number of arguments.	462	
	Rad (<Numeric expression: deg.>)	Converts the angle unit from radian (rad) to degree (deg).	466	
	Sgn (<Numeric expression>)	Checks the sign of the number in the expression	473	
	Sqr (<Numeric expression>)	Calculates the square root	477	
	Rnd (<Numeric expression>)	Produces the random numbers.	468	
	SplSpd(<Spline No.>)	Obtains the maximum speed which can be specified without an error from the path point data registered in a random spline file.	476	
Trigonometric functions	ACos(<Numeric expression>)	Calculates the arc cosine. Unit: radian Definition range: -1.0 to +1.0, Value range: 0 to PI	444	Numeric value
	ASin(<Numeric expression>)	Calculates the arc sine. Unit: radian Definition range: -1.0 to +1.0, Value range: -PI/2 to +PI/2	447	
	Atn(<Numeric expression>)	Calculates the arc tangent. Unit: radian Definition range: Numeric value, Value range: -PI/2 to +PI/2	447	
	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	447	
	Cos(<Numeric expression>)	Calculates the cosine Unit: radian Definition range: Numeric value range, Value range: -1 to +1	451	
	Sin(<Numeric expression>)	Calculates the sine Unit: radian Definition range: Numeric value range, Value range: -1 to +1	473	
	Tan(<Numeric expression>)	Calculates the tangent. Unit: radian Definition range: Numeric value range, Value range: Range of numeric value	479	

Class	Function name (format)	Functions	Page	Result
Character string functions	Asc(<Character string expression>)	Provides a character code for the first character of the character string in the expression.	446	Numeric value
	Bin$(<Numeric expression>)	Converts numeric expression value into binary character string.	448	Character string
	Chr$(<Numeric expression>)	Provides character having numeric expression value character code.	450	
	Cvi(<Character string expression>)	Converts a 2-byte character string into integers.	452	Numeric value
	Cvs(<Character string expression>)	Converts a 4-byte character string into a single-precision real number.	452	
	Cvd(<Character string expression>)	Converts an 8-byte character string into a double-precision real number.	453	
	Hex$(<Numeric expression>)	Converts numeric expression value into hexadecimal character string.	457	Character string
	Left$(<Character string expression>,<Numeric expression>)	Obtains character string having length designated with 2nd argument from left side of 1st argument character string.	459	
	Mid$(<Character string expression>, <Numeric expression> <Numeric expression>)	Obtains character string having length designated with 3rd argument from the position designated with the 2nd argument in the 1st argument character string.	461	
	Mirror$(<Character string expression>)	Mirror reversal of the character string binary bit is carried out.	462	
	Mki$(<Numeric expression>)	Converts numeric expression value into 2-byte character string.	463	
	Mks$(<Numeric expression>)	Converts numeric expression value into 4-byte character string.	463	
	Mkd$(<Numeric expression>)	Converts numeric expression value into 8-byte character string.	464	
	Right$(<Character string expression>,<Numeric expression>)	Obtains character string having length designated with 2nd argument from right side of 1st argument character string.	468	
	Val(<Character string expression>)	Converts a character string into a numeric value.	480	Numeric value
	Str$(<Numeric expression>)	Converts the numeric expression value into a decimal character string.	479	Character string
	Strpos(<Character string expression>, <Character string expression>)	Obtains the 2nd argument character string position in the 1st argument character string.	478	Numeric value
	StrLwr(<Character string>)	Decapitalizes a <Character string>.	477	Character string
	StrUpr(<Character string>)	Capitalizes a <Character string>.	478	
	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.	451	Numeric value
Position variables	Dist(<Position>,<Position>)	Obtains the distance between two points.	454	Position
	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.	456	
	Rdfl1(<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.	466	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)	469	
	Rdfl2(<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).	467	Numeric value
	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.	470	
	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.	445	

Class	Function name (format)	Functions	Page	Result
Position variables	Inv(<Position>)	Obtains the reverse matrix.	458	Position
	PtoJ(<Position>)	Converts the position data into joint data.	465	Joint
	JtoP(<Position>)	Converts the joint data into position data.	458	Position
	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	481	Numeric value
	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.	482	Numeric value
	Zone3 (<Position 1>,<Position 2>,<Position 3>,<Position 4> <Numeric value1>, <Numeric value2>, <Numeric value3>)	Checks whether position 1 is within the space (cube) created by four positions (position 2, position 3, position 4) and three values (integer 1, integer 2, integer 3). 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.	483	Numeric value
	PosCq(<Position>)	Checks whether <position> is within the movement range.	464	Numeric value
	PosMid (<Position1>,<Position2>, <Numeric value1>, <Numeric value2>)	Calculates the middle position between <position 1> and <position 2>.	465	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>.	449	Numeric value
	SetJnt (<J1 axis>,<J2 axis>,<J3 axis>,<J4 axis> <J5 axis>,<J6 axis>,<J7 axis>,<J8 axis>)	Sets values in joint variables.	471	Joint
	SetPos (<X axis>,<Y axis>,<Z axis>,<A axis> <B axis>,<C axis>,<L1 axis>,<L2 axis>)	Sets values in position variables.	472	Position
	SplPos (<Spline No.>, <Frame transformation>, <Path point No.>)	Loads the path point data registered in a random spline file.	475	Position
	SplECord (<Spline No.>, <Frame transformation>	Loads the Ex-T coordinate system origin data registered in a random spline file.	474	Position

4.7 List of Command

A list of pages with description of each command is shown below. They are listed in the order of presumed usage frequency.

(1) Command related to movement control

Command	Explanation	Page
Mov (Move)	Joint interpolation	267
Mvs (Move S)	Linear interpolation	280
Mvr (Move R)	Circular interpolation	274
Mvr2 (Move R 2)	Circular interpolation 2	276
Mvr3 (Move R 3)	Circular interpolation 3	278
Mvc (Move C)	Circular interpolation	273
Mva (Move Arch)	Arch motion interpolation	268
MvSpl (Move Spline)	Spline interpolation	283
Mxt (Move External)	Optimum acceleration/deceleration rate specification	288

Command	Explanation	Page
Mv Tune (Move Tune)	Specification of the moving characteristics mode	286
Ovrd (Override)	Overall speed specification	304
Spd (Speed)	Speed specification during linear or circular interpolation movement	325
JOvrd (J Override)	Speed specification during joint interpolation movement	263
Cnt (Continuous)	Continuous path mode specification	192
Accel (Accelerate)	Acceleration/deceleration rate specification	171
ColChk (Col Check)	Collision detection function	195
CavChk On (CavChk On)	Interference avoidance function	181
Cmp Jnt (Compliance Joint)	Specification of compliance in the JOINT coordinate system	184
Cmp Pos (Compliance Posture)	Specification of compliance in the XYZ coordinate system	186
Cmp Tool (Compliance Tool)	Specification of compliance in the Tool coordinate system	188
Cmp Off (Compliance OFF)	Compliance setting invalid	190
CmpG (Compliance Gain)	Compliance gain specification	191
Oadl (Optimal Acceleration)	Sets the optimum acceleration/deceleration	297
Loadset (Load Set)	Hand's optional condition specification	266
Prec (Precision)	High accuracy mode specification	306
Torq (Torque)	Torque specification of each axis	334
JRC (Joint Roll Change)	Enables multiple rotation of the tip axis	264
Fine (Fine)	Robot's positioning range specification	244
Fine J (Fine Joint)	Robot's positioning range specification by joint interpolation	245
Fine P (Fine Pause)	Robot's positioning range specification by distance in a straight line	246
Servo (Servo)	Servo motor power ON/OFF	321
Wth (With)	Addition instruction of movement instruction	337
WthIf (With If)	Additional conditional instruction of movement instruction	338
EMvs (E Move S)	Ex-T control linear interpolation	236
EMvc (E Move C)	Ex-T control circular interpolation	228
EMvr (E Move R)	Ex-T control circular arc interpolation	230
EMvr2 (E Move R 2)	Ex-T control circular arc interpolation 2	232
EMvr3 (E Move R 3)	Ex-T control circular arc interpolation 3	234
EMvSpl (E Move Spline)	Ex-T spline interpolation	238

(2) Command related to program control

Command	Explanation	Page
Rem (Remarks)	Comment(')	313
If...Then...ElseIf...Else...EndIf (If Then Else)	Conditional branching	259
Select Case (Select Case)	Enables multiple branching	319
GoTo (Go To)	Jump	254
GoSub (Return)(Go Subroutine)	Subroutine jump	253
Reset Err (Reset Error)	Resets an error (use of default is not allowed)	315
CallP (Call P)	Program call	178
FPrm (FPRM)	Program call argument definition	248
Dly (Delay)	Timer	222
Function... FEnd	Function procedure definition	249
Exit	Exits the procedure/repeat processing	243
Hlt (Halt)	Suspends a program	256
Include	#Include statement declaration	261
End (End)	End a program	241

Command	Explanation	Page
On ... GoSub (ON Go Subroutine)	Subroutine jump according to the value	300
On ... GoTo (On Go To)	Jump according to the value	301
For - Next (For-next)	Repeat	247
While-WEnd (While End)	Conditional repeat	336
Open (Open)	Opens a file or communication line	302
Print (Print)	Outputs data	307
Input (Input)	Inputs data	262
Close (Close)	Closes a file or communication line	182
Remove (Remove)	Deletes a file.	314
Save	Saves a program	318
ColChk (Col Check)	Enables or disables the collision detection function	195
On Com GoSub (ON Communication Go Subroutine)	Communication interrupt subroutine jump	299
Com On/Com Off/Com Stop (Communication ON/OFF/Stop)	Allows/prohibits/stops communication interrupts	199
HOpen / HClose (Hand Open/Hand Close)	Hand's open/close	257
Error (error)	User error	242
Skip (Skip)	Skip while moving	324
Wait (Wait)	Waiting for conditions	335
Clr (Clear)	Signal clear	183
SplWrt (Spline Write)	Creates the spline file	329
SplFWrt (Spline Frame Write)	Rewrites the frame transportation information registered into the spline file.	328
GpsChk (Get position check)	Starts/stops monitoring the condition defined in the Def Gps command or Def Map command	255

(3) Definition commands

Command	Explanation	Page
Dim (Dim)	Array variable declaration	220
Def Plt (Define pallet)	Pallet declaration	216
Plt (Pallet)	Pallet position calculation	305
Def Act (Define act)	Interrupt definition	202
Act (Act)	Starts or ends interrupt monitoring	173
Def Arch (Define arch)	Definition of arch shape for arch motion	205
Def Jnt (Define Joint)	Joint type position variable definition	212
Def Pos (Define Position)	XYZ type position variable definition	218
Def Work	Work coordinate variable definition	219
Def Inte/Def Long/Def Float/Def Double (Define Integer/Long/Float/Double)	Integer or real number variable definition	210
Def Char (Define Character)	Character variable definition	206
Def IO (Define IO)	Signal variable definition	211
Def FN (Define function)	User function definition	207
Const	Constant definition	200
Static	Static variable declaration	331
Title (Title)	Program title setting	332
Base (Base)	Robot base position setting	175
Tool(Tool)	Tool length setting	333
SetCalFrm (Set Calibration Frame)	Coordinate system setting used for frame transformation	322
Def Gps (Define get position)	Definition of the monitored condition for the position data acquisition using the get-position-quick function (GPS function)	208

Command	Explanation	Page
Def Map (Define mapping)	Definition of the monitored condition for the mapping (the workpiece presence recognition) in a cassette using the get-position-quick function (GPS function)	213

(4) Multi-task related

Command	Explanation	Page
XLoad (X Load)	Loads a program to another task slot	340
XRun (X Run)	Execute the program in another task slot	342
XStp (X Stop)	Stop the program in another task slot	343
XRst (X Reset)	Resets the program in another task slot being suspended	341
XClr (X Clear)	Cancels the loading of the program from the specified task slot	339
GetM (Get Mechanism)	Obtains mechanical control right	252
RelM (Release Mechanism)	Releases mechanical control right	312
Priority (Priority)	Changes the task slot priority	308
Reset Err (Reset Error)	Resets an error (use of default is not allowed)	315

(5) Communications

Command	Explanation	Page
Com Off (Communication OFF)	Prohibits an interruption from the communication line.	199
Com On (Communication ON)	Allows an interruption from the communication line.	199
Com Stop (Communication Stop)	Suspends an interruption from the communication line. (Data is received.)	199
On Com GoSub (ON Communication Go Subroutine)	Communication interruption definition	299
Open (Open)	Opens a file.	302
Close (Close)	Closes a file.	182
Input (Input)	Inputs text data.	262
Print (Print)	Outputs text data.	307
NVOpen (network vision sensor line open)	Connects with the vision sensor and logs on to the vision sensor.	291
NVClose (network vision sensor line close)	Cuts off the connection with vision sensor.	289
NVLoad (network vision sensor load)	Puts the specified vision program into the state in which it can be started.	290
NVRun (network vision sensor run)	Starts the specified vision program.	293
NVTrg (network vision sensor trigger)	Requests the vision sensor to capture an image and acquires the encoder value after the specified time.	295
EBRead (EasyBuilder read)	Reads the data for which the tag name of the vision sensor is specified.	223
EBWrite (EasyBuilder write)	Writes the data for which the tag name of the vision sensor is specified.	226

(6) Others

Command	Explanation	Page
ChrSrch (Character search)	Searches the character string out of the character array.	180
PrmRead (Parameter Read)	Reads a parameter value	309
PrmWrite (Parameter Write)	Writes a parameter	310
Get Pos (Get Position)	Reserved.	-
PVSCal (PVS calibration)	Changes the vision sensor image coordinate to the robot world coordinate using the vision sensor calibration data.	311

4.8 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
	15	AND	256	15
	(Numeric value type)		(Numeric value type)	(Numeric value type)
	P1	*	M1	P2
	(Position type)		(Numeric value type)	(Position type)
	M1	*	P1	
	(Numeric value type)		(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	-	-	-	-
	Addition +	Character string	-	-	-	-
	Comparison (Comparison operators)	Integer	-	-	-	-
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	Real number	-	-
	Substitution =	-	Integer	Integer	-	-
	Comparison (Comparison operators)	-	Integer	Integer	-	-
	Logic (Logic operators)	-	Integer	Integer	-	-
Real number	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 ^	-	Real number	Real number	-	-
	Substitution =	-	Integer	Real number	-	-
	Comparison (Comparison operators)	-	Integer	Integer	-	-
	Logic (Logic operators)	-	Integer	Integer	-	-
Position	Addition +	-	-	-	Position	-
	Subtract -	-	-	-	Position	-
	Multiplication *	-	Position	Position	Position	-
	Division /	-	Position	Position	Position	-
	Integer division \	-	-	-	-	-
	Remainder MOD	-	-	-	-	-
	Exponent ^	-	-	-	-	-
	Substitution =	-	-	-	Position	-
	Comparison (Comparison operators)	-	-	-	-	-
	Logic (Logic operators)	-	-	-	-	-
Joint	Addition +	-	-	-	-	Joint
	Subtract -	-	-	-	-	Joint
	Multiplication *	-	Joint	Joint	-	-
	Division /	-	Joint	Joint	-	-
	Integer division \	-	-	-	-	-
	Remainder MOD	-	-	-	-	-
	Exponent ^	-	-	-	-	-
	Substitution =	-	-	-	-	Joint
	Comparison (Comparison operators)	-	-	-	-	-
	Logic (Logic operators)	-	-	-	-	-
Right argument only (Single arugument)	eversal -	-	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.

4.9 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

4.10 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 4-13, 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.

4.11 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.

4.12 Detailed explanation of command words

4.12.1 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.

4.12.2 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 downward>]
```

[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) 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.
(4) It is also valid during optimum acceleration/deceleration control (Oadl On).

[Related instructions]
Oadl (Optimal Acceleration), Loadset (Load Set)

[Related system variables]
M_Acl/M_DAcl/M_NAcl/M_NDAcl/M_AclSts

[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/-1>

[Terminology]

<Priority No.>	0: Either enables or disables the entire interrupt. 1 - 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>	1: Allows interrupts 0:Prohibits interrupts. -1: If the <Priority No.> is 1 to 8, the interrupt in progress is canceled. And interrupt monitoring function is stopped.

[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                                ' Enable interrupt 1.
 4 Mov P2
 5 Act 1=0                                ' Disable interrupt 1.
    :
10 *INTR                                  '
11 IF M_In(1)=1 GoTo 110                  ' Loops until the M_In(1) signal becomes 0.
12 Return 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                              ' Enable interrupt 1.
 4 Mov P2
  :
10 *INTR
11 Act 1=0                              ' Disable interrupt 1.
12 M_Out(10)=1                          ' Turn on the output signal 10
13 Return 1                             ' Returns to the next step which interrupted
```

(3) When the input signal 1 turns on (set to 1)while moving from P1 to P2, Operation is interrupted and wait until the signal 0 turns on then return to the first step of the program.

```
 1 *Start
  :
11 Def Act1, M_In(1)=1 Goto *INTR2      ' Assign input signal 1 to the interrupt 1 condition
12 Mov P1
13 Act 1=1                              ' Enable interrupt 1.
14 Mov P2
  :
21 *INTR2
22 Act 1=0                              ' Disable interrupt 1.
23 *Loop
```

```
24 If M_In(1) <> 0 Then Goto *Loop
25 Act 1=-1                          ' The interrupt in progress is canceled. And interrupt
                                       monitoring function is stopped.
26 Goto *Start
```

[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)
(9) The error L3110 will occur when Act 0=-1.

[Related instructions]

Def Act (Define act), Return (Return)

Base (Base)

[Function]

Changes (relocation and rotation) can be made to the world coordinate system which is the basis for the control of the robot's current position. There are two alternative methods to achieve this. One is to directly specify base conversion data and the other, to specify a predefined work coordinate system number. This function has significant influences on teaching data for and jog operation of the robot. Read instructions given in "4.4Coordinate system description of the robot" and proceed with care.

[Format]

Base[]<Base conversion data>	' Specifying base conversion data directly
Base[]<Base coordinate number>	' Specifying base conversion data indirectly by a base coordinate number (work coordinate system number).

[Terminology]

<Base conversion data>

Base conversion data is specified with a position constant or a position variable.
Values to be specified (coordinate values) represent position data for the origin point of the base coordinate system as viewed from a world coordinate system which is newly furnished.

<Base coordinate number>

The system's initial value or value set in the parameter concerned (work coordinate system) is designated as base conversion data.
This value is a constant in numerical form or a variable which is chosen from 0 through 8.

0: P_NBase (system's initial value) is specified.
(Because P_NBase=(0,0,0,0,0,0), this value clears base conversion settings.)

1 - 8:Each value corresponds to parameter/work coordinate system (WK1CORD~WK8CORD).

Note) When a real number or a double-precision real number is specified, the fractional portion is round down.

[Reference Program]

Specify by base conversion data

```
1 Base (50,100,0,0,0,90)   ' A new world coordinate system is defined by conversion data in the form of a constant.
2 Mvs P1                   'A move to P1 is made in the new world coordinate system.
3 Base P2                  ' A new world coordinate system is defined by conversion data in the form of a constant.
4 Mvs P1                   ' move to P1 is made in the new world coordinate system.
5 Base 0                   ' World coordinate system is returned to an initial value.
                            (P_NBase(ÉVsystem's initial value)is set for base conversion data)
```

Specify by the base coordinates number

```
1 Base 1                   ' Work coordinate system 1 (parameter: WK1CORD) is defined as a new world coordinate system.
2 Mvs P1                   'A move to P1 is made in the new world coordinate system.
3 Base 2                   ' Work coordinate system 2 (parameter: WK2CORD) is defined as a new world coordinate system.
4 Mvs P1                   'A move to P1 is made in the new world coordinate system.
5 Base 0                   ' World coordinate system is returned to an initial value.
                            (P_NBase(ÉVsystem's initial value)is set for base conversion data)
```

[Explanation]

(1) Values subject to base conversion (coordinate values) represent position data for the origin point of the base coordinate system as viewed from a world coordinate system which is newly defined. Therefore, when you use the robot's current position to specify base conversion data with coordinate values defined by a Fram function or the like, do so by inversely converting the coordinate values [for example, Base Inv(P1)]. [for example, Base Inv(P1)].
Note that when you specify a work coordinate system number, the above inverse conversion is accomplished automatically in an internal process.
Elements X, Y and Z of position data indicate the amount of translation from the origin point of the world coordinate system to that of the base coordinate system. Also, elements A, B and C indicate how much the base coordinate system is tilted relative to the robot's 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

Elements A, B, and C are set to take a clockwise move as a forward rotation looking at the plus side from the origin point of the world coordinate system.

(2) The contents of the structural flag have no meaning.

(3) "Base conversion data" changed with this command is saved in parameter MEXBS, and "base coordinate number" in parameter MEXBSNO. The saved values are retained even after the controller is turned off.
For programs in which Base command, Tool command, and M_Tool are executed repeatedly, parameters may not be saved in time, causing error C7091 (parameter save error). If error C7091 occurs, change the order of Base command and Tool command as shown in the following example.

Example)

```
*MAIN                          Base PB
  Base PB                      Tool PT
  Tool PT                 →    *MAIN
  If M_In(20)=1 Then             If M_In(20)=1 Then
    GoSub *SUB1                    GoSub *SUB1
  EndIf                          EndIf
  GoTo *MAIN                     GoTo *MAIN
```

(4) Performing a base conversion changes the robot's current position to values that refer to the newly established world coordinate system or work coordinate system. The target position specified by the movement command will also be treated as a position in the newly established world coordinate system or work coordinate system.

(5) You should note that the base conversion data differs in the valid axial elements depending on the robot's type (structure of the robot arm). Refer to Page 517, "5.6 Standard Tool Coordinates".
Also, refer to the following sections for more information relative to this command.
Page 149, "4.4.1 About the robot's coordinate system"
Page 150, "4.4.2 About base conversion".

(6) The system's default value for this data is P_NBase=(0,0,0,0,0,0) (0,0).

World coordinate system: Xw, Yw, Zw
Base coordinate system: Xb, Yb, Zb

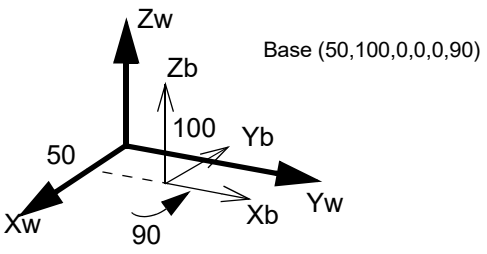


Fig.4-16:Conceptual diagram of the base coordinate system

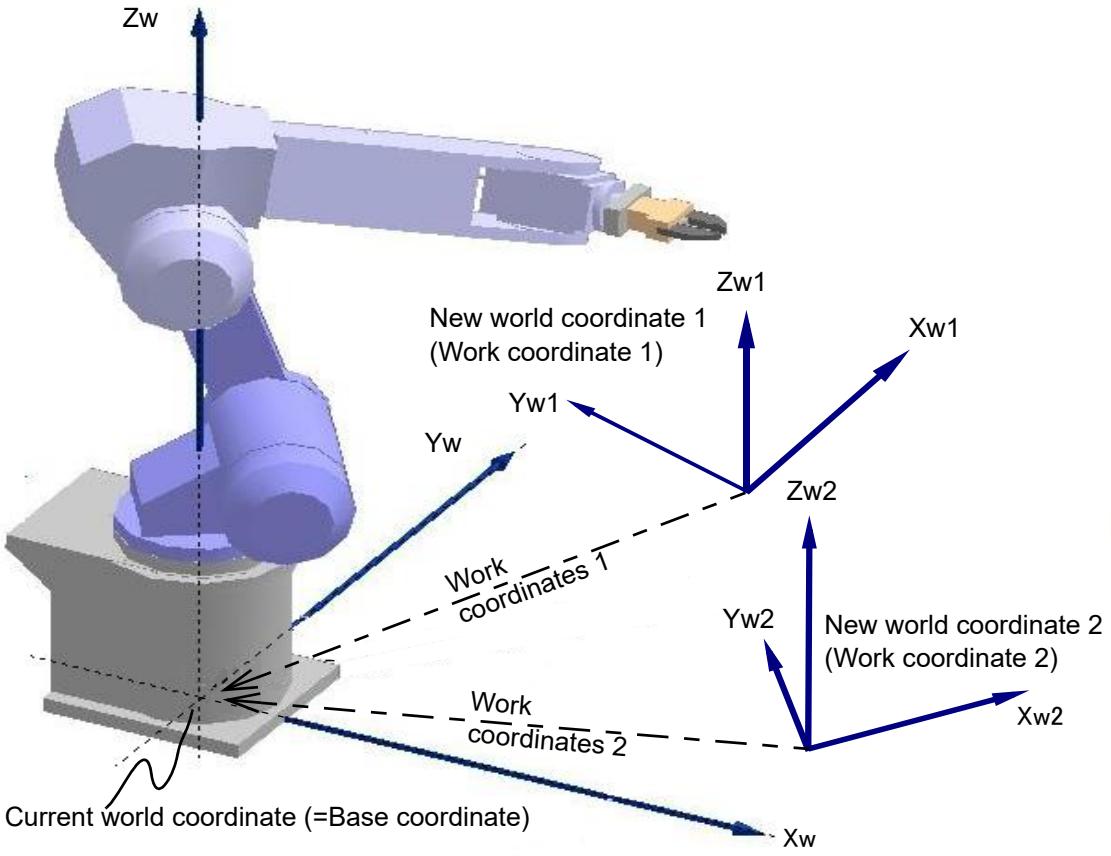


Fig.4-17:Base conversion with a work coordinate system number being specified

CAUTION

Performing a base conversion changes the robot's current position to values that refer to the newly established world coordinate system or work coordinate system. The target position specified by the movement command will also be treated as a position in the newly established world coordinate system or work coordinate system.
As a result, data taught till then may become unusable as it is. The coordinate system when taught needs to be the same as the newly established coordinate system.
If they do not match, the robot can stray to unexpected positions, possibly resulting in property damage or personal injury.
When using the base conversion function, be sure to maintain positive control over relation between the base coordinate system subject to conversion and the position which the robot is taught to take so that a proper robot operation and an effective use of the base conversion function are insured.

[Related parameter]
MEXBS,WKnCORD ("n" is 1 to 8), MEXBSNO

[Related system variables]
M_BsNo, P_Base/P_NBase, P_WkCord

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 132, "(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
```

Sub program side

```
 1 FPrm M01, P01,P02
 2 If M01<>0 Then GoTo *LBL1
 3 Mov P01
 4 *LBL1
 5 Mvs P02
 6 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 10 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) Tool conversion data is valid in a sub program. Values of Accel, Spd, Ovrd, and JOvrd 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.

CavChk On (CavChk On)

[Function]

Activates the stop function of the interference avoidance function.
This function is only available for certain models. For details, refer to Page 567, "5.24 Interference avoidance function".

[Format]

```
CavChk[]<On/Off>[,<Robot CPU No>[,NOErr]]
```

[Terminology]

<On/Off>	On: Activates the stop function of the interference avoidance function. Off: Deactivates the stop function of the interference avoidance function.
<Robot CPU No.>	Specifies the target robot CPU for the activation/deactivation of the interference avoidance. When this argument is omitted or 0 is specified, all connected robots are the target. Specify as a value 0 to 3, constant, or variable. Specify 0 for the CR800-D series. An alarm (L4930) would occur if a value other than 0 is specified.
NOErr	Disables the interference prediction alarm. An alarm (L240n) would occur if this argument is omitted.

[Reference Program]

Refer to Page 589, "5.24.10 Sample programs".

[Explanation]

(1) This command activates the stop function of the interference avoidance function. The parameter: CAV setting determines the initial status of the interference avoidance function.
Activating the interference avoidance function while the function is set disabled by the parameter: CAV will cause an alarm (L4930).
(2) If the robot CPU number is omitted, interference avoidance operation is performed for the avoidance-function-enabled CPUs only.
(3) Specifying the robot CPU number of the robot CPU where the interference avoidance function is set disabled will cause an alarm (L4930).
(4) Use the LoadSet command to specify hand and workpiece models.
(5) The interference avoidance function cannot be activated during tracking. (Alarm (L4936) would occur.)

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 ORS18160.
Refer to Page 532, "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) though (3) above are performed simultaneously.

```
1 Clr 0
```

[Related parameter]

ORST0 to ORS18160

[Related system variables]

M_In/M_Inb/M_In8/M_Inw/M_In16, M_Out/M_Outb/M_Out8/M_Outw/M_Out16

Cmp Jnt (Compliance 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-FR/RH-CR series, 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.

(12) While operation is performed in the compliance mode of the joint coordinate system, if the Excessive error 1 (H096n) occurs, increase the set value of parameter CMPJCLL to suppress the error.
(Setting range: 1 to 10)
Gradually increase the set value of this parameter until no Excessive error 1 occurs.

⚠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.

⚠CAUTION Start the compliance mode when the robot is in stop. If the compliance mode is started while the robot is operating, the control may be disabled when the compliance mode is canceled.

[Available robot type]

RH-FR series, RH-CR series

[Related system variables]

M_CmpDst

[Related instructions]

Cmp Off (Compliance OFF), CmpG (Compliance Gain), Cmp Pos (Compliance Posture), Cmp Tool (Compliance Tool)

[Related parameter]

CMPJCLL

Cmp Pos (Compliance Posture)

[Function]

Start the soft control mode (compliance mode) of the specified axis in the XYZ coordinates system.

[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.

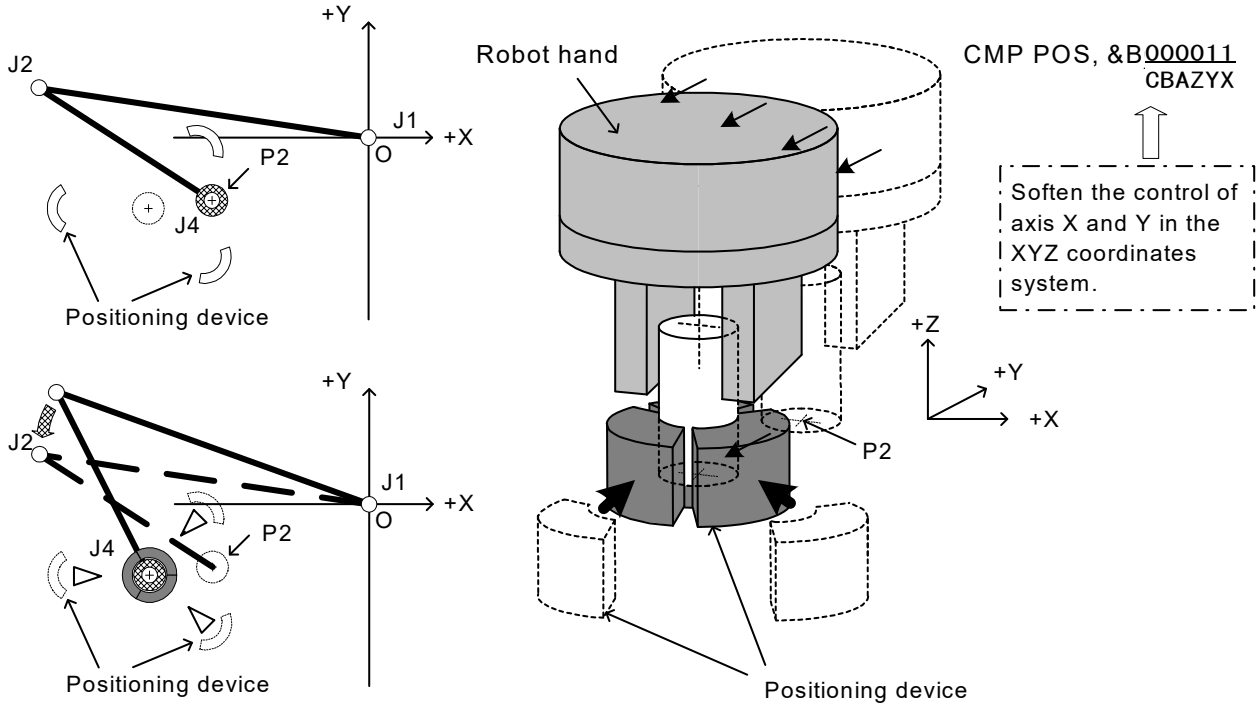


Fig.4-18:The example of compliance mode use

⚠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.

⚠CAUTION 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.

⚠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.

⚠CAUTION Start the compliance mode when the robot is in stop. If the compliance mode is started while the robot is operating, the control may be disabled when the compliance mode is canceled.

[Related system variables]
M_CmpDst

[Related instructions]
Cmp Off (Compliance OFF), CmpG (Compliance Gain), Cmp Tool (Compliance Tool), Cmp Jnt (Compliance Joint)

Cmp Tool (Compliance Tool)

[Function]
Start the soft control mode (compliance mode) of the specified axis in the Tool coordinates system.

[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 517, "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 Cmp G 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 type robot (such as the RV-4FRJL), 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 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.

(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.

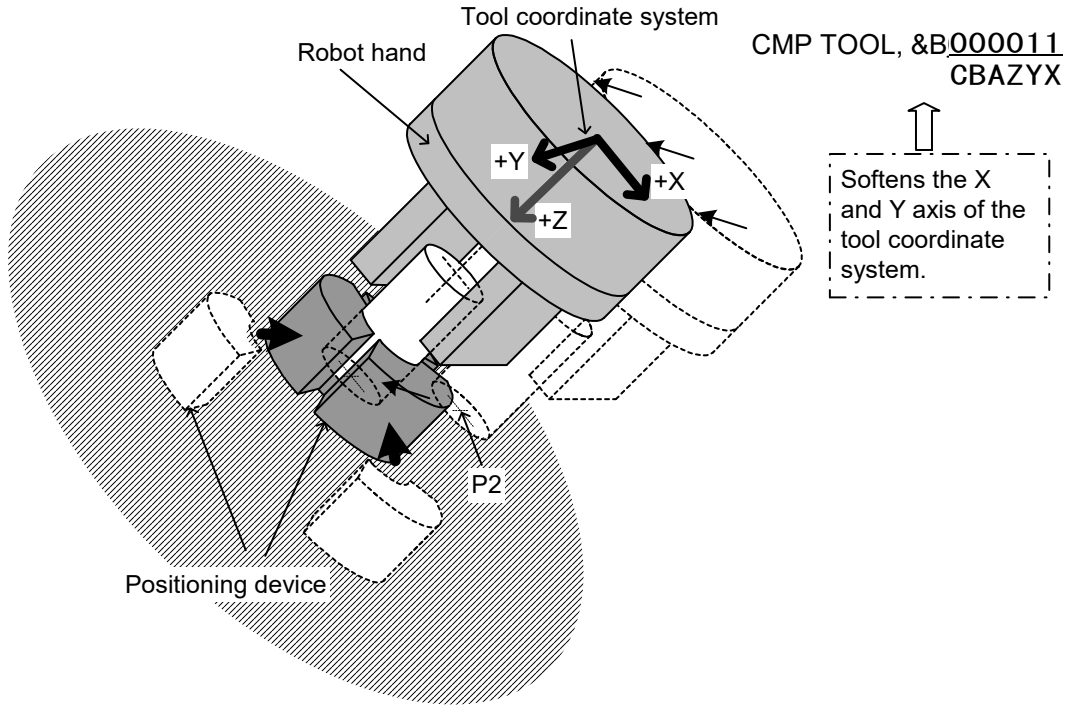


Fig.4-19:The example of using the compliance mode

⚠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.

⚠CAUTION 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.

⚠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.

⚠CAUTION Start the compliance mode when the robot is in stop. If the compliance mode is started while the robot is operating, the control may be disabled when the compliance mode is canceled.

[Related system variables]
M_CmpDst

[Related instructions]
Cmp Off (Compliance OFF), CmpG (Compliance Gain), Cmp Pos (Compliance Posture), Cmp Jnt (Compliance Joint)

Cmp Off (Compliance OFF)

[Function]
Release the soft control mode (compliance mode).

[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 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(10110)=1                  ' Instructs to close the chuck for positioning.
6 Dly 0.5                         ' Waits for the completion of chuck closing.
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.
(3) If the error H115n (Large command position) occurs at the time of execution of the Cmp Off command, executing Mov P_Fbc before the Cmp Off may prevent the error occurrence.
(4) If the compliance mode is started when the robot is operating, the control may be disabled when the Cmp Off command is executed. Start the compliance mode when the robot is in stop.

[Related instructions]
CmpG (Compliance Gain), Cmp Tool (Compliance Tool), Cmp Pos (Compliance Posture), Cmp Jnt (Compliance Joint)

CmpG (Compliance Gain)

[Function]
Specify the softness of robot control.

[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) Even if value of less than minimum is set up, the gain is minimum value. Also, two or more decimal positions can be set for gain values.

Type	Cmp Pos, Cmp Tool	Cmp Jnt
RH-FR/RH-CR series	0.20, 0.20, 0.20, 0.20, 0.20, 0.20	0.00, 0.00, 0.20, 0.00, 1.00, 1.00
RV-FR series	0.01, 0.01, 0.01, 0.01, 0.01, 0.01	-

(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.

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
```

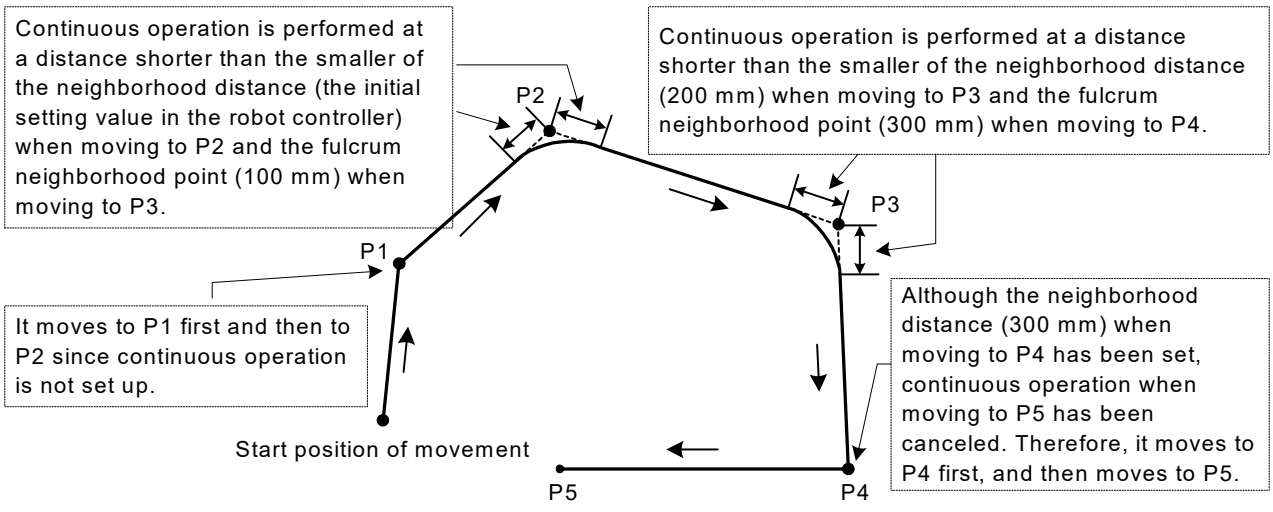


Fig.4-20:Example of continuous path operation

⚠CAUTION The robot's locus of movement may change with specified speed.
Especially as for the corner section, short cut distance may change. Therefore, when beginning automatic operation, moves at low speed at first, and you should gather speed slowly with being careful of interference with peripheral equipment.

[Explanation]

(1) The interpolation (4 step to 8 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. 4-21, 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.

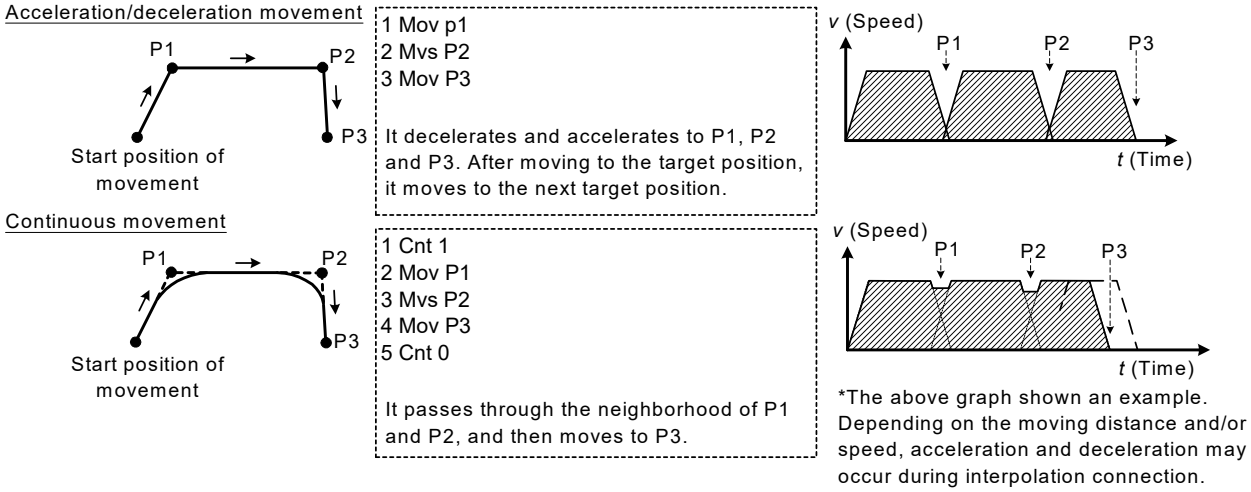


Fig.4-21:Acceleration/deceleration movement and continuous movement

(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).

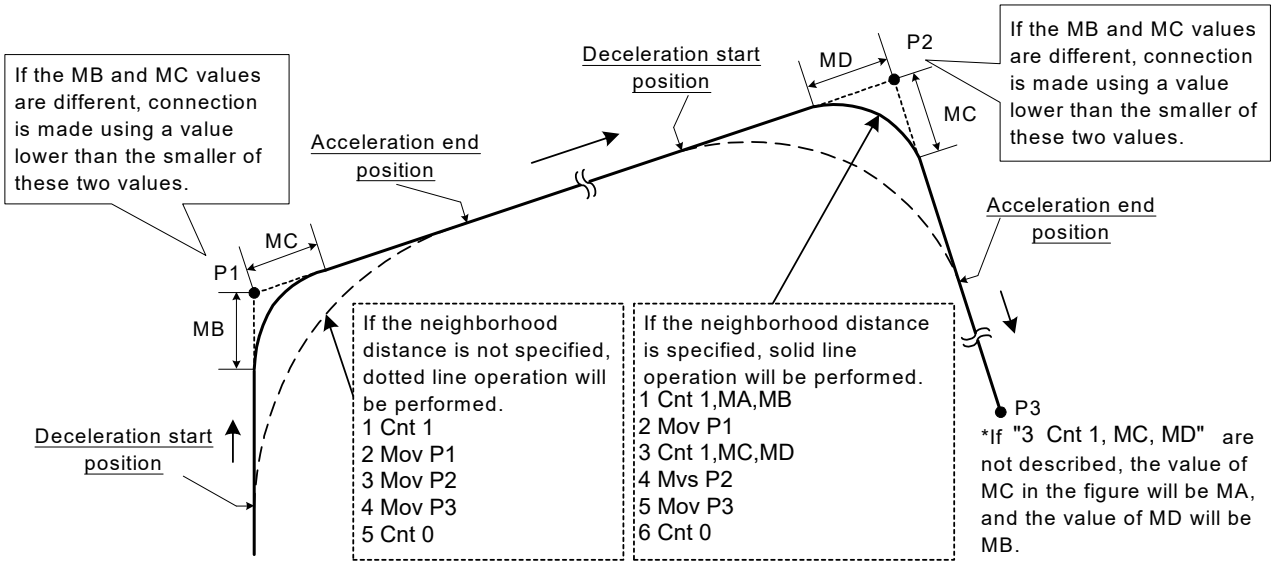


Fig.4-22:Setting Up the Neighborhood Distance

(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 command 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 collision detection function in automatic operation.
The collision 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.

[Format]

```
ColChk[]On [, NOErr] / Off
```

[Terminology]

On	Enable the collision detection function. Once an collision is detected, it immediately stops the robot, issues an error numbered in 1010's, and turns OFF the servo.
Off	Disable the collision detection function
NOErr	Even if an collision is detected, no error is issued. (If omitted, an error will occur.)

[Reference Program 1]

If an error is set in the case of collision

```
1 ColLvl 80,80,80,80,80,80,,        'Specify the allowable level for collision detection.
2 ColChk On                         'Enable the collision 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 collision detection function.
7 Mov P3
```

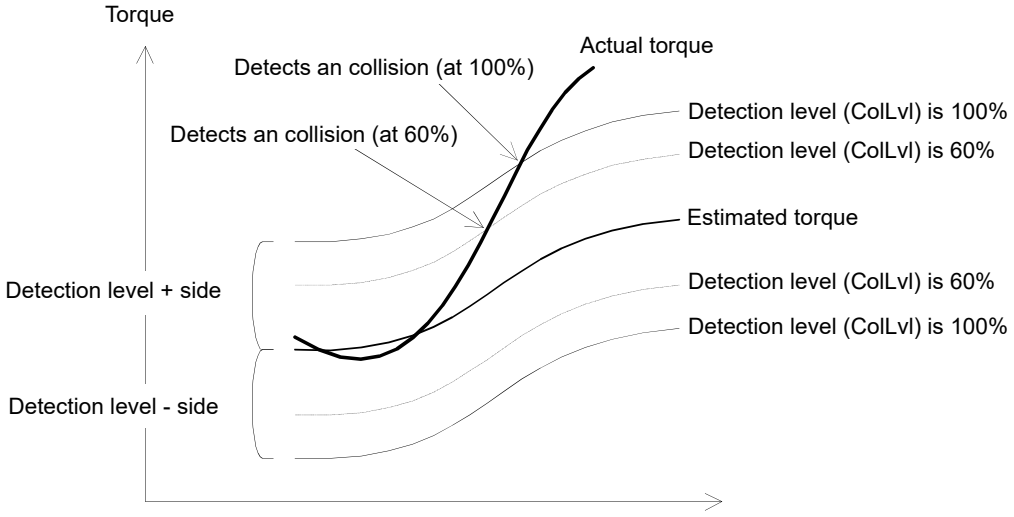
[Reference Program 2]

If interrupt processing is used in the case of collision

```
1 Def Act 1,M_ColSts(1)=1 GoTo *HOME,S   'Define the processing to be executed when an collision is
                                          detected using an interrupt.
2 Act 1=1
3 ColChk On,NOErr                   'Enable the collision detection function in the error non-
                                     occurrence mode.
4 Mov P1
5 Mov P2                            'If an collision is detected while executing steps 4 through 7, it
                                     jumps to interrupt processing.
6 Mov P3
7 Mov P4
8 Act 1=0
    :
10 *HOME                            'Interrupt processing during collision detection
11 ColChk Off                       'Disable the collision 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 safe position.
15 Mvs PDst                         'Move to the safe position.
16 Error 9100                       'Stop operation by generating a user-defined L level error.
```

[Explanation]

(1) The collision 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 collision 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 collision detection function is disabled. Enable the Col parameter before using. This instruction specifies whether to enable or disable the collision 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 collision 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 collision detection function is disabled by this instruction, the collision detection level set by a ColLvl instruction is retained.

(6) 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.

(7) If an collision 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 collision was detected. (The recording into the log is done only if no other errors occur simultaneously.)

(8) If an attempt is made to execute ColChk On and ColChk On,NOErr on a robot that cannot use the collision detection function, low level error 3970 occurs. In the case of ColChk Off, neither error occurs nor processing is performed.

(9) The collision 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 collision detection function. Conversely, error 3930 will occur if an attempt is made to enable a Cmp or Torq instruction while collision detection is being enabled.

(10) If ColChk Off is described immediately after an operation instruction, collision detection may not work near the last stop position of a given operation. As shown in reference program 1, execute ColChk Off upon completion of positioning by a Dly or Fine instruction between an operation instruction and a ColChk Off instruction.

(11) The collision 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.

(12) If the collision detection function is enabled by this instruction, the execution time (tact time) may become long for some programs. Use the collision detection function only for operations that may interfere with peripheral devices, rather than enabling it for the entire program.
(13) This function cannot be used together with the multi-mechanism control function.
(14) To continue the servo-on after the detection, set "1" in the parameter COLSERVO and designate NOErr (error non-occurrence mode). However, an overload error may be generated depending on the pressing amount at collision.

[Related variables]
M_ColSts, J_ColMxl, P_ColDir

[Related instructions]
ColLvl (Col Level)

[Related parameter]
COL, COLLVL, COLLVLJG, HNDDATn, WRKDATn, COLSERVO

ColLvl (Col Level)

[Function]
Set the detection level of the collision detection function in automatic operation.

[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 80,80,80,80,80,80,,    'Specify the allowable level for collision detection.
2 ColChk On                     'Enable the collision detection function.
3 Mov P1
4 ColLvl ,50,50,,,,,            'Change the allowable level of the J2 and J3 axes for collision detection.
5 Mov P2
6 Dly 0.2                       'After arriving at P2, disable collision detection.
7 ColChk Off                    'Disable the collision detection function.
8 Mov P3
```

[Explanation]
(1) Set the allowable level of each axis for the collision detection function during program operation.
(2) This instruction affects the collision 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. The initial value of parameter differ by each type.
(4) If this value is increased, the detection level (sensitivity) lowers; if this value is lowered, the detection level increases.
(5) 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.
(6) It is possible to choose "Ref. value of COL level" (reference value of the collision detection level) with the oscillograph function of RT ToolBox3. This function can be easy to adjust the detection level of ColLvl instruction. Refer to Page 557, "5.21 About the collision detection function" for details.
(7) The collision 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) The allowable level is reset to the setting value of the COLLVL parameter when a program reset or an End instruction is executed.
(9) Even if an attempt is made to execute this instruction on robots that cannot use the collision detection function, the instruction is ignored and thus no error occurs.
(10) The collision detection function is not valid for the J7 and J8 axes.
(11) 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 variables]
M_ColSts, J_ColMxl, P_ColDir

[Related instructions and variables]
ColChk (Col Check)

[Related parameter]
COL,COLLVL, HNDDATn, WRKDATn

* Refer to Page 557, "5.21 About the collision detection function" for details.
And, the sample program which automatically sets up the collision detection level is shown in J_ColMxl.

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 299, "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 302, "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.

Const

[Function]
Defines a constant (unchangeable variable).

[Format]

```
Const[]Def[]Inte[]<Numeric value variable name>[ = <Numeric value>]
                [, <Numeric value variable name>[ = <Numeric value>][, ...]]
Const[]Def[]Long[]<Numeric value variable name>[ = <Numeric value>]
                 [, <Numeric value variable name>[ = <Numeric value>][, ...]]
Const[]Def[]Float[]<Numeric value variable name>[ = <Numeric value>]
                 [, <Numeric value variable name>[ = <Numeric value>][, ...]]
Const[]Def[]Double[]<Numeric value variable name>[ = <Numeric value>]
                   [, <Numeric value variable name>[ = <Numeric value>][, ...]]
```

<Numeric value variable name> Designate a variable name.
<Numeric value> Designate a value to be set to the numeric value variable.

```
Const[]Def[]Char[]<Character string variable name>[ = <Character string>]
                 [, <Character string variable name>[ = <Character string>][, ...]]
```

<Character string variable name> Designate a variable name.
<Character string> Designate a character string to be set to the character string variable.

```
Const[]Def[]Jnt[]<Joint variable name>[, <Joint variable name>[, ...]]
Const[]Def[]Pos[]<XYZ variable name>[, <XYZ variable name>[, ...]]
Const[]Def[]Work[]<Work coordinate variable name>[, <Work coordinate variable name>[, ...]]
```

<Joint variable name> Designate a variable name.
<XYZ variable name> Designate a variable name.
<Work coordinate variable name> Designate a variable name.

```
Const[]Dim[]<Variable name>(<Eelement Value>[, <Eelement Value>[, <Eelement Value>]])
        [, <Variable name>(<Eelement Value>[, <Eelement Value>[, <Eelement Value>]])[, ...]]
```

<Variable name> Designate an array variable name.
<Eelement Value> Designate the number of elements in an array variable with a constant.

```
Const[]<Variable name> = <Expression>
```

<Variable name> Designate a variable name.
<Expression> Designate a substitution value and a numeric operation expression.

[Reference Program]

```
1 Const Def Inte M1 = 10000, M2 = 200      ' Define M1 with the constant 10000 and M2 with the
                                             constant 200.
2 Const Def Float M3 = 3.3                 ' Define M3 with the constant 3.3.
3 Const Def Char Message = "Error"         ' Define Message with the character string constant "Error".
4 Const Dim cmd$(5)                        ' Define cmd$ with a constant.
5 Const cmd$(1) = "Cmd01"                  ' Define cmd$(1) with the character string constant
                                             "Cmd01".
6 Const Def P1                             ' Define P1 with a constant.
7 Const P1 = (100, 0, 0)                   ' Define P1 with the XYZ constant (100.00, 0.00, 0.00).
```

[Explanation]

(1) The Const variables are regarded as unchangeable variables (constants).
Writing any value to a Const variable causes the error L4320 (00000). However, a substitution (writing) can be performed without an error when Const is described.
(2) As is the case with variable substitution, the range check is performed according to the type when a Const variable is defined.
When a variable of integer type is initialized with a real number, the rounded value is defined.
(3) Const variables are not cleared to zero even if the Clr instruction is executed.
(4) User-defined external variables can also be defined with Const.
To set a value to a user-defined external variable in a user base program, Def specification is required.
Substitution of values in array or XYZ/joint/work coordinate variables needs to be executed in a program other than the user base program.
(5) If single variables are defined with Const in a user base program, they can be used as Const variables without definition in the referenced program.
(6) If array variables are defined with Const in a user base program, they are handled as Const variables without definition in the referenced program.

[Related instructions]

Def Inte/Def Long/Def Float/Def Double (Define Integer/Long/Float/Double), Def Char (Define Character), Def Jnt (Define Joint), Def Pos (Define Position), Def Work, Dim (Dim), Static

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>	Describe a GoTo statement or GoSub statement to process an interrupt.
<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 *L100, S        ' Calls the subroutine of *L100 when the general
2                                               ' purpose input signal No. 17 turns on.
3 Def Act 2, MFG1 AND MFG2 GoTo *L200           ' Jumps to *L200 when the AND operation of MFG1
4                                               ' and MFG2 results in "true".
5 Def Act 3, M_Timer(1) > 10500 GoSub *L300, S  ' Calls the subroutine of *L300 after 10.5 seconds
6                                               ' have passed.
7 Act 1=1                                       ' Enables Act 1.
8 Act 2=1                                       ' Enables Act 2.
9 Dly 10
10 End
11 ' Process of the priority No. 1
12 *L100: M_Timer(1) = 0                        ' Resets the timer to zero.
13 Act 3 = 1                                    ' Enables Act 3.
14 Return 0
15 ' Process of the priority No. 2
16 *L200: Mov P_Safe
17 End
18 ' Process of the priority No. 3
19 *L300: M_Timer(1) = 0                        ' Resets the timer to zero.
20 Act 3 = 0                                    ' Disables Act 3.
21 Return 0
```

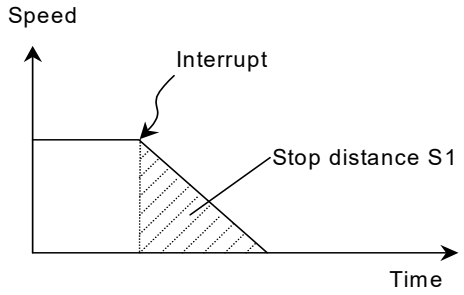
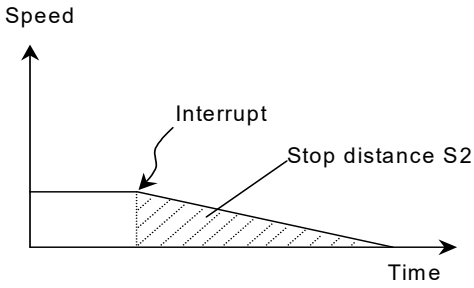
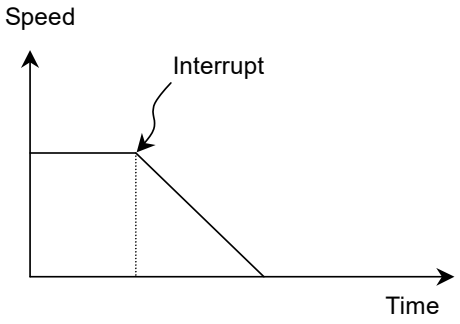
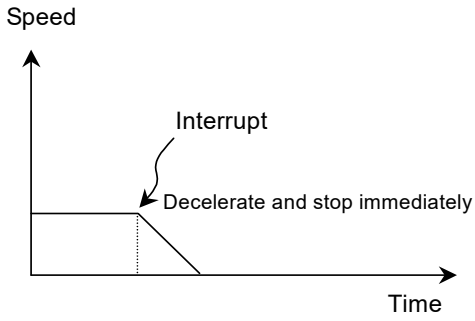
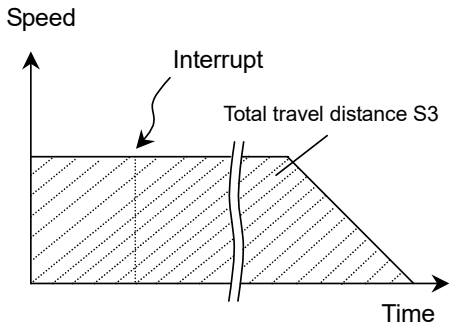
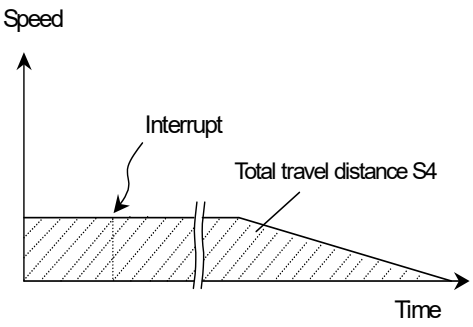
[Explanation]

(1) Writes the Return command at the end of the jump destination processing called up in the interrupt.
(2) 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 173, "Act (Act)" for the interrupt processing.
(3) The priority level for the interrupts is decided by the <Priority No.>, and the priority level, from the highest ranges from 1 to 8.
(4) There can be up to 8 settings for the interrupts. Use the <Priority No.> to differentiate them.
(5) An <expression> should be either a simple logical operation or a comparison operation (one operator). Parentheses cannot be used either.
(6) If two Def Act commands with the same priority number are included in a program, the latter one defined becomes valid.
(7) Since Def Act defines only the interrupt, always use the Act command to designate the enable/disable status of the interrupt.
(8) The communications interrupt (Com) has a higher priority level than any of the interrupts defined by Def Act.
(9) The Def Act command is only enabled in the defined program. Def Act must be redefined with a sub program or specified program when using an interrupt process in a program (sub program) called with the CallP command (call between program), or while a defined function is being executed by a program specified by the #Include statement.
(10) 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.
(11) Expressions containing conditional expressions combined with logical operations, such as (M1 AND &H001) = 1, are not allowed.
(12) When an interrupt occurs during execution of the circular or the arc interpolation (Mvc, Mvr, Mvr2, Mvr3), and then control returns to the interrupted step by Return 0, the robot returns to the start point of the circle or the arc before executing the circular or the arc interpolation again.
(13) When an interrupt occurs during execution of the arch interpolation, and then control returns to the interrupted step by Return 0, the robot executes the arch interpolation from the current position.

⚠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.

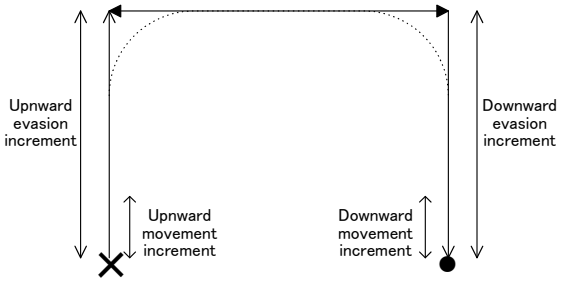
[Format].

```
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 >,
<Upward evasion increment>,
<Downward evasion increment>
Refer to figure at right.
It is possible to specify either a constant or a variable.



<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-FR series)

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-FR/RH-CR series)

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 the shape definition in step 1.
3 Dly 0.3
4 Mva P2,2      'The robot moves according to the default values specified by the parameters.
5 Dly 0.3
```

[Explanation]

(1) If the Mva instruction is executed without the Def Arch command, the robot moves according to the arch shape specified by the parameters.
(2) Used to change the increments in a program, etc.

⚠CAUTION The robot's locus of movement may change with specified speed.
Especially as for the corner section, short cut distance may change. Therefore, when beginning automatic operation, moves at low speed at first, and you should gather speed slowly with being careful of interference with peripheral equipment.

[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 16 characters. Refer to the Page 135, "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 (240 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 145, "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 Work coordinate type: W or w
<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.
Similarly, a user-defined function which is defined in a procedure is valid only in the procedure where it is defined. When using a user-defined function common in multiple procedures, define the function in the global scope (out of the procedure).

Def Gps (Define get position)

[Function]

This command defines the monitored condition for the position data acquisition using the fast-response position data import function (GPS function) for the monitoring.

[Format]

Def[]Gps[] <Monitoring No.>, <Input No.>, <Condition>, [<Mechanism No.>]

[Terminology]

<Monitoring No.> The target monitoring number. Set it with a constant number from 1 to 8.

<Input No.> Set the target input signal number for the monitoring.
SKIP input: 801 to 803

<Condition> Specify the signal-on/off status as a trigger.
The robot position data is stored when the signal set in <Input No.> is turned on/off.

Set value	Conditions
On	On the rising edge of target input signal
Off	On the falling edge of target input signal

<Mechanism No.> Set the target mechanism number for the position data acquisition. (If the argument is omitted, 1 is set as the default value.)

[Reference Program]

```
1 Def Gps 1,801,On,1     ' The position data of the mechanism No. 1 is recorded for the monitoring No. 1
                           when the signal No. 801 is turned on.
2 GpsChk On,1            ' Monitoring a condition for the monitoring No. 1 is started.
3 Mvs P1                 ' Moves to P1
4 GpsChk Off,1           ' Monitoring a condition for the monitoring No. 1 is stopped (the position data
                           obtained is stored).
5 M1 = M_Gps(1)          ' The number of the position data recorded in the P_Gps1 is stored.
6 If M1 = 0 Then Error 9000 ' The error 9000 is generated if no position data is recorded.
7 Mvs P_Gps1(1)          ' The robot moves its arm to the position at the first time when the signal No.
                           801 is turned on.
8 Hlt                    ' Halt program
```

[Explanation]

(1) The current position data of the robot set in <Mechanism No.> is imported when the signal set in <Input No.> satisfies the condition specified in <Condition>. However, the monitoring of the condition defined in this command is not started until the GpsChk On command is executed.

(2) The imported position data of the robot is stored in the P_GpsX() ("X" indicates the same number as the target monitoring number from 1 to 8) when the GpsChk Off command is executed.

(3) The position data stored in the P_GpsX() is cleared to zero when the GpsChk On command for the target monitoring number is executed.

(4) During multitask operation, the monitored condition is independently defined for each task slot. Each defined condition is valid for the target slot only. Up to 8 conditions including the one defined in the Def Map command can be defined for all slots.

(5) This command is not available for the programs in which the starting condition is set to ALWAYS or ERROR.

(6) If several conditions are set for one monitoring number, the previously-executed condition is overwritten by the condition to be executed subsequently.

(7) The condition setting defined in this command is cleared to zero when the program is reset, the End command is executed, or the controller is turned on next time.

(8) The condition defined in this command is retained in a sub program called up by the CallP command. When a new setting for this condition is defined in a sub program, it overwrites the previous setting.

(9) When the same number is set in <Monitoring No.> of this command as in <Monitoring No.> of the Def Map command, the monitored condition set in the command which is executed last is valid (the one set in the command which was executed previously is deleted).
(10) During monitoring by executing the GpsChk On command, the setting of target condition cannot be changed in this command. The setting of condition can be changed after stopping monitoring by executing the GpsChk Off command.

[Related instructions]
GpsChk (Get position check), Def Map (Define mapping)

[Related system status variables]
M_Gps, P_Gps1 to P_Gps8

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 WORK4 = 123.468            ' Numerical "123.468000" is set to WORK4.
```

(4) Definition of the double precision type real number variable.

```
1 Def Double WORK4
2 WORK5 = 100/3              ' Numerical "33.333332061767599" is set to WORK5.
```

[Explanation]

(1) The variable name can have up to 16 characters. Refer to the Page 135, "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 (240 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, M_Inw and M_Outw are used in the case of 16-bit words, and M_In32, M_Out32 are used in the case of 32-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 DWord(32bit).
<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.
```

(4) Assign the input variable named PORT4 to input/output signal number 16 in dword type, and specify the mask information as 3FFFFF in hexadecimal.

```
 1 Def IO PORT4 = DWord, 16, &H3FFFFF
   :
10 PORT4 = 65536                   ' Output signal number 32 turns on.
```

[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 16 characters. Refer to the Page 135, "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 reference program (2) 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.)
0 0 0 0 1 1 1 1
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.)
* * * * 1 1 1 1
(a) (b)
(a) Retains the current output status
(b) 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 16 characters. Refer to the Page 135, "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 (240 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 145, "4.3.24 User-defined external variables" for details.

Def Map (Define mapping)

[Function]

This command defines the monitored condition for the mapping (the workpiece presence recognition) in a cassette, using the get-position-quick function (GPS function).

[Format]

```
Def[]Map[] <Monitoring No.>, <Input No.>, <Condition>, [<Mechanism No.>],
        <Lowest position>, <Highest position>, <Number of segments>, <Sensitivity>
```

[Terminology]

<Monitoring No.> The target monitoring number. Set it with a constant number from 1 to 8.

<Input No.> Set the target input signal number for the monitoring.
SKIP input: 801 to 803

<Condition> Specify the signal-on/off status as a trigger.
The robot position data is stored when the signal set in <Input No.> is turned on/off.

Set value	Conditions
On	On the rising edge of target input signal
Off	On the falling edge of target input signal

<Mechanism No.> Set the target mechanism number for the position data acquisition. (If the argument is omitted, 1 is set as the default value.)

<Lowest position>/<Highest position> Set the lowest/highest position to detect a workpiece in a cassette in <Lowest position>/<Highest position>.
The set positions must be in the path of robot's linear movement for detection.
The position set in <Lowest position> is defined as the first segment in the cassette. If the lowest/highest position is set oppositely, the highest position set in <Lowest position> is defined as the first segment.

<Number of segments> Set the number of segments in the cassette.
Based on the set number of segments, each space divided equally between the lowest and highest positions is regarded as a segment.
Setting range: 1 to 130 (segments)

<Sensitivity> Set the sensitive area of a sensor for improvements in accuracy to detect the presence of a workpiece. (Refer to Fig. 4-23.)
When a position outside the set sensitive area is detected, the detection data is not used for the workpiece presence recognition because the detection information is canceled.
Setting range: 0.1 to 100 (mm)
(0.1 to 100 (deg.) for a multi-mechanism with single axis of rotation)

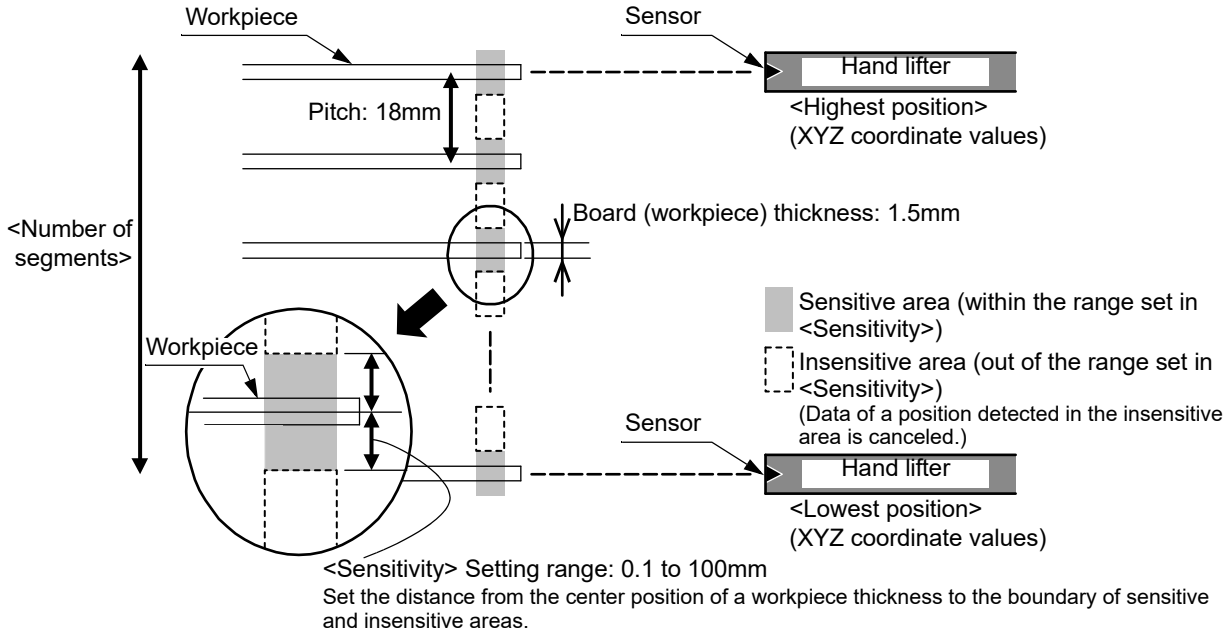


Fig.4-23: Sensitive area of a sensor

[Reference Program]

```
1 Def Map 3,801,On,1,PC1,PC2,20,10 ' The position data of the mechanism No. 1 is recorded for the
                                     monitoring No. 3 when the signal No. 801 is turned on, and the
                                     mapping is executed according to the defined condition.
                                     PC1: Lowest position (first segment) in a cassette, PC2: Highest
                                     position (last segment) in a cassette, 20: the number of
                                     segments in a cassette (20 segments), 10: a sensitive area of a
                                     sensor (10 mm)
2 Mov PM1                          ' The robot moves its arm to the mapping start position.
3 GpsChk On,3                      ' Monitoring a condition for the monitoring No. 3 is started.
                                   ' The position data of the mechanism No. 1 is recorded when the
                                     signal No. 801 is turned on.
                                     Based on the position data, the segment number in which the
                                     workpiece is present is shown as "M_Map3(130)".
4 Mvs PM2                          ' The robot moves its arm to the mapping stop position.
5 GpsChk Off,3                     ' Monitoring a condition for the monitoring No. 1 is stopped. Using
                                     the recorded position data, the segment number in which the
                                     workpiece is present is stored in the M_Map3.
6 M1=M_Gps(3)                      ' The number of the position data recorded in the P_Gps3 is
                                     stored.
7 If M1 = 0 Then Error 9000        ' The error 9000 is generated if no position data is recorded.
8 For M2=1 To 20
9  M_Out(6100+M2)=M_Map3(M2)       ' The results of the mapping are output with a signal (the signal
                                     number "6101" is assigned for the first result).
10 Next M2                         ' The process performed for the first segment is repeated for the
                                     remaining segments.
```

[Explanation]

(1) When the GpsChk Off command is executed, the segment number in which the workpiece is present is calculated from the position data at the time when the target signal is inputted, based on the values set in <Lowest position>, <Highest position>, <Number of segments>, and <Sensitivity>. As a result of the calculation, "1" (the presence of the workpiece) or "0" (the absence of the workpiece) is stored in the status variable M_MapX() ("X" indicates the same number as the target monitoring number from 1 to 8). The segment number, an element in the M_MapX(), is counted from the position set in <Lowest posiiton>, and "1" is output for the first segment (Data of the first segment is shown as "M_MapX(1)".)

(2) The current position data of the robot set in <Mechanism No.> is recorded in the P_GpsX(), and the number of the position data obtained is recorded in the in the M_Gps(n), when the signal set in <Input No.> satisfies the condition specified in <Condition>. However, monitoring the condition defined in this command is not started until the GpsChk On command is executed.
(All of the position data and the number of its data responding to the input signal is stored in the P_GpsX() and M_Gps(n), respectively. They are not linked to the settings in <Sensitivity> and <Number of segments>.)
(3) The position data stored in the P_GpsX(), M_MapX(), and M_Gps(n) is cleared to zero when the GpsChk On command for the target monitoring number is executed.
(4) When the position settings in <Lowest position> and <Highest position> are equal, the number of segments is defined as one, and the presence or absence of the workpiece in that segment is checked (calculated).
(5) During multitask operation, the monitored condition is independently defined for each task slot. Each defined condition is valid for the target slot only. Up to 8 conditions including the one defined in the Def Gps command can be defined for all slots.
(6) This command is not available for the programs in which the starting condition is set to ALWAYS or ERROR.
(7) If several conditions are set for one monitoring number, the previously-executed condition is overwritten by the condition to be executed subsequently.
(8) The condition setting defined in this command is cleared to zero when the program is reset, the End command is executed, or the controller is turned on next time.
(9) The condition defined in this command is retained in a sub program called up by the CallP command. When a new setting for this condition is defined in a sub program, it overwrites the previous setting.
(10) When the same number is set in <Monitoring No.> of this command as in <Monitoring No.> of the Def Gps command, the monitored condition set in the command which is executed last is valid (the one set in the command which was executed previously is deleted).
(11) During monitoring by executing the GpsChk On command, the setting of target condition cannot be changed in this command. The setting of condition can be changed after stopping monitoring by executing the GpsChk Off command.

[Related instructions]
GpsChk (Get position check), Def Gps (Define get position)

[Related system status variables]
M_Gps, P_Gps1 to P_Gps8, M_Map1 to M_Map8

Def Plt (Define pallet)

[Function]

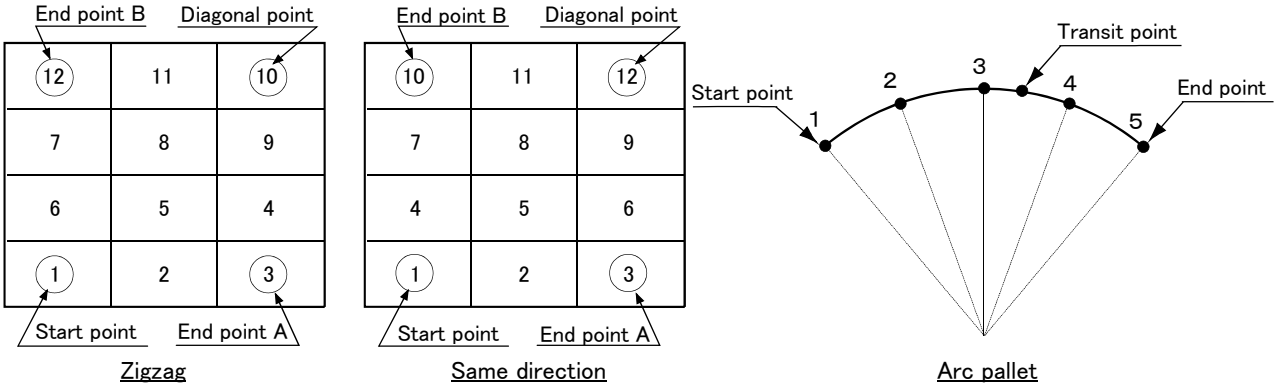
Defines the pallet. (3-point pallet, 4-point pallet)

[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.
(8) In the robot types in which the J1 axis or the J4 axis can exceed the +/-180 degrees, the palette that the joint angle of the J1 axis or the J4 axis straddles the +/-180 degrees cannot be specified. The alarm will occur, if such position were defined.
If you use the pallet in such a position, please divide and define the palette. Refer to Page 108, "4.1.2 Pallet operation" for details.
To operate the robot in the operating range over ±180 degrees, change the parameter setting. For details, refer to the description of the parameter PLTSPEC.
(9) When this command is executed, a check is performed to determine whether the rectangle composed of the <End Point A> and <End Point B> for the specified <Start Point>, and their diagonal points (optional) falls within the robot movement range.
If alarm L.3110 occurs when executing the command, ensure that the above mentioned quadrilaterals lie within the movement range.
(10) If using as a line pallet, define with <Start Point> and <End Point B> as the same point (or with <Start Point> and <End Point A> as the same point).

⚠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 108, "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 108, "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 16 characters. Refer to the Page 135, "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 (240 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 145, "4.3.24 User-defined external variables" for details.

Def Work

[Function]

Declares a work coordinate variable. Use this instruction when the variable name starts with a character other than 'W' (or 'w').

[Format]

```
Def[]Work[]<Work coordinate variable name>[, <Work coordinate variable name>[, ...]]
```

[Terminology]

<Work coordinate variable name> Designate a variable name.

[Reference Program]

```
1 Def Work Box1          ' Defines Box1 as a work coordinate variable.
```

[Explanation]

(1) Use this instruction to define a work coordinate variable with a name starting with a character other than 'W' (or 'w').

(2) The variable name can have up to 32 characters. Refer to the Page 135, "4.3.6 Types of characters that can be used in program" for the characters that can be used.

(3) A variable becomes a user-defined external variable (shared among programs) by adding '_' after 'W' (or 'w') in the variable name and registering it in a user base program.
When using this variable, define it with this instruction.
Refer to "4.3.24User-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> (<Eelement Value> [, <Eelement Value> [, <Eelement Value>]])
        [, <Variable name> (<Eelement Value> [, <Eelement Value> [, <Eelement Value>]])]...
```

[Terminology]

<Variable name>	Describe the name of the array variable.
<Eelement Value>	Describe in terms of constants, the number of elements in an array variable. (1 to 2500)

[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 M3&(5)                 ' Define long-precision real number type array variable M3& having the
                               five elements.
6 Dim CMOJI(7)               ' Define the character-string type variable CMOJI having the seven
                               elements.
7 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) Eelement number start from 1 when actually referencing array variables. For PDATA on step 1 of the statement example, the element number will be 1 to 10.

(4) <Eelement Value> can be described with numeric constants from 1 to 2500. 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 the specified number of elements is larger than the defined number of elements, the error L4370 will occur at 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 command.

(8) The arrays defined by the Dim command are valid only in the program where they are defined. To use these arrays by a sub program called by the CallP command, 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 145, "4.3.24 User-defined external variables" for details.

[Related instructions]
Const

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 (2) in [Reference 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 2 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.

EBRead (EasyBuilder read)

[Function]

Reads out the data by specifying the tag name of the vision sensor.

The data read from the vision sensor is stored in the specified variable.
Please read out data specifying the tag name by using this command when the vision program (job) is made with the vision tool EasyBuilder made by Cognex Corporation.

[Format]

1) When used as a single command

```
EBRead[]#<Vision sensor number>,[<Tag name>],<variable name 1> [,<variable name 2>]..[,<Time out>]
```

[Terminology]

<Vision sensor number>(Can not be omitted)	This specifies the number of the vision sensor to control. Setting range:1 - 8
<Tag name> (Can be omitted)	Specifies the name of symbolic tag where data read out by the vision sensor is stored. When omitting it, the value of paraemter EBRDTAG (initial value is the custom format tag name "Job.Robot.FormatString") is set to it.
<Variable name>(Can not be omitted)	Specifies the variable where the data read from the vision sensor is stored. It is possible to use two or more variables by delimited with commas. It is possible to specify the Numeric value variable, Position variable or String variable. When the Position variable is specified, the value is set to X, Y, and C element, and 0 is set to other elements.
<Time out> (If omitted, 10)	Specifies the time-out time (in seconds). Specification range: Integer 1-32767

[Reference Program]

```
100 If M_NvOpen(1)<>1 Then          ' If vision sensor number 1 log on is not complete
110  NVOpen "COM2:" As #1           ' Connects with the vision sensor connected to COM2.
120 End If
130 Wait M_NvOpen(1)=1              ' Connects with vision sensor number 1 and waits for logon to be
                                      completed.
140 NVLoad #1,"TEST"                ' Loads the "Test" program.
150 NVRun #1,"TEST"                 ' Starts the "Test" program.
160 EBRead #1,,MNUM,PVS1,PVS2       ' The data of "Job.Robot.FormatString" tag is read, and they are
                                      preserved in the variable MNUM, PVS1, and PVS2.
170 --------
  :
300 NVClose #1                      ' Cuts the line with the vision sensor connected to COM2.
```

[Explanation]

(1) Gets the data by specifying the tag name from an active vision program in the specified vision sensor.
(2) The data read from the vision sensor is stored in the specified variable.
(3) When the specified variable identifier is delimited by comma and enumerated when the data of the vision sensor is two or more values (character string) delimited by comma, data is stored in order of describing the variable identifier. In this case, the type of the object data should be the same as the type of the variable.
(4) When the position variable is specified, the vision data is stored in X, Y, and C element. And the value of other elements are 0.
The value converted into the radian is set to C element.
(5) The value of receiving data are set only to the specified variables when the number of specified variables is less than that of receive data.
(6) The variable more than the number of receiving data is not updated when the number of specified variables is more than that of receive data.
(7) When the tag name is omitted, the value of parameter EBRDTAG is set instead of the tag name. (The factory shipment setting is " Job.Robot.FormatString".)

(8) It is possible to specify the timeout time by the numerical value. Within the timeout time, does not move to the next step until the results are received from the vision sensor. However, if the robot program is stopped, this command is immediately cancelled. Processing is continued with a restart.
(9) When this command is used with multi-tasking, it is necessary to execute the NVOpen command and the NVRun command in the task using this command. In this case, use the <vision sensor number> specified with the NVOpen command.
(10) A program start condition of "Always" is not supported.
(11) If an interrupt condition is established while this command is being executed, the interrupt processing is executed immediately even during processing of this command. The processing is executed after completing the interrupt processing.
(12) In order to shorten the tact time, it is possible to do other work after executing the NVRun command and execute EBRead when it is required.
(13) Please set 1 to the parameter NVTRGTMG if the EBRead command is executed immediately after the NVRun command.
When parameter NVTRGTMG is factory shipment setting,the next process of the NVRun command is executed without waiting for the completion of the vision recognition processing. Therefore, the last recognition might be taken when the EBRead command is continuously executed.
(14) Note that if the program stops between NVRun and EBRead, the results when NVRun is executed and the results when EBRead is executed may be different.

<Value of the variable>
The variable by executing the EBRead command is as follows.
(1) Content of specified tag (Pattern_1.Number_Found) is 10
(a) The value when "EBRead #1,"Pattern_1.Number_Found",MNUM" is executed is :
-> MNUM=10

(b) The value when "EBRead #1,"Pattern_1.Number_Found",CNUM" is executed is :
-> CMNUM="10"

(2) Content of specified tag (Job.Robot.FormatString) is 2, 125.75, 130.5, -117.2, 55.1, 0, 16.2
(a) The value when "EBRead #1,,MNUM,PVS1,PVS2" is executed is :
-> MNUM=2
PVS1.X=125.75 PVS1.Y=130.5 PVS1.C=-117.2
PVS2.X=55.1 PVS2.Y=0, PVS2.C=16.2
* The element (Excluding X and Y element) that the vision data is not set is 0.

(b) The value when "EBRead #1,,MNUM,MX1,MY1,MC1,MX2,MY2,MC2" is executed is :
-> MNUM=2
MX1=125.75 MY1=130.5 MC1=-117.2
MX2=55.1 MY2=0 MC2=16.2

(c) The value when "EBRead #1,,CNUM,CX1,CY1,CC1,CX2,CY2,CC2" is executed is :
-> CNUM="2"
CX1="125.75" CY1="130.5" CC1="-117.2"
CX2="55.1" CY2="0" CC2="16.2"

(3) Content of specified tag (Job.Robot.FormatString) is 2, 125.75, 130.5
(a) The value when "EBRead #1,,MNUM,PVS1" is executed is :
-> MNUM=2
PVS1.X=125.75 PVS1.Y=130.5
* The element (Excluding X and Y element) that the vision data is not set is 0.

[Errors]

(1) If data type for an argument is incorrect, a "syntax error in input command statement" error is generated.
(2) If there is an abnormal number of command arguments (too many or too few), an "incorrect argument count" error occurs.
(3) If the <vision sensor number> is anything other than "1" through "8", an "argument out of range" error occurs.
(4) If the NVOpen command is not opened with the number specified as the <vision sensor number>, an "The NVOpen command is not executed." error occurs.
(5) If data type of the strings data received from the vision sensor and the variable substituted for it is difference, a " Illegal Receive data(EBREAD) " error is generated.
(6) If the <Timeout> is other than "1" - "32767", an "argument out of range" error occurs.
(7) If the vision sensor does not respond without the time specified as the <Timeout> or within the first 10 seconds if the <Timeout> parameter is omitted, a "vision sensor response timeout" error occurs.
(8) If the communications line is cut while this command is being executed, an "abnormal communications" error occurs and the robot controller side line is closed.
(9) If the specified tag name does not exist in the active vision program, a “Vision Tag name is abnormal” error is generated.
(10) Please specify 31 variables or less
('number of the recognition' +' position in the coordinate (X,Y,Z)' x 10) .
If 32 variables or more are specified, a "syntax error in input command statement" error is generated.
(11) If the <vision program name> exceeds 15 characters, an "abnormal vision program name" error occurs.
(12) If a <vision program name> uses a character other than "0" - "9", "A" - "Z", "-", or "_" (including lowercase letters), an "abnormal vision program name" error occurs.
(13) If the program specified in the <vision program name> is not in the vision sensor, a "vision program does not exist" error occurs.
(14) If the program specified in the <vision program name> is not started by an NVRun command, a "abnormal vision program name" error occurs.
(15) If the <Recognition count cell>, <Start cell>, or <End cell> contains a number other than "0" - "399" or a letter other than "A - "Z", an "argument out of range" error occurs.
(16) If there is no value in the cell specified in "Recognition count cell", an "invalid value in specified for recognition count cell" error occurs.
(17) If the <Start cell> and <End cell> are reversed, a "specified cell value out of range" error occurs.
(18) If the number of data included in the cell which specifies it by <Start cell> and <End cell> exceeds 90, a"specified cell value out of range" error occurs.
(19) If the range specified by <Star cell> and <End cell> exceeds line 30 and row 10, a"specified cell value out of range" error occurs.
(20) If the <Type> is other than "0" - "7", an "argument out of range" error occurs.

⚠CAUTION

Please set parameter NVTRGTMG.
Please set 1 to parameter NVTRGTMG when EBRead command is executed immediately after NVRun command.
When parameter NVTRGTMG is factory shipment setting (NVTRGTMG=2), the next command is executed without waiting the completion of the vision recognition processing. Therefore, there is a possibility being gotten the last recognition result when EBRead command is continuously executed.

EBWrite (EasyBuilder write)

[Function]

Specifies the tag name of the vision sensor to write data.
When the vision program (job) is made with the vision tool EasyBuilder made by Cognex Corporation, you can write data to the cell specified with the tag name by using this command.

[Format]

```
EBWrite[]#<Vision sensor number>,[<Tag name>],<Writing data> [,<Time out>]
```

[Terminology]

<Vision sensor number>	Specifies the number of the vision sensor to control with a numeric constant. Setting range: 1 to 8
<Tag name>	Specifies the name of the symbolic tag for the cell to which data is written. If the name is not specified, the value of parameter EBWRTAG is set.
<Writing data>	Specifies the data to be written to the vision sensor. Numeric constants, numerical variables, character string constants, character string variables, position element data, joint element data, numeric expressions, or character string expressions can be used.
<Time out>	Specifies the time-out time (in seconds) with a numeric constant. If the time is not specified, a timeout of 10 seconds is set. Setting range: 1 to 32767 (integers)

[Reference Program]

```
1  If M_NvOpen(1)<>1 Then          ' If vision sensor number 1 is not logged on
2  NVOpen "COM2:" As #1            ' Connects with the vision sensor connected to COM2, and the
                                     sensor is numbered 1.
3  Wait M_NvOpen(1) = 1            ' Waits until vision sensor number 1 logs on.
4  End If
5  NVOpen #1, "TEST"               ' Loads the "TEST" program (job).
6  EBWrite #1, "Sample.Float", 5   ' Rewrites "Sample.Float" tag data as 5.
7  EBWrite #1, "Sample.String", "Test"  ' Rewrites "Sample.String" tag data as "Test".
8  NVRun #1, "TEST"                ' Starts "TEST" program (job).
  :
  :
20 End
```

[Explanation]

(1) Writes data to the cell specified with the tag name in the active vision program (job) of the specified vision sensor.
(2) The error (L.3141) occurs if no NVOpen command is executed for the vision sensor specified with <Vision sensor number>.
(3) If <Tag name> is not specified, the value of parameter EBWRTAG is set. (The factory shipment setting is ""(NULL).)
(4) The error (L.8637) occurs if the active vision program does not have the specified <Tag name>.
(5) The type of the data written to the cell of the vision sensor varies depending on the type of <Writing data>. (When a double-precision real number is specified, the single-precision real number converted from the double-precision real number is used.

<Writing data>	Type of data written to cells
Numeric value type (integer)	Integer (Int)
Numeric value type (real number)	Single-precision real number (Float)
Character string type	Character string (String)

(6) Processes are performed according to the combinations of the types of <Writing data> and the cell value types of the vision program specified with <Tag name> as shown below.

<Writing data>	Cell value type	Process
Numeric value type (integer)	Boolean value editing control	Cell value update (integer)
	Integer editing control	Cell value update (integer)
	Floating-point number editing control	Cell value update (single-precision real number)
	Text editing control	Execution error (L.8637)

<Writing data>	Cell value type	Process
Numeric value type (real number)	Boolean value editing control	Cell value update (integer, rounding down decimals)
	Integer editing control	Cell value update (integer, rounding down decimals)
	Floating-point number editing control	Cell value update (single-precision real number)
	Text editing control	Execution error (L.8637)
Character string type	Boolean value editing control	Execution error (L.8637)
	Integer editing control	Execution error (L.8637)
	Floating-point number editing control	Execution error (L.8637)
	Text editing control	Cell value update (character string)

(7) You can specify the time-out time with a numeric constant. During the time-out time, the next step is not performed until data containing writing results is received from the vision sensor.
(8) When the execution of a robot program is stopped, the processing of this command is interrupted. The interrupted processing restarts by executing the program again.
(9) When this command is used with multi-tasking, it is necessary to execute the NVOpen command in the task slot you use. Use the number specified with NVOpen command for <Vision sensor number>.
(10) This command cannot be used if ALWAYS is specified in the start conditions of the task slot.
(11) If interruption conditions have been satisfied during this command, interruption processing will be executed immediately.

EMvc (E Move C)

[Function]

Carries out 3-dimensional circular interpolation movement along the work coordinates system (Ex-T coordinates system) in the order of start point, transit point 1, transit point 2, and start point.
(For the outline of the function, refer to Page 750, "7.3 Ex-T control".)

[Format]

```
EMvc[]<Work Coordinates Number>, <Start Point>, <Transit Point 1>, <Transit Point 2>
        [[]<Appended Conditions>]
```

[Terminology]

<Work Coordinates Number>	A work coordinate number which is chosen from 1 through 8.
<Start Point>	This is the final position for interpolation operation. Describe a position using a position type variable or constant, logic/arithmetic expressions, functions, or a joint variable.
<Transit Point 1>	Transit point 1 for a circular arc. Describe a position using a position type variable or constant, logic/arithmetic expressions, functions, or a joint variable.
<Transit Point 2>	Transit point 2 for a circular arc. Describe a position using a position type variable or constant, logic/arithmetic expressions, functions, or a joint variable.
<Appended Conditions>	The Wth and WthIf statements can be used.

[Reference Program]

(1) Moves to the positions in the order of P1, P2, P3, and P1 along the work coordinate 1 by circular interpolation.

```
1 EMvc 1, P1, P2, P3
```

(2) Moves to the positions in the order of P1, J2, P3, and P1 along the work coordinate 2 by circular interpolation (when the joint variable is used).

```
2 EMvc 2, P1, J2, P3
```

(3) Moves to the positions in the order of P1, P2, P3, and P1 along the work coordinate 2 by circular interpolation, and simultaneously turns on the output signal 17.

```
3 EMvc 2, P1, P2, P3 Wth M_Out(17)=1
```

(4) Turns on the output signal 21 if the input signal 20 is turned on during movement to the positions in the order of P3, (Plt 1, 5), P4, and P3 along the work coordinate 4 by circular interpolation.

```
4 EMvc 4, P3, (Plt 1, 5), P4 WthIf M_In(20)=1, M_Out(21)=1
```

[Explanation]

(1) In Ex-T control circular interpolation motion, a circle along the work coordinates is formed with the 3 given points, and the circumference is moved. (360 degrees)
(2) The posture at the start point is maintained during Ex-T circular 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 Ex-T control linear interpolation, and then performs the Ex-T control circular interpolation.
(4) If paused during execution of a EMvc command and restarted after jog feed, the robot returns to the interrupted position 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 528, "5.10 Automatic return setting after jog feed at pause")
(5) This cannot be used in programs set to ALWAYS or ERROR. (The error L3287 will occur.)
(6) When the work coordinates system is not specified, the initial value is 0 (0, 0, 0, 0, 0, 0) for all elements. If the Ex-T control linear interpolation is executed in this condition, the movement is performed along the world coordinates system origin.
(7) If any additional axis is provided, the additional axis also moves. However, the additional axis is not included in the Ex-T control. Therefore, the robot position may be deviated from its original working position when the robot arm is moved with the additional axis such as the travel axis and the work coordinates is also moved along with the robot. Accordingly, operating an additional axis using this command is not recommended. Also, do not perform synchronous control of additional axis while using this command.

[Related instructions]
EMvr (E Move R), EMvr2 (E Move R 2), EMvr3 (E Move R 3), EMvs (E Move S)

[Related system variables]
P_WkCord (Work coordinates data)

[Related parameter]
WK1CORD to WK8CORD

EMvr (E Move R)

[Function]

Carries out 3-dimensional circular arc interpolation movement along the work coordinates system (Ex-T coordinates system) from the start point to the end point via transit points.
(For the outline of the function, refer to Page 750, "7.3 Ex-T control".)

[Format]

```
EMvr[]<Work Coordinates Number>, <Start Point>, <Transit Point>, <End Point>
        [[]Type[] <Constants 1>, <Constants 2>][[]<Appended Conditions>]
```

[Terminology]

<Work Coordinates Number>	A work coordinate number which is chosen from 1 through 8.
<Start Point>	Start point for the arc. Describe a position using a position type variable or constant, logic/arithmetic expressions, functions, or a joint variable.
<Transit Point>	Transit point for the arc. Describe a position using a position type variable or constant, logic/arithmetic expressions, functions, or a joint variable.
<End Point>	End point for the arc. Describe a position using a position type variable or constant, logic/arithmetic expressions, functions, or a joint variable.
<Constants 1>	Specify the posture rotation. Short cut = 0, The default value is 0.
<Constants 2>	Specify the posture interpolation method. Equivalent rotation/3-axis XYZ = 0/1. The default value is 0.
<Appended Conditions>	The Wth and WthIf statements can be used.

[Reference Program]

(1) Moves to P3 from P1 via P2 along the work coordinate 1 by circular arc interpolation.
```
1 EMvr 1, P1, P2, P3
```
(2) Moves to P3 from P1 via J2 along the work coordinate 2 by circular arc interpolation.
```
2 EMvr 2, P1, J2, P3
```
(3) Moves to P3 from P1 via P2 along the work coordinate 2 by circular arc interpolation, and simultaneously turns on the output signal 17.
```
3 EMvr 2, P1, P2, P3 Wth M_Out(17)=1
```
(4) Turns on the output signal 21 if the input signal 20 is turned on during movement from P3 to P4 via (Plt 1, 5) along the work coordinate 4 by circular arc interpolation.
```
4 EMvr 4, P3, (Plt 1, 5), P4 WthIf M_In(20)=1, M_Out(21)=1
```

[Explanation]

(1) In Ex-T control circular arc interpolation motion, a circular arc along the work coordinates is formed with the 3 given points, and the movement is performed along the circular arc.
(2) Interpolation is performed for the posture at the start point and the end point. The posture at the transit point does not affect the interpolation.
(3) If the current position and the starting position do not match, the robot automatically moves to the starting point based on the Ex-T control linear interpolation, and then performs the Ex-T control circular arc interpolation.
(4) If paused during execution of a EMvr command and restarted after jog feed, the robot returns to the interrupted position and restarts the remaining circular arc interpolation. The interpolation method (JOINT interpolation / XYZ interpolation) which returns to the interrupted position can be changed by the "RETPATH" parameter. (Refer to Page 528, "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, the Ex-T control 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) This cannot be used in programs set to ALWAYS or ERROR. (The error L3287 will occur.)
(9) When the work coordinates system is not specified, the initial value is 0 (0, 0, 0, 0, 0, 0) for all elements. If the Ex-T control linear interpolation is executed in this condition, the movement is performed along the world coordinates system origin.

(10) If any additional axis is provided, the additional axis also moves. However, the additional axis is not included in the Ex-T control. Therefore, the robot position may be deviated from its original working position when the robot arm is moved with the additional axis such as the travel axis and the work coordinates is also moved along with the robot. Accordingly, operating an additional axis using this command is not recommended. Also, do not perform synchronous control of additional axis while using this command.
(11) If 1 (3-axis XYZ) is specified to <Constants 2> of this command, movement of the robot is the same as 1 (3-axis XYZ) is specified to <Constants 2> of Mvr command.

[Related instructions]
EMvc (E Move C), EMvr2 (E Move R 2), EMvr3 (E Move R 3), EMvs (E Move S)

[Related system variables]
P_WkCord (Work coordinates data)

[Related parameter]
WK1CORD to WK8CORD

EMvr2 (E Move R 2)

[Function]

Carries out 3-dimensional circular arc interpolation movement along the work coordinates system (Ex-T coordinates system) from the start point to the end point on the arc composed of the start point, end point, and reference points. The movement direction does not pass through the reference points.
(For the outline of the function, refer to Page 750, "7.3 Ex-T control".)

[Format]

```
EMvr2[]<Work Coordinates Number>, <Start Point>, <End Point>, <Reference Point>
        [[]Type[] <Constants 1>, <Constants 2>][[]<Appended Conditions>]
```

[Terminology]

<Work Coordinates Number>	A work coordinate number which is chosen from 1 through 8.
<Start Point>	Start point for the arc. Describe a position using a position type variable or constant, logic/arithmetic expressions, functions, or a joint variable.
<End Point>	End point for the arc. Describe a position using a position type variable or constant, logic/arithmetic expressions, functions, or a joint variable.
<Reference Point>	Reference point for a circular arc. Describe a position using a position type variable or constant, logic/arithmetic expressions, functions, or a joint variable.
<Constants 1>	Specify the posture rotation. Short cut = 0, The default value is 0.
<Constants 2>	Specify the posture interpolation method. Equivalent rotation/3-axis XYZ = 0/1. The default value is 0.
<Appended Conditions>	The Wth and WthIf statements can be used.

[Reference Program]

```
1 EMvr2 1, P1, P2, P3
2 EMvr2 2, P1, J2, P3
3 EMvr2 2, P1, P2, P3 Wth M_Out(17)=1
4 EMvr2 4, P3, (Plt 1, 5), P4 WthIf M_In(20)=1, M_Out(21)=1
```

[Explanation]

(1) In circular arc 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 528, "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) This cannot be used in programs set to ALWAYS or ERROR. (The error L3287 will occur.)
(10) When the work coordinates system is not specified, the initial value is 0 (0, 0, 0, 0, 0, 0) for all elements. If the Ex-T control linear interpolation is executed in this condition, the movement is performed along the world coordinates system origin.
(11) If any additional axis is provided, the additional axis also moves. However, the additional axis is not included in the Ex-T control. Therefore, the robot position may be deviated from its original working position when the robot arm is moved with the additional axis such as the travel axis and the work

coordinates is also moved along with the robot. Accordingly, operating an additional axis using this command is not recommended. Also, do not perform synchronous control of additional axis while using this command.

(12) If 1 (3-axis XYZ) is specified to <Constants 2> of this command, movement of the robot is the same as 1 (3-axis XYZ) is specified to <Constants 2> of Mvr command.

[Related instructions]

EMvc (E Move C), EMvr (E Move R), EMvr3 (E Move R 3), EMvs (E Move S)

[Related system variables]

P_WkCord (Work coordinates data)

[Related parameter]

WK1CORD to WK8CORD

EMvr3 (E Move R 3)

[Function]

Carries out 3-dimensional circular arc interpolation movement along the work coordinates system from the start point to the end point on the arc composed of the start point, end point, and center point.
(For the outline of the function, refer to Page 750, "7.3 Ex-T control".)

[Format]

```
EMvr3[]<Work Coordinates Number>, <Start Point>, <End Point>, <Center Point>
        [[]Type[] <Constants 1>, <Constants 2>][[]<Appended Conditions>]
```

[Terminology]

<Work Coordinates Number>	A work coordinates number which is chosen from 1 through 8.
<Start Point>	Start point for the arc. Describe a position using a position type variable or constant, logic/arithmetic expressions, functions, or a joint variable.
<End Point>	End point for the arc. Describe a position using a position type variable or constant, logic/arithmetic expressions, functions, or a joint variable.
<Center Point>	Center point for the arc. Describe a position using a position type variable or constant, logic/arithmetic expressions, functions, or a joint variable.
<Constants 1>	Specify the posture rotation. Short cut/detour = 0, The default value is 0.
<Constants 2>	Specify the posture interpolation method. Equivalent rotation/3-axis XYZ = 0/1. The default value is 0.
<Appended Conditions>	The Wth and WthIf statements can be used.

[Reference Program]

```
1 EMvr3 1, P1, P2, P3
2 EMvr3 2, P1, J2, P3
3 EMvr3 2, P1, P2, P3 Wth M_Out(17)=1
4 EMvr3 4, P3, (Plt 1, 5), P4 WthIf M_In(20)=1, M_Out(21)=1
```

[Explanation]

(1) In circular arc 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 528, "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) The central angle from the start point to the end point always satisfies 0 < central angle < 180 degrees.
(8) 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.
(9) 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.
(10) 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.
(11) This cannot be used in programs set to ALWAYS or ERROR. (The error L3287 will occur.)
(12) When the work coordinates system is not specified, the initial value is 0 (0, 0, 0, 0, 0, 0) for all elements. If the Ex-T control linear interpolation is executed in this condition, the movement is performed along the world coordinates system origin.

(13) If any additional axis is provided, the additional axis also moves. However, the additional axis is not included in the Ex-T control. Therefore, the robot position may be deviated from its original working position when the robot arm is moved with the additional axis such as the travel axis and the work coordinates is also moved along with the robot. Accordingly, operating an additional axis using this command is not recommended. Also, do not perform synchronous control of additional axis while using this command.
(14) If 1 (3-axis XYZ) is specified to <Constants 2> of this command, movement of the robot is the same as 1 (3-axis XYZ) is specified to <Constants 2> of Mvr command.

[Related instructions]
EMvc (E Move C), EMvr (E Move R), EMvr2 (E Move R 2), EMvs (E Move S)

[Related system variables]
P_WkCord (Work coordinates data)

[Related parameter]
WK1CORD to WK8CORD

EMvs (E Move S)

[Function]

Carries out linear interpolation movement along the work coordinates system (Ex-T coordinates system) from the current position to the movement target position.

(For the outline of the function, refer to Page 750, "7.3 Ex-T control".)

[Format]

```
EMvs[]<Work Coordinates Number>, <Target Position> [[]Type[]<Constants 1>,
        <Constants 2>][[]<Appended conditions>]
```

[Terminology]

<Work Coordinates Number>	A work coordinate number which is chosen from 1 through 8.
<Target Position>	This is the final position for interpolation operation. This position may be specified using a position type variable and constant, logic/arithmetic expressions, functions, or a joint variable. (Input the position data of the robot control point.)
<Constants 1>	Specify the posture rotation. Short cut = 0, The default value is 0.
<Constants 2>	Specify the posture interpolation method. Equivalent rotation/3-axis XYZ = 0/1. The default value is 0.
<Appended Conditions>	The Wth and WthIf statements can be used.

[Reference Program]

(1) Moves to the target position P1 along the work coordinate 1 by linear interpolation.

```
1 EMvs 1,P1
```

(2) Moves to the target position (Plt 1,5) along the work coordinate 3 by linear interpolation.

```
1 EMvs 3,(Plt 1,5)
```

(3) Moves to the target position P1 along the work coordinate 2 by linear interpolation, and simultaneously turns on the output signal 17.

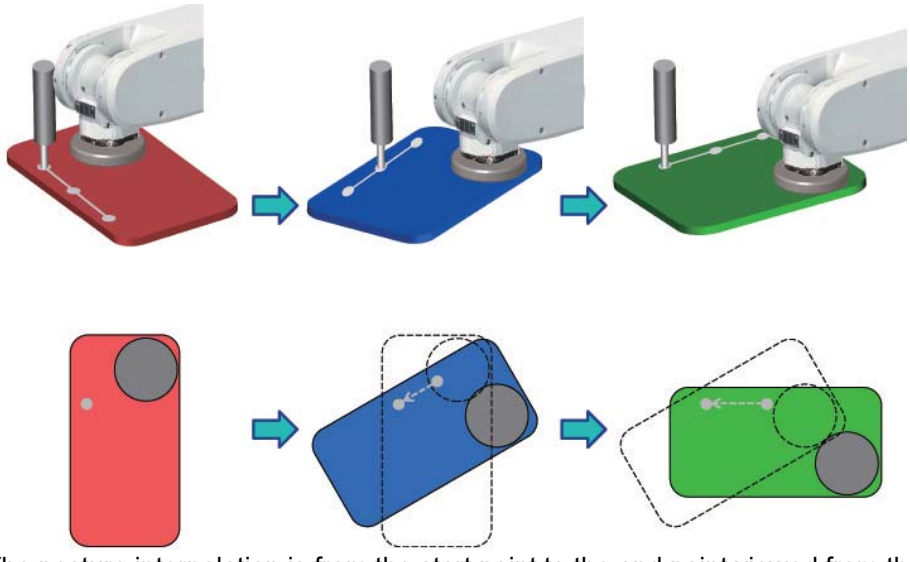
```
1 EMvs 2,P1 Wth M_Out(17)=1
```

(4) Turns on the output signal 20 if the input signal 18 is turned on during movement to the target position P4 along the work coordinate 1 by linear interpolation.

```
1 EMvs 1,P4 WthIf M_In(18)=1, M_Out(20)=1
```

[Explanation]

(1) The EMvs command (Ex-T control linear interpolation) is for a type of movement in which the robot moves from its current position to the movement target position along the specified work coordinates system.
The figure shows an example of movement by the EMvs command. The example shows linear interpolation along the Ex-T control point (Ex-T coordinates system origin) while the posture is changed.



(2) The posture interpolation is from the start point to the end point viewed from the work coordinates.

(3) The work coordinate numbers 1 to 8 correspond to WK1CORD to WK8CORD.
(4) MELFA-BASIC V/VI can be used.
(5) The movement speed during Ex-T control linear interpolation can be also specified by the Spd command. The Ovrd command or the O/P override setting also changes the speed.
(6) The acceleration/deceleration of the Ex-T control linear interpolation is optimum when Oadl ON is set (initial value). The Accel command is valid.
(7) The MvTune command can switch the operation mode.
(8) Path connection by the Cnt command and the positioning accuracy specification by the Fine command are also available.
(9) The Ex-T control linear interpolation can be interrupted or resumed during operation. Returning to the interrupted position is performed by the normal interpolation operation, not by the Ex-T control linear interpolation. The interpolation type to the interrupted position depends on the RETPATH parameter setting. The initial setting is "returning to the interrupted position by JOINT interpolation".
(10) This cannot be used in programs set to ALWAYS or ERROR. (The error L3287 will occur.)
(11) When the work coordinates system is not specified, the initial value is 0 (0, 0, 0, 0, 0, 0) for all elements. If the Ex-T control linear interpolation is executed in this condition, the movement is performed along the world coordinates system origin.
(12) If any additional axis is provided, the additional axis also moves. However, the additional axis is not included in the Ex-T control. Therefore, the robot position may be deviated from its original working position when the robot arm is moved with the additional axis such as the travel axis and the work coordinates is also moved along with the robot. Accordingly, operating an additional axis using this command is not recommended. Also, do not perform synchronous control of additional axis while using this command.
(13) If 1 (3-axis XYZ) is specified to <Constants 2> of this command, movement of the robot is the same as 1 (3-axis XYZ) is specified to <Constants 2> of Mvs command.

[Related instructions]
EMvc (E Move C), EMvr (E Move R), EMvr2 (E Move R 2), EMvr3 (E Move R 3)

[Related system variables]
P_WkCord (Work coordinates data)

[Related parameter]
WK1CORD to WK8CORD

EMvSpl (E Move Spline)

[Function]

Spline interpolation in alignment with the Ex-T coordinate system origin is executed based on the information registered in the designated spline file.
(Refer to Page 680, "7.2 Spline interpolation" for details on spline interpolation.)

[Format]

```
EMvSpl []<Ex-T coordinates number>, <Spline No.>, <Speed>,
        <Acceleration/deceleration distance> [, <Frame transformation>
        [, <Posture interpolation type> [, <Filter length>]]]
```

[Terminology]

<Ex-T coordinates number>	The number of Ex-T coordinates as a control point is designated with a constant or numerical variable. Setting range: 0: The Ex-T coordinate system origin data registered into the spline file is used. 1 to 8: The origin data of work coordinate system corresponding to the number is used.
<Spline No.>	The number of the spline file holding information on the path to be moved is designated with a constant or numerical variable. Setting range: 1 to 99
<Speed>	The spline interpolation movement speed is designated with a constant or numerical variable. Setting range: Value larger than 0.0 (mm/s).
<Acceleration/deceleration distance>	The spline interpolation acceleration/deceleration distance is designated with a constant or numerical variable. The acceleration distance is the distance required to accelerate from the start or movement to the designated speed. The deceleration distance is the distance required to decelerate from the designated speed to the end position. With spline interpolation, the acceleration distance and deceleration distance are a common setting. Setting range: Value larger than 0.0 (mm)
<Frame transformation>	The details for executing frame transformation are designated with a constant or numerical variable. Setting range: 0: Frame transformation is not executed. 1: Frame transformation is executed using coordinate system set in spline file. 2: Frame transformation is executed using coordinate system set with SetCalFrm command. When omitted: Frame transformation is not executed.
<Posture interpolation type>	The posture interpolation type is designated with a constant or numerical variable. Setting range: 0 (equivalent rotation movement)/1 (3-axis orthogonal movement) When omitted: Equivalent rotation movement
<Filter length>	The filter length applied on the acceleration/deceleration movement is designated with a constant or numerical variable. Setting range: 0.0 to 1000.0 (ms) When omitted: 100.0 ms

[Reference Program]

```
1 EMvSpl 0,2,20,10    ' Executes Ex-T spline interpolation to shift the spline curve generated with the
                        path point data set in spline file 2 along the Ex-T coordinate system origin
                        registered in the spline file.
2 EMvSpl 7,2,30,10    ' Executes Ex-T spline interpolation to shift the spline curve generated with the
                        path point data set in spline file 2 along work coordinate 7.
```

[Explanation]

(1) If 1 to 8 is specified for the <Ex-T coordinate No.>, spline interpolation along the coordinate system data set in WK1CORD to WK8CORD is executed. If 0 is specified, spline interpolation along the Ex-T coordinate system origin registered in the spline file is executed.

(2) If the coordinate system is not set in the spline file, the error L2610 (The setting of Ex-T is illegal) will occur.

(3) Ex-T spline interpolation is carried out based on the path point data, etc., registered in the spline file corresponding to <Spline No.>.

(4) The Ex-T coordinate system origin viewed from the workpiece being grasped moves so that the curve is smooth.

(5) If the robot's current position at the start of EMvSpl command execution and the Ex-T spline interpolation start position are deviated, the robot will move to the start position with Ex-T linear interpolation (EMvs command) and then start spline interpolation.

(6) If a spline file corresponding to the <Spline No.> is not saved in the controller, the error L2610 (Can't open spline file) will occur.

(7) The Ex-T spline interpolation command speed (Ex-T coordinate system speed viewed from workpiece) is specified with the following equation.
Speed command = (EMvSpl command's <Speed>) × (Ovrd command) × (operation panel override)
The Spd command and JOvrd command settings are not used. Even if the status variable M_NSpd is set for <Speed>, the optimum speed control mode does not function.

(8) The Ex-T spline interpolation acceleration/deceleration is designated with <Acceleration/deceleration distance>. The Accel command setting is not used. Even if the optimum acceleration/deceleration control is enabled with the Oadl command, it will not be applied to Ex-T spline interpolation.

(9) The acceleration movement generates speed to reach the <Speed> when the <Acceleration/deceleration distance> is moved from the Ex-T spline interpolation start position. The deceleration movement generates the speed to move the <Acceleration/deceleration distance> from the <Speed> and stop at the end position.
If the Ovrd command and operation panel's override is set to a value smaller than 100%, the commanded speed will be reached at a shorter travel rate than <Acceleration/deceleration distance>. (The effect of filters, etc., can cause the acceleration/deceleration distance to be actually longer than the value set with <Acceleration/deceleration distance>.

(10) If "1 (execute using coordinate system set in spline file)" or "2 (execute using coordinate system set with SetCalFrm command) is designated in <Frame transformation>, the path point data is converted sequentially based on the designated method, and Ex-T spline interpolation is executed to pass through those path points.

(11) If the coordinate system is not set in the spline file even though "1 (execute using coordinate system set in spline file)" is designated in <Frame transformation>, error L2042 (Frame transformation coordinates are not set) will occur.

(12) If the coordinate system cannot be calculated when "1 (execute using coordinate system set in spline file)" is designated in <Frame transformation>, error L2041 (Can't calculate frame transformation coordinates) will occur.

(13) If "2 (execute using coordinate system set with SetCalFrm command)" is designated for <Frame transformation>, frame conversion is executed using the coordinate system set with the SetCalFrm command executed last. If the SetCalFrm command has not been executed even once and the coordinate system is not set, error L2042 (Frame transformation coordinates are not set) will occur.

(14) Switch between "0 (equivalent rotation movement)" and "1 (3-axis orthogonal movement)" by designating <Posture interpolation type>. The equivalent rotation movement operates so that the posture change between the path points has the minimum angle.
The 3-axis orthogonal motion generates the posture data with the J4, J5 and J6 axis angles instead of the A, B and C axis coordinate values. Thus, even if the path point posture data (A, B, C axis coordinate values) are the same, the robot's posture will change during the movement.

(15) Transfer of the configuration flag (peculiar point transit) is not supported. If the configuration flag differs between path points, error L2611 (Path point configuration flag is different) will occur.
(16) The filter characteristics applied on the acceleration/deceleration movement can be changed with <Filter length>. Increase the filter length to make the movement during acceleration and deceleration smoother. Note that if the filter length is too high, the movement will slow down and it will take a long time for the Ex-T spline interpolation to end.
(17) If a signal output is set for the path point data, the general purpose output signal will be output with the designated conditions when passing through the target path point.
(18) If the robot positioning complete condition is specified (Fine command, Fine J command, or Fine P command), positioning completion will be confirmed at the Ex-T spline interpolation end position.
(19) Even if continuous movement is designated with the Cnt command, continuous movement will not take place at the Ex-T spline interpolation start position and end position.
(20) Even if "Stop type 1" is designated for the Def Act command (interrupt condition and process definition), if an interrupt occurs during the Ex-T spline interpolation, the robot will stop with the same deceleration as "Stop type 2".
(21) The additional axis will not move from the Ex-T spline interpolation start position. Even if an additional axis position is set for one of the interim path points, it will not be used during Ex-T spline interpolation.
(22) Step feed can be executed for the EMvSpl command, but step return cannot be executed. Error L2612 (Cannot execute step return) will occur.
(23) The mechanism's control rights (GetM command) are required to execute the EMvSpl command.
(24) The EMvSPl command cannot be executed in a slot where the start conditions are ALWAYS or ERROR. Error L3287 (Cannot execute (ERROR ALWAYS)) will occur.
(25) If a value outside the setting range is set for the EMvSpl command argument, error L3110 (Argument value is out of range (EMvSpl)) will occur.

⚠CAUTION If an <Acceleration/deceleration distance> too short for the <Speed> is designated, sudden acceleration/deceleration will take place and could cause robot vibration or a servo error. Always set an appropriate acceleration/deceleration distance.

Related instructions]
SetCalFrm (Set Calibration Frame)

[Related system variables]
M_SplPno, M_SplVar, P_WkCord, P_ECord

[Related parameter]
WKnCORD("n" is 1 to 8), SPLOPTGC

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]
Normal programs

```
Error[]<Error No.>
```

Constantly-executed programs (ALWAYS attribute)

```
Error[]<Error No.>, <Conditions>
```

[Terminology]
<Error No.> Either a constant or numeric operation expression can be set. Designate the No. within the range of 9000 to 9299.
<Conditions> 0: Execution of the program which executed this Error command stops with error.
1: Execution of the program which executed this Error command does not stop with error.

[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 command 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

Exit

[Function]
Exits from a Function procedure.

[Format]

Exit[]Function

[Reference Program]

```
1 Function V Func
2 ' :
3  Exit Function                    ' Exits from the function processing.
4 FEnd
```

[Explanation]
(1) The program exits from the function processing (Function procedure).

[Related instructions]
Function... FEnd

Fine (Fine)

[Function]

This instruction specifies completion conditions of the robot's positioning with number of encoder pulses. 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 command.

[Format]

Fine[]<No. of pulses> [, <Axis No.>]

[Terminology]

<No. of pulses>	Specify the positioning encoder 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 command 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.

```
1 Mov P1
2 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.

(8) The Fine command is affected by the encoder resolution or the gear ratio because positioning judgment is made using encoder pulses of the motor axis. The output angle per pulse differs depending on the model or the axis. To monitor positioning using the unit of the output axis (in units of mm or degree), use the Fine P command or the Fine J command.

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]

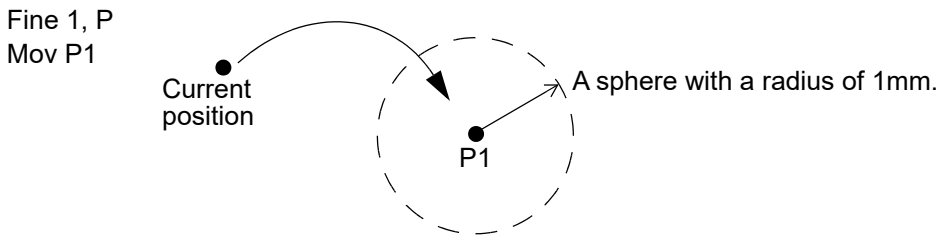
```
1 Fine 1, P          'Specifies the positioning linear distance to 1 mm.
2 Mov P1
3 Fine 0, P          'Disables the positioning linear distance specification.
4 Mov P2
```

[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.



Positioning is completed when the current position moves into the sphere with a radius of 1mm with P1 set to the center.

Fig.4-24:The example of Fine P command use

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 x 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.
```

(3) Process of the For-Next can be skipped by Break

```
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   If M_In(8)=1 Then Break ' If the input signal 8 is turned on, jump to Step 6.
5 Next                      ' Return to step 2.
6 If M_BrkCq=1 Then Hlt
```

[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.
(6) In the For, it can escape to the next step of the Next by Break. That is, process of the For-Next can be skipped.

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 *LBL
3 Mov P1
4 *LBL
5 Mvs P2                        ' Return to the main program.
6 End
```

[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)

Function... FEnd

[Function]
Defines a Function procedure (function).

[Format]

```
Function[]<Return value type>[]<Procedure name>([<Dummy Argument>[, <Dummy Argument>]...])
  <Function processing>
  <Procedure name> = <Return value>
  Exit[]Function
FEnd
```

[Terminology]

<Return value type>	Designate a return value type. P: XYZ type J: Joint type W: Work coordinate type M: Numeric value type (Designate a type with a postfix. When a postfix is omitted, the single-precision real number type is designated.) C: Character string type V: No return value (void type)
<Procedure name>	A user-selected character string (Up to 32 alphanumeric characters)
<Dummy Argument>	The argument names are the same as the variable names corresponding to the five data types in "4.3.8Data type". To pass an argument, add the keyword of "ByVal" (call by value, it can be omitted)/ "ByRef"(call by reference) before the argument name.
<Function processing>	Describe the function processing.
<Return value>	Describe the function return value in a specified type.

[Reference Program]

```
1 Function Main
2    Dim Mary(10)
3    For Idx=1 To 10 Step 1  ' Processing for storing a value in Mary
4       Mary(Idx) = Idx
5    Next
6    Msum = Sum(Mary, 10) ' Calculates the total value of Mary.
7 FEnd
8 ' ----------------------------------------
9 ' Calculates the total value of the first dimensional array specified in the first argument.
10 ' Return value: Total value
11 ' First argument: First dimensional array
12 ' Second argument: Number of elements
13 Function M Sum(ByRef Mary(), ByVal Melem)
14    Msum = 0
15    For Idx=1 To Melem Step 1
16       Msum = Msum + Mary(Idx)
17    Next
18    Sum = Msum            ' Returns the total value of Mary.
19    Exit Function
20 FEnd
```

[Explanation]

(1) Up to 256 Function procedures can be defined.
(2) Specification of the return value can be omitted. When the specification of the return value is omitted, the void type is set.
(3) Up to 16 dummy arguments can be defined.
(4) When a procedure is defined, a Function Main procedure needs to be defined as an entry point in the program file. (Write the program conventionally in a program file in which a Function Main procedure does not exist.)
(5) When the program in which a Function Main procedure exists is called with the CallP instruction, the argument of the CallP instruction is passed to the dummy argument of the procedure. (The FPrm instruction is not required.)
Note that a Function Main procedure without dummy arguments cannot be defined when a Function Main procedure with dummy arguments is defined.
(6) In a program with a Function Main procedure, instruction statements (except for instructions that can be executed by the preprocessor) must be described in the procedure.
*Instructions that can be executed by the preprocessor are as follows.
• #Include statement
• Definition instruction Note 1)
• Conditional branching/Repetition instruction
• Arithmetic operation/Comparison operation
• Tool/Base instruction
Note1) Any processing to call a subroutine (GoSub...Return statement) or GoTo statement cannot be described with instructions (including the Def Act instruction) which are defined outside the procedure.
(7) All the variables defined in a program in which a Function procedure does not exist are handled as global variables in the program.
(8) To call a Function procedure, parentheses must be described after <Procedure name>.
(9) To specify an array for an argument, add "(<Number of elements>[, <Number of elements>[, <Number of elements>]])" to the argument name and declare the function.
Omitting <Number of elements> can pass an array whose number of elements and dimensions is undefined. (* Up to third dimensions)
For correspondence between dummy arguments and actual arguments when an array is specified, refer to the following table.

		Actual argument		
		Not an array	Array (The number of elements is determined.)	Array (The number of elements is not determined.)
Dummy argument	Not an array	OK	NG	NG
	Array (The number of elements is determined.)	NG	OK (Only when the number of elements is the same as the number of dimensions.)	NG
	Array (The number of elements is not determined.)	NG	OK	OK

(10) When an array is specified for an argument and <Number of elements> is omitted, the <Number of elements> in the later dimensions is not determined.
(11) When element data is specified for an actual argument, call by value is always used.
(12) Condition variables cannot be specified for arguments.
(13) The variables declared and defined in the function are handled as local variables, which are valid only in the function.
(14) If a variable with the same name as a local variable has already been declared and defined outside the function, the local variable takes precedence.
(15) Function can be saved in the static variables.
In the variable editor, the program is displayed as follows.
"<Procedure name>::<Variable name>"

(16) The local variable value is discarded after the function is ended.
(17) To hold the local variable value even after the function is ended, add a Static keyword before the variable name at variable declaration to change the local variable to a static variable.
(18) If no processing is performed for the return value, the default value of the return value type is returned. The following shows the default values of each data type.
* XYZ/Work coordinate type: (0,0,0,0,0,0,0,0)(,)
* Joint type: (0,0,0,0,0,0,0,0)
* Numeric value type: 0
* Character string type: ""
(19) The Function exits from the function processing with Exit Function.
(20) The following values take over the set values even at the function call destination (and the function call source).
Cnt, Fine, MvTnue, Tool, Base, Accel, Spd, Ovrd, JOvrd
(21) Set parameter FUNCSPEC to "1" when combining other program read commands (CallP, XLoad, XRun, #Include).

[Related parameter]
FUNCSPEC

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.

Commands	Accel, Base, CavChk, Cmp Jnt, Cmp Pos, Cmp Tool, Cmp Off, CmpG, Cnt, ColChk, ColLvl, Def Plt, Plt, Fine, Fsc, FsGChg, HOpen, HClose, JRC, LoadSet, Mov, Mva, Mvc, Mvr, Mvr2, Mvr3, Mvs, MvSpl, MvTune, Mxt, Oadl, P_Base, Prec, Servo, SetCalFrm, Spd, SpdOpt, Tool, Torq, Trk, EMvc, EMvr, EMvr2, EMvr3, EMvs
Functions	Align, PosCq, PosMid * The following commands do not require GetM for Mechanism 1 if the controller software version is the one mentioned below or later. In this case, the commands can be executed without generating an error. JtoP, PtoJ ..... Ver.N8/P8 or later Inv ..... Ver.R3a/S3a(Special ES Ver.R3dZ) Fram ..... Ver.R3a/S3a(Special ES Ver.R3dZ)
Robot status variables	J_Curr, J/Fbc, J_Origin, P_Curr, P_CurrR, P_Fbc, P_Safe, P_Tool, M_Rspd, M_SetAdl

(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.

GpsChk (Get position check)

[Function]

This command starts/stops monitoring the condition defined in the Def Gps command or Def Map command using the get-position-quick function (GPS function).
Before executing this command, define the monitored condition in the Def Gps command or Def Map command.

[Format]

GpsChk[] <On/Off>, <Monitoring No.>

[Terminology]

<On/Off>	On: Monitoring of the condition defined in the Def Gps command or Def Map command is started. The data stored in the M_Gps(n), P_GpsX(), and M_MapX() corresponding the target monitoring number is cleared to zero ("X" indicates the same number as the target monitoring number from 1 to 8). Off: Monitoring of the target input signal is stopped. The data being stored in the M_Gps(n), P_GpsX(), and M_MapX() corresponding to the target monitoring number is imported.
<Monitoring No.>	Set the same monitoring number as set in the Def Gps command or Def Map command.

[Reference Program]

```
1  Def Gps 7,852,Off,1    ' The position data of the mechanism No. 1 is recorded for the monitoring No. 7
                          when the signal No. 852 is turned off.
2  GpsChk On,7            ' Monitoring a condition for the monitoring No. 7 is started.
3  Mvs P1                 ' Moves to P1
4  GpsChk Off,7           ' Monitoring a condition for the monitoring No. 7 is stopped.
```

[Explanation]

(1) This command starts/stops monitoring the condition defined in the Def Gps command or Def Map command.
(2) The data set in <Monitoring No.> in the P_GpsX() and M_Gps(n) is cleared to zero when the GpsChk On command is executed.
(3) The data set in <Monitoring No.> in the P_GpsX() and M_Gps(n) is created when the GpsChk Off command is executed. Before using the data obtained using the get-position-quick function (GPS function), execute the GpsChk Off command.
(4) This command is not available for the programs in which the starting condition is set to ALWAYS or ERROR.

[Related instructions]

Def Gps (Define get position), Def Map (Define mapping)

[Related system status variables]

M_Gps, P_Gps1 to P_Gps8, M_Map1 to M_Map8

Hlt (Halt)

[Function]
Interrupts the execution of the program which executed this Hlt command.
In use of the multitasking function, the executing status of other programs is not affected.

[Format]

```
Hlt
```

[Reference Program]
(1) 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.
```

(2) Stop the robot without condition during program execution.

```
15 Hlt                             ' Stop the program without condition.
```

[Explanation]
(1) Interrupts the execution of the program which executed this Hlt command, and will be waiting state.
(2) In use of the multitasking function, only the task slot which executed this command interrupts execution.
(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)

⚠CAUTION When using the tracking function
When this Hlt command is executed during tracking movement, tracking movement will be stopped (an equivalent for the Trk Off command) and execution of the program will be interrupted. In use of the multi-mechanism, tracking movement is stopped to the robot of the mechanism number got by the GetM command. When you continue tracking movement by the restart (continuation), please create the program to execute the Trk On command.

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 forcer> *1	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> *1	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> *1	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.

*1) It is valid only in our company electric hand.

[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).

```
1 HClose 1
2 *LBL: If M_HndCq(1)<>1 Then GoTo *LBL
3 Mov P1
```

(5) There are related parameters. Refer to Page 528, "5.10 Automatic return setting after jog feed at pause" and, Page 531, "5.13 About default hand status" of this manual.

[Related system variables]
M_In/M_Inb/M_In8/M_Inw/M_In16 (900s number), M_Out/M_Outb/M_Out8/M_Outw/M_Out16 (900s number), M_HndCq

[Related instructions]
Loadset (Load Set), Oadl (Optimal Acceleration)

[Related parameter]
HANDTYPE, HANDINIT
Refer to Page 528, "5.10 Automatic return setting after jog feed at pause"and, Page 531, "5.13 About default hand status".

If...Then...ElseIf...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
     :
[ElseIf[]<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) Describes If...Then...Else... in a line.

```
10 If M1>10 Then *L100                    'When M1 is larger than 10, jump to the step *L100.
11 If M1>10 Then GoTo *L20 Else GoTo *L30 ' If M1 is larger than 10, it jumps to step *L20;
                                           if smaller than 10, it jumps to label *L30.
                                           The "GoTo" after" Then" or "Else" can be
                                           omitted.
```

(2) Using a block structure of If...Then...Else...EndIf.

```
10 If M1>10 Then
11   M1=10
12   Mov P1
13 Else
14   M1=-10
15   Mov P2
16 EndIf
```

(3) Describes If...Then...Else...EndIf block inside Then or Else.

```
30 If M1>10 Then
31   If M2 > 20 Then
32     M1 = 10
33     M2 = 10
34   Else
35     M1 = 0
36     M2 = 0
37   EndIf
38 Else
39    M1 = -10
```

```
400  M2 = -10
410 EndIf
```

(4) Describes the Break structure in the Then or Else, it can escape to the next step of EndIf.

```
30 If M1>10 Then
31   If M2 > 20 Then Break                ' Once the designated conditions are
                                            established, the process escapes to step 39.
32   M1 = 10
33   M2 = 10
34 Else
35   M1 = -10
36   If M2 > 20 Then Break                ' Once the designated conditions are
                                            established, the process escapes to step 39.
37   M2 = -10
38 EndIf
39 If M_BrkCq=1 Then Hlt
40 Mov P1
```

[Explanation]

(1) The If .. Then .. Else .. statements should be contained in one step.
(2) It is allowed to split an If .. Then .. ElseIf .. Else .. EndIf block over several steps.
(3) Else can be omitted.
(4) Make sure to include the EndIf statement in the If .. Then .. ElseIf .. Else .. EndIf block.
(5) It is possible to describe the ElseIf statement in multiple lines.
(6) If the GoTo instruction is used to jump out from inside an If .. Then .. ElseIf .. Else .. EndIf block, an error will occur when the memory for control structure (stack memory) becomes insufficient.
(7) For If .. Then .. ElseIf .. Else .. EndIf, it is possible to describe If .. Then .. ElseIf .. Else .. EndIf inside Then or Else. (UP to eight levels of nesting is allowed.)
(8) GoTo following Then or Else may be omitted.
Example) If M1 > 10 Then *L200 Else *L300
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 *L200 (The program at left can be rewritten as shown below.)
→ If M1 > 10 Then *L200
→ If M1 > 10 GoTo *L200
(9) 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.
(10) Logic numbers can be described in the condition expression. If the logic number is not 0, then it is true (the condition is met), and if 0, it is false. Therefore, a description such as one below can be used.
Example) If M_IN(900) Then M_Out(30)=1
When the input signal 900 is on, it is processed as true (M_In(900) = 1). Consequently, the description following Then is executed.
When the input signal 900 is off, it is processed as false (M_In(900) = 0). Consequently, the description following Then is not executed.
(11) If the Else statement is described before ElseIf statement, an error will occur.

Include

[Function]
Reads a specified program.

[Format]

```
#Include[]"<Program Name>
```

[Terminology]

<Program Name>	Designate a program name to be read.

[Reference Program]

```
1 #Include "LIB1"          ' Reads the program.
2 '
3 Function Main
4    Mvs PData1            ' Uses the common variables of the read program.
5    Func1()               ' Calls the functions of the read program.
6 FEnd
```

```
1 ' LIB1.MB6
2 Def Pos PData1, PData2
3 Function V Func1
4    Mvs P1
5    Mvs P2
6    Mvs P3
7 FEnd
```

[Explanation]

(1) Up to 64 #Include statements (when a further #Include statement is declared in a program read by an #Include statement) can be declared including multiple includes.
(2) The specified program is read in the step in which the #Include statement is described.
(3) When an #Include statement is declared in a Function procedure, an error will occur at execution.
(4) When the same file is specified for several times, the file is not read after the second reading.
(The program which has been read in the program specified with the #Include statement is included.)
(5) When the read source program (where the #Include statement is described) is specified, the specification is ignored.
(6) If a variable or function with the same name has existed in the read destination program (specified with the #Include statement), the variable or function in the read source program (where the #Include statement is described) takes precedence.
(7) If a variable or function with the same name exists among the read destination programs, the variable or function in the program read later (the #Include statement is written later) takes precedence.
(8) Functions with the same name among programs are regarded as overloaded functions if the signatures of the functions are different.

Input (Input)

[Function]

Inputs data into a file (including communication lines). Only AscII character data can be received.

[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 "CPRC232" 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 "CPRC232" parameter is 0)

```
PRN MELFA,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_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 < [+] <Numeric Value> / -<Numeric Value> / 0 > [, < Axis No>]

[Terminology]

<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.
<+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

RH-FR/RH-CR series: J4 axis (priority axis)
RV-FR series: J6 axis (priority axis)

(2)Additional axes of all models
(3)All axes of user defined mechanisms

[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 ^ (Number of encoder bits + 15) - 1. For example, with a 13-bit encoder (8192 pulses), this will be Max. = 2 ^ (13+15)-1 = 0x0fffffff, and for a 14-bit encoder (16384 pulses), this will be Max. 2 ^ (14+15)-1 = 0x1fffffff.
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 command 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 484, "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.
And, when using the interference avoidance function, specify the hand number and the work number.
(Specify the model which is the target of the interference check)

[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 541, "5.16 Hand and Workpiece Conditions (optimum acceleration/deceleration settings)" for details on the optimum acceleration/deceleration.
Refer to Page 567, "5.24 Interference avoidance function" for details of the interference avoidance function.

[Related instructions]

Oadl (Optimal Acceleration), HOpen / HClose (Hand Open/Hand Close)

[Related parameter]

HNDDAT1 to 8, WRKDAT1 to 8, HNDHOLD1 to 8
Refer to Page 541, "5.16 Hand and Workpiece Conditions (optimum acceleration/deceleration settings)".
Refer to Page 484, "Table 5-1: List Movement parameter" for the ACCMODE.
Refer to Page 567, "5.24 Interference avoidance function" for details of the parameter about interference avoidance function.

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 WthIF 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 528, "5.10 Automatic return setting after jog feed at pause")

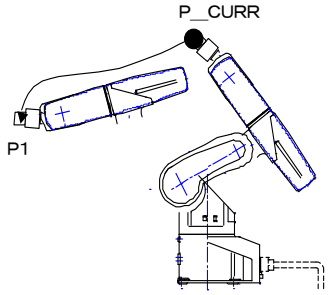


Fig.4-25:Example of joint interpolation motion path

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).

[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 205, " 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 205, " Def Arch (Define arch)".

(4) This command 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 528, "5.10 Automatic return setting after jog feed at pause").

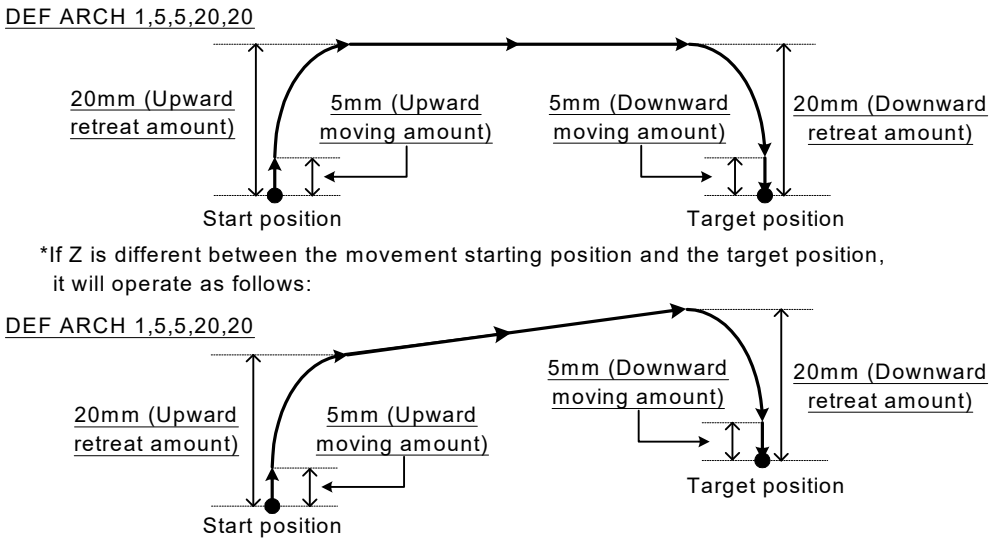


Fig.4-26:Example of arch interpolation motion path (seen from the side)

⚠CAUTION The robot's locus of movement may change with specified speed.
Especially as for the corner section, short cut distance may change. Therefore, when beginning automatic operation, moves at low speed at first, and you should gather speed slowly with being careful of interference with peripheral equipment.

[Related instructions]
Def Arch (Define arch), Accel (Accelerate), Ovrd (Override)

Mva2 (Move Arch 2)

[Function]

This instruction moves the robot from the current position to the target position with an arch movement (arch interpolation).

Note) The available robot type is limited. Refer to "[Available robot type]".

[Format]

```
Mva2 <Target Position> [ , <Downward Increment> [ , <Neighborhood Distance 1>
                    [, <Neighborhood Distance 2> [ , <Positioning Distance> [ , <Settling Time> ]]]]]
```

[Terminology]

<Target Position>	Final position of interpolation movement. This position may be specified using a position type variable and constant, or a joint variable.
<Downward Increment>	Specify the downward increment from the operation limit of the Z axis. Unit: [mm] Setting range: Real value less than or equal to 0 When 0 is specified or this argument is omitted, the robot moves upward to the operation limit of the Z axis.
<Neighborhood Distance 1>	The neighborhood distance at the 1st corner can be specified using a constant or a variable. Unit: [mm] Setting range: Real value more than or equal to -1/0 When 0 is specified, the robot switches to the next operation after the completion of the operation to the upward position. When -1 is specified, the operation is the same as Cnt1. When this argument is omitted, the value in the 1st element of the parameter "MVACNT1" is used.
<Neighborhood Distance 2>	The neighborhood distance at the 2nd corner can be specified using a constant or a variable. Unit: [mm] Setting range: Real value more than or equal to -1/0 When 0 is specified, the robot switches to the next operation after the completion of the operation to a position above the target position. When -1 is specified, the operation is the same as Cnt1. When this argument is omitted, the value in the 2nd element of the parameter "MVACNT1" is used.
<Positioning Distance>	Specify the positioning complete conditions to the target position with a linear distance. Unit: [mm] Setting range: Real value more than or equal to 0 (The minimum value that can be specified is 0.001.) When 0 is specified, the positioning will be disabled. When this argument is omitted, the value in the 3rd element of the parameter "MVACNT1" is used.
<Settling Time>	Specify the settling time after the completion of arrival to the target position. Unit: [Seconds] Setting range: Real value more than or equal to 0 When this argument is omitted, the value in the 4th element of the parameter "MVACNT1" is used.

[Reference Program]

```
1 Mov P0                    ' Moves to the start position of the arch motion movement.
2 Mva2 P1                   ' Moves upward to the operation limit of the Z axis, and moves to P1 with
                              the arch motion.
3 Mva2 P2,-20               ' Moves upward to the position which is -20mm from the operation limit
                              of the Z axis, and moves to P2 with the arch motion.
5 Mva2 P3,0,20,20,0.1,0     ' Moves upward to the operation limit of the Z axis, and moves to P3 with
                              the arch motion.
                            ' The positioning distance of the target position is 0.1mm.
```

[Explanation]

(1) The robot moves upward in 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 arch motion movement is performed with one instruction.

(2) <Downward Increment>, <Neighborhood Distance 1>, <Neighborhood Distance 2>, <Positioning Distance>, and <Settling Time> can be omitted.

(3) When <Neighborhood Distance 1>, <Neighborhood Distance 2>, <Positioning Distance>, and <Settling Time> are omitted, the value in the parameter "MVACNT1" is used. For details, refer to the explanation of the parameter "MVACNT1".

(4) The interpolation method of the Mva2 instruction is the joint interpolation, short cut, or equivalent rotation.

(5) The Cnt command is temporarily ignored (disabled, the status is maintained) when the Mva2 command is executed.

(6) The Fine command is temporarily ignored (disabled, the status is maintained) when the Mva2 command is executed.

(7) If <Positioning Distance> is specified while the compliance function is enabled, the robot may be unable to reach the positioning completion condition of the target position depending on the operation, and will wait indefinitely for the completion of the operation instruction. As a result, the program execution comes to a halt. Do not specify <Positioning Distance> while the compliance function is enabled.

(8) The available robot type is limited. Using this instruction for other types of robot generates an error H3391. For the supported robot, refer to "[Available robot type]".

(9) This instruction cannot be used in a constantly executed program.

(10) If paused during execution of an Mva2 instruction and restarted after jog feed, the robot returns to the interrupted position and restarts the Mva2 instruction. The interpolation method (JOINT interpolation/ XYZ interpolation) which returns to the interrupted position can be changed by the RETPATH parameter.

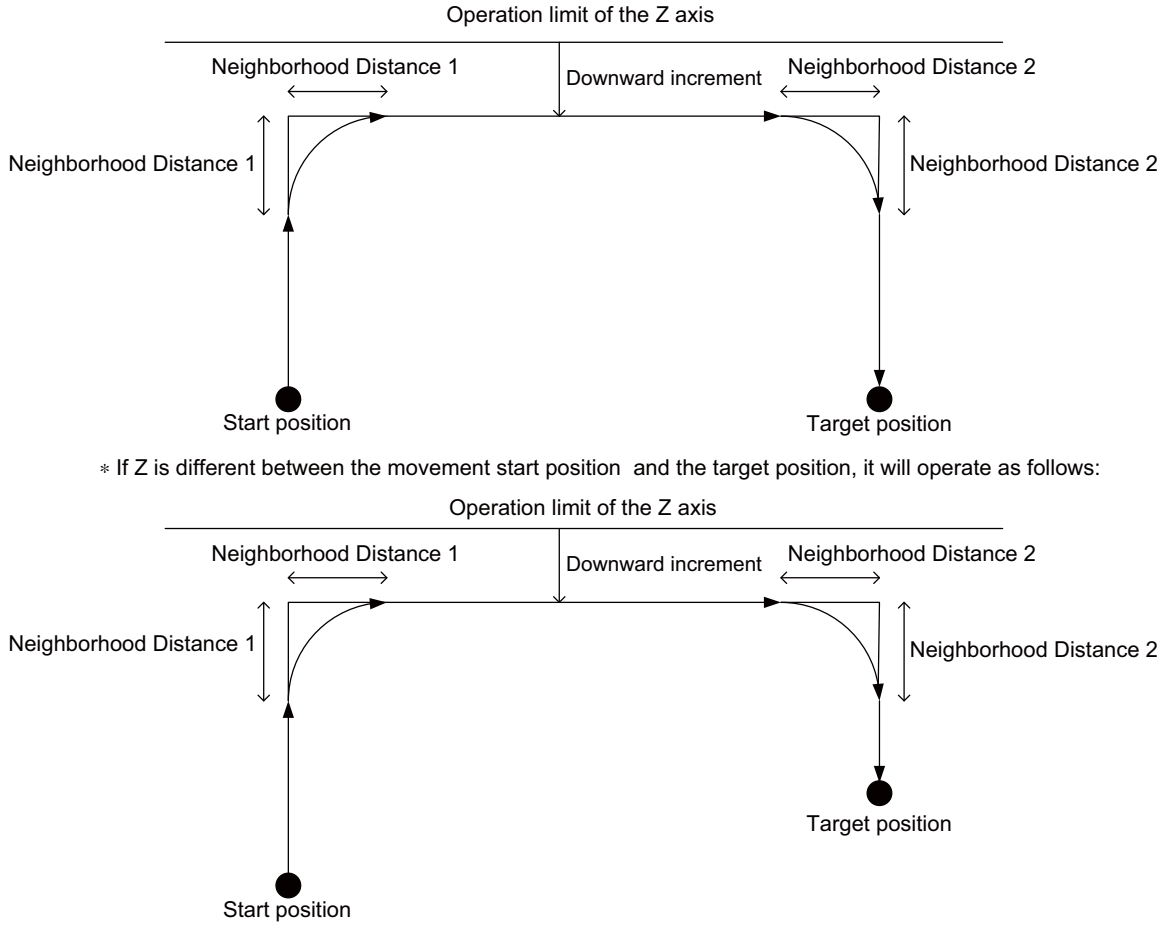
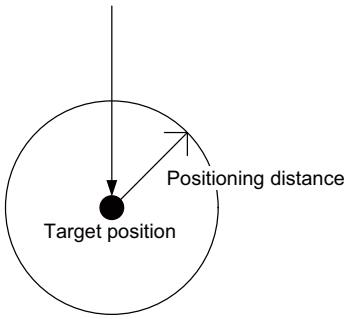


Fig.4-27:Example of Mva2 instruction motion path (seen from the side)



"Positioning is completed when the current position moves into the sphere with the radius specified in <Positioning Distance>."

Fig.4-28:Positioning distance of Mva2 instruction

[Available robot type]
RH-FR series (Without RH-3FRHR series)

[Related instructions]
MVACNT1

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 using a position type variable or constant, or a joint variable.
<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 WthIf 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 WthIf 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 528, "5.10 Automatic return setting after jog feed at pause")

(5) This instruction cannot be used in a constantly executed program.

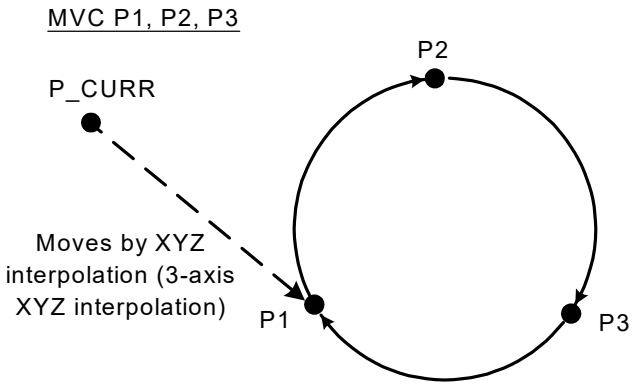


Fig.4-29:Example of circle interpolation motion path

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 using a position type variable or constant, or a joint variable.
<Transit Point>	Transit point for the arc. Describe a position using a position type variable or constant, or a joint variable.
<End Point>	End point for the arc. Describe a position using a position type variable or 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 WthIf 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 WthIf 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 528, "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. The singular point passage function is supporting only certain models. Refer to Page 552, "5.20 About singular point passage function" in details.

(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.

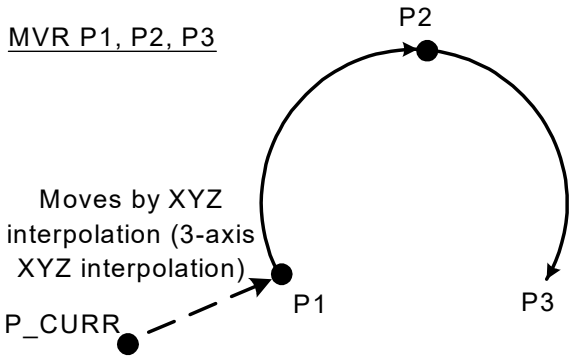


Fig.4-30:Example of circular interpolation motion path 1

Precautions for registration of position data after execution of linear (circular arc) interpolation for vertical 5-axis robots

In the linear (circular arc) interpolation, the posture data at the operation start position and the posture data at the movement target position are different by +180 degrees or -180 degrees or more, the robot performs shortcut operation. For vertical 5-axis robots (RV-4FRJL), the registered posture data is shown at the movement target position, but the actual posture is deviated by +360 degrees or -360 degrees.
For example, when linear interpolation is performed from the 0-degree position to the +200-degree position of the A axis, the posture data at the current position is shown as +200 degrees, but the robot operates to the -160-degree position.
When the position data is registered under such a condition, the posture data deviated by +360 or -360 degrees from the actual position is registered. Be careful when the position data is compensated after linear (circular arc) compensation movement for debugging operation (step feed, MS position movement) of the program.
This phenomenon can be avoided by specifying the roundabout operation in the argument type of the linear (circular arc) interpolation, or by the servo OFF operation before the position data is registered.

Mvr2 (Move R 2)

[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. Describe a position using a position type variable or constant, or a joint variable.
<End Point>	End point for the arc. Describe a position using a position type variable or constant, or a joint variable.
<Reference point>	Reference point for a circular arc. Describe a position using a position type variable or constant, or a joint variable.
<Constants 1>	Short cut/detour = 0/1, 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 WthIf 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 WthIf 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 528, "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. The singular point passage function is supporting only certain models. Refer to Page 552, "5.20 About singular point passage function" in details.

(10) 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.

(11) This instruction cannot be used in a constantly executed program.

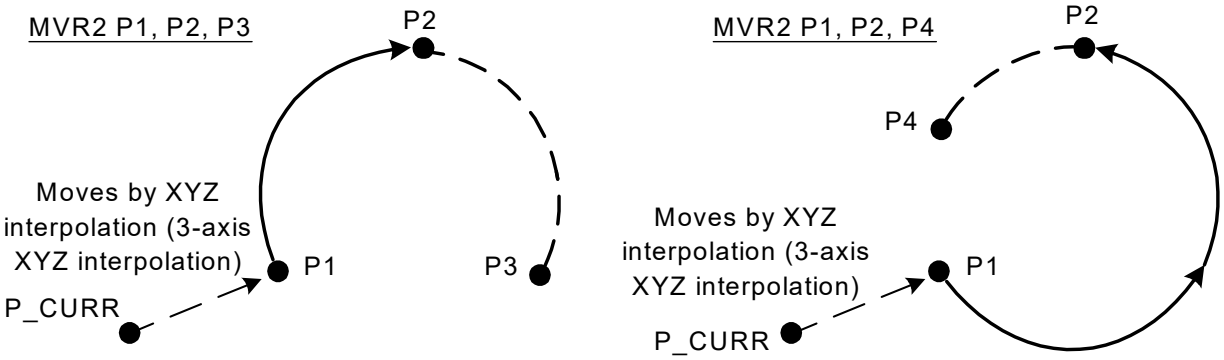


Fig.4-31:Example of circular interpolation motion path 2

Precautions for registration of position data after execution of linear (circular arc) interpolation for vertical 5-axis robots

In the linear (circular arc) interpolation, the posture data at the operation start position and the posture data at the movement target position are different by +180 degrees or -180 degrees or more, the robot performs shortcut operation. For vertical 5-axis robots (RV-4FRJL), the registered posture data is shown at the movement target position, but the actual posture is deviated by +360 degrees or -360 degrees.

For example, when linear interpolation is performed from the 0-degree position to the +200-degree position of the A axis, the posture data at the current position is shown as +200 degrees, but the robot operates to the -160-degree position.

When the position data is registered under such a condition, the posture data deviated by +360 or -360 degrees from the actual position is registered. Be careful when the position data is compensated after linear (circular arc) compensation movement for debugging operation (step feed, MS position movement) of the program.

This phenomenon can be avoided by specifying the roundabout operation in the argument type of the linear (circular arc) interpolation, or by the servo OFF operation before the position data is registered.

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>, <Constants 2>][] [<Appended Condition>]
```

[Terminology]

<Start Point>	Start point for the arc. Describe a position using a position type variable or constant, or a joint variable.
<End Point>	End point for the arc. Describe a position using a position type variable or constant, or a joint variable.
<Center Point>	Center point for the arc. Describe a position using a position type variable or constant, or a joint variable.
<Constants 1>	Short cut/detour = 0/1, 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 WthIf 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 WthIf 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 528, "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. The singular point passage function is supporting only certain models. Refer to Page 552, "5.20 About singular point passage function" in details.
(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) The central angle from the start point to the end point always satisfies 0 < central angle < 180 degrees.
(10) 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.
(11) 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.
(12) 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.
(13) This instruction cannot be used in a constantly executed program.

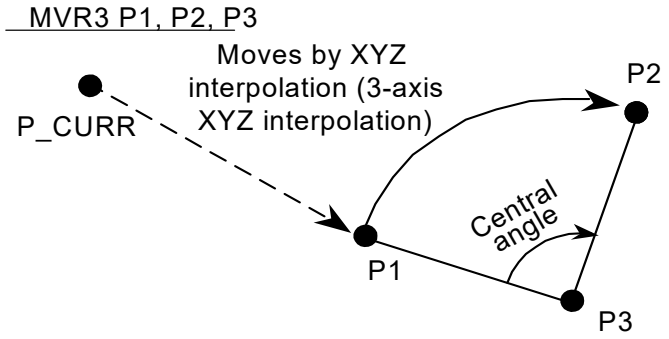


Fig.4-32:Example of circular interpolation motion path 3

Precautions for registration of position data after execution of linear (circular arc) interpolation for vertical 5-axis robots

In the linear (circular arc) interpolation, the posture data at the operation start position and the posture data at the movement target position are different by +180 degrees or -180 degrees or more, the robot performs shortcut operation. For vertical 5-axis robots (RV-4FRJL), the registered posture data is shown at the movement target position, but the actual posture is deviated by +360 degrees or -360 degrees.
For example, when linear interpolation is performed from the 0-degree position to the +200-degree position of the A axis, the posture data at the current position is shown as +200 degrees, but the robot operates to the -160-degree position.
When the position data is registered under such a condition, the posture data deviated by +360 or -360 degrees from the actual position is registered. Be careful when the position data is compensated after linear (circular arc) compensation movement for debugging operation (step feed, MS position movement) of the program.
This phenomenon can be avoided by specifying the roundabout operation in the argument type of the linear (circular arc) interpolation, or by the servo OFF operation before the position data is registered.

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. Describe a position using a position type variable or 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 = 0/1, 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 WthIf statements can be used.
<Separation Distance>	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 WthIf 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 517, "5.6 Standard Tool Coordinates" for detail. The "Fig.4-33:Example of movement at linear interpolation" is an example of movement.

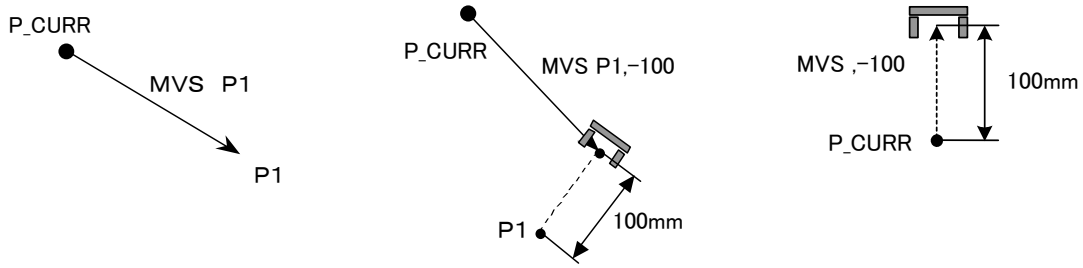


Fig.4-33:Example of movement at linear interpolation

(4) Short cut operation refers to posture interpolation between the start point and end point in the direction with less motion.

(5) Detour operation refers to posture interpolation between the start point and end point in the direction with more motion.

(6) When detour/short cut is designated, if the target position is outside the motion range, the error of outside the motion range will occur.

(7) 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 528, "5.10 Automatic return setting after jog feed at pause".

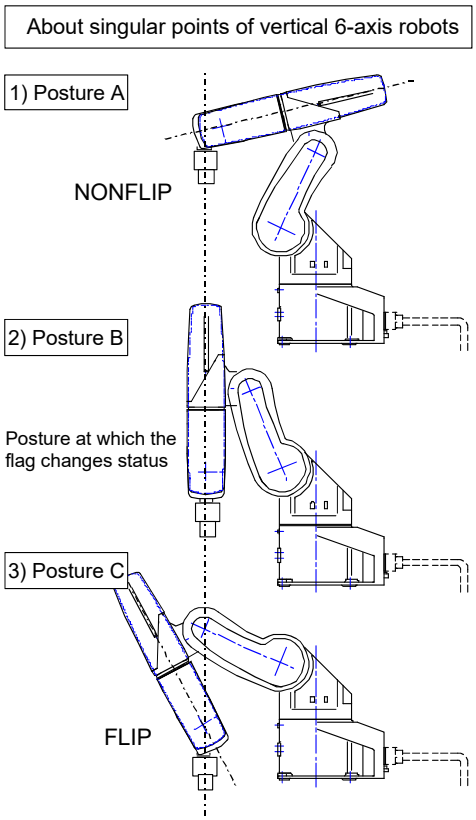
(8) This instruction cannot be used in a constantly executed program.

(9) 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.

(10) 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.

(11) 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. The singular point passage function is supporting only certain models. Refer to Page 552, "5.20 About singular point passage function" in details.

(12) Description of singular points.



<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.4-34:Singular point 1

Precautions for registration of position data after execution of linear (circular arc) interpolation for vertical 5-axis robots

In the linear (circular arc) interpolation, the posture data at the operation start position and the posture data at the movement target position are different by +180 degrees or -180 degrees or more, the robot performs shortcut operation. For vertical 5-axis robots (RV-FR series), the registered posture data is shown at the movement target position, but the actual posture is deviated by +360 degrees or -360 degrees.
For example, when linear interpolation is performed from the 0-degree position to the +200-degree position of the A axis, the posture data at the current position is shown as +200 degrees, but the robot operates to the -160-degree position.
When the position data is registered under such a condition, the posture data deviated by +360 or -360 degrees from the actual position is registered. Be careful when the position data is compensated after linear (circular arc) compensation movement for debugging operation (step feed, MS position movement) of the program.
This phenomenon can be avoided by specifying the roundabout operation in the argument type of the linear (circular arc) interpolation, or by the servo OFF operation before the position data is registered.

MvSpl (Move Spline)

[Function]

Spline interpolation is executed based on the information registered in the designated spline file.
(Refer to Page 680, "7.2 Spline interpolation" for details on spline interpolation.)

[Format]

```
MvSpl []<Spline No.>, <Speed>, <Acceleration/deceleration distance>
        [, <Frame transformation> [, <Posture interpolation type> [, <Filter length>]]]
```

[Terminology]

<Spline No.>	The number of the spline file holding information on the path to be moved is designated with a constant or numerical variable. Setting range: 1 to 99
<Speed>	The spline interpolation movement speed is designated with a constant or numerical variable. Setting range: Value larger than 0.0 (mm/s). Refer to [Explanation] "(5)" for details.
<Acceleration/deceleration distance>	The spline interpolation acceleration/deceleration distance is designated with a constant or numerical variable. The acceleration distance is the distance required to accelerate from the start or movement to the designated speed. The deceleration distance is the distance required to decelerate from the designated speed to the end position. With spline interpolation, the acceleration distance and deceleration distance are a common setting. Setting range: Value larger than 0.0 (mm)
<Frame transformation>	The details for executing frame transformation are designated with a constant or numerical variable. Setting range: 0: Frame transformation is not executed. 1: Frame transformation is executed using coordinate system set in spline file. 2: Frame transformation is executed using coordinate system set with SetCalFrm command. When omitted: Frame transformation is not executed.
<Posture interpolation type>	The posture interpolation type is designated with a constant or numerical variable. Setting range: 0 (equivalent rotation movement)/1 (3-axis orthogonal movement) When omitted: Equivalent rotation movement
<Filter length>	The filter length applied on the acceleration/deceleration movement is designated with a constant or numerical variable. Setting range: 0.0 to 1000.0 (ms) When omitted: 100.0 ms

[Reference Program]

```
1 Ovrd 70             ' Set program override to 70%
2 Mov P1              ' Move to position P1 with joint interpolation
3 MvSpl 2, 50, 10     ' Execute spline interpolation that passes through spline file 2 path point
4 Mvs P2              ' Move to position P2 with linear interpolation
5 MvSpl 2, 50, 10, 1  ' Pass through path point for which frame transformation has been executed on
                        spline file 2 path point
                      ' Execute spline interpolation
```

[Explanation]

(1) Spline interpolation is carried out based on the path point data, etc., registered in the spline file corresponding to <Spline No.>. A smooth curve (spline curve) is generated to pass through the robot position and posture that are registered as path points. The robot movement follows that curve.

(2) If the robot's current position at the start of MvSpl command execution and the spline interpolation start position are deviated, the robot will move to the start position with linear interpolation and then start spline interpolation.

(3) If a spline file corresponding to the <Spline No.> is not saved in the controller, the error L2610 (Can't open spline file) will occur.

(4) The spline interpolation command speed is designated with the following expression.

Speed command = (MvSpl command's <Speed>) × (Ovrd command) × (operation panel override)

The Spd command and JOvrd command settings are not used. Even if the status variable M_NSpd is set for <Speed>, the optimum speed control mode does not function.

(5) The setting range of <speed> changes according to a distance between path points that registered into the spline file. Refer to Page 685, "Table 7-5: Check related to path points" for details.

(6) The spline interpolation acceleration/deceleration is designated with <Acceleration/deceleration distance>. The Accel command setting is not used. Even if the optimum acceleration/deceleration control is enabled with the Oadl command, it will not be applied to spline interpolation.

(7) The acceleration movement generates speed to reach the <Speed> when the <Acceleration/deceleration distance> is moved from the spline interpolation start position. The deceleration movement generates the speed to move the <Acceleration/deceleration distance> from the <Speed> and stop at the end position.

If the Ovrd command and operation panel's override is set to a value smaller than 100%, the commanded speed will be reached at a shorter travel rate than <Acceleration/deceleration distance>. (The effect of filters, etc., can cause the acceleration/deceleration distance to be actually longer than the value set with <Acceleration/deceleration distance>.

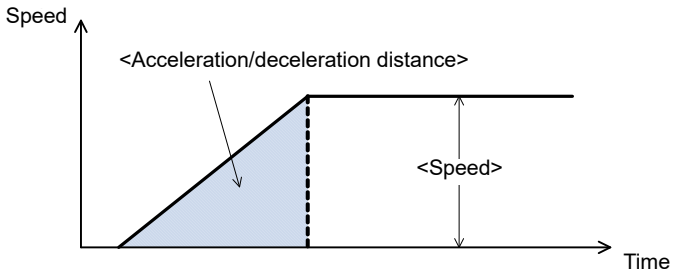


Fig.4-35: Acceleration with <Acceleration/deceleration distance>

(8) If "1 (execute using coordinate system set in spline file)" or "2 (execute using coordinate system set with SetCalFrm command) is designated in <Frame transformation>, the path point data is converted sequentially based on the designated method, and spline interpolation is executed to pass through those path points.

(9) If the coordinate system is not set in the spline file even though "1 (execute using coordinate system set in spline file)" is designated in <Frame transformation>, error L2042 (Frame transformation coordinates are not set) will occur.

(10) If the coordinate system cannot be calculated when "1 (execute using coordinate system set in spline file)" is designated in <Frame transformation>, error L2041 (Can't calculate frame transformation coordinates) will occur.

(11) If "2 (execute using coordinate system set with SetCalFrm command)" is designated for <Frame transformation>, frame conversion is executed using the coordinate system set with the SetCalFrm command executed last. If the SetCalFrm command has not been executed even once and the coordinate system is not set, error L2042 (Frame transformation coordinates are not set) will occur.

(12) Switch between "0 (equivalent rotation movement)" and "1 (3-axis orthogonal movement)" by designating <Posture interpolation type>. The equivalent rotation movement operates so that the posture change between the path points has the minimum angle.

The 3-axis orthogonal motion generates the posture data with the J4, J5 and J6 axis angles instead of the A, B and C axis coordinate values. Thus, even if the path point posture data (A, B, C axis coordinate values) are the same, the robot's posture will change during the movement.

(13) Transfer of the configuration flag (peculiar point transit) is not supported. If the configuration flag differs between path points, error L2611 (Path point configuration flag is different) will occur.

(14) The filter characteristics applied on the acceleration/deceleration movement can be changed with <Filter length>. Increase the filter length to make the movement during acceleration and deceleration smoother. Note that if the filter length is too high, the movement will slow down and it will take a long time for the spline interpolation to end.
(15) If a signal output is set for the path point data, the general purpose output signal will be output with the designated conditions when passing through the target path point.
(16) If the robot positioning complete condition is specified (Fine command, Fine J command, or Fine P command), positioning completion will be confirmed at the spline interpolation end position.
(17) Even if continuous movement is designated with the Cnt command, continuous movement will not take place at the spline interpolation start position and end position.
(18) Even if "Stop type 1" is designated for the Def Act command (interrupt condition and process definition), if an interrupt occurs during the spline interpolation, the robot will stop with the same deceleration as "Stop type 2".
(19) The additional axis will not move from the spline interpolation start position. Even if an additional axis position is set for one of the interim path points, it will not be used during spline interpolation.
(20) Step feed can be executed for the MvSpl command, but step return cannot be executed. Error L2612 (Cannot execute step return) will occur.
(21) The mechanism's control rights (GetM command) are required to execute the MvSpl command.
(22) The MvSPl command cannot be executed in a slot where the start conditions are ALWAYS or ERROR. Error L3287 (Cannot execute (ERROR ALWAYS)) will occur.
(23) If a value outside the setting range is set for the MvSpl command argument, error L3110 (Argument value is out of range (MvSpl)) will occur.

⚠CAUTION If an <Acceleration/deceleration distance> too short for the <Speed> is designated, sudden acceleration/deceleration will take place and could cause robot vibration or a servo error. Always set an appropriate acceleration/deceleration distance.

[Related instructions]
SetCalFrm (Set Calibration Frame)

[Related system variables]
M_SplPno, M_SplVar

Mv Tune (Move Tune)

[Function]

Select the robot operating characteristics from one of the following four 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.

[Format]

```
MvTune[]<Operating Characteristics Mode>
```

[Terminology]

<Operating Characteristics Mode > The robot operating characteristics mode (1 to 4) is specified with either a constant or numeric value variable.
1: Standard mode (default)
2: High-speed positioning mode
3: Trajectory priority mode
4: Vibration suppression mode

Table 4-17:Movement mode of MvTune

Operating 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 it takes to reach the target position. Use this mode where it is desired to shorten positioning time and improve work efficiency. (Applications: tracking operation, palletizing operation, etc.)
3	Trajectory priority mode	This mode improves the trajectory accuracy in an interpolating operation. Use this mode when performing any operation in which trajectory accuracy is an important consideration. (Applications: sealing operation, welding operation, deburring operation, etc.)
4	Vibration suppression mode Note1)	This mode is effective in suppressing vibration (resonance) of the robot arm. Use this mode where vibration is encountered during the transfer of work. (Applications: wafer transfer, precision component transfer, etc.)

Note1) The vibration suppression mode (MvTune 4) is usable with software version N7 (CRnQ-700 series)/P7 (CRnD-700 series) or later.

[Reference Program]

```
LoadSet 1,1        'Sets to hand 1/workpiece 1.
MvTune 2           'Changes the operating characteristics mode to high-speed positioning
Mov P1             'Operates in the high-speed positioning mode
Mvs P1             'Operates in the high-speed positioning mode
MvTune 3           'Changes operating mode to the trajectory priority variation
Mvs P3             'Operates in the trajectory priority mode
```

[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 541, "5.16 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 differences between the standard mode and the other operating characteristics modes are as follows.

Table 4-18:By-operating mode Performance Comparison

Operating mode	Items of comparison			
	Time required to reach target	Trajectory accuracy (*1)	Vibration suppression (*2)	Load factor (*3)
Standard mode	○	○	○	○
High-speed position mode	◎	○ +	○	△
Trajectory priority mode	△	◎	○	○ +
Vibration suppression mode	○ -	○ -	◎	○

(Note) Symbols in the table indicate relative performance rating.

◎ Improved, ○ + Somewhat improved, ○ Same, ○ -Somewhat degraded, △ Degraded

*1: For comparing the robot's ability to respond to operating command value

*2: For comparing the capability of suppressing external perturbations which induce vibration

*3: For comparing the amount of heat generated by the motor

(5) 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.)

(6) High-speed positioning mode may allow vibration during acceleration or deceleration to become higher as compared with the standard mode. If this is inconvenient, select the standard mode.

(7) The trajectory priority mode is adjusted so as to achieve maximum effect at operating speeds in medium- to low-speed range. Therefore, when a motion involved is such that a small circle is drawn, vibration may increase as compared with the standard mode. In this case, use the Spd command to slow operating speed and thus decrease vibration.

(8) Use of the vibration-suppressing mode may lead to an increased overshoot in the "time required to reach target position" depending on the operating condition that is used. In such an operation that starts pending the completion of the positioning task (e.g. operation which uses Fine command), tact time may become delayed. (The greater the mass of work, the greater the overshoot.)

(9) This command does not function for the jog operation.

[Related instructions]

Loadset (Load Set), Oadl (Optimal Acceleration), Prec (Precision)

[Related parameter]

ACCMODE, HNDDAT 0 to 8, WRKDAT 0 to 8

Mxt (Move External)

[Function]

The real-time external control function by ethernet I/F

The absolute position data is retrieved from an external personal computer connected via Ethernet network at each controller control time, and the robot is directly moved. The controller control time is approx. 7.1 ms with the CR800-Q and approx. 3.5 ms with the CR800-D/R (approx. 7.1 ms when user mechanism is set).

[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.
<Replay 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]

(1) 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)

(2) One position command can be retrieved and operated at the operation control time. The operation control time is approx. 7.1 ms with the CR800-Q and approx. 3.5 ms with the CR800-D/R (approx. 7.1 ms when user mechanism is set).

(3) 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.

(4) The timeout is designated with the parameter MXTTOUT.

(5) One randomly designated (head bit, bit width) input/output signal can be transmitted and received simultaneously with the position data.

(6) A personal computer with sufficient processing speed must be used to command movement in the movement control time.

(7) Refer to Page 534, "5.15 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)

NVClose (network vision sensor line close)

[Function]
Cuts the line with the specified vision sensor.

[Format]

NVClose□[[#]<Vision sensor number>[,[[#]<Vision sensor number>・・・]

[Terminology]

<Vision sensor number> (Can be omitted)

Specifies a constant from 1 to 8 (the vision sensor number). Indicates the number for the vision sensor connection to the COM specified with the <COM number>.
When this parameter is omitted, all the lines (vision sensor lines) opened with an NVOpen command are closed. Also, up to 8 <vision sensor numbers> can be specified. They are delimited with commas.
Setting range: 1 - 8

[Reference Program]

```
1 If M_NVOpen(1)<>1 Then      ' When logon has not been completed for vision sensor number 1
2   NVOpen "COM2:" ASs#1      ' Connects with the vision sensor connected to COM2 and sets its
                                number as number 1.
3 End If
4 Wait M_NVOpen(1)=1          ' Connects with vision sensor number 1 and waits for logon to be
                                completed.
5 ・・・・・
  :
100 NVClose #1                ' Cuts the line with the vision sensor connected to COM2.
```

[Explanation]

(1) Cuts the line with the vision sensor connected with the NVOpen command.
(2) If the <vision sensor number> is omitted, cuts the line with all the vision sensors.
(3) If a line is already cut, execution shifts to the next step.
(4) Because up to seven vision sensors can be connected at the same time, <Vision sensor numbers> are used in order to identify which vision sensor to close the line for.
(5) If the program is cancelled while this command is being executed, execution continues until processing of this command is complete.
(6) When this command is used with multi-tasking, in the task using this command, it is necessary to close only the lines opened by executing an NVOpen command . At this time, use the <Vision sensor number> specified with the NVOpen command.
(7) A program start condition of "Always" is not supported.
(8) If an End command is used, all the lines opened with an NVOpen command or Open command are closed. However, lines are not closed with an End command in a program called out with a CAllp command.
Lines are also closed by a program reset, so when an End command or a program reset is executed, it is not necessary to close lines with this command.
(9) If an interrupt condition is established while this command is being executed, the interrupt processing is executed after this command is completed.

[Errors]

(1) If the value specified as the <vision sensor number> is anything other than "1" through "8", the "argument out of range" error occurs.
(2) If there are more than eight command arguments, an "incorrect argument count" error occurs.

NVLoad (network vision sensor load)

[Function]
Loads the specified vision program into the vision sensor.

[Format]

```
NVLoad□#<Vision sensor number>,"<Vision program (job) name>"
```

[Terminology]

<Vision sensor number> (Can not be omitted)
This specifies the number of the vision sensor to control.
Setting range:1 - 8

<Vision program (job) name> (Can not be omitted)
Specifies the name of the vision program to start.
The vision program extension (.job) can be omitted.
The only characters that can be used are "0" - "9", "A" - "Z", "a" - "z", "-", and "_".

[Reference Program]

```
100 If M_NVOpen(1)<>1 Then    ' If vision sensor number 1 log on is not complete
110   NVOpen "COM2:" As #1    ' Connects with the vision sensor connected to COM2.
120 End If
130 Wait M_NVOpen(1)=1        ' Connects with vision sensor number 1 and waits for logon to be
                                completed.
140 NVLoad #1,"TEST"          ' Loads the "Test".
150 NVPst #1, "","E76","J81","L84",0,10
                              ' Receives the recognition count recognized with the "Test" program
                                from the E76 cell and the recognition results from cells J81 through
                                L84, and stores them in P_NvS1().
160 ・・・・
 :
300 NVClose #1                ' Cuts the line with the vision sensor connected to COM2.
```

[Explanation]

(1) Loads the specified vision program into the specified vision sensor.
(2) This command moves to the next step at the point in time when the vision program is loaded into the vision sensor.
(3) If the program is cancelled while this command is being executed, it stops immediately.
(4) If the specified <vision program name> is already loaded, the command ends with no processing.
(5) When this command is used with multi-tasking, it is necessary to execute the NVOpen command in the task using this command. Also, use the <vision sensor number> specified with the NVOpen command.
(6) A program start condition of "Always" is not supported.
(7) If an interrupt condition is established while this command is being executed, the interrupt processing is executed immediately.

[Errors]

(1) If data type for an argument is incorrect, a "syntax error in input command statement" error is generated.
(2) If there is an abnormal number of command arguments (too many or too few), an "incorrect argument count" error occurs.
(3) If the <vision sensor number> is anything other than "1" through "8", an "argument out of range" error occurs.
(4) If the NVOpen command is not opened with the number specified as the <vision sensor number>, an "abnormal vision sensor number specification" error occurs.
(5) If the <vision program name> exceeds 15 characters, an "abnormal vision program name" error occurs.
(6) If a <vision program name> uses a character other than "0" - "9", "A" - "Z", "-", or "_" (including lowercase letters), an "abnormal vision program name" error occurs.
(7) If the program specified in the <vision program name> is not in the vision sensor, a "vision program does not exist" error occurs.
(8) If the vision sensor is "offline", the "Put online" error occurs, so put the vision sensor "Online".
(9) If the communications line is cut while this command is being executed, an "abnormal communications" error occurs and the robot controller side line is closed.

NVOpen (network vision sensor line open)

[Function]
Connects with the specified vision sensor and logs on to that vision sensor.

[Format]

NVOpen□"<COM number>"□As□#<Vision sensor number>

[Terminology]

<Com number> (Can not be omitted):
Specify the communications line number in the same way as for the Open command. "COM1:" can not be specified by it is monopolized by the operation panel front RS-232C.
Setting range: "COM2:" - "COM8:"

<Vision sensor number> (Can not be omitted)
Specifies a constant from 1 to 8 (the vision sensor number). Indicates the number for the vision sensor connection to the COM specified with the <COM number>.
Be careful. This number is shared with the <file number> of the Open command.
Setting range: 1 - 8

[Reference Program]

```
1 If M_NVOpen(1)<>1 Then      ' If vision sensor number 1 log on is not complete
2   NVOpen "COM2:" As#1       ' Connects with the vision sensor connected to COM2 and sets its
                                number as number 1.
3 End If
4 Wait M_NVOpen(1)=1          ' Connects with vision sensor number 1 and waits for logon to be
                                completed.
```

[Explanation]

(1) Connects with the vision sensor connected to the line specified with the <COM number> and logs on to that vision sensor.

(2) It is possible to connect to a maximum of 7 vision sensors at the same time. <Vision sensor numbers> are used in order to identify which vision sensor is being communicated with.

(3) When used together with the Open command, the Open command <COM number> and <File number> and the <COM number> and <Vision sensor number> of this command are shared, so use numbers other than those specified with the Open command <COM number> and <File number>.

Example: Normal example	Error example
1 Open “COM1:” As #1	1 Open “COM2:” As #1
2 NVOpen “COM2:” As #2	2 NVOpen “COM2:” As #2 => <COM number> used
3 NVOpen “COM3:” As #3	3 NVOpen “COM3:” Ass#1 => <Vision sensor number> Used

It is not possible to open more than one line in a configuration with one robot controller and one vision sensor. If the same IP address is set as when the [NETHSTIP] parameter was set, an "Ethernet parameter NETHSTIP setting" error occurs.

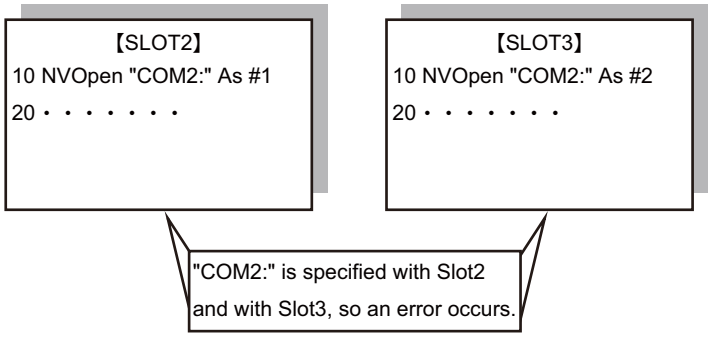
(4) Logging on to the vision sensor requires the "User name" and "Password". It is necessary to set a user name for which full access is set in the vision sensor and the password in the robot controller [NVUSER] and [NVPSWD] parameters.
The user name and password can each be any combination of up to 15 numbers (0-9) and letters (A-Z). (T/B only supports uppercase letters, so when using a new user, set the password set in the vision sensor with uppercase letters.)
The user name with full access rights when the network vision sensor is purchased is "admin". The password is "". Therefore, the default values for the [NVUSER] and [NVPSWD] parameters are [NVUSER] = "admin" and [NVPSWD] = "".
When the "admin" password is changed with MELFA-Vision or a new user is registered, change the [NVUSER] and [NVPSWD] parameters. When such a change is made, when the content of the [NVPSWD] parameter is displayed, "****" is displayed. If the vision sensor side password is changed, open the [NVPSWD] parameter and directly change the displayed "****" value. After the making the change, reset the robot controller power.
[Caution]
When multiple vision sensors are connected to one robot controller, set the same user name and password for all of them.

(5) The state of communications with the network vision sensor when this command is executed can be checked with M_NVOpen. For details, see the explanation of M_NVOpen.

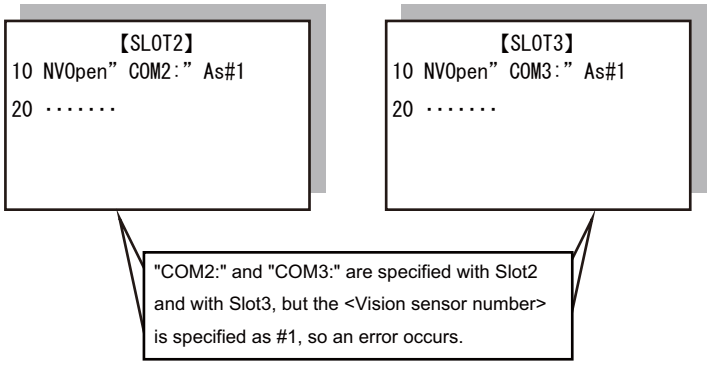
(6) If the program is cancelled while this command is being executed, it stops immediately. In order to log on to the vision sensor, it is necessary to reset the robot program, then start.

(7) When this command is used with multi-tasking, there are the following restrictions.
The <COM number> and <Vision sensor number> must not be duplicated in different tasks.

(a) If the same <COM number> is used in another task, the "attempt was made to open an already open communication file" error occurs.



(b) If the same vision sensor number is used in another task, the "attempt was made to open an already open communication file" error occurs.



(8) A program start condition of "Always" is not supported.
(9) Three robots can control the same vision sensor at the same time. If a fourth robot logs on, the line for the first robot is cut off, so be careful when constructing the system.
(10) The line is not closed with an End command in a program called out with a Callp command, but the line is closed with a main program End command. The line is also closed by a program reset.
(11) If an interrupt condition is established while this command is being executed, the interrupt processing is executed immediately even during processing of this command.

[Errors]
(1) If data type for an argument is incorrect, the "syntax error in input command" error is generated.
(2) If there is an abnormal number of command arguments (too many or too few), the "incorrect argument count" error occurs.
(3) If the character specified in <COM number> is anything other than "COM2:" through "COM8:", the "argument out of range" error occurs.
(4) If the value specified as the <vision sensor number> is anything other than "1" through "8", the "argument out of range" error occurs.
(5) If a <COM number> for which the line is already connected is specified (including the <File number> for which the line has been opened with an Open command), the "attempt was made to open an already open communication file" error occurs.
(6) If the vision sensor is not connected before the line is opened, the "vision sensor not connected" error occurs. (The same set manufacturer parameter [COMTIMER] as in the Ethernet specifications is used. Currently "1s")
(7) If the same <COM number> or the same <vision sensor number> is specified in another task, the "attempt was made to open an already open communication file" error occurs.
(8) If the user name or password specified in the [NVUSER] parameter (user name) and [NVPSWD] (password) is wrong, the "wrong password" error occurs.
(9) If the communications line is cut while this command is being executed, the "abnormal communications" error occurs and the robot controller side line is closed.
(10) If a program is used for which the starting condition is "Always", the "this command can not be used if the start condition is ERR or ALW" error occurs.

NVRun (network vision sensor run)

[Function]
Starts the specified vision program.

[Format]

NVRun□#<Vision sensor number >,< Vision program (job) name >

[Terminology]
<Vision sensor number> (Can not be omitted)
This specifies the number of the vision sensor to control.
Setting range:1 - 8
<Vision program (job) name> (Can not be omitted)
Specifies the name of the vision program to start.
The vision program extension (.job) can be omitted.
The only characters that can be used are "0" - "9", "A" - "Z", "a" - "z", "-", and "_".

[Reference Program]

```
100 If M_NvOpen(1)<>1 Then     ' If vision sensor number 1 log on is not complete
110   NVOpen "COM2:" As #1     ' Connects with the vision sensor connected to COM2.
120 End If
130 Wait M_NvOpen(1)=1         ' Connects with vision sensor number 1 and waits for logon to be
                                 completed.
140 NVLoad #1,"TEST"           ' Loads the "Test" program.
150 NVRun #1,"TEST"            ' Starts the "Test" program.
160 EBRead #1,,MNUM,PVS1,PVS2
                                'The data of "Job.Robot.FormatString" tag is read, and they are
                                preserved in the variable MNUM, PVS1, and PVS2.
170 ・・・
 :
300 NVClose #1                 ' Cuts the line with the vision sensor connected to COM2.
```

[Explanation]
(1) Loads the specified vision program into the specified vision sensor.
(2) The timing of completing this command processing is different depending on the value of parameter NVTRGTMG. When the parameter NVTRGTMG is a factory shipment setting, the next command is executed after completing the communication of image processing instruction (taking picture demand) between the vision sensor and robot controller.
(3) If the program is cancelled while this command is being executed, it stops immediately.
(4) If the specified <vision program name> is already loaded, only image capture and image processing are executed. (The vision program is not loaded.)?
(5) For receiving data from the vision sensor, use the EBRead command.
(6) When this command is used with multi-tasking, it is necessary to execute the NVOpen command in the task using this command. Also, use the <vision sensor number> specified with the NVOpen command.
(7) A program start condition of "Always" is not supported.
(8) Please set the trigger of EasyBuilder's image capture to "External trigger", "Manual triggeroperation" or "Network".
(It is possible to use "Camera" when the value of paramete NVTRGTMG is 0 or 2.)
(9) Up to three robots can control the same vision sensor at the same time, but this command can not be used by more than one robot at the same time. Use this command on any one of the robots.
(10) If an interrupt condition is established while this command is being executed, the interrupt processing is executed immediately.

[Errors]

(1) If data type for an argument is incorrect, a "syntax error in input command statement" error is generated.
(2) If there is an abnormal number of command arguments (too many or too few), an "incorrect argument count" error occurs.
(3) If the <vision sensor number> is anything other than "1" through "8", an "argument out of range" error occurs.
(4) If the NVOpen command is not opened with the number specified as the <vision sensor number>, an "abnormal vision sensor number specification" error occurs.
(5) If the <vision program name> exceeds 15 characters, an "abnormal vision program name" error occurs.
(6) If a <vision program name> uses a character other than "0" - "9", "A" - "Z", "-", or "_" (including lowercase letters), an "abnormal vision program name" error occurs.
(7) If the program specified in the <vision program name> is not in the vision sensor, a "vision program does not exist" error occurs.
(8) If EasyBuilder's image capture specification is set to anything other than "Camera" (all trigger command), "External trigger", or "Manual trigger", an "abnormal image capture specification" error occurs.
(When the value of parameter NVTRGTMG is 1 and the image capture specification is set to "Camera", the same error occurs.)
(9) If the vision sensor is "offline", the "Put online" error occurs, so put the vision sensor "Online".
(10) If the communications line is cut while this command is being executed, an "abnormal communications" error occurs and the robot controller side line is closed.

NVTrg (network vision sensor trigger)

[Function]

Requests the specified vision program to capture an image.

[Format]

```
NVTrg□#<Vision sensor number>,<delay time>, <encoder 1 value read-out variable>
                [,[<encoder 2 read-out variable>][,[<encoder 3 value read-out variable>]
                [,[<encoder 4 read-out variable>][,[<encoder 5 value read-out variable>]
                [,[<encoder 6 read-out variable>][,[<encoder 7 value read-out variable>]
                [,[<encoder 8 read-out variable>]
```

[Terminology]

<Vision sensor number> (Can not be omitted)
This specifies the number of the vision sensor to control.
Setting range:1 - 8

<Delay time> (Can not be omitted)
This specifies the delay time (in ms) from when the image capture request is output to the vision sensor until the encoder value is obtained.
Setting range:0 - 150 ms

<Encoder n value read-out variable> (Can be omitted from the second one on)
Specifies the double precision numeric variable into which the read out external encoder n value is set.
Note: n is 1 - 8.

[Reference Program]

```
1 If M_NVOpen(1)<>1 Then        ' If vision sensor number 1 logon is not complete
2   NVOpen "COM2:" As #1        ' Connects with the vision sensor connected to COM2.
3 EndIf
4 Wait M_NVOpen(1)=1            ' Connects with vision sensor number 1 and waits for logon to be
                                  completed.
5 NVRun #1,"TEST"               ' Starts the "Test" program.
6 NVTrg #1,15,M1#,M2#           ' Requests the vision sensor to capture an image and acquires encoders
                                  1 and 2 after 15 ms.
7 NVIn #1, "TEST","E76","J81","L84",0,10
                                 'Receives the recognition count recognized with the "Test" program
                                  from the E76 cell and the recognition results from cells J81 through L84,
                                  and stores this in P_NvS1 ().
8 ・・・
  :
100 NVClose #1                  ' Cuts the line with the vision sensor connected to COM2.
```

[Explanation]

(1) Outputs the image capture request to the specified vision sensor and acquires the encoder value after the specified time. The acquired encoder value is stored in the specified numeric variable.
(2) This command moves to the next step at the point in time when the encoder value is acquired the specified time after the image capture request to the vision sensor.
(3) If the program is cancelled while this command is being executed, it stops immediately.
(4) For receiving data from the vision sensor, use the NVIn command.
(5) When this command is used with multi-tasking, it is necessary to execute the NVOpen command in the task using this command. Also, use the <vision sensor number> specified with the NVOpen command.
(6) A program start condition of "Always" is not supported.
(7) Up to three robots can control the same vision sensor at the same time, but this command can not be used by more than one robot at the same time. Use this command on any one of the robots.
(8) If an interrupt condition is established while this command is being executed, the interrupt processing is executed immediately.

[Errors]

(1) If data type for an argument is incorrect, a "syntax error in input command statement" error is generated.
(2) If there is an abnormal number of command arguments (too many or too few), an "incorrect argument count" error occurs.
(3) If the <vision sensor number> is anything other than "1" through "8", an "argument out of range" error occurs.
(4) If the NVOpen command is not opened with the number specified as the <vision sensor number>, an "abnormal vision sensor number specification" error occurs.
(5) If the vision program's image capture specification is set to anything other than "Camera" (all trigger command), "External trigger", or "Manual trigger", an "abnormal image capture specification" error occurs.
(6) If the vision sensor is "offline", the "Put online" error occurs, so put the vision sensor "Online".
(7) If the communications line is cut while this command is being executed, an "abnormal communications" error occurs and the robot controller side line is closed.

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
2 Mov P1          ' Move with maximum load.
3 LoadSet 1,1     ' Set hand 1 and workpiece 1.
4 Mov P2          ' Move with hand 1 + workpiece 1 load.
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 484, "Table 5-1: List Movement 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 541, "5.16 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 Page 494, "JADL" parameter.

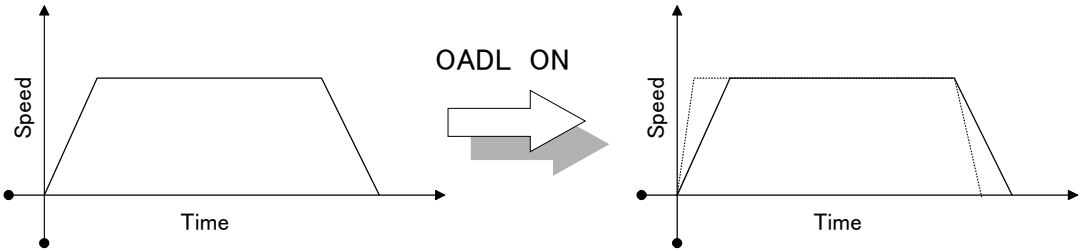


Fig.4-36:Acceleration/deceleration pattern at light load

[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 RECV process.

```
 1 Open "COM1:" AS #1          ' Communication line opening.
 2 On Com(1) GoSub *RECV       ' 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 *RECV.>>
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 *RECV.>>
23                             '
24 Com(1) Off                  ' Disable interrupt from file No. 1 communication line.
25 Close #1
26 End
  :
  :
30 *RECV                       ' 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 command. 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 label L1 if M1 is 1, label LSUB if M1 is 2, label L2 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 abel 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 command. 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 label L1 if M1 is 1, to label LJMP if M1 is 2, to label L2 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, Random, Binary
Communication line	COM1: The setting in the "COMDEV" parameter. : COM8: The setting in the "COMDEV" parameter.	Omitted = TCP communication UDP = UDP communication
	ENET:192.168.0.2 Note1)	Mxt command
	MXT: QRBUS<CPU No.2 to 4>" Select the number among 2 to 4 for the CPU module No. to specify the master robot CPU.	Mxt command

Note1) 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 (TCP communication).
*Input = 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.
*Binary = Designate the binary file mode.
*UDP = Performs UDP communication.

<File No.> Specify a constant from 1 to 8.
To interrupt from communication line: 1 to 3.

[Reference Program]

(1) File operation (Create a file named "temp.txt" and write "abc" in the file.)

```
1 Open "temp.txt" For Append As #1    ' Opens the temp.txt file as the file number 1 in Append mode.
2 Print #1, "abc"                     ' Writes "abc" in the file.
3 Close #1                            ' Closes the file.
```

(2) File operation (Writing to the binary file)

```
1 Open "temp.dat" For Binary As #1    ' Opens the temp.dat file as the file number 1 in binary file mode.
2 BPrint #1, P_Curr.X, 4              ' Writes the current position in binary format in increments of four
                                        bytes.
3 BPrint #1, P_Curr.Y, 4
4 BPrint #1, P_Curr.Z, 4
5 BPrint #1, P_Curr.A, 4
6 BPrint #1, P_Curr.B, 4
7 BPrint #1, P_Curr.C, 4
8 Close #1                            ' Closes the file.
```

(3) Communication line

```
1 Open "COM1:" As #1    ' Opens the communication line set with the first element of the parameter
                        ' COMDEV as the file number 1.
2 Mov P_01
3 Print #1, P_Curr      ' Outputs the current position to the external source. (In the following format)
                        ' "(100.00,200.00,300.00,400.00,500.00,600.00,700.00,800.00)(7,0)"
4 Input #1, M1, M2, M3  ' Receives the current position in ASCII format of "101.00,202.00,303.00" from
                          the external source.
5 P_01.X = M1
6 P_01.Y = M2
7 P_01.C = Rad(M3)      ' Copies the current position to global data.
8 Close                 ' Closes all the opened files.
9 End
```

(4) Communication line (UDP communication)

```
1 Open "COM1:" For UDP As #1          ' Opens the communication line set with the first element
                                      ' of the parameter COMDEV as the file number 1 using
                                      ' the UDP communication.
2 Dim PRob(4)                         ' Registers the teaching point for calibration in advance.
3 For Mno=1 To 4
4    Mov PRob(Mno)
5    Print #1, "TRIGGER"              ' Sends a trigger to the sensor.
6    BInput #1, MX, 4                 ' Acquires sensor recognition data in binary format.
7    BInput #1, MY, 4
8    BInput #1, MC, 4
9    PCam.X = MX
10   PCam.Y = MY
11   PCam.C = MC
12   VSSetCP 1, Mno, PCam, PRob(Mno)  ' Sets the corresponding points of the sensor coordinate
                                        system and the robot coordinate system.
13 Next Mno
14 VSRegCD 1                          ' Calculates calibration data.
15 Close #1                           ' Closes the communication line.
16 End
```

[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.

(2) A communication line is handled as a file.

(3) Use the BInput/BPrint instruction to read/write binary data.

[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 0.01 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: 0.01 to 100)
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.
7 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.
Mov (Plt 1, 5)
Refer to Page 108, "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.

[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 command if it is desired to perform interpolation movement with increased path accuracy.
(2) When this command 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 command.
(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.

[Related instructions]

Loadset (Load Set), Mv Tune (Move Tune)

[Related system variables]

HNDDAT 0 to 8, WRKDAT 0 to 8

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.

PrmRead (Parameter Read)

[Function]
Reads a parameter value.

[Format]

[Format 1] PrmRead [<Mechanism Number>], <Parameter name>, <Storage destination variable>

[Terminology]

<Mechanism Number>	Specify the mechanism No. of a parameter to be read. Parameters independent from the mechanism can also be read by specifying 1, 2, or 3 as the mechanism No. Specify the mechanism No. with a constant or variable. (The mechanism No. can be omitted. The common parameter and mechanism No. 1 can be read.) Setting range: 0 to 3, 0: Common parameter (Mechanism No. 1 can also be read by specifying 0.)
<Parameter name>	Specify the name of the parameter to be read.
<Storage destination variable>	Specify a variable in which the read parameter value is stored. Numeric value type: Only the 1st element is read and the read value is round off for the integer type. Position type: The 1st to 8th elements are read. Joint type: The 1st to 8th elements are read. Character type: All the elements are read as characters. The maximum number of characters is 127.

[Reference Program]

```
1 PrmRead 0,"LNG",CLNG$          ' Reads the parameter LNG.
2 PrmRead 1,"LNG",CLNG$          ' The parameter can also be read by specifying the mechanism
                                   No. 1.
3 PrmRead 1,"MEJAR",CMEJAR$      ' Reads the parameter MEJAR. However, the characters after the
                                   128th character are not read.
4 Dim CDIM$(16)
5 PrmRead 0,"ROMDRV",MDRV        ' Reads the parameter after converting it into a numeric value.
6 PrmRead 0,"AREA1P1",PAREA1     ' Reads the parameter AREA1P1 to the P variable.
```

[Explanation]
(1) Only the parameters disclosed to customers can be read.

[Related instructions]
PrmWrite (Parameter Write)

PrmWrite (Parameter Write)

[Function]
Writes a parameter value.

[Format]

```
PrmWrite [<Mechanism Number>], <Parameter name>, <Parameter value>
```

[Terminology]

<Mechanism Number>	Specify the mechanism No. of the parameter to be written. Parameters independent from the mechanism can also be written by specifying 1, 2, or 3 as the mechanism No. Specify the mechanism No. with a constant or variable. (The mechanism No. can be omitted. The common parameter and mechanism No. 1 can be written.) Setting range: 0 to 3, 0: Common parameter (Mechanism No. 1 can also be read by specifying 0.)
<Parameter name>	Specify the name of the parameter to be written.
<Parameter value>	Specify a parameter value to be written. The value can be also specified with a numeric value. However, if the target parameter is the character type, the value is converted into characters. Numeric value type: Only the 1st element is written and the written value is round off for the integer type. Position type: The 1st to 8th elements are written. Joint type: The 1st to 8th elements are written. Character type: All the elements are written as characters. The maximum number of characters is 127.

[Reference Program]

```
1 PrmWrite 0,"LNG",CLNG$          ' Writes the parameter LNG.
2 PrmWrite 1,"LNG",CLNG$          ' The parameter can also be written by specifying the
                                    mechanism No. 1.
3 PrmWrite 1,"MEJAR",CMEJAR$      ' Writes the parameter MEJAR.
4 PrmWrite 0,"ROMDRV",MDRV        ' Writes the parameter after converting it into a numeric
                                    value.
5 PrmWrite 0,"AREA1P1",PAREA1     ' Writes the parameter AREA1P1 to the P variable.
```

[Explanation]
(1) Only the parameters disclosed to customers can be written.
(2) It is necessary to turn the robot controller's power off and then on again after write a parameter.

[Related instructions]
PrmRead (Parameter Read)

PVSCal (PVS calibration)

[Function]

Changes the vision sensor image coordinate to the robot world coordinate using the vision sensor calibration data (parameters from VSCALB 1 to VSCALB 8) set by 2D vision calibration function of RT ToolBox3.

[Format].

```
<Position Variables>=PVSCal<Calibration Number>, <Vision X>,
                                    <Vision Y>, <Vision θ> [, <Reference Position Variables>]
```

[Terminology]

<Position Variables>	Specifies the position variable to assign. Returns the robot world coordinate for the calculation result of the coordinate conversions.
<Calibration Number>	Specify the vision sensor calibration data to be used for the coordinate conversion with the number of parameters from VSCALB 1 to VSCALB 8.
<Vision X>	X pixel coordinate of the vision sensor. [pixel]
<Vision Y>	Y pixel coordinate of the vision sensor. [pixel]
<Vision θ>	θ pixel coordinate of the vision sensor. [deg.]
<Reference Position Variables>	Specify the reference position as s position constant or position variable. When attaching a camera to the hand, specify the robot position (at the time of the image recognition) where the image is recognized with the vision sensor as a reference position. The relative calculation is performed as follows. <Robot position at the time of the image recognition>*<Calculation result of the coordinate conversion> When omitted, the absolute coordinate is set.

[Reference Program]

```
1 ' Start the target vision sensor with Open/Print/Input command to substitute the acquired image
    coordinate [pixel] for numeric variable.
2 ' MX= X [pixcel] of the vision sensor
3 ' MY= Y [pixcel] of the vision sensor
4 ' MT= θ [deg.] of the vision sensor
5 PVS=PVSCal(1,MX,MY,MT)        ' Changes the image coordinate to the robot (world) coordinate
                                  with Calibration 1.
6 PVS.Z=PDST.Z                  ' Specifies Z height.
7 Mov PVS,-50                   ' Moves to a position 50 mm above the calculated position.
8 Mvs PVS                       ' Moves to the calculated position.
```

[Explanation]

(1) Changes the vision sensor image coordinate to the robot world coordinate using the vision sensor calibration data (parameters from VSCALB 1 to VSCALB 8).
Use the 2D vision calibration function of RT ToolBox3 to set the parameters from VSCALB 1 to VSCALB 8 to be used for the coordinate conversion for the advanced settings.
(2) If the calibration numbers other than 1 to 8 are set, error L3110 (Arg. value range over) occurs.
(3) If the numbers of argument are not either four or five, error L3120 (No. of arg. is over) occurs.
(4) If the type of arguments is different, error L3810 (Different argument type) occurs.

[Related parameter]

VSCALB1 to 8

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 3 step in reference program.

Remove (Remove)

[Function]
Deletes a file.

[Format]

```
Remove□"<File Name>"
```

[Terminology]

<File Name>	Specifies a file name to be deleted.

[Reference Program]

```
1 Remove "temp.txt"    ' Deletes the specified file.
```

[Explanation]
(1) A file created with the Open command is deleted.
(2) When the delete target file does not exist, the error L7010 occurs at execution.

[Related instructions]
Open (Open)

Reset Err (Reset Error)

[Function]

This command 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 1.

[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
  :
10 ' ***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 command at the end of the jump destination processing called up by the GoSub command.

(2) An error occurs if the Return command is executed without being called by the GoSub command.

(3) Always use the Return command to return from a subroutine when called by the GoSub command. An error occurs if the GoTo command 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 173, "Act (Act)" for the interrupt processing.

(6) When an interrupt occurs during execution of the circular or the arc interpolation (Mvc, Mvr, Mvr2, Mvr3), and then control returns to the interrupted step by Return 0, the robot returns to the start point of the circle or the arc before executing the circular or the arc interpolation again.

(7) When an interrupt occurs during execution of the arch interpolation, and then control returns to the interrupted step by Return 0, the robot executes the arch interpolation from the current position.

[Related instructions]

Act (Act), GoSub (Return)(Go Subroutine), On ... GoSub (ON Go Subroutine), On Com GoSub (ON Communication Go Subroutine), Def Act (Define act)

Save

[Function]
Saves a program.

[Format]

Save[]<Save type>[, <Slot number>]

[Terminology]

<Save type>	0: Saves the slot 1 program and the user base program. 1: Saves the user base program. 2: Saves the program of a specified slot.
<Slot number>	Specify a slot number. (1 to 32) Specify the value with a constant or variable.

[Reference Program]

```
1 Save 0        ' Saves the slot 1 program and the user base program.
2 Save 1        ' Saves the user base program.
3 Save 2, 3     ' Saves the program of the slot 3.
```

[Explanation]
(1) Variable values changed during high-speed DRAM operation cannot be saved. However, using this instruction can save the values.
(2) The program waits in the executed step until it has been saved.

⚠CAUTION Since execution of the Save instruction takes time, using this instruction affects the tact time.

[Related parameter]
AUTOSAVE

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 or character string 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> *<Character string constant>
<Process>	Writes any command (other than the GoTo command) 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
6 Case 12
7  Mov P2
8  Break
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 End Select. 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 describe Select Case inside Select Case. (UP to eight levels of nesting is allowed.)

(6) It is possible to write While-WEnd and For-Next within a Case block.

(7) Use "Case IS", when using the comparison operators (<, =, >, etc.) for the "<Expression>".

(8) The Brake statement can be omitted. (Descriptions in a Case block is processed according to (1).)

(9) If the type of <Condition> and <Expression> do not match, error L3810 (Different argument type) occurs.

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

SetCalFrm (Set Calibration Frame)

[Function]

Sets the reference coordinate system used for frame transformation.
Two types of reference coordinate systems (pre-frame transformation and post-frame transformation) are set with this command.

[Format]

SetCalFrm []<Position 1>, <Position 2>, <Position 3>, <Position 4>, <Position 5>, <Position 6>

[Terminology]

<Position 1>	The position data used as the origin for the X, Y and Z axes in the pre-frame transformation reference coordinate system is designated with a position variable or position constant.
<Position 2>	The position data used as the origin for the +X axis in the pre-frame transformation reference coordinate system is designated with a position variable or position constant.
<Position 3>	The position data used as the +Y direction point in the X-Y plane of the pre-frame transformation reference coordinate system is designated with a position variable or numeric constant.
<Position 4>	The position data used as the origin for the X, Y and Z axes in the post-frame transformation reference coordinate system is designated with a position variable or position constant.
<Position 5>	The position data used as the origin for the +X axis in the post-frame transformation reference coordinate system is designated with a position variable or position constant.
<Position 6>	The position data used as the +Y direction point in the X-Y plane of the post-frame transformation reference coordinate system is designated with a position variable or numeric constant.

[Reference Program]

```
1 PR1=(0, 0, 0, 0, 0, 0)(0, 0)                ' Origin position of pre-frame transformation reference
                                                coordinate system
2 PR2=(1, 0, 0, 0, 0, 0)(0, 0)                ' +X axis position of pre-frame transformation reference
                                                coordinate system
3 PR3=(0, 1, 0, 0, 0, 0)(0, 0)                ' +Y direction position of pre-frame transformation
                                                reference coordinate system
4 PC1=(0, 0, 0, 0, 0, 0)(0, 0)                ' Origin position of post-frame transformation reference
                                                coordinate system
5 PC2=(1, 1, 0, 0, 0, 0)(0, 0)                ' +X axis position of post-frame transformation reference
                                                coordinate system
6 PC3=(-1, 1, 0, 0, 0, 0)(0, 0)               ' +Y direction position of post-frame transformation
                                                reference coordinate system
7 SetCalFrm PR1, PR2, PR3, PC1, PC2, PC3      ' Sets the reference coordinate system used for frame
                                                transformation
                                              ‘ Reference coordinate system after frame transformation
                                                is coordinate system rotated +45 degrees around Z axis
8 MvSpl 5, 50, 20, 2                          ' Using the reference coordinate system set with the
                                                SetCalFrm command, the spline file 5 path points are
                                                frame-transformed, and spline interpolation passing
                                                through those path points is executed
```

[Explanation]

(1) Position data for defining the two types of reference coordinate systems (pre-frame transformation and post-frame transformation) used for frame transformation are set. Three position data items are required to define one coordinate system, so a total of six points of position data is set. In this example of the following figure, the pre-frame transformation reference coordinate system "Xfr-Zfr-Yfr" is defined

with position data PR1, PR2 and PR3, and the post-frame transformation reference coordinate system "Xfc-Zfc-Yfc" is defined with position data PC1, PC2 and PC3.

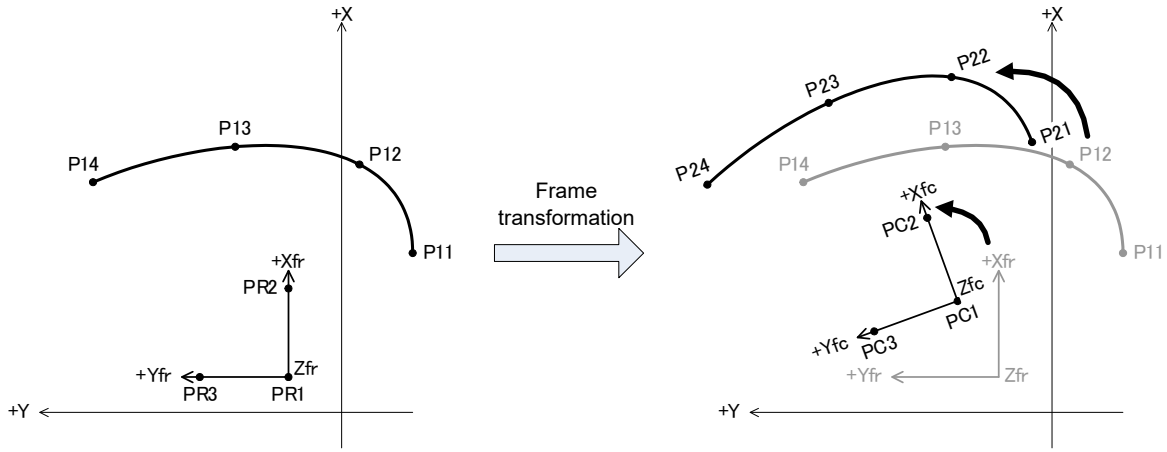


Fig.4-37: Example of frame transformation

(2) The X, Y and Z axis coordinate values of each position data are used to define the coordinate system. Other element data such as the A, B and C axis coordinate values are not used.
(3) The coordinate system cannot be calculated if the same point is found in three position data items defining the coordinate system, or if the three position data items are arranged on the same line. In this case, error L2041 (Can't calculate frame transformation coordinates) will occur.
(4) The reference coordinate system is not set immediately after the controller power is turned ON. The reference coordinate system is returned to the unset state with the main program's END command or program reset operation.
(5) The mechanism control rights (GetM command) are required to execute the SetCalFrm command.
(6) The SetCalFrm command cannot be executed in a slot for which the start conditions are ALWAYS or ERROR. Error L3287 (Can't use this command when start condition is ERR or ALW) will occur.

[Related instructions]
MvSpl (Move Spline)

[Related status variable]
Fram

Skip (Skip)

[Function]
Transfers control of the program to the next step.

[Format]

Skip

[Reference Program]

```
1 Mov P1 WthIf 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 Wth or WthIf 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_SkipCq (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. Setting range: 0.01 to 10000 [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 command in front of the error causing operation command 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 or the program is reset.

[Related system variables]

M_Spd/M_NSpd/M_RSpd

SpdOpt (Speed Optimize)

[Function]

Adjusts the speed so that the speed does not exceed during the linear interpolation operation in the horizontal direction which passes through near the singular point (X=Y=0: one of the robot's singular points).

Note) This command is limited to the corresponding robot models.
Robot model: RH-3FRHR series

[Format]

```
SpdOpt[]<On/Off>
```

[Terminology]

<On/Off>	ON: Enable the speed-optimization function. OFF: Disable the speed-optimization function.

[Reference Program]

```
1 Mov P1
2 SpdOpt On          ' Enable the speed-optimization function.
3 Mvs P2
4 Mvs P3
5 SpdOpt Off         ' Disable the speed-optimization function.
6 Mvs P6
```

[Explanation]

(1) When performing a XYZ interpolation operation while maintaining the speed of the control point, the J1 axis must rotate at a faster speed when passing through a point near the origin point O (one of the robot's singular points) as shown in Fig. 4-38, causing an excessive speed error depending on the specified speed. If SpdOpt On is executed, the speed is adjusted automatically in order to prevent an excessive speed error from occurring.
For example, while in operation at the command speed V, it approaches the origin point O, and the speed will be exceeded if it continues to operate at the current speed, the speed is decreased automatically as shown by A in Fig. 4-39 in order to prevent the speed to be exceeded. Then, when it has passed near the origin point O and it becomes possible to increase the speed, it starts accelerating to reach command speed V as shown by B in Fig. 4-39.

(2) Relation between Ovrd or Spd command
Moves at one of late speed.
a) If the specified speed (Ovrd or Spd command) is slower .....Moves at specified speed.
b) If the specified speed (Ovrd or Spd command) is faster .....Moves at the adjustment speed in this function.

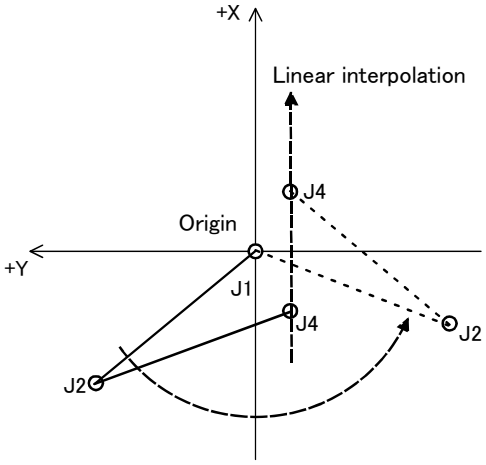


Fig.4-38:Passing through near the origin point

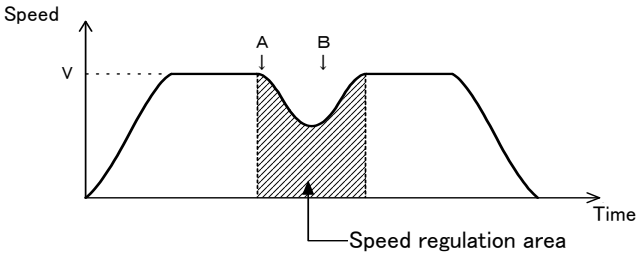
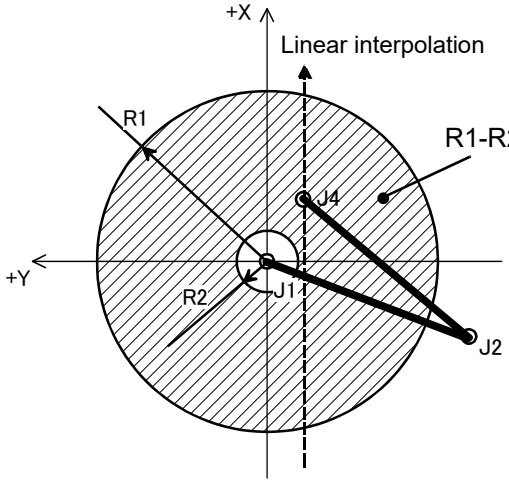


Fig.4-39:The situation of speed at speed-optimization.

(3) This command is valid only in linear interpolation movement. It functions in neither joint interpolation nor circle interpolation. And, even if the speed optimization function is valid, if the J4 axis does not pass through the area shown in "Fig. 4-40Speed regulation area and singular point area" by linear interpolation, this does not function.



Distance of R1 and R2
RH-3FRHR series
R1=240.0mm, R2=1.0mm
Inside of R2:Singular point area (positioning and passage only by joint interpolation are possible)

Fig.4-40:Speed regulation area and singular point area

(4) The initial condition of the speed optimization function at turning ON the power supply can be changed with parameter SPDOPT. This parameter also limits the models which can be used.
The initial value in the target models is SPDOPT=1 (speed optimization valid).
(5) If the End instruction or a program reset operation is executed, the status of the speed adjustment function returns to the initial state immediately after the power is turned on.
(6) When the speed adjustment function is enabled, error 2804 will be generated if the XYZ interpolation by which the J4 axis passes through a singular point area shown in Fig. 4-40 is executed, and the operation is then suspended.
(7) Even if this instruction is described in a program, it is ignored on models other than the applicable models.
(8) Even if the speed adjustment function is enabled, an exceeded speed error may be generated if a path is connected by enabling the Cnt instruction near the origin point, or a XYZ interpolation operation that drastically changes the posture is executed. In such a case, move the position where a path is connected away from the origin point, or adjust the speed by using the Ovrd instruction.
(9) In the case of a XYZ interpolation that operates slightly in the horizontal direction but operates significantly in the vertical direction, the operation speed may degrade drastically when the speed adjustment function is enabled vs. when it is disabled. In such a case, disable the speed adjustment function, or operate by using a JOINT interpolation (Mov command).

[The available robot type]

RH-3FRHR series

[Related parameter]

SPDOPT

SplFWrt (Spline Frame Write)

[Function]

Register the frame transformation information to specified spline file.

[Format]

```
SplFWrt []<Spline No.>, <Position 1>, <Position 2>, <Position 3>, <Position 4>, <Position 5>,
        <Position 6>
```

[Terminology]

<Spline No.>	The number of the spline file to register the frame transformation information is designated. Setting range: 1 to 99
<Position 1>	The origin of reference coordinate system used for frame transformation is designated.
<Position 2>	The position on the X axis of reference coordinate system used for frame transformation is designated.
<Position 3>	The position of the +Y direction point in the X-Y plane of the reference coordinate system used for frame transformation is designated.
<Position 4>	The origin after adjustment used for frame transformation is designated.
<Position 5>	The position on the X axis after adjustment used for frame transformation is designated.
<Position 6>	The position of the +Y direction point in the X-Y plane after adjustment used for frame transformation is designated.

[Reference Program]

```
1 SplFWrt 1, PR1, PR2, PR3, PC1, PC2, PC3   ' Set the reference coordinate system used for frame
                                               transformation and the coordinate system after
                                               adjustment to the spline file 1.
```

[Explanation]

(1) Registers the frame transformation information to the spline file specified by <Spline No.>.
(2) Position data for defining the two types of reference coordinate systems (pre-frame transformation and post-frame transformation) used for frame transformation are set. Three position data items are required to define one coordinate system.
(3) The X, Y and Z axis coordinate values of each position data are used to define the coordinate system. Other element data such as the A, B and C axis coordinate values are not used.
(4) The coordinate system cannot be calculated if the same point is found in three position data items defining the coordinate system, or if the three position data items are arranged on the same line. In this case, error L2041 (Can't calculate frame transformation coordinates) will occur.
(5) If a spline file corresponding to the <Spline No.> is not saved in the controller, the error L2610 (Can't open spline file) will occur.

SplWrt (Spline Write)

[Function]
Creates a spline file that includes information of the specified file.

[Format]

```
SplWrt []<Spline No.>, <File name 1> [, <Operation mode> [, <Spline cancel angle>
        [, <Block length ratio> [, <Ex-T coordinates number>[, <File version>]]]]]
```

[Terminology]

<Spline No.>	The number of the spline file to create is designated. Setting range: 1 to 99
<File name>	The file name that includes the path point information to register is designated. Describe with 16 characters or less. The position data used as the origin for the +X axis in the pre-frame transformation reference coordinate system is designated with a position variable or position constant.
<Operation mode>	The operation mode (constant linear speed/variable linear speed) during the spline interpolation is designated. Setting range: 0: Constant linear speed mode 1: Variable linear speed mode When omitted: Creates the spline file in constant linear speed mode.
<Spline cancel angle>	The angle that cancels the spline interpolation is designated. Setting range: 0 to 180 (degree) When omitted: 120 (degree)
<Block length ratio>	The block length ratio where the robot moves as a linear block instead of a spline curve is designated. Setting range: 0 to 100 (times) When omitted: 8 times
<Ex-T coordinates number>	The number of the work coordinate to be a control point. Setting range: 0 to 8 When omitted: Ex-T spline function is invalid.
<File version>	The version of the spline file to be created is designated. Setting range: 1 to 2 When omitted: Create the spline file of version 2. (Creates the latest version of the spline file corresponding to the controller.)

[Reference Program]

```
1 SplWrt 1,"01.csv"                    ' Creates spline file 1 based on the path point information
                                         registered in "01.csv".
2 SplWrt 2,"02.csv", 0, 120, 8, 1, 2   ' Creates spline file 2 for the Ex-T spline based on both the path
                                         point information registered in "02.csv" and the coordinate data
                                         registered in the work coordinate number 1.
```

[Explanation]
(1) Creates a spline file specified by <Spline No.>.
(2) Registers the path point information registered in the file specified by <File name> to the spline file.
(3) Registers the mode specified by <Operation mode> to the spline file. When omitted, constant linear speed mode is registered.
(4) Registers the angle specified by <Spline cancel angle> to the spline file. When omitted, 120 (degree) is registered. Spline cancel does not function when 0 (degree) is set.
(5) Registers the ratio specified by <Block length ratio> to the spline file. When omitted, 8 (times) is registered. Block length ratio does not function when 0 (times) is set.

(6) Registers the coordinate system data registered in WKnCord (n=1 to8) to the spline file when 1 to 8 is specified by <Ex-T coordinates number>. Ex-T spline function is invalid when 0 is set. When omitted, 0 (Ex-T spline function is invalid) is registered.
(7) Creates the spline file the version is specified by <File version>. When omitted, 2 (Creates the latest version of the spline file corresponding to the controller.) is registered.
(8) After this command is executed, the file specified by <File name> is deleted from the controller.
(9) If this command is executed with the target spline file opened with a spline related command (MvSpl, EMvSpl, SplPos, SplSpd, SplECord) for another slot, error L2610 (cannot make the spline file) will occur.
(10) If the path point file specified by <File name> cannot be opened, error L2611 (Cannot open the path point file) will occur.
(11) If the path point information registered in <File name> is of a different format (different number of tags, different tag name), error L2611 (The format of file is different) will occur.
(12) If the number of path points registered in <File name> is abnormal (less than 4 points/5001 points or more), error L2611 (The number of point is illegal) will occur.
(13) If a value outside the setting range is set for the M_SplVar function registered in <File name>, error L2615 (M_SplVar setting range exceeds setting range) will occur.
(14) If a value outside the setting range is set for the tolerance registered in <File name>, error L2615 (Tolerance setting value exceeds range) will occur.
(15) If a value outside the setting range is set for the output signal registered in <File name>, error L2615 (Output signal exceeds range) will occur.
(16) If a value outside the setting range is set for the argument, error L3110 (Arg. value range over) will occur.
(17) If the numbers of argument are abnormal (too much or too little), error L4220 (Syntax error) will occur.
(18) If the controller has no available capacity to create the spline file, error C7070 (Memory area is full) will occur.

[Related status variable]
P_WkCord

[Related parameter]
WKnCORD (n = 1 to 8)

Static

[Function]
Declares a Static value.

[Format]

```
Static[]Def[]Inte[]<Numeric value variable name>[, <Numeric value variable name>[, ...]]
Static[]Def[]Long[]<Numeric value variable name>[, <Numeric value variable name>[, ...]]
Static[]Def[]Float[]<Numeric value variable name>[, <Numeric value variable name>[, ...]]
Static[]Def[]Double[]<Numeric value variable name>[, <Numeric value variable name>[, ...]]
```

<Numeric value variable name> Designate a variable name.

```
Static[]Def[]Char[]<Character string variable name>[, <Character string variable name>[, ...]]
```

<Character string variable name> Designate a variable name.

```
Static[]Def[]Jnt[]<Joint variable name>[, <XYZ variable name>[, ...]]
Static[]Def[]Pos[]<XYZ variable name>[, <Joint variable name>[, ...]]
Static[]Def[]Work[]<Work coordinate variable name>[, <Work coordinate variable name>[, ...]]
```

<Joint variable name> Designate a variable name.
<XYZ variable name> Designate a variable name.
<Work coordinate variable name> Designate a variable name.

```
Static[]Dim[]<Variable Name>(<Eelement Value>[, <Eelement Value>[, <Eelement Value>]])
        [, <Variable Name>(<Eelement Value>[, <Eelement Value>[, <Eelement Value>]]) [, ...]]
```

<Variable Name> Designate an array variable name.
<Eelement Value> Designate the number of elements in an array variable with a constant.

[Reference Program]

```
1 Static Def Inte M1          ' Declares M1 as a static variable of numeric value type.
2 Static Def Char Message     '  Declares Message as a static variable of character string type.
3 Static Def Pos P1           ' Declares P1 as a static variable of XYZ type.
4 Static Dim cmd$(5)          ' Declares cmd as a static array variable of character string type.
```

[Explanation]
(1) Declares a static value.
(2) The static values declared in a Function procedure are held even after the function ends.
(3) Variables defined with Const can be used together.

[Related instructions]
Const, Def Inte/Def Long/Def Float/Def Double (Define Integer/Long/Float/Double), Def Char (Define Character), Def Jnt (Define Joint), Def Pos (Define Position)

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 Mvs 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 (100,0,100,0,0,0,0,0) are set in PTL01, it will have the same meaning as (1).)

```
1 Tool PTL01
2 Mvs P1
```

[Explanation]

(1) The Tool command 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 command.

(2) "Tool conversion data" changed with this command is saved in parameter MEXTL. The saved values are retained even after the controller is turned off.
For programs in which Base command, Tool command, and M_Tool are executed repeatedly, parameters may not be saved in time, causing error C7091 (parameter save error). If error C7091 occurs, change the order of Base command and Tool command as shown in the following example.

Example)

```
*MAIN                          Base PB
 Base PB                       Tool PT
 Tool PT                  ➡    *MAIN
 If M_In(20)=1 Then             If M_In(20)=1 Then
  GoSub *SUB1                    GoSub *SUB1
 EndIf                          EndIf
 GoTo *MAIN                     GoTo *MAIN
```

(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 518, "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 16 parameters as tool conversion data.

[Related parameter]
MEXTL, MEXTL 1 to 16 Refer to Page 517, "5.6 Standard Tool Coordinates" for detail.

[Related system variables]
P_Tool/P_NTool, M_Tool

Torq (Torque)

[Function]
Designates the torque limit for each axis. By specifying the torque limit, an excessive load (overload) on works and so froth 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
                                    ' Generate an interrupt when the difference between the
                                      command position and the feedback position reaches 10 mm or
                                      more.
2 Act 1=1                           ' Enable the interrupt 1
3 Torq 3,10                         ' Set the torque limit of the three axes to 10% of the normal torque
                                      using the torque instruction.
4 Mvs P1                            ' Moves
5 Mov P2
:
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) 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

```
1 Wait M_In(1)=1          ' The same meaning as 1 *L10:If M_In(1)=0 Then GoTo *L10.
2 Wait M_In(3)=0
```

(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.
(4) Number of conditions which may be included in a Wait command is one. If more than one is included, an erroneous judgment or an error in execution process can result.

An example of wrong indication) Wait M_In(38)=1 Or M_In(39)=1
→There are two ways to avoid it.
1) Avoid using the Wait command and use the "If-Then" statement instead.
Example) *Loop
If M_In(38)=1 Or M_In(39)=1 Then *Next Else *Loop
*Next
2) Set 0 (conditional command) to the parameter PRSPEC.

(5) If Wait and CallP commands are executed at the same time in a multitask program, it may affect the system. For a possibility that these commands are executed at the same time in a multitask program, set 0 (conditional command) in the parameter PRSPEC.

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 Page 168, "4.8 Operators")

[Reference Program]

(1) 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.
3   Print# 1, M1                  ' Output the M1 value.
4 WEnd                            ' Return to the While statement (step 1)
5 End                             ' End the program.
```

(2) Process of the While-WEnd can be skipped by Break

```
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   M_Out(8)=M1                   ' Output the numeric variable M1.
4   If M_In(8)=1 Then Break       ' If the input signal 8 is turned on, jump to Step 6.
5 WEnd                            ' Return to the While statement (step 1)
6 If M_BrkCq=1 Then Hlt
```

[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.
(5) In the While, it can escape to the next step of the WEnd by Break. That is, process of the While-WEnd can be skipped.

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 Page 168, "4.8 Operators".)]

[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)

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 command at the processing part.
(4) If the robot is stopped using the Hlt or the Skip command, it decelerates and stops in the same way as for "Stop type 1" or "Stop type 2" of the Def Act command. (Refer to Page 202, " Def Act (Define act)")
The Hlt command interrupts the execution of the program. The Skip command executes the next step and continues the program.
When 0 is set to the parameter WTHFUNC, the robot stops by "stop type 1". When 1 is set to the parameter WTHFUNC, the robot stops by "stop type 2". The initial setting is "stop type 1" (WTHFUNC=0).

XClr (X Clear)

[Function]

This instruction cancels the program selection status of the specified task slot from within a program. Enabling the execution of a new program in the specified task slot. 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.
13 Wait M_Psa(2)=1        ' Waits until selection of a program is possible in task slot 2.
14 XClr 2                 ' Cancels the program selection status of task slot 2.
15 End
```

[Explanation]

(1) If the specified slot does not select the program, the error (L3370) will occur.
(2) If the specified slot is operating, the error (L3380) will occur.
(3) If the specified slot is being paused, the error (L3380) will occur.
(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. Specify with the character string constant. (The character string variables cannot be used.)

[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) If the specified program does not exist, the error (L4140) will occur.
(2) If the specified program is already selected for another slot, the error (L3360) will occur.
(3) If the specified program is being edited, the error (L3360) will occur.
(4) If the specified slot is operating, the error (L3360) will occur.
(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, this instruction may fail while loading a program. If necessary, perform a load completion check as shown on the 3rd step of the reference program.

[Related instructions]

XClr (X Clear), XRst (X Reset), XRun (X Run), XStp (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.
(If the slot is operating or no program is selected, the error (L3340, L3350) will occur.)

(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. Specify with either the character string constant or character string variables.
<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) If the specified program does not exist, the error (L4140) will occur.
(2) If the specified slot No. is already in use, the error (L3310) will occur.
(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, this instruction may fail while loading a program. If necessary, perform a load completion check as shown on the 2nd step of both the reference programs (3) and (4).

[Related instructions]
XClr (X Clear), XLoad (X Load), XRst (X Reset), XStp (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 no program is selected, the error (L3330) will occur. If the program is already stopped, the execution does not fail.
(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 Page 168, "4.8 Operators" 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 16 characters can be used. (Up to 17 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(10)=0 Then GoTo *LBL1     ' Wait by the step 3 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) The underscore ("_") cannot be specified as the 2nd character. This form is for the system external variable. For example, "*A_", "*B_", "*Z_", etc. are the syntax error.
When using the "*L_", the error occur at execution. This form is reservation for the system.
In addition, the underscore ("_") can be used in the 3rd character or later.
(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.

4.13 Detailed explanation of Robot Status Variable

4.13.1 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.

4.13.2 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 (Open) command. 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 command (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 Cose #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 (Open) command 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 command, the current open status will not change. In such a case, it is necessary to close the line with the Close (Close) command once, and then execute the OPEN command 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 function returns the mechanism name (robot type name) for which control right has been acquired.

[Format]

Example) <Character String Variable >=C_Mecha[(<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, current task slot number is set as the default value.

[Reference Program]

```
1 C1$=C_Mecha(1)        ' "RH-3FRH5515-D" is assigned to C1$.
                          (When slot 1 has acquired control right of mechanism 1 ("RH-3FRH5515-D"))
```

[Explanation]
(1) The mechanism name (robot type name), for which the specified task slot has acquired control right, is returned.
(2) When a task slot, which has not acquired control right of the mechanism, is specified, "" (none) is returned.
(3) This variable only reads the data.

[Related instructions]
GetM (Get Mechanism), RelM (Release Mechanism)

C_Prg

[Function]
This variable returns the selected program name (number).

[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, current task slot number is set as the default value.

[Reference Program]

```
1 C1$=C_Prg(1)          ' "10" is assigned to C1$. (if the program name is "10".)
```

[Explanation]
(1) The program name (number) 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 program name (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: econd (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_Date

C_User

[Function]
This variable returns the data registered in the "USERMSG" parameter.

[Format]

Example) <Character String Variable >=C_User

[Reference Program]
```
1 C1$=C_User              ' The characters registered in "USERMSG" are assigned to C1$.
```

[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_ColMxl

[Function]

Return the maximum value of the differences between the estimated torque and actual torque while the collision detection function is being enabled.

[Format]

Example) <Joint Type Variable>=J_ColMxl [(<Mechanism Number>)]

[Terminology]

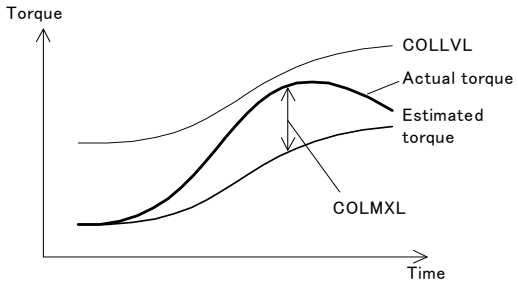
<Joint Type Variable>	Specify a joint type variable to be assigned.(Joint type variable will be used even if this is a pulse value.)
<Mechanism Number>	Enter the mechanism number. 1 to 3, If the argument is omitted, 1 is set as the default value.

[Reference Program]

```
1 M1=100                          'Set the initial value of the allowable collision 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 collision level of each axis.
8 ColChk On                       'Enable the collision detection function.
                                  (Start the calculation of the maximum value of torque error.)
9 Mov P1
  :
  :
50 ColChk Off                     'Disable the collision detection function.
                                  (End the calculation of the maximum value of torque error.)
51 M1=J_ColMxl(1).J1+10           'For each axis, the allowable collision 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_ColMxl(1).J2+10
53 M3=J_ColMxl(1).J3+10
54 M4=J_ColMxl(1).J4+10
55 M5=J_ColMxl(1).J5+10
56 M6=J_ColMxl(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 collision 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 collision level.

(3) For robots that prohibit the use of collision 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.

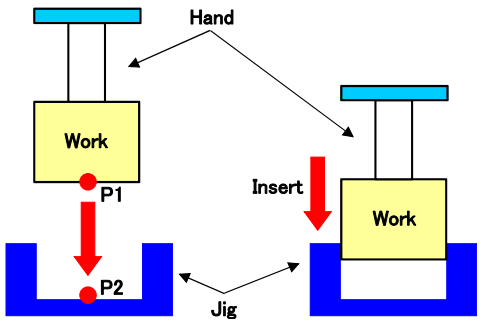
(6) This variable only reads the data.

[Reference]

CavChk On (CavChk On), ColLvl (Col Level), M_ColSts, P_ColDir

[Sample program]

The program which sets the collision detection level automatically is shown below.



Sets up the collision detection level automatically in the moving to P2 from P1.

Sample program	Explanations
'********** Collision detection level automatic setup ********** 'GoSub *LEVEL ' Collision detection level automatic setting program 'HLT '********** *MAIN Oadl ON LoadSet 2,2 Collvl M_01,M_02,M_03,M_04,M_05,M_06,, Mov PHOME Mov P1 Dly 0.5 ColChk ON Mvs P2 Dly 0.5 ColChk OFF Mov PHOME End '************************ LEVEL FIX ************************ *LEVEL Mov PHOME M1=0 ' Collision detection level of the J1 axis (initialization) M2=0 ' Collision detection level of the J2 axis (initialization) M3=0 ' Collision detection level of the J3 axis (initialization) M4=0 ' Collision detection level of the J4 axis (initialization) M5=0 ' Collision detection level of the J5 axis (initialization) M6=0 ' Collision detection level of the J6 axis (initialization) ColLvl 500,500,500,500,500,500,, ' Set the collision detection level to 500% For MCHK=1 To 10 Dly 0.3 Mov P1 Dly 0.3 Colhk ON 'Enable the collision detection. Mvs P2 Dly 0.3 ColChk OFF 'Disable the collision detection. If M1<J_ColMxl(1).J1 Then M1=J_ColMxl(1).J1 If M2<J_ColMxl(1).J2 Then M2=J_ColMxl(1).J2 If M3<J_ColMxl(1).J3 Then M3=J_ColMxl(1).J3 If M4<J_ColMxl(1).J4 Then M4=J_ColMxl(1).J4 If M5<J_ColMxl(1).J5 Then M5=J_ColMxl(1).J5 If M6<J_ColMxl(1).J6 Then M6=J_ColMxl(1).J6 Next MCHK M_01=M1+10 M_02=M2+10 M_03=M3+10 M_04=M4+10 M_05=M5+10 M_06=M6+10 ColLvl M_01,M_02,M_03,M_04,M_05,M_06,, Mvs P1 Mov PHOME RETURN '**********	Is the command which executes the collision detection level automatic setting subroutine. Remove the comment out of the head when set up automatically. Moves in the optimal acceleration and deceleration. Reads the information on the hand and the work-piece. (For the system optimization of the acceleration-and-deceleration hours) Re-set up the collision detection level. Moves to PHOME (standby position). Moves to P1 (starting position). Enable the collision detection. Disable the collision detection. Return to PHOME (standby point) End of program line. The collision detection level automatic setting subroutine. Set the collision detection level to 500% (maximum value). (Before starting movement, confirms that there is no obstacle on the course) Although the collision detection level is automatically detectable, please execute two or more times in consideration of the dispersion etc. The ten of repeat number in the sample is reference values. J_COLMXL is the maximum value of differences between the estimated torque and actual torque while the collision detection function is being enabled. Memorizes the maximum values in the ten times measured for consideration of the dispersion. Usually, as for the detection level, the value of the parameter "ColLvl" will be set up after the power supply ON. Therefore, the value set up automatically should be recorded on external variable. "10" is added with the sample, because 10% of circular land is given for the value searched for by automatic detection. * 10% is the reference value. Depending on the system to be used, it may not operate normally. Please confirm with the system and adjust to the optimal value. Refer to "ColLvl (Col Level)" for details.

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) J_AmpFbc is a percentage of the rated current (increment of 0.1% (1000 = 100%)).
(5) 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 ration 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_BsNo

[Function]
Returns a current base coordinate system number.

[Format]

Example) <Numerical variable>=M_BsNo[(<mechanism number>)]

[Terminology]
<Numerical variable> A numerical variable to which a value is to be assigned is designated.
<Mechanism number> A mechanism number which is chosen from 1 through 3.
(1 is chosen to indicate omission.)
Constants, variables, logic/arithmetic expressions, and functions are usable.
When a real number or a double-precision real number is specified, the fractional portion of 0.5 or over of the number is counted as one and the rest is cut away.

[Reference Program]

```
1 M1=M_BsNo      'Assign base coordinate number for Mechanism No. 1 to variable M1.
2 If M1=1 Then   'If base coordinate number is one, move to P1.
3 Mov P1
4 Else           'If base coordinate number is other than one, move to P2.
5 Mov P2
6 EndIf
```

[Explanation]
(1) Base coordinate number being currently specified (parameter: MEXBSNO) is read.
(2) The following coordinate system is set according to the value that is read.
a) 0: System's initial value (P_Nbase)
b) 1~8: Work coordinate system number 1 through 8 (parameter: WK1CORD~WK8CORD)
c) -1: Base conversion setting is made by other than the above options.
(Base conversion data is specified by a base command, or parameter MEXBS is directly edited.)
(3) If reference is made to the M_BrkCq variable even for once, the existing "break" condition is cleared (relevant value goes to zero). When you want to retain the condition, therefore, save it by assigning an appropriate value to the numerical variable.
(4) You can clear the "break" condition via the T/B monitor screen, as well.

[Related instructions]
Base (Base)

[Related parameter]
MEXBSNO, WKnCORD("n" is 1 to 8), MEXBS

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_CavSts

[Function]

Returns the robot CPU number where an interference is predicted.
1 to 3: Interference predicted.
0: Interference not predicted.

This function is only available for certain models. For details, refer to Page 567, "5.24 Interference avoidance function".

[Format]

Example) Def Act 1,M_CavSts [(<Mechanism No.>)] <> 0 GoTo *LCAV,S

[Terminology]

<Mechanism No.> Enter the mechanism number, 1 to 3. If the argument is omitted, 1 is set.

[Reference Program]

Refer to Page 589, "5.24.10 Sample programs".

[Explanation]

(1) When an interference between robots is predicted, the robot CPU number (1 to 3) where the interference is predicted is written. When an interference with a free plane limit is predicted, 1 is written.
(2) The value is not cleared to 0 until an END command is executed. To clear the value to 0, use an interrupt processing.
(3) The value is retained until the execution of the End command, program reset, or execution of "CavChk Off" with the CavChk On (CavChk On) command (disables the stop function of the interference avoidance function.)
(4) This command is used as an interrupt condition with the Def Act command in the NoErr mode operation.
(5) This command can be used to read and write.

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 collision detection status..
1: Detecting an collision
0: No collision has been detected

[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 collision
                                             is detected using an interrupt.
2 Act 1=1
3 ColChk ON,NOErr                 'Enable the collision detection function in the error non-occurrence
                                   mode.
4 Mov P1
5 Mov P2                          'If an collision 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 collision detection.
101 ColChk Off                    'Disable the collision 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 safe position.
105 Mvs PDst                      'Move to the safe position.
106 Error 9100                    'Stop operation by generating a user-defined L level error.
```

[Explanation]
(1) When an collision is detected, it is set to 1. When the servo is turned off and the collision 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.

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 (When CYC is set to the slot table parameter.)
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 CR800-R/Q 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_DIn32

[Function]
Obtains 32-bit data via the CC-Link register from an external device.

[Format]

Example)<Numeric Variable>=M_DIn32 (<Equation/Inputting register number>)

[Terminology]

<Numeric Variable>	Specifies a long-precision integer number variable to be substituted.
<Equation/Inputting register number>	The range of CC-Link register numbers that can be specified is from 6000 to 6254.

[Reference Program]

```
1 M1=M_DIn(6000)                ' Obtains a value of 32 bits from CC-Link register number 6000
                                  and 6001, and assigns that value to M1&.
2 M2%=6002                      ' Assigns 6002 to M2%.
3 M3&=M_DIn32(M2%) And &H7FFFF  ' Obtains a value of 32 bits from the CC-Link register number
                                  specified by M2%, and assigns the lowest 19-bit value from that
                                  value to M3&.
```

[Explanation]
(1) Two points are used for obtaining 32-bit data from the specified CC-Link register number.
(2) Use a long-precision integer number variable for <Numeric Variable>. If any other variable is used, the correct information cannot be obtained.
(3) Specify a CC-Link register number between 6000 and 6254. If a number outside the range is specified, an all-zero value will be returned and no error will be generated.
(4) M_DIn32 is read-only.

M_DOut32

[Function]

Outputs 32-bit data to an external device via the CC-Link register. Alternatively, checks the current output information.

[Format]

Reference

Example)<Numeric Variable>=M_DOut32 (<Equation/Inputting register number>)

Write

Example)M_DOut32 (<Equation/Inputting register number>)=<Numeric Variable>

[Terminology]

<Numeric Variable>	Specifies a long-precision integer number variable to be substituted.
<Equation/Inputting register number>	The range of CC-Link register numbers that can be specified is from 6000 to 6254.
<Numeric>	Specifies the data to be output. The range of the numerical value is from -2147483648 to 2147483647 (&H80000000 to &H7FFFFFFF).

[Reference Program]

```
1 M_DOut32(6000)=&H12345678      ' Outputs 12345678 in hexadecimal notation to CC-Link
                                   register number 6000 and 6001.
                                   &H4567 will be output to register number 6000, and
                                   &H1234 will be output to 6001.
2 M1&=M_DOut32(6002) And &H7FFFF ' The lowest 19-bit value from the 32-bit data output to CC-
                                   Link register number 6002 and 6003 will be assigned to
                                   M1&.
```

[Explanation]

(1) The specified 32-bit data is output to two points from the specified CC-Link register number.
(2) The current information (32-bit data) output to two points is obtained from the specified CC-Link register number.
(3) Use a long-precision integer number variable for <Numeric Variable> and <Numerical Value>. If any other variable is used, the information may not be processed.
(4) Specify a CC-Link register number between 6000 and 6254. If a number outside the range is specified, an all-zero value will be returned when referenced and no processing will be performed when written, resulting in no error being generated. However, if a minus value is specified, error L3110 will be generated.
(5) By setting the SYNCIO parameter to the high-speed mode, the updating cycle of the register output can be sped up. It is recommended to use high-speed mode with signals interlocked for synchronization in order to keep the timing of the I/O signals correct.
(6) When the specified register number is not occupied by a robot, although data is displayed on the output register monitor, such as RT ToolBox3, as output information, the data is not actually output via the CC-Link. When the last register occupied is specified, although data is displayed on the monitor as output information, only the information of the last register occupied (lowest 16 bits) is output.
(7) The pulse output which combines the Dly command cannot be used. If the Dly command is used, error L4220 (syntax error) occurs.
(8) When data is output to a register number assigned to the dedicated output of the DIODATA and SVDATA parameters, error L0091 (signal already assigned to dedicated output) will occur.

M_ErCode

[Function]
Returns the detailed error number of the error currently generated.

[Format]

Example) <Numeric Variable>=M_ErCode

[Terminology]

<Numeric Variable>	Specifies a 32-bit long-precision integer number. (Specifying a 16-bit long-precision integer number will cause an out-of-range error when substituted.) Refer to (2) in [Explanation] for the meaning of the read value.

[Reference Program]

```
1 *LBL: If M_Err=0 Then *LBL        ' Waits until an error is generated.
2 MD&=M_ErCode                      ' Reads the detailed error number (substitutes for the long-
                                      precision integer number).
3 MS%=Int(MD&/1000) Mod 100         ' Obtains two digits of the detailed error number.
```

[Explanation]
(1) If two or more errors occur, returns the information on the highest level error. (Only one error)
(2) The detailed error number is a number with a maximum of nine digits.
XXXXYYYYY..... XXXX: Four-digit error number displayed by the O/P and T/B
YYYYY: Detailed error number (Content determined by error.)

M_Err/M_ErrLvl/M_Errno

[Function]
Returns information regarding the error generated from the robot.
M_Err : Error occurrence condition
M_ErrLvl : The level of the occurrence error
M_Errno : Error number

[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.
The value assigned and meaning.
M_Err : 0/1 = No error/Under error occurrence
M_ErrLvl : 0/1/2/3/4/5/6=No error / Caution / Low / High / Caution1 / Low1 / High1
Note) The meaning of each terminology is shown in Table 4-19.
M_ErrNo : Error number

[Reference Program]

```
1 *LBL: If M_Err=0 Then *LBL    ' 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) If two or more errors occur, returns the information on the high error of the error level most.
(3) The error level which M_ErrLvl returns, and its meaning are shown below.

Table 4-19:The error level and meaning

Error level	Terminology	Meaning	Error reset
0	No error	The error has not occurred.	-
1	Caution	Program is continued.	[RESET] Key
2	Low	The program under execution is interrupted.	[RESET] Key
3	High	The program under execution is interrupted and turns off the servo power.	[RESET] Key
4	Caution1	Program is continued.	Power supply reset
5	Low1	The program under execution is interrupted.	Power supply reset
6	High1	The program under execution is interrupted and turns off the servo power.	Power supply reset

[Related instructions]
Error (error), Reset Err (Reset Error)

M_ESpd

[Function]

Returns the transit speed [mm/s] of Ex-T coordinate system currently used during the Ex-T control/Ex-T spline interpolation

[Format]

Example)<Position Variables>=M_ESpd [(<Mechanism Number>)]

[Terminology]

<Position Variables> Designates the numerical variable substituted for the reference results.

<Mechanism Number> Sets the mechanism No. that executes Ex-T control/Ex-T spline interpolation.
Setting range: 1 to 3
When omitted: 1
If a non-existent mechanism No. is designated, the error L3870 (designated mechanism No. invalid) will occur when execution is started.

[Reference Program]

```
1 Wait M_00=1              ' Wait for spline interpolation to start
2 *L1:If M_ESpd < 50 Then GoTo *L1
3 M_Out(100)=1             ' When the transit speed is over 50 [mm/s], turns output signal 100 ON.
```

[Explanation]

(1) During Ex-T control/Ex-T spline interpolation, the transit speed of the Ex-T coordinate is returned.

(2) By referring to the M_ESpd value with multi-task, the operation process or signal output, etc., can be executed according to the transit speed of Ex-T spline interpolation.

(3) When the Ex-T control/Ex-T spline interpolation is not executed, 0.0 will be returned.

(4) M_ESpd returns the value indicated in the following table according to the state.

Table 4-20:Value returned by M_ESpd

Status	Value returned by M_ESpd
Immediately after power ON	0.0
During Ex-T control/Ex-T spline interpolation execution	Transit speed of the Ex-T coordinate during executing interpolation.
During interpolation execution without Ex-T control/Ex-T spline interpolation	0.0
During interpolation is paused	0.0
Immediately after main program's End command is executed	0.0
Immediately after program reset operation	0.0
Mechanism not compatible with spline interpolation	0 (0 even after write is executed)

(5) M_ESpd is a read-only status variable.

[Related instructions]

EMvc (E Move C), EMvr (E Move R), EMvr2 (E Move R 2), EMvr3 (E Move R 3), EMvs (E Move S), EMvSpl (E Move Spline)

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_GDev/M_GDevW/M_GDevD

[Function]

Writes to or directly references CPU buffer memory with the CR800-R/Q series controllers.
(CR800-R/Q series controllers only)

M_GDev: Writes or References in bits (one bit).
M_GDevW: Reads/ Writes per word. (16 bits)
M_GDevD: Reads/ Writes per double word. (32 bits)

[Format]

ExampleReference

<Numeric Variable> = M_GDev(<Top input output number>, < Buffer memory address >, < Bit number>)
<Numeric Variable> = M_GDevW(<Top input output number>, < Buffer memory address >)
<Numeric Variable> = M_GDevD(<Top input output number>, < Buffer memory address >)

Writing

M_GDev(<Top input output number>, < Buffer memory address >, < Bit number>) = < Numeric value >
M_GDevW(<Top input output number>, < Buffer memory address >) = < Numeric value >
M_GDevD(<Top input output number>, < Buffer memory address >) = < Numeric value >

[Terminology]

<Numeric Variable>	Specifies the numerical variable to assign.
<Top input output number>	Enter the I/O number of the CPU module. (The value to specify omits the single digit low order of the top input output number in the hexadecimal number.) Range : "&H3E0" to "&H3E" in hexadecimal expression. ("992" to "995" in the decimal number) No. 1: &H3E0 (992 in the decimal number) No. 2: &H3E1 (993 in the decimal number) No. 3: &H3E2 (994 in the decimal number) No. 4: &H3E3 (995 in the decimal number) Note) The write function is allowed to host CPU only.
<Buffer memory address>	Specifies CPU module buffer memory addresses using constants or numeric variables. The effective range varies for each state variable. (Decimal) M_GDev: 10000 to 24335 M_GDevW: 10000 to 24335 M_GDevD: 10000 to 24334
<Bit number>	Specifies the bit number of CPU module buffer memory addresses using constants or numeric variables. These number must be specified in decimal numbers. The setting range is as follows. M_GDev: 0 to 15
<Numeric value>	Specify the data to write in with the constant or the numeric variable. The effective range varies for each state variable. M_GDev: 0, 1 M_GDevW: -32768 to 32767 (&H8000 to &H7FFF) M_GDevD: -2147483648 to 2147483647 (&H80000000 to &H7FFFFFFF)

[Reference Program]

```
1 M_GDev(&H3E1, 10010, 2)=1                  'Sets the value of bit number 2 in buffer memory address
                                              10010 of CPU No. 2 to 1.
2 M_GDevW(&H3E1, 10010)=&HFFFF               'Sets buffer memory address 10010 of CPU No. 2 to FFFF.
3 M_GDevD(&H3E1, 10011)=P1.X * 1000          'Multiplies the X coordinate value of position variable P1 by
                                              1000 and sets buffer memory addresses 10011 and 10012
                                              (2 words) of CPU No. 2 with this value.
4 M1%=M_GDevW(&H3E2, 10001) And &H7 'Sets M1 to the values of the lower 3 bits in buffer memory
                                              address 10001 of CPU No. 3.
```

[Explanation]

(1) This variable is used to write or reference PLC buffer memory.
(2) Buffer memory addresses are specified using start I/O numbers and the actual buffer memory addresses.
(3) Both write and reference data strings are integers.
(4) Variable M_GDevW uses 1 word, or 16 bits, of data in the specified buffer memory address. Variable M_GDevD uses 2 words, or 32 bits, of data.
(5) Specify start I/O numbers in hexadecimal format. The range of numbers is between &H3E0 to &H3E3, which is equivalent to 992 to 995 in decimal. Specify buffer memory addresses in decimal numbers within a range of addresses between 10000 to 24335.
(6) Write operations can only be performed for buffer memory addresses of the target CPU. Attempting to write data to addresses for other CPUs will not result in a successful change to the data stored by the address.
(7) To prevent 32-bit data strings from being fragmented with M_GDevD, specify buffer memory addresses starting with the even-numbered address first. Only CR800-R/Q series devices are supported. (Statements executed using CR800-D series devices result in unsuccessful write operations, and the reference returns zero value.)

[Supplementary]

Table 4-21:<Numeric variable>

O: The available, X: unavailable

Bit width	Numeric variables types				Other variables		
	Integer Ex.)M1%	Long-precision integer number Ex.)M1%	Single-precision real number Ex.)M1&	Double-precision real number Ex.)M1	Position Ex.)P1.X	Joint Ex.)J1.J1	Character string Ex.)C1$
M_GDev	O	O	O	O	O	O	X
M_GDevW	O	O	O	O	O	O	X
M_GDevD	X	O	O	O	O	O	X

Table 4-22:Constants and variables usable as <Start I/O number>, <Buffer memory address>, and <Bit number>

O: The available, X: unavailable

Bit width	Constant types			Numeric variables types				Other variables		
	Numeric value Ex.)12	Binary number Ex.)&B1100	Hexadecimal number Ex.)&HC	Integer Ex.)M1%	Long-precision integer number Ex.)M1%	Single-precision real number Note1) Ex.)M1&	Double-precision real number Note1) Ex.)M1	Position Note1) Ex.)P1.X	Joint Note1) Ex.)J1.J1	Character string Ex.)C1$
M_GDev	O	O	O	O	O	O	O	O	O	X
M_GDevW	O	O	O	O	O	O	O	O	O	X
M_GDevD	O	O	O	O	O	O	O	O	O	X

Note1) The real value is rounded off.

Table 4-23:<Numeric value>

O: The available, X: unavailable

Bit width	constant types			Numeric variables types				Other variables		
	Numeric value Note1) Ex.)12	Binary number Ex.)&B1100	Hexadecimal number Ex.)&HC	Integer Ex.)M1%	Long-precision integer number Ex.)M1%	Single-precision real number Ex.)M1&	Double-precision real number Ex.)M1	Position Note1) Ex.)P1.X	Joint Note1) Ex.)J1.J1	Character string Ex.)C1$
M_GDev	O	O	O	O	X	X	X	X	X	X
M_GDevW	O	O	O	O	X	X	X	X	X	X
M_GDevD	O Note2)	O Note2)	O Note2)	O Note2)	O	O	O Note3)	O	O	X

Note1) The real value is rounded off.

Note2) ⚠CAUTION For the numerical value of the less than 16 bits of the binary number (-32768 to +32767), the specified constant will handle as a negative numerical value, if the bit 15 (the 16th bit) turns on. Therefore, please be careful of turning on all of upper 16 bits. (The sign bit is extended)

Example)

Designation of "-32768(&B1000000000000000)" will output the "&B11111111111111111000000000000000."

[Measures]

After substituting the constant for the long-precision integer number variable as follows, when substituting to this robot status variable M_GDevD, &B00000000000000001000000000000000 (binary number) can be outputted.

1 M1&=-32768

2 M_GDevD(&H20)=M1&

Note3) The ranges of the numerical value which can be outputted are -2147483648 to 2147483647.

M_Gps

[Function]

This command returns the number of the position data stored in the P_GpsX() for the monitoring number defined in the Def Gps command, using the get-position-quick function (GPS function). ("X" indicates the same number as the target monitoring number from 1 to 8.)

[Format]

Example) <Numeric Variable>=M_Gps[(<Numeric>)]

[Terminology]

<Numeric Variable>	Specifies the numerical variable to assign. The number of the current position data stored in the status variable P_GpsX() corresponding to the monitoring number set in <Numeric value> is returned. The default value is set to 0. The value is cleared to zero when the GpsChk On command is executed.
<Numeric>	Set the monitoring number from 1 to 8 defined in the Def Gps command. If the argument is omitted, 1 is set as the default value.

[Reference Program]

```
1 Def Gps1,851,1          ' The position data of the mechanism No. 1 is recorded for the
                           monitoring No. 1 when the signal No. 851 is turned on.
2 GpsChk On,1             ' Monitoring a condition for the monitoring No. 1 is started.
3 Mvs P1                  ' Moves to P1
4 GpsChk Off,1            ' Monitoring a condition for the monitoring No. 1 is stopped.
5 M1=M_Gps(1)             ' The number of the position data recorded in the P_Gps1 is stored.
6 If M1=0 Then Error 9000 ' The error 9000 is generated if no position data is recorded.
```

[Explanation]

(1) The current position data of the robot is stored in the P_GpsX() when the condition defined in the Def Gps command is met. The number of position data stored in the P_GpsX() is stored in this status variable.
(2) The number of times when the condition defined in the Def Gps command is met can be confirmed.
(3) The value is cleared to zero when the GpsChk On command is executed.
(4) The number of times can be confirmed after the GpsChk Off command is executed.
(5) This variable only reads the data.
(6) The number of times is cleared to zero when the controller is turned on.

[Related instructions]

Def Gps (Define get position), Def Map (Define mapping), GpsChk (Get position check)

M_HGDev/M_HGDevW/M_HGDevD

[Function]

Writes or directly references the periodical communication area of CPU buffer memory with the CR800-R series controllers.
(CR800-R series controllers only)
(This enables multiple robot interfaces such as interlocks to be accessed more quickly without having to use ladder program of the PLC. CPU buffer memory information other than robot CPUs, such as motion CPU, can also be accessed. (Refer to Page 622, "5.26 Direct communication with robot CPUs"))

M_HGDev: Writes or References in bits (one bit).
M_HGDevW: Reads/ Writes per word. (16 bits)
M_HGDevD: Reads/ Writes per double word. (32 bits)

[Format]

ExampleReference

<Numeric Variable> = M_HGDev(<Top input output number>, < Buffer memory address >, < Bit number >)
<Numeric Variable> = M_HGDevW(<Top input output number>, < Buffer memory address >)
<Numeric Variable> = M_HGDevD(<Top input output number>, < Buffer memory address >)

Writing

M_HGDev(<Top input output number>, < Buffer memory address >, < Bit number >) = < Numeric value >
M_HGDevW(<Top input output number>, < Buffer memory address >) = < Numeric value >
M_HGDevD(<Top input output number>, < Buffer memory address >) = < Numeric value >

[Terminology]

<Numeric Variable>	Specifies the numerical variable to assign.
<Top input output number>	Enter the I/O number of the CPU module. (The value to specify omits the single digit low order of the top input output number in the hexadecimal number.) Range : "&H3E0" to "&H3E" in hexadecimal expression. ("992" to "995" in the decimal number) No. 1: &H3E0 (992 in the decimal number) No. 2: &H3E1 (993 in the decimal number) No. 3: &H3E2 (994 in the decimal number) No. 4: &H3E3 (995 in the decimal number) Note) The write function is allowed to host CPU only.
<Buffer memory address>	Specifies CPU module buffer memory addresses using constants or numeric variables. The effective range varies for each state variable. (Decimal) M_HGDev: 0 to 12287 M_HGDevW: 0 to 12287 M_HGDevD: 0 to 12286
<Bit number>	Specifies the bit number of CPU module buffer memory addresses using constants or numeric variables. These number must be specified in decimal numbers. The setting range is as follows. M_HGDev: 0 to 15
<Numeric value>	Specifies write data using constants or numeric variables. The effective range varies for each state variable. M_HGDev: 0, 1 M_HGDevW: -32768 to 32767 (&H8000 to &H7FFF) M_HGDevD: -2147483648 to 2147483647 (&H80000000 to &H7FFFFFFF)

[Reference Program]

```
1 M_HGDev(&H3E1,30,2)=1          'Sets the value of bit number 2 in buffer memory
                                  address 30 of CPU No. 2 to 1.
2 M_HGDevW(&H3E1,10)=&HFFFF      'Sets buffer memory address 10 of CPU No. 2 to FFFF.
3 M_HGDevD(&H3E1,11)=P1.X * 1000 'Multiplies the X coordinate value of position variable P1
                                  by 1000 and sets buffer memory addresses 11 and 12
                                  (2 words) of CPU No. 2 with this value.
4 M1%=M_HGDevW(&H3E2,1) And &H7  'Sets M1 to the values of the lower 3 bits in buffer
                                  memory address 1 of CPU No. 3.
```

[Explanation]

(1) This variable is used to write to or reference the periodical communication area of PLC buffer memory.
(2) Buffer memory addresses are specified using start I/O numbers and the actual buffer memory addresses.
(3) Both write and reference data strings are integers.
(4) Variable M_HGDevW uses 1 word, or 16 bits, of data in the specified buffer memory address. Variable M_HGDevD uses 2 words, or 32 bits, of data.
(5) Specify start I/O numbers in hexadecimal format. The range of numbers is between &H3E0 to &H3E3, which is equivalent to 992 to 995 in decimal. Specify buffer memory addresses in decimal numbers within a range of addresses between 0 to 12287.
(6) Write operations can only be performed for buffer memory addresses of the target CPU. Attempting to write data to addresses for other CPUs will not result in a successful change to the data stored by the address.
(7) Only CR800-R series devices are supported. (Statements executed using CR800-D/CR800-Q series devices result in unsuccessful write operations, and the reference returns zero value.)
(8) To prevent 32-bit data strings from being fragmented with M_HGDevD, specify buffer memory addresses starting with the even-numbered address first.

[Supplementary]

Table 4-24:<Numeric variable>

O: The available, X: unavailable

Bit width	Numeric variables types				Other variables		
	Integer Ex.)M1%	Long-precision integer number Ex.)M1%	Single-precision real number Ex.)M1&	Double-precision real number Ex.)M1	Position Ex.)P1.X	Joint Ex.)J1.J1	Character string Ex.)C1$
M_HGDev	O	O	O	O	O	O	X
M_HGDevW	O	O	O	O	O	O	X
M_HGDevD	X	O	O	O	O	O	X

Table 4-25:<Constants and variables usable as <Start I/O number>, <Buffer memory address>, and <Bit number>

O: The available, X: unavailable

Bit width	constant types			Numeric variables types				Other variables		
	Numeric value[Note1)] Ex.)12	Binary number Ex.)&B1100	Hexadecimal number Ex.)&HC	Integer Ex.)M1%	Long-precision integer number Ex.)M1%	Single-precision real number[Note1)] Ex.)M1&	Double-precision real number[Note1)] Ex.)M1	Position[Note1)] Ex.)P1.X	Joint[Note1)] Ex.)J1.J1	Character string Ex.)C1$
M_HGDev	O	O	O	O	O	O	O	O	O	X
M_HGDevW	O	O	O	O	O	O	O	O	O	X
M_HGDevD	O	O	O	O	O	O	O	O	O	X

Note1) The real value is rounded off.

Table 4-26:<Numeric value>

O: The available, X: unavailable

Bit width	constant types			Numeric variables types				Other variables		
	Numeric value Note1) Ex.)12	Binary number Ex.)&B1100	Hexadecimal number Ex.)&HC	Integer Ex.)M1%	Long-precision integer number Ex.)M1%	Single-precision real number Ex.)M1&	Double-precision real number Ex.)M1	Position Note1) Ex.)P1.X	Joint Note1) Ex.)J1.J1	Character string Ex.)C1$
M_HGDev	O	O	O	O	X	X	X	X	X	X
M_HGDevW	O	O	O	O	X	X	X	X	X	X
M_HGDevD	O Note2)	O Note2)	O Note2)	O Note2)	O	O	O Note3)	O	O	X

Note1) The real value is rounded off.

Note2) ⚠CAUTION For the numerical value of the less than 16 bits of the binary number (-32768 to +32767), the specified constant will handle as a negative numerical value, if the bit 15 (the 16th bit) turns on. Therefore, please be careful of turning on all of upper 16 bits. (The sign bit is extended)

Example)

Designation of "-32768(&B1000000000000000)" will output the "&B11111111111111111000000000000000."

[Measures]

After substituting the constant for the long-precision integer number variable as follows, when substituting to this robot status variable M_HGDevD, &B00000000000000001000000000000000 (binary number) can be outputted.

```
1 M1&=-32768
2 M_HGDevD(&H20)=M1&
```

Note3) The ranges of the numerical value which can be outputted are -2147483648 to 2147483647.

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_In8/M_Inw/M_In16

[Function]

Returns the value of the input signal.
M_In: Returns a bit.
M_Inb or M_In8: Returns a byte (8 bits).
M_Inw or M_In16: Returns a word (16 bits).

[Format]

Example) <Numeric Variable>=M_In(<Equation>)

Example) <Numeric Variable>=M_Inb(<Equation>) or M_In8(<Equation>)

Example) <Numeric Variable>=M_Inw(<Equation>) or M_In16(<Equation>)

[Terminology]

<Numeric Variable> Specifies the numerical variable to assign. Supplementary explanation is shown in Table 4-27.

<Equation> Enter the input signal number. Supplementary explanation is shown in Table 4-28.

(1)CR800-R/Q series
10000 to 18191: Multi-CPU share device
716 to 731: Multi-hand input.
900 to 907: Hand input.

(2)CR800-D series
0 to 255: Standard remote inputs.
716 to 731: Multi-hand input.
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/M_In8 and M_Inw/M_In16 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.

⚠CAUTION Always make interlock of signal to take synchronization. Failure to observe this could lead to cause of malfunction by the signal transmitted incorrectly.

[Supplement]

Table 4-27:<Numeric Variable>

O: The available, X: unavailable

Bit width	Numeric variables types				Other variables		
	Integer Ex.)M1%	Long-precision integer number Ex.)M1&	Single-precision real number Ex.)M1!	Double-precision real number Ex.)M1#	Position Note1) Ex.)P1.X	Joint Note1) Ex.)J1.J1	Character string Ex.)C1$
M_In	O	O	O	O	O	O	X
M_Inb/M_In8	O	O	O	O	O	O	X
M_Inw/M_In16	O	O	O	O	O	O	X

Note1) The unit is the radian if the value of variable is the angle. (The elements of A, B and C of position variable, and all elements of joint variable) The display of the monitor etc. is converted into the degree.
Example) If the input signal 8 is ON in P1.A=M_In (8), P1.A is displayed as "57.296." Therefore, ON/OFF status does not look clear. Because the unit of the element X, Y, and Z of the position variable is "mm", there is no such condition.

Table 4-28:<Equation>

O: The available, X: unavailable

Bit width	constant types			Numeric variables types				Other variables		
	Numeric value Note1) Ex.)12	Binary number Ex.)&B1100	Hexadecimal number Ex.)&HC	Integer Ex.)M1%	Long-precision integer number Ex.)M1&	Single-precision real number Note1) Ex.)M1!	Double-precision real numbe Note1) Ex.)M1#	Positio Note1) Note2) Ex.)P1.X	Joint Note1) Note2) Ex.)J1.J1	Character string Ex.)C1$
M_In	O	O	O	O	O	O	O	O	O	X
M_Inb/M_In8	O	O	O	O	O	O	O	O	O	X
M_Inw/M_In16	O	O	O	O	O	O	O	O	O	X

Note1) The real value is rounded off.

Note2) If the value of the variable is the angle, the unit will be processed by the radian. (The elements of A, B and C of position variable, and all elements of joint variable) Therefore, designation of the signal number is very difficult. The display of the monitor etc. is converted into the degree, and the same value as the setting value displayed.
Example) It is processed by value "0", even if it sets "8" as the value of P1.A (The input in the key by T/B etc.) to specify the input signal No.8. The result is "0" when 8 degree is converted to radian (0.14) and rounded off.Because the unit of the element X, Y, and Z of the position variable is "mm", there is no such condition.

[Reference status variable]

M_Out32

[Related instructions]

Def IO (Define IO)

M_In32

[Function]

Returns the value of the input signal of 32-bit width as a value.

[Format]

Example) <Numeric Variable>=M_In32(<Equation>)

[Terminology]

<Numeric Variable> Specifies the numerical variable to assign. Supplementary explanation is shown in Table 4-29.

<Equation> Enter the input signal number. Supplementary explanation is shown in Table 4-30.

(1)CR800-R/Q series

10000 to 18191: Multi-CPU share device

716 to 731: Multi-hand input.

900 to 907: Hand input.

(2)CR800-D series

0 to 255: Standard remote inputs.

716 to 731: Multi-hand input.

900 to 907: Hand input.

2000 to 5071: Input signal of PROFIBUS.

6000 to 8047: Remote input for CC-Link.

[Reference Program]

```
1 *ack_wait
2 If M_In(7)=0 Then *ack_wait      'Wait until the input signal 7 turns on (interlock).
3 M1&=M_In32(10000)                'Store the data of 32-bit width to the long precision numeric variable
                                     M1 as a value from the input signal 10000.
4 P1.Y=M_In32(10100)/1000.0        'The data of 32-bit width is inputted as a value from the input signal
                                     10100, divide by 1000, and store it to Y element of position variable
                                     P1.
                                     (The example that at the external equipment side, the real number
                                     multiplied by 1000 and converted into integer)
```

[Explanation]

(1) Return the input-signal data of 32-bit width as a numerical value.

(2) 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.

(3) Specify the long precision integer type or the real-number type variable as the <Numeric Variable>.

(4) This variable only reads the data.

⚠CAUTION Always make interlock of signal to take synchronization. Failure to observe this could lead to cause of malfunction by the signal transmitted incorrectly.

[Supplement]

Table 4-29:<Numeric Variable>

O: The available, X: unavailable

Bit width	Numeric variables types				Other variables		
	Integer Ex.)M1%	Long-precision integer number Ex.)M1&	Single-precision real number Ex.)M1!	Double-precision real number Ex.)M1#	Position Note1) Ex.)P1.X	Joint Note1) Ex.)J1.J1	Character string Ex.)C1$
M_In32	X	O	O	O	O	O	X

Note1) The unit is the radian if the value of variable is the angle. The 32-bit input signal is stored without converting as a numerical value of the radian unit. (The elements of A, B and C of position variable, and all elements of joint variable) The display of the monitor etc. is converted into the degree.
Example) If the input data is 5 (decimal number) in P1.A=M_In32(16), P1.A is displayed as "286.479" (deg).

Table 4-30:<Equation>

O: The available, X: unavailable

Bit width	constant types			Numeric variables types				Other variables		
	Numeric value Note1) Ex.)12	Binary number Ex.)&B1100	Hexadecimal number Ex.)&HC	Integer Ex.)M1%	Long-precision integer number Ex.)M1&	Single-precision real number Note1) Ex.)M1!	Double-precision real number Note1) Ex.)M1#	Position Note1) Note2) Ex.)P1.X	Joint Note1) Note2) Ex.)J1.J1	Character string Ex.)C1$
M_In32	O	O	O	O	O	O	O	O	O	X

Note1) The real value is rounded off.
Note2) If the value of the variable is the angle, the unit will be processed by the radian. (The elements of A, B and C of position variable, and all elements of joint variable) Therefore, designation of the signal number is very difficult. The display of the monitor etc. is converted into the degree, and the same value as the setting value displayed.
Example) It is processed by value "0", even if it sets "8" as the value of P1.A (The input in the key by T/B etc.) to specify the input signal No.8. The result is "0" when 8 degree is converted to radian (0.14) and rounded off.Because the unit of the element X, Y, and Z of the position variable is "mm", there is no such condition.

[Reference status variable]

M_In/M_Inb/M_In8/M_Inw/M_In16

[Related instructions]

Def IO (Define IO)

M_JOvrd/M_NJOvrd/M_OPOvrd/M_Ovrd/M_NOvrd

[Function]

Returns override value.
M_JOvrd: Value specified by the override JOvrd command 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 command.
M_NOvrd: Initial override value (100%).

[Format]

Example)<Numeric Variable>=M_JOvrd [(<Equation>)]

Example)<Numeric Variable>=M_NJOvrd[(<Equation>)]

Example)<Numeric Variable>=M_OPOvrd

Example)M_OPOvrd=<Equation>

Example)<Numeric Variable>=M_Ovrd[(<Equation>)]

Example)<Numeric Variable>=M_NOvrd[(<Equation>)]

[Terminology]

<Numeric Variable>	Specifies the numerical variable to assign.
<Equation>	In M_JOvrd/M_NJOvrd/M_Ovrd/M_NOvrd, 1 to 32, enter the task slot number. If these parameters are omitted, the current slot will be used as the default. In M_OPOvrd, enter an override value.

[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.
7 M_OPOvrd=20        ' The OP (operation panel) override value is set in 20.
```

[Explanation]

(1) If the argument is omitted, the current slot status will be returned.
(2) In M_JOvrd/M_NJOvrd/M_Ovrd/M_NOvrd, only read the data.
In M_OPOvrd, writes/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 100%.
2 *Label
3 Mov P1
4 Mov P2
5 If M_LdFact(2)>90 Then
6 Accel 50,50                ' Lower the acceleration/deceleration ratio to 50%.
7 M_SetAdl(2)=50             ' Furthermore, lower the acceleration/deceleration ratio of the J2 axis to
                               50%. (In actuality, 50% x 50% = 25%)
8 Else
9 Accel 100,100              ' Return the acceleration/deceleration time.
10 EndIf
11 GoTo *Label
```

[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 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.
(6) The initial value of the target mechanism number is "1". Therefore, when mechanism number 1 is targeted, after executing the RelM command, or the program slot is other than 1, execution of the GetM command is unnecessary. If target mechanism is other than 1, execute the GetM command beforehand.

[Related instructions]
Accel (Accelerate), Ovrd (Override)

M_LdFMax

[Function]
The maximum load ratio for each joint axis can be referenced.

[Format]

Example)<Numeric Variable>=M_LdFMax(<Axis Number>)

[Terminology]
<Axis Number> 1 to 8, Specifies the axis number.
<Numeric Variable> The load ratio of each axis is substituted. The range is 0 to 100%.

[Reference Program]

```
1 Accel 100,100                    ' Lower the overall deceleration time to 100%.
2 *Label
3 Mov P1
4 Mov P2
5 If M_LdFMax(2)>90 Then Goto *LdErr Else Goto *Label
6
7 *LdErr
8 Error 9101                       ' User error output
9 End
```

[Explanation]
(1) The maximum load ratio of each axis can be referenced. The severity of a robot can be checked.
(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 alarm 9101 is generated.
(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.
(6) The initial value of the target mechanism number is "1". Therefore, when mechanism number 1 is targeted, after executing the RelM command, or the program slot is other than 1, execution of the GetM command is unnecessary. If target mechanism is other than 1, execute the GetM command beforehand.

[Related instructions]
Accel (Accelerate), Ovrd (Override)

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_Map1 to M_Map8

[Function]
This command returns the segment number in which a workpiece is present in the condition defined in the Def Map command, using the get-position-quick function (GPS function).
The information of the workpiece presence recognition obtained by the execution of the mapping (the recording of the workpiece presence in a segment to which a sensor attached on the hand reacts by sliding the hand vertically at the front of a cassette) can be checked.

[Format]

Example) <Numeric Variable>=M_Map1[(<Numeric value>)]

[Terminology]

<Numeric Variable>	Specifies the numerical variable to assign. The information of the workpiece presence recognition in the segment set in <Numeric value> is returned to the position variable. The default value is set to 0. "1" is stored for the segment in which the workpiece is present. The value is cleared to zero when the GpsChk On command is executed.
<Numeric value>	Set the segment number to check for the workpiece presence in a cassette. The lowest segment in a cassette is defined as the first segment. Setting range: 1 to 130 (segments) (A setting value of 0 or a negative value is regarded as 1.)

[Reference Program]

```
1  Def Map 3,851,On,1,PC1,PC2,20,10
                              ' The position data of the mechanism No. 1 is recorded for the
                                monitoring No. 3 when the signal No. 851 is turned on, and the
                                mapping is executed according to the defined condition.
                                PC1: Lowest position (first segment) in a cassette, PC2: Highest
                                position (last segment) in a cassette, 20: the number of segments in a
                                cassette (20 segments), 10: a sensitive area of a sensor (10 mm)
2  Mov PM1                    ' The robot moves its arm to the mapping start position.
```

```
3  GpsChk On,3                  ' Monitoring a condition for the monitoring No. 3 is started. The position
                                  data is recorded when the signal No. 851 is turned on, and the
                                  segment number in which the workpiece is present is calculated from
                                  that position data, which is shown as "M_Map3(130)".
4  Mvs PM2                      ' The robot moves its arm to the mapping stop position.
5  GpsChk Off,3                 ' Monitoring a condition for the monitoring No. 3 is stopped.
6  M1=M_Gps(3)                  ' The number of the position data recorded in the P_Gps3 is stored.
7  If M1=0 Then Error 9000      ' The error 9000 is generated if no position data is recorded.
8  For M2=1 To 20
9  M_Out(6100+M2) = M_Map3(M2)
                                ' The results of the mapping are output with a signal (the signal number
                                  "6101" is assigned for the first result).
10 Next M2                      ' The process performed for the first segment is repeated for the
                                  remaining segments.
```

[Explanation]

(1) The current position data of the robot is stored in the P_GpsX() when the condition defined in the Def Map command is met. The segment number in which the workpiece is present is calculated from that position data and the condition defined in the Def Map command, and stored in this status variable.
(2) To send the data of workpiece position to an external device, output the signal including the information stored in this status variable.
(3) The value is cleared to zero when the GpsChk On command is executed.
(4) The number of times can be confirmed after the GpsChk Off command is executed.
(5) This variable only reads the data.
(6) The number of times is cleared to zero when the controller is turned on.

[Related instructions]

Def Map (Define mapping), GpsChk (Get position check)

M_Mode

[Function]

Returns the key switch mode of the operation panel.
1: MANUAL
2: AUTOMATIC (O/P)
3: AUTOMATIC (External)

[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_Mxt

[Function]

Use this command for synchronization of the robot applications between the master and slave robots for cooperative operation.

[Format]

```
Example)<Numeric Variable>=M_Mxt(<Target CPU No.>)
```

[Terminology]

<Numeric Variable>	Specifies the numerical variable to assign.
<Target CPU No.>	Enter the CPU module No. of the master robot. 2 to 4.

[Reference Program]

```
M_Mxt(2)=1          ' The robot CPU No. 2 starts cooperation.
Wait M_Mxt(2)=1     ' Wait for the robot CPU No. 2 to start cooperation.
```

[Explanation]

(1) This can be used for synchronization of the applications with the cooperative operation function.
(2) In the target CPU, this variable reads and writes the data. In other CPU, this variable only reads the data.
(3) If the status variable is changed from 1 to 0 at the master robot during the cooperative operation, the slave robot finishes Mxt command automatically.
(4) The initial value after power ON is 0.

M_NvOpen

[Function]
Indicates the vision sensor line connection status.

[Array meaning]
Array elements (1 - 8): Vision sensor numbers

[Explanation of values returned]
0: Line connecting (logon not complete) 1: Logon complete -1: Not connected

[Usage]
After an NVOpen command is executed, checks whether or not the line with the vision sensor is connected and the vision sensor logged onto.

[Reference Program]

```
1 If M_NVOpen(1)<>1 Then      ' If vision sensor number 1 is not connected
2   NVOpen "COM2:" As #1      ' Connects with the vision sensor connected to COM2 and sets its
                                number as number 1.
3 EndIf
4 Wait M_NVOpen(1)=1          ' Connects with vision sensor number 1 and waits for the logon state.
5 ・・・・・
  :
100 NVClose #1                ' Cuts the line with the vision sensor connected to COM2.
```

[Explanation]
(1) Indicates the status of a line connected with a network vision sensor with an NVOpen command when the line is opened.
(2) The initial value is "-1". At the point in time that the NVOpen command is executed and the line is connected, the value becomes "0" (line connecting). At the point in time that the network vision sensor logon is completed, the value becomes "1" (logon complete).
(3) This variable strongly resembles the status of status variable M_OPEN, but whereas M_Open
(4) becomes "1" when the connection is verified, M_NVOpen becomes "1" when the vision sensor logon is complete.

[Errors]
(1) If the type of data specified as an array element is incorrect, a "syntax error in input command statement" error occurs.
(2) If there is an abnormal number of array elements (too many or too few), an "incorrect argument type" error occurs.
(3) If an array element other than "1" through "8" is specified, an **"array element mistake"** error occurs.

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 or communication line is opened.

[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 "temp.txt" For Append As #1          ' Open "temp.txt" as the file number 1.
2 *LBL:If M_Open(1)<>1 Then GoTo *LBL       ' Wait until the file number 1 opens.
```

[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 Ethernet	Indicates whether or not connection is made with the counterpart.	For server setting	1: Client is already connected. 0: Client is not connected. -1: The file number is undefined (not opened).
		For client setting	1: Already connected to the server. (Connection has been made.) 0: Not connected to the server. (Connection has not been made. Equivalent to when the server is down after being opened.) -1: The file number is undefined. (When the file has not been opened, or has been opened while the server is down.)

[Related instructions]
Open (Open)

[Related parameter]
COMDEV, CPRE**, NETMODE

M_Out/M_Outb/M_Out8/M_Outw/M_Out16

[Function]

Writes or references external output signal.
M_Out:Output signal bit.
M_Outb or M_Out8:Output signal byte (8 bits).
M_Outw or M_Out16:Output signal word (16 bits).

[Format]

Example)M_Out(<Numeric value 1>)=<Numeric value 2>

Example)M_Outb(<Numeric value 1>) or M_Out8(<Numeric value 1>)=<Numeric value 3>

Example)M_Outw(<Numeric value 1>) or M_Out16(<Numeric value 1>)=<Numeric value 4>

Example)M_Out(<Numeric value 1>)=<Numeric value 2> Dly <Time>

Example)<Numeric Variable>=M_Out(<Numeric value 1>)

[Terminology]

<Numeric value 1> Specify the output signal number. Supplementary explanation is shown in Table 4-31.
(1)CR800-R/Q series
10000 to 18191: Multi-CPU share device
716 to 723: Multi-hand output.
900 to 907: Hand output.
(2)CR800-D series
0 to 255: Standard remote outputs.
716 to 723: Multi-hand output.
900 to 907: Hand output.
2000 to 5071: Input signal of PROFIBUS.
6000 to 8047: Remote input for CC-Link.

<Numeric Variable> Specifies the numerical variable to assign.

<Numeric value 2>, <Numeric value 3>, <Numeric value 4>
Describe the value to output by the numeric variable, the constant, or numerical arithmetic expression.Supplementary explanation is shown in Table 4-32.
Numerical range
<Numeric value 2>: 0 or 1 (&H0 or &H1)
<Numeric value 3>: -128 to +127 (&H80 to &H7F)
<Numeric value 4>: -32768 to +32767 (&H8000 to &H7FFF)

<Time> Describe the output time for the pulse output as a constant or numeric operation expression. Unit: [Seconds]

<Numeric Variable> Specifies the numerical variable to assign. Supplementary explanation is shown in Table 4-33.

[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 6000 and beyond will be referenced/assigned to the CC-Link (optional).
(3) Refer to Page 222, "Dly (Delay)" for the explanation of pulse output.
(4) By high-speed mode setting of parameter: SYNCIO, the updating cycle to the external output signal can be made speedy. However, always make interlock of signal to take synchronization. Because to make the timing of the I/O signal correct. Refer to SYNCIO in Page 502, "SYNCIO"

⚠CAUTION Always make interlock of signal to take synchronization. Failure to observe this could lead to cause of malfunction by the signal transmitted incorrectly.

[Supplement]

Table 4-31:<Numeric value 1>

O: The available, X: unavailable

Bit width	constant types			Numeric variables types				Other variables		
	Numeric value Note1) Ex.)12	Binary number Ex.)&B1100	Hexadecimal number Ex.)&HC	Integer Ex.)M1%	Long-precision integer number Ex.)M1&	Single-precision real number Note1) Ex.)M1!	Double-precision real numbe Note1) Ex.)M1#	Positio Note1) Note2) Ex.)P1.X	Joint Note1) Note2) Ex.)J1.J1	Character string Ex.)C1$
M_Out	O	O	O	O	O	O	O	O	X	X
M_Outb/M_Out8	O	O	O	O	O	O	O	O	X	X
M_Outw/M_Out16	O	O	O	O	O	O	O	O	X	X

Note1) The real value is rounded off.

Note2) If the value of the variable is the angle, the unit will be processed by the radian. (The elements of A, B and C of position variable, and all elements of joint variable) Therefore, designation of the signal number is very difficult. The display of the monitor etc. is converted into the degree, and the same value as the setting value displayed.

Example) It is processed by value "0", even if it sets "8" as the value of P1.A (The input in the key by T/B etc.) to specify the input signal No.8. The result is "0" when 8 degree is converted to radian (0.14) and rounded off. Because the unit of the element X, Y, and Z of the position variable is "mm", there is no such condition.

Table 4-32:<Numeric value 2>, <Numeric value 3>, <Numeric value 4>

O: The available, X: unavailable

Bit width	constant types			Numeric variables types				Other variables		
	Numeric value Ex.)12	Binary number Ex.)&B1100	Hexadecimal number Ex.)&HC	Integer Ex.)M1%	Long-precision integer number Ex.)M1&	Single-precision real number Ex.)M1!	Double-precision real numbe Ex.)M1#	Positio Ex.)P1.X	Joint Ex.)J1.J1	Character string Ex.)C1$
M_Out	O	O	O	O	X	X	X	X	X	X
M_Outb/M_Out8	O	O	O	O	X	X	X	X	X	X
M_Outw/M_Out16	O	O	O	O	X	X	X	X	X	X

Table 4-33:<Numeric value>

O: The available, X: unavailable

Bit width	Numeric variables types				Other variables		
	Integer Ex.)M1%	Long-precision integer number Ex.)M1&	Single-precision real number Ex.)M1!	Double-precision real numbe Ex.)M1#	Positio Note1) Ex.)P1.X	Joint Note1) Ex.)J1.J1	Character string Ex.)C1$
M_Out	O	O	O	O	O	O	X
M_Outb/M_Out8	O	O	O	O	O	O	X
M_Outw/M_Out16	O	O	O	O	O	O	X

Note1) If the value of the variable is the angle, the unit will be processed by the radian. (The elements of A, B and C of position variable, and all elements of joint variable) The display of the monitor etc. is converted into the degree and displayed

[Related status variable]

M_Out32

[Related instructions]

Def IO (Define IO)

[Related parameter]

SYNCIO

M_Out32

[Function]

Writes or references external output signal of 32-bit width as numerical value.

[Format]

Example)M_Out32(<Numeric value 1>)=<Numeric value>

Example)<Numeric Variable>=M_Out32(<Numeric value 1>)

[Terminology]

<Numeric value 1> Specify the output signal number. Supplementary explanation is shown in Table 4-34.
(1)CR800-R/Q series
10000 to 18191: Multi-CPU share device
716 to 723: Multi-hand output.
900 to 907: Hand output.
(2)CR800-D series
0 to 255: Standard remote outputs.
716 to 723: Multi-hand output.
900 to 907: Hand output.
2000 to 5071: Input signal of PROFIBUS.
6000 to 8047: Remote input for CC-Link.

<Numeric value> Describe the value to output by the numeric variable, the constant, or numerical arithmetic expression. Supplementary explanation is shown in Table 4-35.
Numerical range: -2147483648 to +2147483647 (&H80000000 to &H7FFFFFFF)

<Numeric Variable> Specifies the numerical variable to assign. Supplementary explanation is shown in Table 4-36.

[Reference Program]

```
1 M_Out32(10000)=P1.X * 1000    'Multiply X coordinate value of the P1 by 1000, and write to 32-bit
                                  width from the output signal number 10000.(Integer)
2 *ack_wait
3 If M_In(7)=0 Then *ack_wait   'Wait until the input signal 7 turns on (interlock).
4 P1.Y=M_In32(10100)/1000.0     'The data of 32-bit width is inputted as a value from the input signal
                                  10100, divide by 1000, and store it to Y element of position variable
                                  P1.
                                  (The example that at the external equipment side, the real number
                                  multiplied by 1000 and converted into integer)
```

[Explanation]

(1) This is used when writing or referencing external output signal of 32-bit width as numerical value.
(2) The data is outputted to 32-bit width from the specified signal number.
(3) By high-speed mode setting of parameter: SYNCIO, the updating cycle to the external output signal can be made speedy. However, always make interlock of signal to take synchronization. Because to make the timing of the I/O signal correct. Refer to SYNCIO in Page 499, "5.2 Signal parameter"

⚠CAUTION Always make interlock of signal to take synchronization. Failure to observe this could lead to cause of malfunction by the signal transmitted incorrectly.

[Supplement]

Table 4-34:<Numeric value 1>

O: The available, X: unavailable

Bit width	constant types			Numeric variables types				Other variables		
	Numeric value Note1) Ex.)12	Binary number Ex.)&B1100	Hexadecimal number Ex.)&HC	Integer Ex.)M1%	Long-precision integer number Ex.)M1&	Single-precision real number Note1) Ex.)M1!	Double-precision real numbe Note1) Ex.)M1#	Position Note1) Note2) Ex.)P1.X	Joint Note1) Note2) Ex.)J1.J1	Character string Ex.)C1$
M_Out32	O	O	O	O	O	O	O	O	X	X

Note1) The real value is rounded off.

Note2) If the value of the variable is the angle, the unit will be processed by the radian. (The elements of A, B and C of position variable, and all elements of joint variable) Therefore, designation of the signal number is very difficult. The display of the monitor etc. is converted into the degree, and the same value as the setting value displayed.
Example) It is processed by value "0", even if it sets "8" as the value of P1.A (The input in the key by T/B etc.) to specify the input signal No.8. The result is "0" when 8 degree is converted to radian (0.14) and rounded off. Because the unit of the element X, Y, and Z of the position variable is "mm", there is no such condition.

Table 4-35:<Numeric value>

O: The available, X: unavailable

Bit width	constant types			Numeric variables types				Other variables		
	Numeric value Ex.)12	Binary number Ex.)&B1100	Hexadecimal number Ex.)&HC	Integer Ex.)M1%	Long-precision integer number Ex.)M1&	Single-precision real number Note1) Ex.)M1!	Double-precision real numbe Note1) Ex.)M1#	Position Ex.)P1.X	Joint Ex.)J1.J1	Character string Ex.)C1$
M_Out32	O Note1)	O Note1)	O Note1)	O Note1)	O	O	O Note2)	O	O	X

Note1) CAUTION

For the numerical value of the less than 16 bits of the binary number (-32768 to +32767), the specified constant will handle as a negative numerical value, if the bit 15 (the 16th bit) turns on. Therefore, please be careful of turning on all of upper 16 bits. (The sign bit is extended)
Example)
Designation of "-32768(&B1000000000000000)" will output the "&B11111111111111111000000000000000."
[Measures]
After substituting the constant for the long-precision integer number variable as follows, when substituting to this robot status variable M_YDevD, &B00000000000000001000000000000000 (binary number) can be outputted.
1 M1&=32768
2 M_YDevD(&H20)=M1&

Note2) The ranges of the numerical value which can be outputted are -2147483648 to 2147483647.

Table 4-36:<Numeric value>

O: The available, X: unavailable

Bit width	Numeric variables types				Other variables		
	Integer Ex.)M1%	Long-precision integer number Ex.)M1&	Single-precision real number Note1) Ex.)M1!	Double-precision real numbe Note1) Ex.)M1#	Positio Note1) Ex.)P1.X	Joint Note1) Ex.)J1.J1	Character string Ex.)C1$
M_Out32	X	O	O	O	O	O	X

Note1) The unit is the radian if the value of variable is the angle. The 32-bit current output value is stored without converting as a numerical value of the radian unit. (The elements of A, B and C of position variable, and all elements of joint variable) The display of the monitor etc. is converted into the degree.

[Related status variable]

M_Out/M_Outb/M_Out8/M_Outw/M_Out16

[Related instructions]

Def IO (Define IO)

[Related parameter]

SYNCIO

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_RCInfo

[Function]
This variable returns a variety of information of the robot controller.

[Format]

Example)<Numeric Variable>=M_RCInfo (<Information ID number>, <Element number>)

[Terminology]

<Numeric Variable> Specifies the numerical variable to assign.
<Information ID number> Specifies the information ID number of the information to read. Refer to the table below for details.
<Element number> Specifies the element number corresponding to the information to read. Refer to the table below for details.

Information ID number		Element number	
Setting value	Details of information	Setting value (Setting range)	Meaning
1	Temperature of each axis encoder (°C)	1 to 8	Indicates the axial number of the robot. * As for the axis not existing or the additional axis (7 and 8 axes), zero value will be returned.
2	Maximum temperature of each axis encoder (°C)	1 to 8	
3	Temperature of inside the controller (°C)	1	-

[Reference Program]

```
1 Dim METMP(8)                  ' Variable to which the temperature of the each axis
                                  encoder is assigned.
2 Dim METMPX(8)                 ' Variable to which the maximum temperature of the each
                                  axis encoder is assigned.
3 For MT=1 To 8
4   METMP(MT)=M_RCInfo(1,MT)    ' Read the temperature of the each axis encoder.
5   METMPX(MT)=M_RCInfo(2,MT)   ' Read the maximum temperature of the each axis encoder.
6 Next
7 MRCTMP=M_RCInfo(3,1)          ' Read the temperature of inside the controller.
```

[Explanation]

(1) If the value exceeding the setting value is specified as information ID number and element number, the error L3110 will occur when executed.

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 uniformity, 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.

[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        ' If the load rate of the J2 axis exceeds 90%,
3   M_SetAdl(2)=70              ' set the acceleration/deceleration time distribution rate of the J2
                                  axis to 70%.
4 EndIf                         ' Acceleration 70% (= 100% x 70%), deceleration 35% (= 50% x
                                  70%)
5 Mov P1
6 Mov P2
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

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 *Lskip        ' If Skip instruction has been executed, jump to line 1000.
 ;
10 *Lskip
```

[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. When the servo-off condition, the feedback speed is returned. At this time, even if the robot is stopping, the value may change between - 0.01, to +0.01
(2) This can be used in M_RSpd multitask programs or with Wth and WthIf statements.
(3) This variable only reads the data.

M_SplPno

[Function]

During spline interpolation, the number of the path point passed through most recently is returned.
The spline interpolation start position is 1.

[Format]

Example) <Numeric Variable>=M_SplPno [(<Mechanism Number>)]

[Terminology]

<Numeric Variable>	Designates the numerical variable substituted for the reference results.
<Mechanism Number>	Sets the mechanism No. that executes spline interpolation. Setting range: 1 to 3 When omitted: 1 If a non-existent mechanism No. is designated, the error L3870 (designated mechanism No. invalid) will occur when execution is started.

[Reference Program]

```
1 Wait M_00=1                          ' Wait for spline interpolation to start
2 *L1:If M_SplPno < 5 Then GoTo *L1    ' Wait for path point 5 to be passed
3 M1=M_Inw(100)                        ' After passing through path point 5, get input signal 100 to 115
                                         status
4 *L2:If M_SplPno < 10 Then GoTo *L2   ' Wait for path point 10 to be passed
5 M_01=M1                              ' After passing through path point 10, substitute M1 details in
                                         M_01
```

[Explanation]

(1) During spline interpolation, the number of the path point passed through most recently is returned.
(2) By referring to the M_SplPno value with multi-task, the operation process or signal output, etc., can be executed according to the progress of spline interpolation.
(3) M_SplPno returns the value indicated in the following table according to the state.

Table 4-37:Value returned by M_SplPno

Status	Value returned by M_SplPno
Immediately after power ON	0
During spline interpolation execution	1 to (value corresponding to progress of spline interpolation)
After spline interpolation	Path point No. for end position of spline interpolation
Immediately after main program's End command is executed	0
Immediately after program reset operation	0
Mechanism not compatible with spline interpolation	0

(4) M_SplPno is a read-only status variable.

M_SplVar

[Function]

During spline interpolation, the numerical setting value set for the path point passed through most recently is returned. This value can be changed to a random value by using writing.

[Format]

Example) <Numeric Variable 1>=M_SplVar [(<Mechanism Number>)]

Example) M_SplVar [(<Mechanism Number>)]=<Numeric Variable 2>

[Terminology]

<Numeric Variable 1>	Designates the numerical variable substituted for the reference results.
<Mechanism Number>	Sets the mechanism No. that executes spline interpolation. Setting range: 1 to 3 When omitted: 1 If a non-existent mechanism No. is designated, the error L3870 (designated mechanism No. invalid) will occur when execution is started.
<Numeric Variable 2>	Designates the value set in M_SplVar. Setting range: 0 to 32767 If a value exceeding the range is set, the error L2615 (M_SplVar setting value outside range) or error L3110 (argument value outside range) will occur when execution is started.

[Reference Program]

```
1 Wait M_00=1                          ' Wait for spline interpolation to start
2 *L1
3 Select M_SplVar
4   Case 1                             ' Pass through path point for which M_SplVar value is set to 1
5     M_SplVar=0                       ' Reset M_SplVar value to 0
6     M_Out(100)=1                     ' Turn output signal 100 ON
7     Break
8   Default                            ' M_SplVar value is not 1
9     M_Out(100)=0                     ' Turn output signal 100 OFF
10    Break
11 End Select
12 If M_00=1 Then Goto *L1             ' Repeat until spline interpolation finishes
```

[Explanation]

(1) During spline interpolation, the numerical setting value set for the path point passed through most recently is returned. The numerical setting is designated in the spline file's path point data.

(2) By referring to M_SplVar value multi-task, the operation process or signal output process, etc., can be executed according to the progress of spline interpolation. This differs from M_SplPro in that the same value can be returned even if the path point is different. Thus, the program can be simplified if the same output signal is repeatedly turned ON/OFF as shown in [Reference Program].

(3) M_SplVar returns an integer between 0 and 32767. If the numerical setting for the path point is set to "-1" in the spline file, the M_SplVar value will not change even if that path point is passed through. The value at the time of path point passage will be applied.

(4) M_SplVar returns the value according to the states shown in Table 4-38.

Table 4-38: Value returned by M_SplVar

Status	Value returned by M_SplVar
Immediately after power ON	0
During spline interpolation execution	Value corresponding to progress of spline interpolation (Value set in spline file)
When passing through path point set to "–1"	Value at that time is continuously applied
After spline interpolation	Value at end of spline interpolation
Immediately after main program's END command is executed	0
Immediately after program reset operation	0
Immediately after value is read into M_SplVar	Written value
Mechanism not compatible with spline interpolation	0 (0 even after write is executed)

(5) A random value between 0 and 32767 can be set by writing to M_SplVar. That value is held until the value is changed with spline interpolation or other operations/processes.

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 (ms) required to move from the current
                       position to P1, and from P1 to P2.
                       Example) If the time is 5.346 sec. the value of M1 is 5346.
5 M_Timer(1)=1.5     ' 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 conversion data (MEXTL1 to 16) of the specified number as the current tool conversion 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 16 from 1.

[Reference Program]

Setting Tool Conversion Data

```
1 Tool (0,0,100,0,0,0)          ' Specify tool conversion data (0,0,100,0,0,0), and write it into MEXTL.
2 Mov P1
3 M_Tool=2                      ' Change the tool conversion 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 conversion data by a hand input signal.
2   M_Tool=1                    ' Set tool 1 in tool conversion data.
3 Else
4   M_Tool=2                    ' Set tool 2 in tool conversion data.
5 EndIf
6 Mov P1
```

[Explanation]

(1) The values set in the MEXTL1, MEXTL2, MEXTL3 .... MEXTL16 tool parameters are reflected in the tool conversion data. It is also written into the MEXTL parameter.
For programs in which Base command, Tool command, and M_Tool are executed repeatedly, parameters may not be saved in time, causing error C7091 (parameter save error). If error C7091 occurs, change the order of Base command and M_Tool as shown in the following example.

Example)

```
*MAIN                          Base PB
  Base PB                      M_Tool=2
  M_Tool=2          ➡          *MAIN
  If M_In(20)=1 Then             If M_In(20)=1 Then
    GoSub *SUB1                    GoSub *SUB1
  EndIf                          EndIf
  GoTo *MAIN                     GoTo *MAIN
```

(2) Tool numbers 1 to 16 correspond to MEXTL1 to 16.
(3) While referencing, the currently set tool number is read.
(4) If the reading value is 0, it indicates that tool conversion data other than MEXTL1 to 16 is set as the current tool conversion data.

(5) Setting 0 in the tool number will reset tool parameter MEXTL to the initial values (0,0,0,0,0,0). To retain the current tool parameter with the tool number set to 0, change parameter TOOLSPEC from 0 (initial value) to 1, and turn on the power again.
Supported with controller software version A4 or later.
(6) The same setting can be performed on the Tool Setup screen of the teaching pendant. For more information, see Page 27, "3.2.9 Switching Tool Conversion Data".

[Reference]
Tool(Tool), MEXTL, MEXTL1 to MEXTL16, TOOLSPEC

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) AND &H0004    ' The result for area 3 only is assigned to M1.
2 If M1<>0 Then M_Out(10)=1    ' Output signal 10 turns ON if contained in area 3.
```

[Explanation]
(1) For details on how to use user-defined areas, refer to Page 520, "5.8 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 520, "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	&H00100000
21	22	2097152	&H00200000
22	23	4194304	&H00400000
23	24	8388608	&H00800000
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 H00000200, the value indicating area 10, however, this will be returned as an M_Uar32 value.

M_UDevW/ M_UDevD

[Function]

This function is to exchange the signals directly with two or more robot CPUs in the robot controller of CR800-Q series (Only CR800-Q series)
(Since the rudder program of the PLC is not needed, the exchange of the signal can be executed more speedily. And, the reference of shared memory information other than robot CPUs, such as motion CPU, is also possible. (Page 622, "5.26 Direct communication with robot CPUs"))

M_UDevW: Reads/ Writes per word. (16 bits)
M_UDevD: Reads/ Writes per double word. (32 bits)

[Format]

ExampleReference

<Numeric Variable> = M_UDevW(<Top input output number>, < Shared memory address >)
<Numeric Variable> = M_UDevD(<Top input output number>, < Shared memory address >)

Writing

M_UDevW(<Top input output number>, < Shared memory address >) = < Numeric value >
M_UDevD(<Top input output number>, < Shared memory address >) = < Numeric value >

[Terminology]

<Numeric Variable>	Specify the numeric variable which substitutes the value of the input output signal.
<Top input output number>	1 to 32, Specifies the task slot number. If this parameter is omitted, the current slot will be used as the default. Specify the input output signal number of the CPU unit with the constant or the numeric variable. (The value to specify omits the single digit low order of the top input output number in the hexadecimal number.) Range : "&H3E0" to "&H3E" in hexadecimal expression. ("992" to "995" in the decimal number) No. 1: &H3E0 (992 in the decimal number) No. 2: &H3E1 (993 in the decimal number) No. 3: &H3E2 (994 in the decimal number) No. 4: &H3E3 (995 in the decimal number) Note) The write function is allowed to host CPU only.
<Shared memory address>	Specify the shared memory address of the CPU unit with the constant or the numeric variable. The useful ranges differ for each status variable. (Decimal number) M_UDevW: 10000 to 24335 M_UDevD: 10000 to 24334
<Numeric value>	Specify the data to write in with the constant or the numeric variable. The useful ranges differ for each status variable M_UDevW: -32768 to 32767 (&H8000 to &H7FFF) M_UDevD: -2147483648 to 2147483647 (&H80000000 to &H7FFFFFFF)

[Reference Program]

```
1 M_UDevW(&H3E1, 10010)=&HFFFF          ' The &HFFFF (hexadecimal number) is written to the
                                          shared memory address 10010 of No. 2 CPU (host CPU).
2 M_UDevD(&H3E1, 10011)=P1.X * 1000     ' Calculate the X coordinate value of position variable P1 by
                                          1000. And write the result value to shared memory
                                          addresses of 10011/10012 (two word) on No. 2 CPU
                                          (host CPU).
3 M1%=M_UDevW(&H3E2, 10001) And &H7 ' The value of 3-bit width from 10001 of shared memory
                                          address of No. 3 CPU is substituted to M1.
```

[Explanation]

(1) Exchange directly the signals with two or more robot CPUs.
(2) Specify the shared memory to be used by the top input output signal number and the shared memory address.
(3) Both values (reads/ writes) are the integer values.
(4) Handle the data of the following width about the specified shared memory address.
M_UDevW:16 bit, M_UDevD:32 bit
(5) The range of the top input output signal number is &H3E0-&H3E3 in hexadecimal expression. (992-995 in the decimal number)
And, the range of the shared memory address written in or referred to is 10000-24335 in decimal number.
(6) The write function is allowed to host CPU only. It is not updated, although the address of other CPU units is specified and the data is written in. Only CR800-Q series devices are supported. (Statements executed using CR800-D/CR800-R series devices result in unsuccessful write operations, and the reference returns zero value.)
(7) Accessing to the shared memory with placing the address of even number in front can realize the data consistency for 32 bit data with M_UDevD. Refer to Page 413, "[Reference] Assurance of data sent between CPUs"

[Supplementary]

Table 4-39:<Numeric value>

O: The available, X: unavailable

Bit width	Numeric variables types				Other variables		
	Integer Ex.)M1%	Long-precision integer number Ex.)M1&	Single-precision real number Ex.)M1!	Double-precision real number Ex.)M1#	Position Note1) Ex.)P1.X	Joint Note1) Ex.)J1.J1	Character string Ex.)C1$
M_UDevW	X	O	O	O	O	O	X
M_UDevD	X	O	O	O	O	O	X

Note1) If the value of the variable is the angle, the unit will be processed by the radian. (The elements of A, B and C of position variable, and all elements of joint variable) The display of the monitor etc. is converted into the degree and displayed

Table 4-40:<<PLC input-signal number>

O: The available, X: unavailable

Bit width	constant types			Numeric variables types				Other variables		
	Numeric value Note1) Ex.)12	Binary number Ex.)&B1100	Hexadecimal number Ex.)&HC	Integer Ex.)M1%	Long-precision integer number Ex.)M1&	Single-precision real number Note1) Ex.)M1!	Double-precision real number Note1) Ex.)M1#	Position Note1) Note2) Ex.)P1.X	Joint Note1) Note2) Ex.)J1.J1	Character string Ex.)C1$
M_UDevW	O	O	O	O	O	O	O	O	X	X
M_UDevD	O	O	O	O	O	O	O	O	X	X

Note1) The real value is rounded off.
Note2) If the value of the variable is the angle, the unit will be processed by the radian. (The elements of A, B and C of position variable, and all elements of joint variable) Therefore, designation of the signal number is very difficult. The display of the monitor etc. is converted into the degree, and the same value as the setting value displayed.
Example) It is processed by value "0", even if it sets "8" as the value of P1.A (The input in the key by T/B etc.) to specify the input signal No.8. The result is "0" when 8 degree is converted to radian (0.14) and rounded off. Because the unit of the element X, Y, and Z of the position variable is "mm", there is no such condition.

Table 4-41:<Numeric value>

O: The available, X: unavailable

Bit width	constant types			Numeric variables types				Other variables		
	Numeric value Note1) Ex.)12	Binary number Ex.)&B1100	Hexadecimal number Ex.)&HC	Integer Ex.)M1%	Long-precision integer number Ex.)M1&	Single-precision real number Ex.)M1!	Double-precision real number Ex.)M1#	Position Ex.)P1.X	Joint Ex.)J1.J1	Character string Ex.)C1$
M_UDevW	O	O	O	O	X	X	X	X	X	X
M_UDevD	O Note2)	O Note2)	O Note2)	O Note2)	O	O	O Note3)	O	O	X

Note1) The real value is rounded off.

Note2) CAUTION

For the numerical value of the less than 16 bits of the binary number (-32768 to +32767), the specified constant will handle as a negative numerical value, if the bit 15 (the 16th bit) turns on. Therefore, please be careful of turning on all of upper 16 bits. (The sign bit is extended)

Example)

Designation of "-32768(&B1000000000000000)" will output the "&B11111111111111111000000000000000."

[Measures]

After substituting the constant for the long-precision integer number variable as follows, when substituting to this robot status variable M_UDevD, &B00000000000000001000000000000000 (binary number) can be outputted.

```
1 M1&=-32768
2 M_UDevD(&H20)=M1&
```

Note3) The ranges of the numerical value which can be outputted are -2147483648 to 2147483647.

[Reference] Assurance of data sent between CPUs

The old data and the new data may be mixed (data separation) in each CPU due to the timing of receiving data from the other CPU and reading in the host CPU.

The Fig. 4-41 shows the method to realize the data consistency of the user data for the data transmission in the multiple CPU high speed transmission function.

1) Preventing 32-bit data separation

Accessing to the user setting area of the multiple CPU high speed transmission area with placing the address of even number in front (for example, address 10002) can realize the data consistency for 32 bit data.

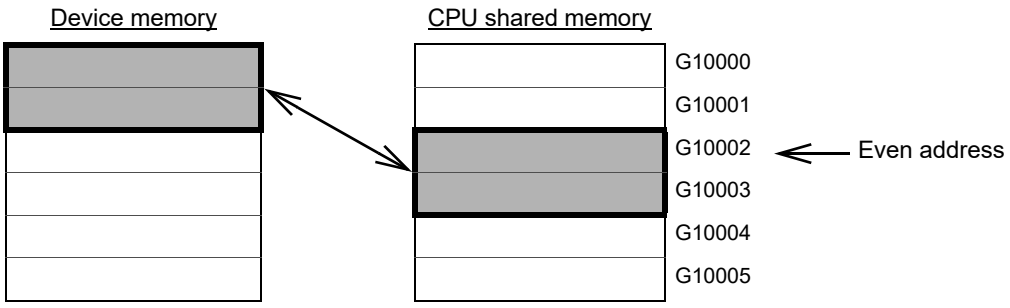


Fig.4-41:Preventing 32-bit data separation

2) Preventing separation for data exceeding 32 bits

Programs are read from the start of user setting area.

With the write instruction, send data are written from the last address to the start address of the user setting area.

Therefore, data separation can be avoided by creating an interlock device at the start of data to be communicated.

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 545, "5.19 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 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 545, "5.19 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 545, "5.19 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.

M_XDev/ M_XDevB/ M_XDevW/ M_XDevD

[Function]

Reads the value of PLC input signal (X) in the robot controller of CR800-R/Q series. (Only CR800-R/Q series)

The direct reference of the input signal of the input output unit / input output mixing unit managed by other CPU is possible. (Refer to Page 591, "5.25 Direct control of the PLC input/output module")

M_XDev: Reads per bit.
M_XDevB: Reads per byte. (8 bits)
M_XDevW: Reads per word. (16 bits)
M_XDevD: Reads per double word. (32 bits)

[Format]

<Numeric Variable> = M_XDev[(<PLC input signal number>)]

<Numeric Variable> = M_XDevB[(<PLC input signal number>)]

<Numeric Variable> = M_XDevW[(<PLC input signal number>)]

<Numeric Variable> = M_XDevD[(<PLC input signal number>)]

[Terminology]

<Numeric Variable>	Specify the numeric variable which substitute the value of the input signal.
<PLC input signal number>	Specify the input signal number to refer to with the constant or the numeric variable. Although the useful range is "&H0" - "&HFFF" (0 - 4095 in decimal number) in hexadecimal expression, it differs for each status variable. M_XDev: &H0 - &HFFF(0 - 4095) M_XDevB: &H0 - &HFF8(0 - 4088) M_XDevW: &H0 - &HFF0(0 - 4080) M_XDevD: &H0 - &HFE0(0 - 4064) Note) The real number is rounded off.

[Reference Program]

```
1 M1%=M_XDev(1)                    ' The value of the PLC input signal 1 (1 or 0) is substituted to M2.
2 M2%=M_XDevB(&H10)                ' The value of 8-bit width from 10 (hexadecimal number) of PLC
                                     input signals is substituted to M2.
3 M3%=M_XDevW(&H20) And &H7        ' The value of 3-bit width from 20 (hexadecimal number) of PLC
                                     input signals is substituted to M3.
4 M4%=M_XDevW(&H20)                ' The value of 16-bit width from 20 (hexadecimal number) of PLC
                                     input signals is substituted to M4.
5 M5&=M_XDevD(&H100)               ' The value of 32-bit width from 100 (hexadecimal number) of PLC
                                     input signals is substituted to M5.
6 P1.Y=M_XDevD(&H100)/1000         ' Input the 32-bit width from 100 (hexadecimal number) as an
                                     integer value, and divide by 1000. (Change into the real number)
                                     And the arithmetic result is substituted to Y coordinate value of
                                     the position variable P1.
```

[Explanation]

(1) Return the condition of PLC input signal (X) by the integer value.
(2) Each return the data of the following width about the specified PLC input-signal number.
M_XDev: 1 bit, M_XDevB:8 bit, M_XDevW:16 bit, M_XDevD:32 bit

(3) The PLC input signal number should be in "&H0"to "&HFFF" in hexadecimal expression. Error L3110 (value of the argument outside of the range) will occur, if it is the abbreviation and outside the range.
(4) It is necessary to set up so that the input signal can be referred to with Parameter QXYREAD previously.
(5) Return 0, when the PLC unit which can correspond is not connected.

[Supplementary]

Table 4-42:<Numeric value>

O: The available, X: unavailable

Bit width	Numeric variables types				Other variables		
	Integer Ex.)M1%	Long-precision integer number Ex.)M1&	Single-precision real number Ex.)M1!	Double-precision real number Ex.)M1#	Position Note1) Ex.)P1.X	Joint Note1) Ex.)J1.J1	Character string Ex.)C1$
M_XDev	O	O	O	O	O	O	X
M_XDevB	O	O	O	O	O	O	X
M_XDevW	O	O	O	O	O	O	X
M_XDevD	X	O	O	O	O	O	X

Note1) If the value of the variable is the angle, the unit will be processed by the radian. (The elements of A, B and C of position variable, and all elements of joint variable) The display of the monitor etc. is converted into the degree and displayed.

Table 4-43:<PLC input-signal number>

O: The available, X: unavailable

Bit width	constant types			Numeric variables types				Other variables		
	Numeric value Note1) Ex.)12	Binary number Ex.)&B1100	Hexadecimal number Ex.)&HC	Integer Ex.)M1%	Long-precision integer number Ex.)M1&	Single-precision real number Note1) Ex.)M1!	Double-precision real number Note1) Ex.)M1#	Position Note1) Note2) Ex.)P1.X	Joint Note1) Note2) Ex.)J1.J1	Character string Ex.)C1$
M_XDev	O	O	O	O	O	O	O	O	X	X
M_XDevB	O	O	O	O	O	O	O	O	X	X
M_XDevW	O	O	O	O	O	O	O	O	X	X
M_XDevD	O	O	O	O	O	O	O	O	X	X

Note1) The real value is rounded off.
Note2) If the value of the variable is the angle, the unit will be processed by the radian. (The elements of A, B and C of position variable, and all elements of joint variable) Therefore, designation of the signal number is very difficult. The display of the monitor etc. is converted into the degree, and the same value as the setting value displayed.
Example) It is processed by value "0", even if it sets "8" as the value of P1.A (The input in the key by T/B etc.) to specify the input signal No.8. The result is "0" when 8 degree is converted to radian (0.14) and rounded off. Because the unit of the element X, Y, and Z of the position variable is "mm", there is no such condition.

M_YDev/ M_YDevB/ M_YDevW/ M_YDevD

[Function]

Reads/ Writes the value of PLC output signal (Y) in the robot controller of CR800-R/Q series. (Only CR800-R/Q series)
(Set up the input output unit / input output mixing unit which robot CPU manages, and execute direct reference or direct write of the output signal. (Refer to Page 591, "5.25 Direct control of the PLC input/output module"))

M_YDev: Reads/ Writes per bit.
M_YDevB: Reads/ Writes per byte. (8 bits)
M_YDevW: Reads/ Writes per word. (16 bits)
M_YDevD: Reads/ Writes per double word. (32 bits)

[Format]

Reference

<Numeric Variable> = M_YDev[(<PLC output signal number>)]
<Numeric Variable> = M_YDevB[(<PLC output signal number>)]
<Numeric Variable> = M_YDevW[(<PLC output signal number>)]
<Numeric Variable> = M_YDevD[(<PLC output signal number>)]

Writing

M_YDev[(<PLC output signal number>)] = < Numeric value>
M_YDevB[(<PLC output signal number>)] = < Numeric value>
M_YDevW[(<PLC output signal number>)] = < Numeric value>
M_YDevD[(<PLC output signal number>)] = < Numeric value>

[Terminology]

<Numeric Variable>	Specify the numeric variable which substitute the value of the output signal to refer to.
<PLC output signal number>	Specify the output signal number to Read/ write with the constant or the numeric variable. Although the useful range is "&H0" - "&HFFF" (0 to 4095 in decimal number) in hexadecimal expression, it differs for each status variable. M_YDev: &H0 - &HFFF(0 - 4095) M_YDevB: &H0 - &HFF8(0 - 4088) M_YDevW: &H0 - &HFF0(0 - 4080) M_YDevD: &H0 - &HFE0(0 - 4064) Note) The real number is rounded off.
<Numeric value>	Describe the value to output by the numeric variable, the constant. The useful ranges differ for each status variable. M_YDev: 0 or 1(&H0 or &H1) M_YDevB: -128 - 127(&H80 - &H7F) M_YDevW: -32768 - 32767(&H8000 - &H7FFF) M_YDevD: -2147483648 - 2147483647 (&H80000000 - &H7FFFFFFF) Note) The real number is rounded off.

[Reference Program]

```
1 M_YDev(1)=1                    ' Turns on the PLC output signal 1
2 M_YDevB(&H10)=&HFF             ' Turns on the 8-bit width from 10 (hexadecimal number) of PLC
                                   output signal.
3 M_YDevW(&H20)=&HFFFF           ' Turns on the 16-bit width from 20 (hexadecimal number) of PLC
                                   output signal.
4 M_YDevD(&H100)=P1.X * 1000     ' Outputs the multiplication result value of X coordinate value of
                                   the position variable P1 by 1000 to 32-bit width from 100
                                   (hexadecimal number) of PLC output signals.
5 M1%=M_YDevW(&H20) And &H7      ' The value of 3-bit width from 20 (hexadecimal number) of PLC
                                   input signals is substituted to M1.
```

[Explanation]

(1) Reads/ Writes the value of PLC output signal (Y).
(2) Both value (Reads/ writes) are the integer values.
(3) Handle the data of the following width about the specified PLC output signal number.
M_YDev: 1 bit, M_YDevB:8 bit, M_YDevW:16 bit, M_YDevD:32 bit
(4) The PLC output signal number should be "&H0"- "&HFFF" in hexadecimal expression. Error L3110 (value of the argument outside of the range) will occur, if it is the abbreviation or outside the range.
(5) The following setup is necessary previously.
* Validate the reference of the input signal with Parameter QXYREAD.
* Set up the input output unit which will be managed by robot CPU with Parameter QXYUNITn (n=1 - 4).
(6) The pulse output which combines the Dly command cannot be used. If the Dly command is used, error L4220 (syntax error) occurs.
(7) If the corresponding PLC unit is not connected at writing the output signal, the signal will not change. If the corresponding PLC unit is not connected at referring to the output signal, the return value will be 0.

[Supplementary]

Table 4-44:<Numeric value>

O: The available, X: unavailable

Bit width	Numeric variables types				Other variables		
	Integer Ex.)M1%	Long-precision integer number Ex.)M1&	Single-precision real number Ex.)M1!	Double-precision real number Ex.)M1#	Position Note1) Ex.)P1.X	Joint Note1) Ex.)J1.J1	Character string Ex.)C1$
M_YDev	O	O	O	O	O	O	X
M_YDevB	O	O	O	O	O	O	X
M_YDevW	O	O	O	O	O	O	X
M_YDevD	X	O	O	O	O	O	X

Note1) If the value of the variable is the angle, the unit will be processed by the radian. (The elements of A, B and C of position variable, and all elements of joint variable) The display of the monitor etc. is converted into the degree and displayed.

Table 4-45:<PLC input-signal number>

O: The available, X: unavailable

Bit width	constant types			Numeric variables types				Other variables		
	Numeric value Note1) Ex.)12	Binary number Ex.)&B1100	Hexadecimal number Ex.)&HC	Integer Ex.)M1%	Long-precision integer number Ex.)M1&	Single-precision real number Note1) Ex.)M1!	Double-precision real numbe Note1) Ex.)M1#	Positio Note1) Note2) Ex.)P1.X	Joint Note1) Note2) Ex.)J1.J1	Character string Ex.)C1$
M_XDev	O	O	O	O	O	O	O	O	X	X
M_XDevB	O	O	O	O	O	O	O	O	X	X
M_XDevW	O	O	O	O	O	O	O	O	X	X
M_XDevD	O	O	O	O	O	O	O	O	X	X

Note1) The real value is rounded off.

Note2) If the value of the variable is the angle, the unit will be processed by the radian. (The elements of A, B and C of position variable, and all elements of joint variable) Therefore, designation of the signal number is very difficult. The display of the monitor etc. is converted into the degree, and the same value as the setting value displayed.

Example) It is processed by value "0", even if it sets "8" as the value of P1.A (The input in the key by T/B etc.) to specify the input signal No.8. The result is "0" when 8 degree is converted to radian (0.14) and rounded off. Because the unit of the element X, Y, and Z of the position variable is "mm", there is no such condition.

Table 4-46:<Numeric value>

O: The available, X: unavailable

Bit width	constant types			Numeric variables types				Other variables		
	Numeric value Note1) Ex.)12	Binary number Ex.)&B1100	Hexadecimal number Ex.)&HC	Integer Ex.)M1%	Long-precision integer number Ex.)M1&	Single-precision real number Ex.)M1!	Double-precision real number Ex.)M1#	Position Ex.)P1.X	Joint Ex.)J1.J1	Character string Ex.)C1$
M_XDev	O	O	O	O	X	X	X	X	X	X
M_XDevB	O	O	O	O	X	X	X	X	X	X
M_XDevW	O	O	O	O	X	X	X	X	X	X
M_XDevD	O Note2)	O Note2)	O Note2)	O Note2)	O	O	O Note3)	O	O	X

Note1) The real value is rounded off.

Note2) ⚠CAUTION For the numerical value of the less than 16 bits of the binary number (-32768 to +32767), the specified constant will handle as a negative numerical value, if the bit 15 (the 16th bit) turns on. Therefore, please be careful of turning on all of upper 16 bits. (The sign bit is extended)

Example)

Designation of "-32768(&B1000000000000000)" will output the "&B11111111111111111000000000000000."

[Measures]

After substituting the constant for the long-precision integer number variable as follows, when substituting to this robot status variable M_YDevD, &B00000000000000001000000000000000 (binary number) can be outputted.

1 M1&=-32768

2 M_YDevD(&H20)=M1&

Note3) The ranges of the numerical value which can be outputted are -2147483648 to 2147483647.

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_CavDir

[Function]

Returns which direction the robot was moving when an interference was predicted during interference check.

This function is only available for certain models. For details, refer to Page 567, "5.24 Interference avoidance function".

[Format]

Example) <Position Variable>=P_CavDir[(<Mechanism No>)]

[Terminology]

<Position Variable> Specifies a position variable to be assigned.

<Mechanism No.> Enter the mechanism number, 1 to 3. If the argument is omitted, 1 is set.

[Reference Program]

Refer to.Page 589, "5.24.10 Sample programs"

[Explanation]

(1) Use this command to check the robot's moving direction in automatic restoration after an interference check.

(2) The robot's moving direction when an interference is predicted is indicated in a ratio, which assuming the maximum locomotive axis value as +/-1.0.
Example) When the robot is being operated at the ratio of (X-axis direction : Y-axis direction) = (2 : -1)
P_CavDir=(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). (* is an arbitrary value.)

(4) A value is calculated when an interference is predicted, and then that value is retained until the next interference is predicted.

(5) If an interference is predicted because of another robot moving while the own robot is in stop, 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 interference occurs at a position near the target position.

(7) This is read only.

(8) For robots that prohibit the use of interference check, 0.0 is always returned for all axes.

(9) Units of the read-enabled values are the same with the ones of P_ColDir.

P_ColDir

[Function]
Return the operation direction of the robot when an collision is detected.

[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 195, "[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 collision detection.
(2) The operation direction of the robot at the very moment of collision 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 collision is detected, and then that value is retained until the next collision is detected.
(5) If an collision 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 collision occurs at a position near the target position.
(7) This is read only.
(8) For robots that prohibit the use of collision detection, 0.0 is always returned for all axes.

[Reference]
ColChk (Col Check), ColLvl (Col Level), M_ColSts, J_ColMxl

P_CordR

[Function]
Returns the base coordinates of the own robot looking from the common coordinates.

This function is only available for certain models. For details, refer to Page 567, "5.24 Interference avoidance function".

[Format]

Example) <Position Variable> = P_CordR [(<Mechanism No>)]

[Terminology]
<Position Variable> Specifies a position variable to be assigned.
<Mechanism No.> Enter the mechanism number, 1 to 3. If the argument is omitted, 1 is set.

[Reference Program]
```
1 P1=P_CordR        ' P1 will contain the base coordinates of the robot from the common coordinates.
```

[Explanation]
(1) The base coordinates of the robot looking from the common coordinates are read. (The setting value of parameter: RBCORD)
(2) All coordinates are read as 0 when the parameter: RBCORD is set to the initial value (0,0,0,0,0,0).
(3) This is read only.
(4) The value "0" is always returned for the user mechanisms.

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 *LACT        ' Defines interrupt.
2 Act 1=1                                ' Enables interrupt.
3 Mov P1
4 Mov P2
5 Act 1=0                                ' Disables interrupt.
 :
100 *LACT
101 P100=P_Curr                          ' Loads the current position when an interrupt signal is
                                           received.
102 Mov P100,-100                        ' Moves 100 mm above P100 (i.e, -100 mm in the Z
                                           direction of the tool).
103 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

P_CurrR

[Function]
Returns the current position of the own robot looking from the common coordinates.

This function is only available for certain models. For details, refer to Page 567, "5.24 Interference avoidance function".

[Format]

Example) <Position Variable> = P_CurrR [(<Mechanism No>)]

[Terminology]
<Position Variable> Specifies a position variable to be assigned.
<Mechanism No.> Enter the mechanism number, 1 to 3. If the argument is omitted, 1 is set.

[Reference Program]

```
1 P1=P_CurrR        ' P1 will contain the current position of the robot looking from the common coordinates.
```

[Explanation]
(1) The current position looking from the common coordinates is read.
(2) The value which is converted from P_Curr by the parameter: RBCORD is returned.
(3) This is read only.
(4) The value "0" is always returned for the user mechanisms.

P_ECord

[Function]

Returns the Ex-T coordinate system origin data currently used during the Ex-T control/Ex-T spline interpolation movement.

[Format]

Example)<Position Variables>=P_ECord [(<Mechanism Number>)]

[Terminology]

<Position Variables> Designates the numerical variable substituted for the reference results.

<Mechanism Number> Sets the mechanism No. that executes Ex-T control/Ex-T spline interpolation.
Setting range: 1 to 3
When omitted: 1
If a non-existent mechanism No. is designated, the error L3870 (designated mechanism No. invalid) will occur when execution is started.

[Reference Program]

```
1 P1 = P_ECord      ' The Ex-T coordinate system origin data that executes interpolation is
                      substituted for position variable P1.
```

[Explanation]

(1) During Ex-T control/Ex-T spline interpolation, the Ex-T coordinate system origin data most recently used is returned.

(2) By referring to the P_ECord value with multi-task, the operation process or signal output, etc., can be executed according to the progress of spline interpolation.

(3) P_ECord returns the value indicated in the following table according to the state.

Table 4-47:Value returned by P_ECord

Status	Value returned by P_ECord
Immediately after power ON	(0.0,0.0,0.0,0.0,0.0,0.0,0.0)(0,0)
During Ex-T control/Ex-T spline interpolation execution	The Ex-T coordinate system origin data used for interpolation
After Ex-T control/Ex-T spline interpolation	The Ex-T coordinate system origin data most recently used
Immediately after main program's End command is executed	(0.0,0.0,0.0,0.0,0.0,0.0,0.0)(0,0)
Immediately after program reset operation	(0.0,0.0,0.0,0.0,0.0,0.0,0.0)(0,0)
Mechanism not compatible with Ex-T control/ Ex-T spline interpolation	(0.0,0.0,0.0,0.0,0.0,0.0,0.0)(0,0)

(4) P_ECord is a read-only status variable.

[Related instructions]

EMvc (E Move C), EMvr (E Move R), EMvr2 (E Move R 2), EMvr3 (E Move R 3), EMvs (E Move S), EMvSpl (E Move Spline)

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_GCurr

[Function]

This command contains a status variable to read the current position of the master robot on starting the cooperative operation between the master and slave robots. Use this command to align the start positions of movement by reading the current position of the master robot before starting the cooperative operation.
As a preparation, it is required to enable the iQ extended function so that the current position of the target CPU can be monitored.

[Format]

```
Example)<Position Variables>=P_GCurr [(<Target CPU No.>)]
```

[Terminology]

<Position Variables>	Specifies the position variable to assign.
<Target CPU No.>	Enter the CPU module No. of the master robot. 2 to 4.

[Reference Program]

```
Open "MXT:QRBUS2" As #1      ' Specify the robot CPU No. 2 in the CPU buffer memory as a master robot.
Mov P_GCurr(2)               ' The start positions of movement at the host CPU is aligned with the robot CPU No. 2.
```

[Explanation]

(1) This command can be used to obtain the start position data of movement in the application with the cooperative operation function.
(2) This variable only reads the data.
(3) The current position of the target CPU at the point of reading this variable is assigned.
If the specified target CPU is not enabled the iQ extended function, all values are always 0 (same as the P_Zero).
(4) The assigned value for the status variable is retained until the controller is powered off. Reading is possible regardless of the Open command.

[Reference]

Torq (Torque), J_Fbc/J_AmpFbc, M_Fbd

P_GDev

[Function]

Writes or directly references positional data to CPU buffer memory with the CR800-R/Q series controllers. (CR800-R/Q series controllers only)
Writing and reading are performed in the position data width of (32 bits x 10).

[Format]

```
Reading
<Position Variable>=P_GDev(<Start I/O Number>, <Buffer memory address>)

Writing
P_GDev(<Start I/O Number>, <Buffer memory address>)=<Position Data>
```

[Terminology]

<Position Variable>	Specifies the position variable to assign.
<Start I/O Number>	Specifies the I/O number of a CPU module as a constant or numeric variable. (The specified value for the start I/O number is in hexadecimal and with its last digit omitted.) Range: &H3E0 to &H3E3 in hexadecimal (992 to 995 in decimal) Module No. 1: &H3E0 (992 in decimal) Module No. 2: &H3E1 (993 in decimal) Module No. 3: &H3E2 (994 in decimal) Module No. 4: &H3E3 (995 in decimal) Note) Data can be written to the host CPU only.
<Buffer memory address>	Specifies CPU module buffer memory addresses using constants or numeric variables. The effective range is between 0 and 524268 in decimal numbers.
<Position Data>	Specifies which position data to write. The data can be specified as a constant, variable, logical/arithmetic expression, or function.

[Reference Program]

```
1 P_GDev(&H3E1, 10)=P_Curr  'Sets buffer memory address 10 of CPU No. 2 to the current location.
2 P1=P_GDev(&H3E2, 1)       'Sets P1 to the positional data in buffer memory address 1 of CPU No. 3.
```

[Explanation]

(1) This variable is used to write or reference positional data to PLC CPU buffer memory.
Both write and reference data strings are positional data.
(2) Buffer memory addresses are specified using start I/O numbers and the actual buffer memory addresses.
(3) This variable uses 20 words (10 32-bit data strings) of data in the specified buffer memory address.
(4) Specify start I/O numbers in hexadecimal format. The range of numbers is between &H3E0 to &H3E3, which is equivalent to 992 to 995 in decimal. Specify buffer memory addresses in decimal numbers within a range of addresses between 0 to 524268.
(5))Write operations can only be performed for buffer memory addresses of the target CPU. Attempting to write data to addresses for other CPUs will not result in a successful change to the data stored by the address.
(6) Only CR800-R/Q series devices are supported. Statements executed using CR800-D series devices result in unsuccessful write operations, and the reference returns zero value.

[Supplement]

Table 4-48:Constants and variables usable as <PLC input signal number> and <Buffer memory address>

Bit width	Constant types			Numeric variable types				Other variables		
	Numeric value Note1) Ex.)12	Binary number Ex.)&B1100	Hexadecimal number Ex.)&HC	Integer Ex.)M1%	Long-precision integer Ex.)M1%	Single-precision real number Note1) Ex.)M1&	Double-precision real number Note1) Ex.)M1	Position Note1) Ex.)P1.X	Joint Note1) Ex.)J1.J1	Character string Ex.)C1$
Availability	O	O	O	O	O	O	O	O	O	X

O: Available, X: Unavailable

Note1) The real value is rounded off.

P_Gps1 to P_Gps8

[Function]

This command returns XYZ coordinate data for the current position data when the condition defined in the Def Gps command is met, using the get-position-quick function (GPS function). (Up to 400 position data can be saved.)

The current position data of the robot at the time when the condition defined in the Def Gps command is met can be checked. (The number of times when the condition defined in the Def Gps command is met can be checked in the M_Gps.)

[Format]

Example) <Position Variable>=P_Gps1[(<Numeric value>)]

[Terminology]

<Position Variable>	Specifies the position variable to assign. The position data corresponding to a position number set in <Numeric value> is returned to <Position variable>. The initial value is "P_Zero" (all elements are zero). The value is cleared to zero when the GpsChk On command is executed, and the current position data at the time when the defined condition is met is stored after the GpsChk Off command is executed.
<Numeric value>	Set a position number obtained using the GPS function. (1 to 400) The first time when the condition defined in the Def Gps command is met is numbered 1, and the following times are numbered 2, 3, ... in turn. (To obtain the position data of the first time, set "1".) 1 will be set when omitted.

[Reference Program]

```
1 Def Gps1,852,On,1        ' The position data of the mechanism No. 1 is recorded for the
                             monitoring No. 1 when the signal No. 852 is turned on.
2 GpsChk On,1              ' Monitoring a condition for the monitoring No. 1 is started.
3 Mvs P1                   ' Moves to P1
4 GpsChk Off,1             ' Monitoring a condition for the monitoring No. 1 is stopped.
5 M1=M_Gps(1)              ' The number of the position data recorded in the P_Gps1 is stored.
6 If M1=0 Then Error 9000  ' The error 9000 is generated if no position data is recorded.
7 Mvs P_Gps1(1)            ' The robot moves its arm to the position at the first time when the signal
                             No. 852 is turned on.
8 Hlt                      ' Halt program
```

[Explanation]

(1) The current position data of the robot is stored in this status variable when the condition defined in the Def Gps command is met. When "1" is set in <Monitoring No.> in the Def Gps command and the condition is met three times, these position data of the robot is stored in the "P_Gps1(1), P_Gps1(2), and P_Gps1(3)". The stored position data of the first time is shown as "P_Gps1(1)".

(2) Up to 400 position data (P_GpsX(1) to P_GpsX(400)) can be saved. When the number of stored position data reaches 400, further position data cannot be stored.

(3) The position data corresponding to a numerical value set in array elements is returned.

(4) The value is cleared to zero when the GpsChk On command is executed.

(5) The number of times can be confirmed after the GpsChk Off command is executed.

(6) This variable only reads the data.

(7) The number of times is cleared to zero when the controller is turned on.

[Related instructions]

Def Gps (Define get position), Def Map (Define mapping), GpsChk (Get position check)

P_HGDev

[Function]

Writes or directly references positional data to the periodical communication area of CPU buffer memory with the CR800-R series controllers. (CR800-R series controllers only)
Writing and reading are performed in the position data width of (32 bits x 10).

[Format]

```
Reading
<Position Variable>=P_HGDev(<Start I/O Number>, <Buffer memory address>)

Writing
P_HGDev(<Start I/O Number>, <Buffer memory address>)=<Position Data>
```

[Terminology]

<Position Variable>	Specifies the position variable to assign.
<Start I/O Number>	Specifies the I/O number of a CPU module as a constant or numeric variable. (The specified value for the start I/O number is in hexadecimal and with its last digit omitted.) Range: &H3E0 to &H3E3 in hexadecimal (992 to 995 in decimal) Module No. 1: &H3E0 (992 in decimal) Module No. 2: &H3E1 (993 in decimal) Module No. 3: &H3E2 (994 in decimal) Module No. 4: &H3E3 (995 in decimal)
<Buffer memory address>	Specifies CPU module buffer memory addresses using constants or numeric variables. The effective range is between 0 and 12268 in decimal numbers.
<Position Data>	Specifies which position data to write. The data can be specified as a constant, variable, logical/arithmetic expression, or function.

[Reference Program]

```
1 P_HGDev(&H3E1,10)=P_Curr      'Sets buffer memory address 10 of CPU No. 2 to the current location.
2 P1=P_HGDev(&H3E2,1)           'Sets P1 to the positional data in buffer memory address 1 of CPU No. 3.
```

[Explanation]

(1) This variable is used to write or reference positional data to the periodical communication area of PLC CPU buffer memory. Both write and reference data strings are positional data.
(2) Buffer memory addresses are specified using start I/O numbers and the actual buffer memory addresses.
(3) This variable uses 20 words (10 32-bit data strings) of data in the specified buffer memory address.
(4) Specify start I/O numbers in hexadecimal format. The range of numbers is between &H3E0 to &H3E3, which is equivalent to 992 to 995 in decimal. Specify buffer memory addresses in decimal numbers within a range of addresses between 0 to 12268.
(5) Write operations can only be performed for buffer memory addresses of the target CPU. Attempting to write data to addresses for other CPUs will not result in a successful change to the data stored by the address.
(6) Reference operations result in a value of 0 when the corresponding CPU buffer memory address (parameter: QMLTCPUn) has not been assigned. Only CR800-R series devices are supported. (Statements executed using CR800-D/CR800-Q series devices result in unsuccessful write operations, and the reference returns zero value.)

[Supplement]

Table 4-49:Constants and variables usable as <PLC input signal number> and <Buffer memory address>

Bit width	Constant types			Numeric variable types				Other variables		
	Numeric value Note1) Ex.)12	Binary number Ex.)&B1100	Hexadecimal number Ex.)&HC	Integer Ex.)M1%	Long-precision integer Ex.)M1%	Single-precision real number Note1) Ex.)M1&	Double-precision real number Note1) Ex.)M1	Position Note1) Ex.)P1.X	Joint Note1) Ex.)J1.J1	Character string Ex.)C1$
Availability	O	O	O	O	O	O	O	O	O	X

O: Available, X: Unavailable

Note1) The real value is rounded off.

P_Safe

[Function]
Writes or references positional data to the safe point (XYZ position of the JSAFE parameter).

[Format]

```
Reading
<Position Variables>=P_Safe [(<Mechanism Number>)]

Writing
P_Safe[(<Mechanism Number>)] = <Position Data>
```

[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.
<Position Data>	Specifies the position data to be written. The data can be specified as a constant, variable, logical/arithmetic expression, or function.

[Reference Program]

```
1 P1=P_Safe          ' P1 will contain the set safe point.
2 P_Safe = P2        ' Change the safe point to P2.
```

[Explanation]
(1) Returns the XYZ position, which has been converted from the joint position registered in parameter JSAFE.
(2) When the data has been written, the value of the parameter JSAFE is also written.

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)(0,0).
[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 conversion data.
(3) This variable only reads the data.

P_UDev

[Function]

Writes/reads the position data to/from the CPU shared memory. (This function is available with the CR800-Q series robot controllers only.)
Writing and reading are performed in the position data width of (32 bits x 10).

[Format]

```
Reading
<Position Variable>=P_UDev(<Start I/O Number>, <Shared Memory Address>)

Writing
P_UDev(<Start I/O Number>, <Shared Memory Address>)=<Position Data>
```

[Terminology]

<Position Variable>	Specifies a numerical variable to substitute.
<Start I/O Number>	Specifies the I/O number of a CPU module as a constant or numeric variable. (The specified value for the start I/O number is in hexadecimal and with its last digit omitted.) Range: &H3E0 to &H3E3 in hexadecimal (992 to 995 in decimal) Module No. 1: &H3E0 (992 in decimal) Module No. 2: &H3E1 (993 in decimal) Module No. 3: &H3E2 (994 in decimal) Module No. 4: &H3E3 (995 in decimal) Note) Data can be written to the host CPU only.
<Shared Memory Address>	Specifies the shared memory address in the CPU module as a constant or numeric variable. The valid range is 10000 to 24316 (decimal).
<Position Data>	Specifies which position data to write. The data can be specified as a constant, variable, logical/arithmetic expression, or function.

[Reference Program]

```
1 P_UDev(&H3E1, 10010)=P_Curr    'Write the current position to the shared memory address 10010
                                 of the CPU module No.2.
2 P1=P_UDev(&H3E2, 10001)        'Read the position data from the shared memory address 10001
                                 of the CPU module No.3, and substitute the read value for the
                                 position variable P1.
```

[Explanation]

(1) This command writes/reads the position data to/from the CPU shared memories of programmable controllers.
Written data and returned read data are both position data.
(2) Use the start I/O number and shared memory address to specify a target shared memory.
(3) The target data is a 20-word width data (32 bits x 10) starting from the specified shared memory address.
(4) Use &H3E0 to &H3E3 in hexadecimal (992 to 995 in decimal) to specify a start I/O number. Use 10000 to 24316 in decimal to specify an address to be written or read in the shared memory address.
(5) Data can be written to the shared memory address of the host CPU only. Even if another CPU address is specified for data writing, the data will not be updated.
(6) If the shared memory address has not been assigned by the Multi CPU quantity setting (Parameter: QMLTCPUn), the value 0 will be returned when position data is read. Only CR800-Q series devices are supported. (Statements executed using CR800-D/CR800-R series devices result in unsuccessful write operations, and the reference returns zero value.)

[Supplement]

Table 4-50:Constants and Variables for <Input Signal Number for Programmable Controller ><Shared Memory Address>

Variable types	Constant types			Numeric variable types				Other variables		
	Numeric value Note1) Ex.)12	Binary number Ex.) &B1100	Hexadecimal number Ex.)&HC	Integer Ex.)M1%	Long-precision integer Ex.)M1&	Single-precision real number Note1) Ex.)M1!	Double-precision real number Note1) Ex.)M1#	Position Note1) , Note2) Ex.)P1.X	Joint Note1) , Note2) Ex.)J1.J1	Character string Ex.)C1$
Availability	○	○	○	○	○	○	○	○	×	×

○ : Available, × : Unavailable

Note1) The real value is rounded off.

Note2) If the value of the variable is an angle, the unit will be processed by the radian. (The elements of A, B and C of position variable, and all elements of joint variable) Therefore, designation of the signal number is very difficult. (The display of the monitor etc. is converted into a degree, and the same value as the setting value is displayed.)
It is processed by value "0", even if it sets "8" as the value of P1.A (The input in the key by T/B etc.) to specify the input signal No.8. The result is "0" when 8 degree is converted to radian (0.14) and rounded off. Because the unit of the element X, Y, and Z of the position variable is "mm", there is no such condition.

[Reference] Data assurance between CPU modules

New and old data may co-exist (data inconsistency may exist) due to difference between data writing/ receiving timings of host and other CPU modules.
Fig. 4-42 shows how to prevent data inconsistency using the program communication between the CPU shared memories.

1) Preventing data inconsistency of 32-bit data
Data inconsistency of 32-bit data can be prevented by accessing a CPU shared memory's user setting area that starts in an even address (for example, address 10002).

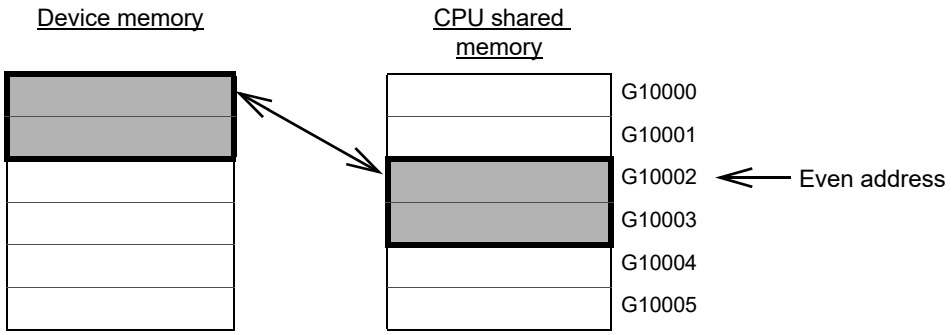


Fig.4-42:Preventing data inconsistency of 32-bit data

2) Preventing data inconsistency of data larger than 32 bits
When data is read by a program, the reading starts from the start of the user setting area.
When data is commanded to be written, the writing of the transmitted data starts from the end of the user setting area to the start of the user setting area.
With these characteristics in mind, provide an interlock device to the head of the transmitted data to prevent data inconsistency between transmitted data.

P_WkCord

[Function]

This function permits you to make reference to the work coordinate data being currently specified or to make a setting for a new work coordinate.
Parameters to be worked with are WK1CORD through WK8CORD.
(For the outline of the function, refer to Page 750, "7.3 Ex-T control".)

[Format]

```
Example)<Position variable>=P_WkCord(<work coordinate number>)        'Reference
        P_WkCord(<work coordinate number>)=<work coordinate data>  'Setting
```

[Terminology]

<Position variable>	A position variable to which a value is to be assigned is designated.
<Work coordinate number>	A work coordinate number which is chosen from 1 through 8. Constants, variables, logic/arithmetic expressions, and functions are usable. When a real number or a double-precision real number is specified, the fractional portion of 0.5 or over of the number is counted as one and the rest is cut away.
<Work coordinate data>	Work coordinate data is specified with a position constant or a position variable. Values to be specified (coordinate values) represent the position of the origin point of a work coordinate system viewed from the base coordinate system.

[Reference Program]

(1) Example when a new work coordinate 2 based on the existing work coordinate 1 is used as a new world coordinates system

```
1 PW=P_WkCord(1)      ' Read work coordinate 1 (set value for parameter: WK1CORD) and assign it to
                        PW.
2 PW.X=PW.X+100       'Add 100 to X coordinate value that has been read.
3 PW.Y=PW.Y+100       'Add 100 to Y coordinate value that has been read.
4 P_WkCord(2)=PW      'Set the results of the above operations for work coordinate 2.
                        (Set them to parameter: WK2CORD).
5 Base 2              'Let work coordinate 2 be a new world coordinate system.
6 Mov P1
```

(2) Specification of PA as a work coordinate 4 and Ex-T control linear interpolation movement along the work coordinate 4

```
1 P_WkCord(4)=PA      ' Specify PA as a work coordinate 4. (Set it to parameter: WK4CORD.)
2 EMvs 4,P1           ' Moves to P1 along the work coordinate 4 by Ex-T control linear interpolation.
```

[Explanation]

(1) By designating a work coordinate number, work coordinate values concerned are read, or work coordinate values are specified. The "1 to 8" specified as work coordinate number correspond to parameter: WK1CORD-WK8CORD.

(2) Elements X, Y and Z of work coordinate data indicate the amount of translation from the origin point of the base coordinate system to that of the work coordinate system.
Elements A, B, and C indicate how much the work coordinates system is tilted relative to the robot's coordinates system.

- X ..... Distance the robot hand translates in the direction of the X axis
- Y..... Distance the robot hand translates in the direction of the Y axis
- Z..... Distance the robot hand translates in the direction of the Z axis
- A..... Angle the robot hand rotates on the X axis
- B..... Angle the robot hand rotates on the Y axis
- C..... Angle the robot hand rotates on the Z axis

Elements A, B, and C are set to take a clockwise move as a forward rotation looking at the plus side from the origin point of the work coordinate system.

(3) There is nothing significant about the structure flag.

(4) Specifying work coordinates by this command clears WO, WX and WY data for the corresponding work coordinate number [coordinate values of 3 points to be taught as work coordinates - parameters: WKnWO, WKnWX, WKnWY (n: 1~8)].
Example) Executing Step 4 (P_WKcORD(2)=PW) of Reference Program <1> which is previously listed causes WK2WO, WK2WX and WK2WY to be set to "0".
(5) Only MELFA-BASIC V/VI can be used.
(6) In the Ex-T control interpolation command, this variable is referred to by the designated work coordinate number and used as the Ex-T control point. Therefore, specifying P_WkCord also specifies the control point coordinates in the Ex-T control.

[Related instructions]
Base (Base), EMvc (E Move C), EMvr (E Move R), EMvr2 (E Move R 2), EMvr3 (E Move R 3), EMvs (E Move S)

[Related parameter]
MEXBSNO, WKnCORD("n" is 1 to 8), WKnWO, WKnWX, WKnWY("n" is 1 to 8)

P_Zero

[Function]
Always returns (0,0,0,0,0,0,0,0)(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)(0,0) is assigned to P1.

[Explanation]
(1) This can be used to initialize the P variable to zeros.
(2) This variable only reads the data.

4.14 Detailed Explanation of Functions

4.14.1 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.

4.14.2 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

ACos

[Function]
Returns the arc cosine from the specified cosine.

[Format]

<Numeric Variable>=ACos(<Equation>)

[Terminology]

<Numeric Variable>	Calculates the arc cosine with specified expression, and returns the result. The unit is radian. Setting range: 0 to PI
<Equation>	Specifies the cosine. Setting range: -1.0 to +1.0

[Reference Program]

```
1 MRad=ACos(0.6)        ' Assigns the arc cosine of 0.6 (0.927295218001612 rad) to the MRad.
```

[Explanation]
(1) Returns the arc cosine from the specified equation. Unit is in radians.

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
```

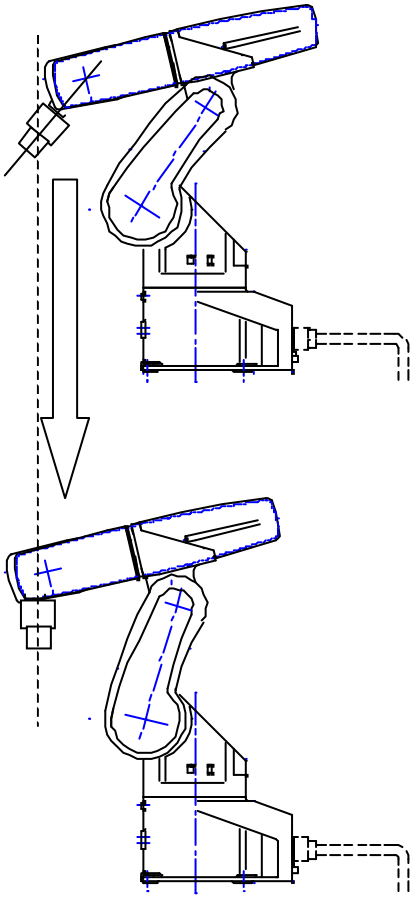
[Explanation]

(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")                ' &H41is 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

ASin

[Function]
Returns the arc sine from the specified sine.

[Format]

```
<Numeric Variable>=ASin(<Equation>)
```

[Terminology]

<Numeric Variable>	Calculates the arc sine with specified expression, and returns the result. The unit is radian. Setting range: -PI/2 to +PI/2
<Equation>	Specifies the sine. Setting range: -1.0 to +1.0

[Reference Program]

```
1 MRad=ASine(-0.4)      ' Assigns the arc sine of -0.4 (-0.411516846067488 rad) to the MRad.
```

[Explanation]
(1) Returns the arc sine from the specified equation. Unit is in radians.

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 < Atn < PI/2.
(3) The range of the returned value for Atn2 is -PI < 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 exampleM1=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 command.
<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: Positions 1, 2, and 3 had the exact same position or all three points were on a straight line. -2: The circular arc generation failed.

[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 an 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("FFFFFFFF")            ' +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)                     ' e^2 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              ' Return base conversion data to the initial value.
2 P10=Fram(P1,P2,P3)        ' Create P10 coordinate system based on P1, P2 and P3 positions.
3 P10=Inv(P10)              ' Convert (inversely convert) P10 to what represents the position of the
                              origin point of base coordinate system viewed from P10.
4 Base P10                  ' Newly establish the position of P10 defined in Step 2 in the world
                              coordinate system.
      :
```

[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(FPrm(P01,P02,P03), P04, P05)

[Reference]

Relative conversion (* operator). Refer to Page 519, "5.7 About Standard Base Coordinates".

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 nquation 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.
(4) The initial value of the target mechanism number is "1". Therefore, when mechanism number 1 is targeted, after executing the RelM command, or the program slot is other than 1, execution of the GetM command is unnecessary. If target mechanism is other than 1, execute the GetM command beforehand.

[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)          ' "KO" 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)          ' "100.1" is assigned to C1$.
2 M1=Cvs(C1$)              ' 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)          ' "10000.1" is assigned to C1$.
2 M1=Cvd(C1$)                ' 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.
(4) The initial value of the target mechanism number is "1" under software version N8(SQ series) and P8(SD series). Therefore, when mechanism number 1 is targeted, after executing the RelM command, or the program slot is other than 1, execution of the GetM command is unnecessary. If target mechanism is other than 1, execute the GetM command beforehand.

[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

Rdfl 1

[Function]
Returns the structure flag of the specified position using character data "R"/"L", "A"/"B", and "N"/"F".

[Format]

```
<Character String Variable >=Rdfl1(<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$=Rdfl1(P1,1)               ' C1$ 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]
Rdfl 2, Setfl 1, Setfl 2

Rdfl 2

[Function]
Returns the multiple rotation information of the specified joint axis.

[Format]

<Numeric Variable>=Rdfl2(<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=Rdfl2(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]
Rdfl 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]

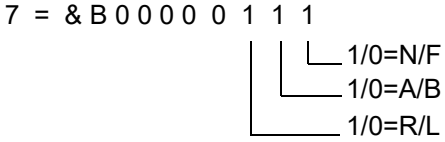
<Position Variables>=Setfl1(<Position Variables>, <Character String>)

[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.



(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]
Rdfl 1, Rdfl 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]

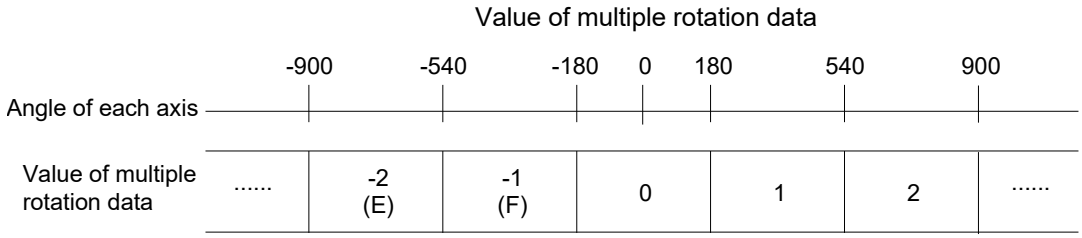
<Position Variables>	Specifies the position variable whose multiple rotation data are to be changed.
<Equation 1>	Specifies the axis number for which the multiple rotation data are to be changed. (1 through 8).
<Equation 2>	Specifies the multiple rotation data value to be changed (-8 through 7).

[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]
Rdfl 1, Rdfl 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

SplECord

[Function]

Saves the Ex-T coordinate system origin data registered in the spline file in the position variable.

[Format]

<Position Variable>=SplECprd(<Spline No.> [, <Frame transformation>])

[Terminology]

<Position Variable> The variables for a result to be substituted.

<Spline No.> The number of the spline file holding information on the path to obtain is designated with a constant or numerical variable.
Setting range: 1 to 99

<Frame transformation> The details for executing frame transformation are designated with a constant or numerical variable.
Setting range:
0: Frame transformation is not executed.
1: Frame transformation is executed using coordinate system set in spline file.
2: Frame transformation is executed using coordinate system set with SetCalFrm command.
When omitted: Frame transformation is not executed.

[Reference Program]

```
1 P_WkCord(1) = SplECord(8)    ' The origin data of the Ex-T coordinate system registered in spline file
                                 8 is substituted to the work coordinate 1.
2 P1 = SplECord(5)             ' The Ex-T coordinate system origin data registered in spline file 5 is
                                 substituted to the position variable P1.
3 P2 = SplECord(10,1)          ' Substitutes frame converted data for the Ex-T coordinate system
                                 origin data registered in spline file 10 for P2.
```

[Explanation]

(1) Returns the Ex-T coordinate system origin data registered in the spline file specified by <Spline No.>.
(2) If a spline file corresponding to the <Spline No.> is not saved in the controller, the error L2610 (Can't open spline file) will occur.
(3) If "1 (execute using coordinate system set in spline file)" or "2 (execute using coordinate system set with SetCalFrm command) is designated in <Frame transformation>, the Ex-T coordinate system origin data is converted based on the designated method, and the adjusted data is returned.
(4) If the coordinate system is not set in the spline file even though "1 (execute using coordinate system set in spline file)" is designated in <Frame transformation>, error L2042 (Frame transformation coordinates are not set) will occur.
(5) If the coordinate system cannot be calculated when "1 (execute using coordinate system set in spline file)" is designated in <Frame transformation>, error L2041 (Can't calculate frame transformation coordinates) will occur.
(6) If "2 (execute using coordinate system set with SetCalFrm command)" is designated for <Frame transformation>, frame conversion is executed using the coordinate system set with the SetCalFrm command executed last. If the SetCalFrm command has not been executed even once and the coordinate system is not set, error L2042 (Frame transformation coordinates are not set) will occur.
(7) If a spline file does not have the Ex-T coordinate settings , the error L2610 (The setting of Ex-T is illegal) will occur.

SplPos

[Function]
Saves a random path point data registered in the spline file in the position variable.

[Format]

<Position Variable>=SplPos(<Spline No.>, <Path point No.> [, <Frame transformation>])

[Terminology]

<Position Variable>	The variables for a result to be substituted. The value of the path point data registered in the specified spline file is substituted to the configuration flag.
<Path point No.>	Specify the path point number to obtain with the constant or the numeric variable. Setting range: 1 to 5000
<Frame transformation>	The details for executing frame transformation are designated with a constant or numerical variable. Setting range: 0: Frame transformation is not executed. 1: Frame transformation is executed using coordinate system set in spline file. 2: Frame transformation is executed using coordinate system set with SetCalFrm command. When omitted: Frame transformation is not executed.

[Reference Program]

```
1 P1 = SplPos(1,5)        ' Substitutes spline file 1 path point 5 position data for position variable P1.
2 P2 = SplPos(1,10,1)     ' Substitutes frame converted data for spline file 1 path point 10 position data for position variable P2.
```

[Explanation]

(1) Returns path point position data corresponding to the <Path point No.> registered in the spline file specified with the <Spline No.>.
(2) The value of the path point data registered in the specified spline file is substituted to the configuration flag.
(3) If a spline file corresponding to the <Spline No.> is not saved in the controller, the error L2610 (Can't open spline file) will occur.
(4) If a path point information corresponding to the <Path point No.> is not saved in the spline file, the error L2610 (Path point is not registered) will occur.
(5) If "1 (execute using coordinate system set in spline file)" or "2 (execute using coordinate system set with SetCalFrm command) is designated in <Frame transformation>, the path point data is converted based on the designated method, and the path point data is returned.
(6) If the coordinate system is not set in the spline file even though "1 (execute using coordinate system set in spline file)" is designated in <Frame transformation>, error L2042 (Frame transformation coordinates are not set) will occur.
(7) If the coordinate system cannot be calculated when "1 (execute using coordinate system set in spline file)" is designated in <Frame transformation>, error L2041 (Can't calculate frame transformation coordinates) will occur.
(8) If "2 (execute using coordinate system set with SetCalFrm command)" is designated for <Frame transformation>, frame conversion is executed using the coordinate system set with the SetCalFrm command executed last. If the SetCalFrm command has not been executed even once and the coordinate system is not set, error L2042 (Frame transformation coordinates are not set) will occur.

SplSpd

[Function]

Calculates the maximum speed which can be specified without an error in the spline interpolation commands (MvSpl, EMvSpl).

[Format]

```
<Numeric Variable>=SplSpd(<Spline No.>)
```

[Terminology]

<Position Variable>	The variables for a result to be substituted.
<Spline No.>	The number of the spline file holding information on the path to obtain is designated with a constant or numerical variable. Setting range: 1 to 99

[Reference Program]

```
1 MS = SplSpd(8)        ' Substitutes the maximum speed that can be specified without an error
                          occurring when spline interpolation is executed using spline file 8 path
                          point data for MS.
2 MvSpl 8,MS,50         ' Executes spline interpolation to pass through the spline file 8 path
                          point at the speed specified with MS.
```

[Explanation]

(1) Calculates the maximum speed which can be specified without an error from the path point data registered in the spline file specified by Spline No.

(2) The calculated maximum speed is the speed when the program override is 100%.

(3) If spline interpolation is executed using the calculated speed, a speed-over error may occur if the robot posture changes greatly.

(4) If a spline file corresponding to the <Spline No.> is not saved in the controller, the error L2610 (Can't open spline file) will occur.

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.

StrLwr

[Function]
Decapitalizes a character string.

[Format]

<Character String Variable> = StrLwr(<Character String>)

[Terminology]

<Character String Variable>	Sets the character string variable to be substituted.
<Character String>	Sets the character string to be decapitalized.

[Reference Program]
```
1 C1$ = StrLwr("ABC")      ' Sets "abc" in C1$.
```

[Explanation]
(1) A character string can be decapitalized.

[Reference]
StrUpr

StrUpr

[Function]
Capitalizes a character string.

[Format]

```
<Character String Variable> = StrUpr(<Character String>)
```

[Terminology]

<Character String Variable>	Sets the character string variable to be substituted.
<Character String>	Sets the character string to be capitalized.

[Reference Program]

```
1 C1$ = StrUpr("abc")       ' Sets "ABC" in C1$.
```

[Explanation]
(1) A character string can be capitalized.

[Reference]
StrLwr

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)

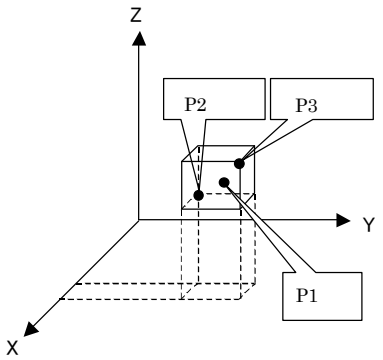
Positions 1 to 3 set the XYZ coordinates variable system (P variable X, Y, Z, A, B, C, L1 and L2).

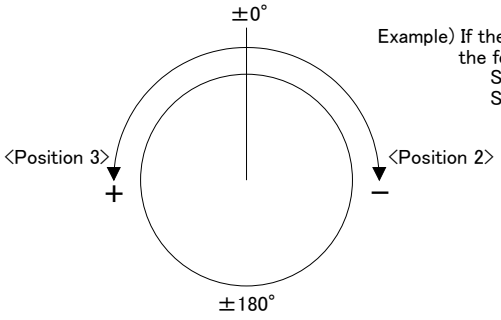
[Reference Program]

```
1 M1=Zone(P1,P2,P3)
2 If M1=1 Then Mov P_Safe Else End
```

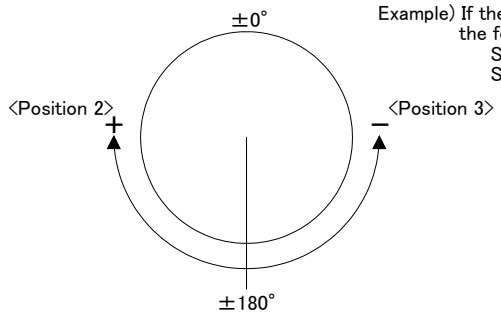
[Explanation]

(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.





Example) If the value passes through 0 from −90 to +90, the following setting is necessity.
Sets the negative value to ABC of <position 2>.
Sets the positive value to ABC of <position 3>.



Example) If the value passes through 180 from −90 to +90, the following setting is necessity.
Sets the positive value to ABC of <position 2>.
Sets the negative value to ABC of <position 3>.

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.

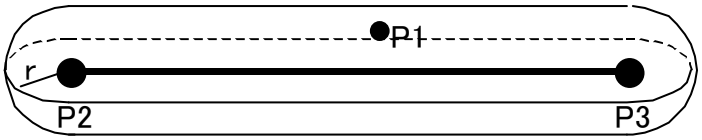
[Reference Program]

```
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>=Zone3(<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]

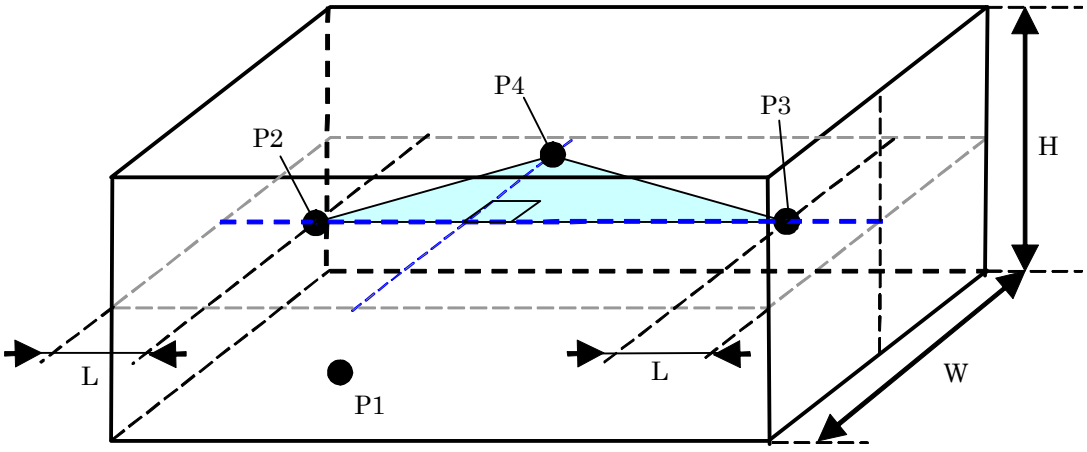
[Reference Program]

```
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 below. 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 484
2	Signal parameter	These parameters set the items pertaining to signals. For the CR800-Q series devices, replace the "CPU buffer memory" described in this chapter with the "CPU shared memory".	Page 499
3	Operation parameter	These parameters set the items pertaining to the operations of the controller, T/B and so forth.	Page 507
4	Command parameter	These parameters set the items pertaining to the robot language.	Page 510
5	Communication parameter	These parameters set the items pertaining to communications.	Page 514

For the parameters regarding dedicated I/O signals, refer to Page 625, "6.2 PLC link I/O function". 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 84, "3.15 Operation of parameter screen".

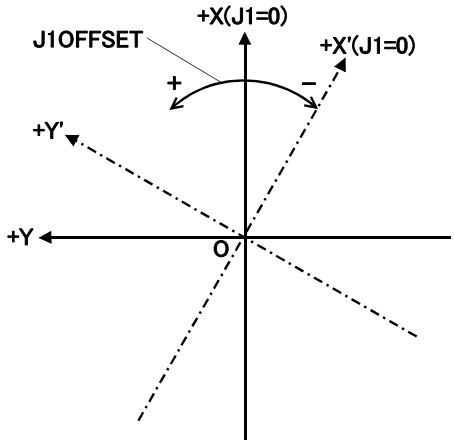
⚠CAUTION When changing parameters, check thoroughly the function and setting values first. Otherwise, the robot may move unexpectedly, which could result in personal injury or property damage.

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. Note) Please note that the joint movement range of J1 axis cannot be changed after the J1 axis offset angle (J1OFFSET) is specified in vertical 5-axis type robot. 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
Narrow angle/Wide angle limit function * Only RV-4FR/ 7FR/13FR/20FR series	MELTEXS	Integer 1	The function protected so that the robot arm head may not interfere with the base is operating. Designate the valid/invalid of this function. Sets "0" to this parameter, when it is necessary to move the head of arm near the base. In this case, take care that the arm head and the base do not interfere. Refer to Operating range diagram in separate manual: “Standard Specifications” about area of Narrow angle/Wide angle limit. 0:Invalid 1: Valid	1(Valid) * Only RV-4FR/ 7FR/13FR/20FR series. Other types are 0 (invalid).

Parameter	Parameter name	No. of arrays No. of characters	Details explanation	Factory setting
J1 axis offset angle	J1OFFSET	Real value 2	Specify the J1 axis offset angle for vertical 5-axis type robot. This setting is invalid for other types of robot, do not change the default setting. Note) Since the J1 axis direction is changed from J1=0 degree (front direction) by specifying parameter J1OFFSET, the joint movement range (MEJAR) is automatically corrected. Please note that the joint movement range of J1 axis cannot be changed while parameter J1OFFSET is set. Element 1: Specify whether to enable/disable this function. Set 1.0 to enable, or 0.0 to disable this function. Element 2: Specify the offset angle. The J1 axis direction is shifted by the set angle from J1=0 degree (X axis). (Unit: deg.) Setting range: -360 <= [Element 2] <= +360  When parameter J1OFFSET is set, the origin setting is required in the J1 axis. Set the origin by procedure below. Refer to the separate manual: "Robot arm setup & maintenance" for details of the origin setting. 1) Set the value to the parameter J1OFFSET. 2) Power ON the controller again. 3) Align the J1 axis with zero degree position. 4) Perform J1 origin setting by jig method 5) Display the joint coordinate values in T/B's jog screen, and confirm that the angle of J1 axis is what was set up at 1). 6) Record the changed origin data to the followings. - Origin data label inside of the robot.	0.0,0.0
Standard tool coordinates Refer to "4.4Coordinate system description of the robot", "5.6Standard 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 to 16 Refer to "M_Tool"	MEXTL1 : MEXTL16	Real value 6 each	If the M_Tool variable is substituted by 1 to 16, the tool conversion data can be switched using this parameter value each.	(X,Y,Z,A,B,C) = 0.0,0.0,0.0,0.0,0.0,0.0
Tool function settings	TOOLSPEC	Integer 1	Select whether to reset tool parameter MEXTL (all components to 0) at the same time as when tool number 0 is selected. 0: Reset 1: Not reset Supported with controller software version A4 or later.	0 (Reset)

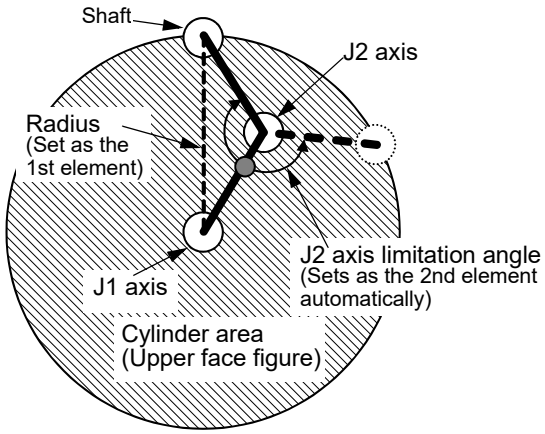
<table>
<tr><th>Parameter</th><th>Parameter name</th><th>No. of arrays
No. of characters</th><th>Details explanation</th><th>Factory setting</th></tr>
<tr><td>Tool base coordinates

Refer to "4.4Coordinate system description of the robot", "5.7About Standard Base Coordinates"</td><td>MEXBS</td><td>Real value 6</td><td>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.
Note) The value cannot be changed during program execution or pausing.</td><td>(X,Y,Z,A,B,C) = 0.0,0.0,0.0,0.0,0.0,0.0</td></tr>
<tr><td>Standard base coordinates

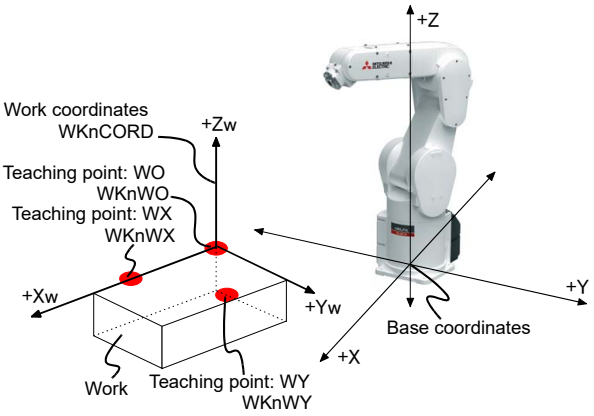
Refer to "4.4Coordinate system description of the robot"</td><td>MEXBSNO</td><td>Real value 1</td><td>Sets world coordinate system by specifying a base coordinate number (base conversion).
Displays current settings, as well.
Description of set values:
0:Designates P_NBase (system's initial value). (Because P_NBase = (0, 0, 0, 0, 0, 0), base conversion is cleared.)
1~8: Designates a set value for work coordinate systems 1 through 8 (parameters: WK1CORD through WK8CORD).
-1:Base conversion data is specified directly by a base command or by a reference base coordinate parameter MEXBS.
(Note: The set value "-1" is valid for read only.)
Note) The value cannot be changed during program execution or pausing.</td><td>-1</td></tr>
<tr><td>Base coordinate data for system</td><td>MEXSBS</td><td>Real value 6</td><td>Sets the positional relationship between the world coordinate system and the base coordinate system.
The world coordinate system and the base coordinate system match at factory default.
This parameter is not changed by the Base command.</td><td>(X,Y,Z,A,B,C) = 0.0, 0.0, 0.0,0.0, 0.0, 0.0</td></tr>
<tr><td>Tool coordinate data for system</td><td>MEXSTL</td><td>Real value 6</td><td>Sets the initial values for the relationship between the hand tip (control point) and the mechanical interface (hand mounting surface).
This parameter is not changed by the Tool command or M_Tool.</td><td>(X,Y,Z,A,B,C) = 0.0, 0.0, 0.0,0.0, 0.0, 0.0</td></tr>
<tr><td rowspan="3">User area

Refer to "5.8About user-defined area"</td><td colspan="3">Specify the user definition area (maximum of 32 area) and the action when the robot enters in the area.</td><td></td></tr>
<tr><td>AREA*CS
* is 1 to 32</td><td>Integer 1</td><td>Specify the coordinate system of the user definition area *.
0: Base coordinate system (conventional compatibility)
1: Robot coordinate system</td><td>0</td></tr>
<tr><td>AREA*P1
* is 1 to 32</td><td>Real value 8</td><td>Designates position coordinates of the diagonal point 1 of the user-defined area n and coordinates of posture data/additional axes. Definitions are given, starting with the 1st element, to X, Y, Z, A, B, C, L1, and L2 in the order listed.
<table>
<tr><th>Item</th><th>Details</th><th>Unit</th></tr>
<tr><td>X, Y, Z elements</td><td>Specify position coordinates of the diagonal point 1.</td><td>mm</td></tr>
<tr><td>A, B, C elements</td><td>Specify posture area.</td><td>deg</td></tr>
<tr><td>L1, L2 elements</td><td>Specify additional axis area.</td><td>mm, deg</td></tr>
</table>
<NOTES>
*Specify values in the coordinate system which was designated by AREA*CS.
*If a posture check is not to be made, set A, B and C coordinates to -360.
*If additional axes are used, specify elements L1 and L2.
*In regard to elements X, Y, Z, L1 and L2, defined area remains unchanged if parameter interchange is made to AREA*P2.</td><td>(X,Y,Z,A,B,C,L1,L2) = 0.0, 0.0, 0.0, -360.0, -360.0, -360.0, 0, 0</td></tr>
</table>

<table>
<tr><th>Parameter</th><th>Parameter name</th><th>No. of arrays
No. of characters</th><th>Details explanation</th><th>Factory setting</th></tr>
<tr><td>User area</td><td>AREA*P2
* is 1 to 32</td><td>Real value 8</td><td>Designates position coordinates of the diagonal point 2 of the user-defined area * and coordinates of posture data/additional axes. Definitions are given, starting with the 1st element, to X, Y, Z, A, B, C, L1, and L2 in the order listed.
<table><tr><th>Item</th><th>Details</th><th>Unit</th></tr><tr><td>X, Y, Z elements</td><td>Specify position coordinates of the diagonal point 2.</td><td>mm</td></tr><tr><td>A, B, C elements</td><td>Specify posture area.</td><td>deg</td></tr><tr><td>L1, L2 elements</td><td>Specify additional axis area.</td><td>mm, deg</td></tr></table><NOTES>
*Specify values in the coordinate system which was designated by AREA*CS.
*If a posture check is not to be made, set A, B and C coordinates to +360.
*If additional axes are used, specify elements L1 and L2.
*In regard to elements X, Y, Z, L1 and L2, defined area remains unchanged if parameter interchange is made to AREA*P1.</td><td>(X,Y,Z,A,B,C,L1,L2) = 0.0, 0.0, 0.0, -360.0, -360.0, -360.0, 0, 0</td></tr>
<tr><td></td><td>AREA*ME
* is 1 to 32</td><td>Integer 1</td><td>Designate the mechanism No. for which the user-defined area* is to be validated.
The mechanism No. is 1 to 3, but normally 1 is set.
0: Invalid (Don't do the area check)
1: Mechanism 1 (usually set up)
2: Mechanism 2
3: Mechanism 3</td><td>0</td></tr>
<tr><td></td><td>AREA*AT
* is 1 to 32</td><td>Integer 1Outside of the area</td><td>Specify desired behavior when the robot enters the user-defined area.
0: Invalid (This function will be invalid)
1: In-zone signal output (The dedicated output and the status variable output)
2: Error output.
<Details of the setting>
<table><tr><th>Setting</th><th>Inside of the area</th><th>Outside of the area</th></tr><tr><td rowspan="2">Signal output</td><td>The dedicated output signal ON (*1)</td><td>The dedicated output signal OFF</td></tr><tr><td>Turn on the correspondence bit of the status variable. (*2)</td><td>Turn off the correspondence bit of the status variable.</td></tr><tr><td>Error output</td><td>The stop by the error output (H2090 error occurrence)</td><td>-</td></tr></table>*1: Set up the signal number of the dedicated I/O by USRARE
*2: System status variable (M_Uar32, M_Uar)
<NOTES>
If error output is opted for, a check is performed only in the position area, ignoring the posture area and additional axis area.</td><td>0(Invalid)</td></tr>
</table>

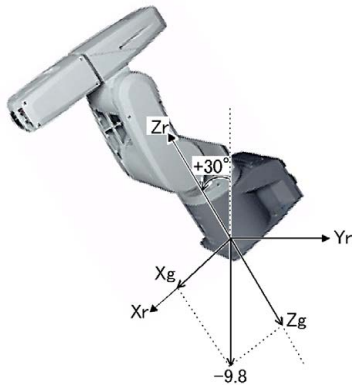
Parameter	Parameter name	No. of arrays No. of characters	Details explanation	Factory setting
Free plane limit Refer to "5.9Free plane limit"	This is the overrun limit set on a free plane. Create a plane with three coordinate points, and set the area that does not include the origin as the outside-movement 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)=0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0
	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)
Cylinder limit *Available robot Robot: RH-3FRHR series	Movement can be limited in the cylinder area centering on the J1 axis. Set the radius centering on the J1 axis as the parameter			
	MECAR	Real value 2	Constitution of the parameter : (The radius, J2 axis limitation angle) Set the radius as the 1st element. Unit: mm(Valid to two digits decimal) If the radius is 0, this function does not function. If the radius is minus value or the distance can not arrive, the error occurs at inputting the value. The cylinder limit makes the limitation of J2 axis operating area The limitation angle of the J2 axis is calculated automatically and sets as the 2nd element in the absolute value. (read only, can not change) * As compared with the Joint movement range parameter (MEJAR), limitation of operation is performed in the smaller one.  Notes) This function is limited to RH-3FRHR series.	0, 0
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
Align type selection	ALIGNTYP	Integer 1	Specifies how to find the hand posture used for the hand alignment movement and the Align function. The robot performs the hand alignment movement toward its hand posture. The Align function returns the XYZ coordinate values set for the hand alignment. 0: Normal For each of the A, B, and C components of the XYZ coordinate posture, find the closest XYZ postures (0 degree, +/-90 degrees, and +/-180 degrees). Specify this method for the RV-type robot. 1: Cylindrical Find the position whose posture axis (C) is changed along the straight line connecting the origin ((x, y)=(0, 0)) and the control point. When the workpiece is arranged cylindrically for the RH-type robot, this method may be useful. Do not specify this method for the RV-type robot. 2 or later: Reserved	0
Origin setting completion status	MEINST	Integer 1	Indicates the origin setting completion status. By setting the origin, this parameter is automatically set.	0
Pulse number for the origin setting	MEINSZ	Integer 8	Sets the number of pulses at 0 position for the origin setting. By setting the origin, this parameter is automatically set.	0,0,0,0,0,0,0,0
Pulse position within one revolution for the origin setting	MEOFFZ	Integer 8	Sets the pulse position within one revolution at 0 position for the origin setting. By setting the origin, this parameter is automatically set.	0,0,0,0,0,0,0,0
Check data for the origin setting	MEINSD	Integer 1	Sets the check data for MEISZ. By setting the origin, this parameter is automatically set.	0
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.
Select the function of singular point adjacent alarm Refer to "5.17About 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

Parameter	Parameter name	No. of arrays No. of characters	Details explanation	Factory setting
Work coordinates	WKnCORD "n" is 1 to 8	Real value 6	The work coordinates for work jog operation (X,Y,Z,A,B,C) Unit: mm or degree It is used as standard coordinates and work coordinate data in the work jog. It is also used as the control point in the Ex-T control (Ex-T coordinates). Refer to Page 750, "7.3 Ex-T control" for Ex-T control. When using it as work coordinate data, the valid axial element differ depending on the robot type. Refer to Page 519, "5.7 About Standard Base Coordinates". The work coordinates defined by operation of T/B are set. However, inputting the coordinate value into this parameter can also define work coordinates. In this case, each coordinate value of the three teaching points for defining the work coordinates is cleared by 0. (Parameter: WKnWO, WKnWX, WKnWY ("n" is 1-8))  Note) To manage easily, you should teach work coordinates in the condition that not convert the base coordinates. (Base coordinates and the world coordinate are in agreement.)Especially, it is necessary when defining two or more work coordinates.	(0.00, 0.00, 0.00, 0.00, 0.00, 0.00)
	WKnWO "n" is 1 to 8	Real value 3	Set the position of the work coordinates origin as a teaching position of work coordinates. (Correspond to "WO" of the teaching operation by T/B. Refer to above figure) (X, Y, Z) Unit: mm Notes) The work coordinates are not defined by merely inputting this coordinate value. Calculate the work coordinates by performing [DEFINE] on the TB work coordinate setting screen or by performing [Write] on the RT ToolBox3 work coordinate parameter screen.	(0.00, 0.00, 0.00)
	WKnWX "n" is 1 to 8	Real value 3	Set the position of "+X" axis of work coordinates as a teaching position of work coordinates. (Correspond to "WX" of the teaching operation by T/B. Refer to above figure) (X, Y, Z) Unit: mm Notes) The work coordinates are not defined by merely inputting this coordinate value. Calculate the work coordinates by performing [DEFINE] on the TB work coordinate setting screen or by performing [Write] on the RT ToolBox3 work coordinate parameter screen.	(0.00, 0.00, 0.00)
	WKnWY "n" is 1 to 8	Real value 3	Set the position at the side of "+Y" axis on the X-Y plane of work coordinates. (Correspond to "WY" of the teaching operation by T/B. Refer to above figure) (X, Y, Z) Unit: mm Notes) The work coordinates are not defined by merely inputting this coordinate value. Calculate the work coordinates by performing [DEFINE] on the TB work coordinate setting screen or by performing [Write] on the RT ToolBox3 work coordinate parameter screen.	(0.00, 0.00, 0.00)

Parameter	Parameter name	No. of arrays No. of characters	Details explanation	Factory setting
WORK jog operation mode	WKnJOGMD "n" is 1 to 8	Integer 1	The operation mode in WORK jog operations is specified for each work coordinate. 0: WORK jog (The operation of A, B, and C elements is rotation around the axis parallel to the X, Y, and Z axes of the work coordinates. The control point position does not change.) 1: Ex-T jog (The operation of A, B, and C elements is rotation around the X, Y, and Z axes of the work coordinates with the control point moving.	0
Selected work coordinate number	WKJOGNO	Integer 1	Sets the selected work coordinate number. By changing the work coordinates, the parameter is automatically set.	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

<table>
<tr><th>Parameter</th><th>Parameter name</th><th>No. of arrays
No. of characters</th><th>Details explanation</th><th>Factory setting</th></tr>
<tr><td>The gravity direction

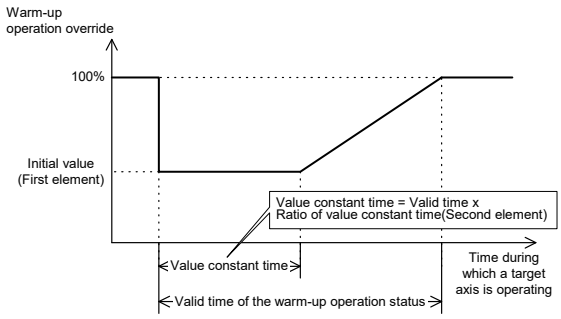
* Only RV-FR series.</td><td>MEGDIR</td><td>Real value 4</td><td>This parameter specifies the direction and magnitude of gravitational acceleration that acts on the robot according to the installation posture for the X, Y, and Z axes of the robot coordinate system, respectively (unit: mm/second^2).
There 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.
<table><tr><th>Installation posture</th><th>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)</th></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 Note1)</td><td>(2.0, 0.0, 0.0, 0.0)</td></tr><tr><td>Optional posture Note2)</td><td>(3.0, ***, ***, ***)</td></tr></table>Note1) In RH-3FRHR series, the current setting of first element (Installation posture; 0.0) should not be changed.
Note2) "***" indicates the numerical value
Refer to the lower example about the set value of the optional postures.
The example of the setting of gravity acceleration is shown below.
Example: If the robot is tilted 30 degrees forward (see the figure below):
The direction gravity acceleration of X axis (Xg) = 9.8 x sin(30 degrees) = 4.9 .
The direction gravity acceleration of the Z axis (Zg) = 9.8 x cos(30 degrees) = 8.5 .
Note that the value is set to -8.5 because the direction is opposite to the Z axis of the robot coordinate system.
The direction gravity acceleration of the Y axis (Yg) = 0.0
Therefore, the set value is (3.0, 4.9, 0.0, and -8.5)
 
</td><td>0.0, 0.0, 0.0, 0.0</td></tr>
<tr><td>Hand initial state

Refer to "5.13About default hand status"</td><td>HANDINIT</td><td>Integer 8</td><td>Set the pneumatic hand I/F output for when the power is turned ON.
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) (specifying a signal other than one in 900’S by the HANDTYPE parameter), do not use this HANDINIT parameter, but use the ORS* parameter.
The value set by the ORS* parameter becomes the initial value of signals at power ON.</td><td>1,0,1,0,1,0,1,0</td></tr>
</table>

Parameter	Parameter name	No. of arrays No. of characters	Details explanation	Factory setting
Hand type Refer to "5.12About the hand type"	HANDTYPE	Character string 8	Set the single/double solenoid hand type and output signal No. (D: double solenoid, S: single solenoid). Set the signal No. after the hand type. When D900 is set, the signal No. 900 and 901 will be output. In the case of D (double solenoid), please configure the setting so that the signals do not overlap	D900,D902,D904,D906,,,,
Hand I/O type	HIOTYPE	Integer 1	Set the electric specifications (sink type/source type) of the built-in air hand interface. Please set the value according to the specification of your solenoid valve, etc. -1: Not set, 0: Source type, 1: Sink type	-1
Hand and workpiece conditions (Used in optimum acceleration/ deceleration and impact detection) Refer to "5.16Hand 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. Note) You should set up the hand and work-piece conditions correctly. If a setting is performed in such a way that it falls below the mounted load actually, the life span of the mechanism elements used in the robot may be shortened.			
	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	Setting value for each mechanism
	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	0.0,0.0,0.0,0.0,0.0,0.0, 0.0
	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	0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0
	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 "5.16Hand 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)	1

Parameter	Parameter name	No. of arrays No. of characters	Details explanation	Factory setting
Optimum acceleration/ deceleration adjustment rate	JADL	Real value 8	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. In the RV-SD series, the initial values have been set so as to prevent overload and overheat errors from occurring. They are applied to both the deceleration and acceleration speeds. * 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. * 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. Note) This function is valid only in the RV-SD series.	Setting value for each mechanism
Acceleration-and-deceleration optimization pattern selection * Only for RH-FR series (Without RH-CR/RH-3FRHR series)	MAPMODE	Integer 1	In RH-F series (without RH-3FRHR series), choose the standard acceleration-and-deceleration rate or the high acceleration-and-deceleration rate for the acceleration-and-deceleration optimization function corresponding to the height of the shaft (J3 axis). 0: Standard acceleration-and-deceleration rate (initial value), 1 : High acceleration-and-deceleration rate. *Initial setting is the standard acceleration-and-deceleration rate, and vibration (remains vibration to include) is suppressed to the minimum. When this vibration does not affect the robot's operations, the high acceleration-and-deceleration rate can be chosen, and the robot can be operated at high speed. Refer to the separate "standard specification" for details of the acceleration-and-deceleration rate.	0
Optimum speed control adjustment coefficient	OPTOVC	Real value 1	Set the adjustment coefficient for the optimum speed control. Setting range: 0.30 to 1.00 The optimum speed control is a function to use the Spd command to designate the optimum speed for the linear and the circular arc interpolation movements that do not specify the speed by themselves. Increasing the setting value enables the robot to operate in high speed. However, when the operation posture of the robot is largely changed near the position such as a singular point, a speed overlimit error occurs easily. In such cases, reducing the setting value suppresses the occurrence of the speed overlimit error but lowers the operation speed of the robot. * Speed overlimit error The error number is H213n. (The letter "n" indicates the axis number 1 to 8.) This error occurs when the setting value of the speed command to the motor exceeds the permissible value.	It varies with models.
Speed optimization interpolation functional switch *Available robot RH-3FRHR series	SPDOPT	Integer 1	Set enable/disable of speed optimization interpolation function just after the power supply turned on 1: Enable (Enable the speed optimization interpolation function at the power on) 0: Disable (Disable the speed optimization interpolation function at the power on) If the value of this parameter is 1 or 0, it is possible to switch between enabling and disabling the speed adjustment interpolation function using the SpdOpt instruction in a program. Note) This function is supported by limited models of RH-3FRHR series.	RH-3FRHR series is 1 (Other model are 0)

Parameter	Parameter name	No. of arrays No. of characters	Details explanation	Factory setting
Impact Detection Note that this parameter cannot be used together with the multi-mechanism control function.	COL	Integer 3	Define whether the impact detection function can/cannot be used, and whether it is enabled/disabled immediately after power ON. Element 1: The impact detection function can (1)/cannot (0) be used. Element 2: Enable (1)/disable (0) as the initial state at automatic operation. Element 3: Enable (1)/disable (0)/NOERR mode (2) during jog operation 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).	RH-3FRH/6FRH/ 12FRH/20FRH series: 1,0,1 RV-FR series: 0,0,1 RH-3FRHR series: 1,1,1
Detection level	COLLVL	Integer 8	Set the initial value of the detection level (sensitivity) of each joint axis during automatic operation. Setting range: 1 to 500, unit: %	The setting varies depending on the model.
Detection level during jog operation	COLLVLJG	Real value 8	Set the detection level (sensitivity) during jog operation (including pause status) for each joint axis. Unit: % Decrease the value to increase the detection level (sensitivity). Increase the value if an impact detection error occurs even though no impact is detected during jog operation. Setting range: 1 to 500, unit: %	The setting varies depending on the model.
Servo status after collision detection	COLSERVO	Integer 1	Specifies the servo status after the collision detection. For this function, using the impact detection in the NOERR mode is required. 0: Servo off 1: Servo on held	0
Selection of wrist rotation angle (axis A) coordinate system	RCD	Integer 1	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. 2: General angle method 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.	2 (general angle method)
Warm-up operation mode setting	WUPENA	Integer 1	Designate the valid/invalid of the Warm-up operation mode. 0:Invalid 1: Valid Note: If a value other than the above is set, everything will be disabled. Note: For multiple mechanisms, this mode is set for each mechanism.	0(Invalid)
Warm-up operation mode target axis	WUPAXIS	Integer 1	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). Bit ON: Target axis Bit OFF: Other than target axis A joint axis that will generate an excessive difference error when operated at low temperature will be a target axis. Note: If the bit of a non-existent axis is set to ON, it will not be a target axis. Note: If there is no target axis, the warm-up operation mode will be disabled. Note: For multiple mechanisms, this mode is set for each mechanism.	0

Parameter	Parameter name	No. of arrays No. of characters	Details explanation	Factory setting
Warm-up operation mode control time	WUPTIME	Real value 2	Specify the time to be used in the processing of warm-up operation mode. (Valid time, resume time) Unit: min. 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) 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 valid time is 0 min, the warm-up operation mode will be disabled. Note: For multiple mechanisms, this mode is set for each mechanism.	1, 60
Warm-up operation override	WUPOVRD	Integer 2	Perform settings pertaining to the speed in the warm-up operation status. (Initial value, ratio of value constant time) Unit: % 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) The correspondence between the values of warm-up operation overrides and the setting values of various elements is shown in the figure below.  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.	70, 50

Parameter	Parameter name	No. of arrays No. of characters	Details explanation	Factory setting
Functional setting of compliance error	CMPERR	Integer 1	Setting this parameter prevents errors 2710 through 2740 (errors that occur if the position command generated in compliance control is abnormal) from occurring. 1: Enable error generation 0: Disable error generation 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 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.	1 (Enable error generation)
Current limit level for Cmp Jnt	CMPJCLL	Integer 1	Change the motor current limit level of each axis in the compliance mode (Cmp Jnt instruction) of the joint coordinate system. Setting range: 1 (High) to 10 (Low). While operation is performed in the compliance mode of the joint coordinate system, if the Excessive error 1 (H096n) occurs, increase the set value of this parameter to suppress the error. (The set valve after the change will be applied from the next execution of Cmp Jnt instruction.)	1
Optimization of overload detection level	OLTMX	Integer 1	Set the upper limit of the ambient temperature for the robot's working environment. The overload detection level for robot movement is optimized based on this setting value. (Unit: °C) Setting range: 0-40 Please refer to Page 565, "5.22 Optimizing the overload level" for details of this parameter's function.	RH-3FRH/6FRH/ 12FRH/20FRH: 40 RV-2FR series: 40 RV-4FR/7FR/13FR/ 20FR series, RH-3FRHR: 30
Motor overheat warning	MOHW		Specifies whether or not to warn before a motor overheat error occurs. 0: Disable 1: Enable	1
Occurrence interval of battery error	ITBATERR	Integer 1	Specifies a time interval at which to generate a battery exhaustion time error alarm (in hours) Setting range: 1 to 336 When a set value is less than 1, "1" is taken as being specified; when a set value is greater than 336, "336" is taken as being specified	24
Synchronize time with PLC (CR800-R/Q only)	TIMESYNC	Integer 1	Choose whether to synchronize time of the robot controller and the PLC. (Synchronize/ Not synchronize = 1/0)	1
Definition of the Wait command function	PRSPEC	Integer 1	Set the execution method of the Wait command. 0: Conditional command (two or more conditional expressions can be written.) 1: Operation command (only one conditional expression can be written.)	0
WthIf command function	WTHFUNC	Integer 1	Specifies a stop type of WthIf command. 0: stop type 1 1: stop type 2 Refer to a explanation of Def Act command about the stop type.	0

Parameter	Parameter name	No. of arrays No. of characters	Details explanation	Factory setting
Specification of the end conditions of an operation command when Cnt command is invalid	MVTERM	Integer 1	Specifies the end conditions of an operation command when Cnt command is invalid. For the FR series (RV-2FR/2FRL and RH-3FRHR are excluded), the completion conditions for movement instructions have been changed in order to achieve higher target position attainability performance when Cnt is set to 0. The operation time may become longer when a program created by any model earlier than the FR series (SD/SQ series) is used by the FR series (RV-2FR/2FRL and RH-3FRHR are excluded) as it is. When priority is given to the operation speed, change this parameter and use the Fine P or Dly instruction, etc., for positions that require positioning. 0: speed command output is completed 1: corresponds to Cnt 1 2: speed command generation is completed	0: FR series (RV-2FR/2FRL and RH-3FRHR are excluded.) 2: RV-2FR/2FRL, RH-3FRHR series
Safe point return mode	ESCMODE	Integer 1	Choose tool defining position or mechanical interface origin position as a current position when returning the safe point. 0: set to TCP position (tool defining position) 1: set to mechanical interface origin position	0
Controller CPU drive mode	DRVMODE	Integer 1	Specifies whether the connection with a robot is required or not when simulating by a software. 0: Normal mode (Connection with robot is required.) 1: Robot servo separation mode (Connection with robot is not required.)	0
Feedback gain adjustment coefficient	FBGAIN	Integer 8	Adjusts the speed loop gain for each joint axis. Unit: [%] Setting range: 50 to 150 Setting a smaller value suppresses the motor oscillation or the disturbance influence on driving units. By setting a larger value, the trajectory accuracy or the settling time will be improved, and yet the motor may oscillate. Increase the value gradually while checking the condition.	100, 100, 100, 100, 100, 100, 100, 100
Fixation of gain	PG1	Integer 1	Disables the active gain control and fixes the gain. Sets this parameter when tracking accuracy is poor in tracking operation, etc. Setting range: 0.00 to 120.00 Less than 1.0: Enables the active gain control. 1.0 to 120.0: Fixes the gain. Normally, set to about 20. Reduces the setting value when the vibration occurs. Increasing the setting value may cause vibrations.	0.0
Servo simulation function	SRVSIM	Integer 1	Specifies enable/disable of the load factor simulation for the RT ToolBox3 simulation. 0: Disable 1: Enable	1
Vision sensor calibration data	VSCALBn n=1 to 8	Integer 12	Stores the vision sensor calibration data defined in the 2D vision calibration function of RT ToolBox3. Corresponds to 8 calibration data of the calibration numbers n=1 to 8.	0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0
Setting value when the MVA2 command argument is omitted	MVACNT1	Real value 4	Specifies the setting value when the MVA2 command argument is omitted. Element 1: Neighborhood distance 1 [mm] Setting range: Real value more than or equal to -1/0 Element 2: Neighborhood distance 2 [mm] Setting range: Real value more than or equal to -1/0 * When -1 is specified, the operation is the same as Cnt1. Element 3: Positioning distance [mm] Setting range: Real value more than or equal to 0 Element 4: Settling time [sec] Setting range: Real value more than or equal to 0	0, 0, 0.1, 0

5.2 Signal parameter

These parameters set the items pertaining to signals.

Table 5-2:List Signal parameter

<table>
<tr><th>Parameter</th><th>Parameter name</th><th>No. of arrays
No. of characters</th><th>Details explanation</th><th>Factory setting</th></tr>
<tr><td>Dedicated I/O signal</td><td></td><td></td><td>For the parameters of the dedicated I/O signal, refer to Page 637, "6.3 Dedicated input/output".</td><td></td></tr>
<tr><td>Reads the program number from the numerical input when the start signal is input.</td><td>PST</td><td>Integer 1</td><td>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)</td><td>0(Invalid)</td></tr>
<tr><td>Stop input normal close designation</td><td>INB</td><td>Integer 1</td><td>Change the dedicated input (stop) between the normal open and normal close.
(Normal open/Normal close = 0/1)
The input signal changed is shown below.
*The dedicated input signal: STOP, STOP2 *The dedicated stop input signal for the controller: SKIP Refer to separate manual: “Standard Specifications or Special Specifications” for details of SKIP.</td><td>0(Normal open)</td></tr>
<tr><td>Robot error output</td><td>ROBOTERR</td><td>Integer 1</td><td>Set the error level that opens the robot error output terminal. When shipped from the factory, this setting is set so that it will open at any error level.
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.
<table>
<tr><th rowspan="2">Setting</th><th colspan="3">Error Level</th></tr>
<tr><th>Warning</th><th>Low</th><th>High</th></tr>
<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>
</table>
Refer to the following manual for details.
Instruction Manual: Refer to “External Emergency Stop Connection” for ROBOT ARM SETUP & MAINTENANCE.</td><td>7 (Open for any error level)</td></tr>
<tr><td>CC-Link error release permission.
(* CR800-D only)</td><td>E7730</td><td>Integer 1</td><td>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.
(Enable temporary error cancellation/disable error cancellation = 1/0)

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.</td><td>0 (disable error cancellation)</td></tr>
</table>

Parameter	Parameter name	No. of arrays No. of characters	Details explanation	Factory setting
Output signal reset pattern Refer to "5.14About 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/*) Note) A bit set to "* (maintain)" is turned off when the power is turned on.			
	ORST0	Character string 4	Set the signal No. 0 to 31.	00000000,00000000,00000000,00000000
	ORST32 : ORST8016	Character string 4	Set the signal No. 32 to 63. : Set the signal No. 8016 to 8047 Note) The output signals of 716 to 723 are used for Open or Close of the multi-hand. And, setting of the output signals of 700 to 715 are impossible for the system reservation area.	00000000,00000000,00000000,00000000 :
	ORS10000 : ORS18160	Character string 4	Set the signal No. 10000 to 10031. : Set the signal No. 18160 to 18191.	00000000,00000000,00000000,00000000 :
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 (* CR800-R/Q only)	QMLTCPUN	Integer 1	At the multi CPU system, set the number of CPU units with which the standard base unit is equipped.	2

<table>
<tr><th>Parameter</th><th>Parameter name</th><th>No. of arrays
No. of characters</th><th>Details explanation</th><th>Factory setting</th></tr>
<tr><td>CPU buffer memory periodical communication area setting
(* CR800-R only)</td><td>QMLTCPUn
n=1 to 4</td><td>Integer 4</td><td>At the multi CPU system, read the number of points transmitting and receiving data by the CPU buffer memory periodical communication area of the multi CPU No. 1 to 4 from the multi CPU No. 1, and set it automatically. It is not necessary to change the value.

First element: Size of the periodical communication area (K word)
Range: 0 to 12
*The total size of all multi CPUs is 24K words at maximum.

Second element: No. of auto refresh points (word)
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.

Third element: System reservation

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.</td><td>1,0,1,1</td></tr>
<tr><td>Multi CPU No. n high-speed communication area setting
(CR800-Q series only)</td><td></td><td></td><td>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 PLC CPU If the parameter settings do not match, and therefore care should be taken to ensure that the parameter settings for each CPU match.

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.
<table>
<tr><th>CPU Qty</th><th>Setting Range</th></tr>
<tr><td>2</td><td>0 to 14K points</td></tr>
<tr><td>3</td><td>0 to 13K points</td></tr>
<tr><td>4</td><td>0 to 12K points</td></tr>
</table>
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.
Third element: System area size (K points)
Range: 1 or 2
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.</td><td></td></tr>
</table>

Parameter	Parameter name	No. of arrays No. of characters	Details explanation	Factory setting
Multi CPU input offset (* CR800-R only)	QMLTCPUS	Integer 1	At the CR800-R controller, set the robot input signal offset for the multi CPU. Specify an offset from HG0 in 1K word units, and read as an R/C input from the specified CPU buffer memory. Set as required if mixing other iQ Platform compatible CPUs (motion CPU or NCCPU) and you wish to prevent the CPU buffer memory used at each CPU from overlapping. Setting range: -1 to 14 (integer value) (-1: Not use / 0 to 14 K words) (A) By setting to -1, the offset will be automatically fixed based on the installed slot. (Compatible with previous versions (N4a and prior).) (B) By setting to 0 to 14, the input offset is set based on the value. * Refer to cases (A) and (B) in "5.2.1About multi CPU input offsets (CR800-R/Q controller only)" on the following pages. 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 PLC to multiple robots almost simultaneously. For the multi CPU system, refer to the manual of the RCPU (MELSEC iQ-R CPU Module User's Manual (Application)).	-1
Multi CPU input offset (* CR800-Q series only)			At the CR800-Q series controller, set the robot input signal offset for the multi CPU. 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 iQ Platform compatible CPUs (motion CPU or NCCPU) and you wish to prevent the shared memory used at each CPU from overlapping. Setting range: -1 to 14 (integer value) (-1: Not use / 0 to 14 K words) (A) By setting to -1, the offset will be automatically fixed based on the installed slot. (Compatible with previous versions (N4a and prior).) (B) By setting to 0 to 14, the input offset is set based on the value. * Refer to cases (A) and (B) in "5.2.1About multi CPU input offsets (CR800-R/Q controller only)" on the following pages. 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 PLC to multiple robots almost simultaneously. Refer to the QCPU User's Manual (Multi CPU System Edition) SH(Name)-080475 for details on the multi CPU system.	
Processing mode of the signal output Note1)	SYNCIO	Integer 1	Specify the processing mode of signal output by M_Out/ M_Outb/M_Out8/M_Outw/M_Out16/M_Out32/M_Dout and Def Io. Compatibility mode/High-speed mode 1/High-speed mode 2 Compatibility mode: Process by compatibility conventional, without accelerating the renewal cycle of the signal. High-speed mode 1: Accelerate the signal output of M_Out/ M_Outb/M_Out8/M_Outw/M_Out16/ M_Out32. High-speed mode 2: In addition to the high-speed mode 1, also accelerate the signal output in M_Dout.	2: High-speed mode 2
Dedicated input filter	IOFIL	Integer 1	Prevents erroneous input due to noise, etc. in the dedicated input signals. 0: Disable filter 1: Enable filter	0

Note1) This parameter makes speedy processing of the external output signal by system status variable M_Out etc.
In the program example 1 shown in the following, output signal processing of Steps 1 and 4 gets speedy.
<Program example 1>

```
1 M_Out(9)=1                          'Turn on the output signal 9.
2 *ack_check                          '
3 If M_In(7)=0 Then *ack_check        ' Wait until the input signal 7 turns on (interlock).
4 M_Out(9)=0                          'Turn off the output signal 9.
5 End
```

* Reference value of speed improvement:
In the above-mentioned program example, processing time is reduced about 80%
However, the CC-Link, profibus and parallel I/O interface (card) are effective in case the command of the signal output of two or more lines is continuing.(CR800-D) In the following program example 2, processing time is reduced about 75%.
<Program example 2>

```
1 M_Out(9)=1                          'Turn on the output signal 9.
2 M_Out8(10)=&H1F                     '&H1F is outputted to 8-bit width from the output signal 10.
3 M_Out16(18)=&H3FFF                  '&H3FFF is outputted to 16-bit width from the output signal 18.
4 M_Out32(33)=&H7FFFFFFF              '&H7FFFFFFF is outputted to 32-bit width from the output signal 33.
5 End
```

⚠CAUTION Always make interlock of signal to take synchronization. Failure to observe this could lead to cause of malfunction by the signal transmitted incorrectly.
In addition, the "Conventional compatibility mode" is prepared for if needing the same processing time as the conventional. The initial value of SD series is Conventional compatibility mode. However, sure under the interlocking of the signal, because of to performance improvement, recommends use in the High-speed mode

5.2.1 About multi CPU input offsets (CR800-R/Q controller only)

(1) Case (A)

When using no offset for input (Parameter: when QMLTCPUS = -1)

Table 5-3:CPU buffer memory and robot I/O signal compatibility

PLC (word device)			Robot (bit device)	
Controller	CR800-R	CR800-Q		
Output	U3E0\HG0 to U3E0\HG511	U3E0\G10000 to U3E0\G10511	Input	Robot CPU No.1 / 10000 to 18191
	U3E0\HG512 to U3E0\HG1023	U3E0\G10512 to U3E0\G11023		Robot CPU No.2 / 10000 to 18191
	U3E0\HG1024 to U3E0\HG1535	U3E0\G11024 to U3E0\G11535		Robot CPU No.3 / 10000 to 18191
Input	U3E1\HG0 to U3E1\HG511	U3E1\G10000 to U3E1\G10511	Output	Robot CPU No.1 / 10000 to 18191
	U3E2\HG0 to U3E2\HG511	U3E2\G10000 to U3E2\G10511		Robot CPU No.2 / 10000 to 18191
	U3E3\HG0 to U3E3\HG511	U3E3\G10000 to U3E3\G10511		Robot CPU No.3 / 10000 to 18191

The following figure shows the memory map for the CR800-R series. For the CR800-Q series, refer to the Table 5-3 and replace the device numbers with the corresponding device numbers in the figure.

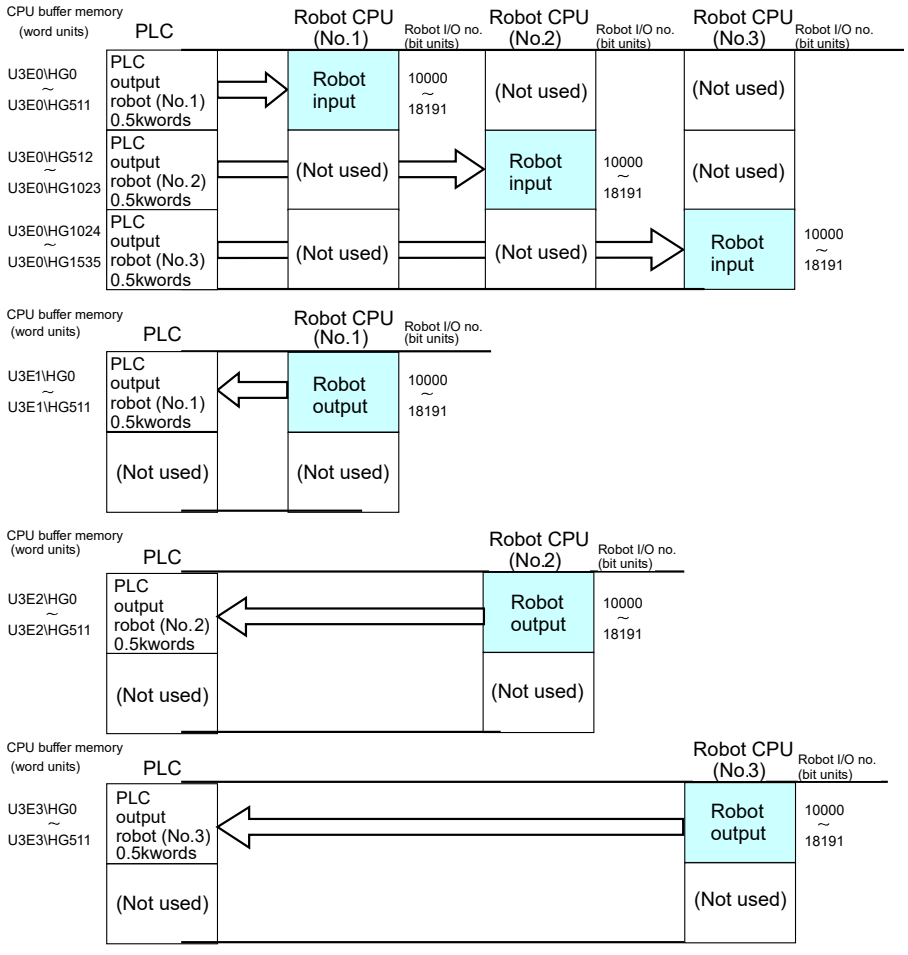


Fig.5-1:CPU buffer memory and robot I/O signal compatibility (Case (A))

(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 buffer memory and robot I/O signal compatibility

PLC (word device)			Robot (bit device)	
Controller	CR800-R	CR800-Q		
Output	U3E0\HG0+(*1) to U3E0\HG511+(*1)	U3E0\G10000+(*1) to U3E0\G10511+(*1)	Input	Robot CPU No.1 / 10000 to 18191
	U3E0\HG0+(*2) to U3E0\HG511+(*2)	U3E0\G10512+(*2) to U3E0\G11023+(*2)		Robot CPU No.2 / 10000 to 18191
	U3E0\HG0+(*3) to U3E0\HG511+(*3)	U3E0\G11024+(*3) to U3E0\G11535+(*3)		Robot CPU No.3 / 10000 to 18191
Input	U3E1\HG0 to U3E1\HG511	U3E1\G10000 to U3E1\G10511	Output	Robot CPU No.1 / 10000 to 18191
	U3E2\HG0 to U3E2\HG511	U3E2\G10000 to U3E2\G10511		Robot CPU No.2 / 10000 to 18191
	U3E3\HG0 to U3E3\HG511	U3E3\G10000 to U3E3\G10511		Robot CPU No.3 / 10000 to 18191

The following figure shows the memory map for the CR800-R series. For the CR800-Q series, refer to the Table 5-4 and replace the device numbers with the corresponding device numbers in the figure.

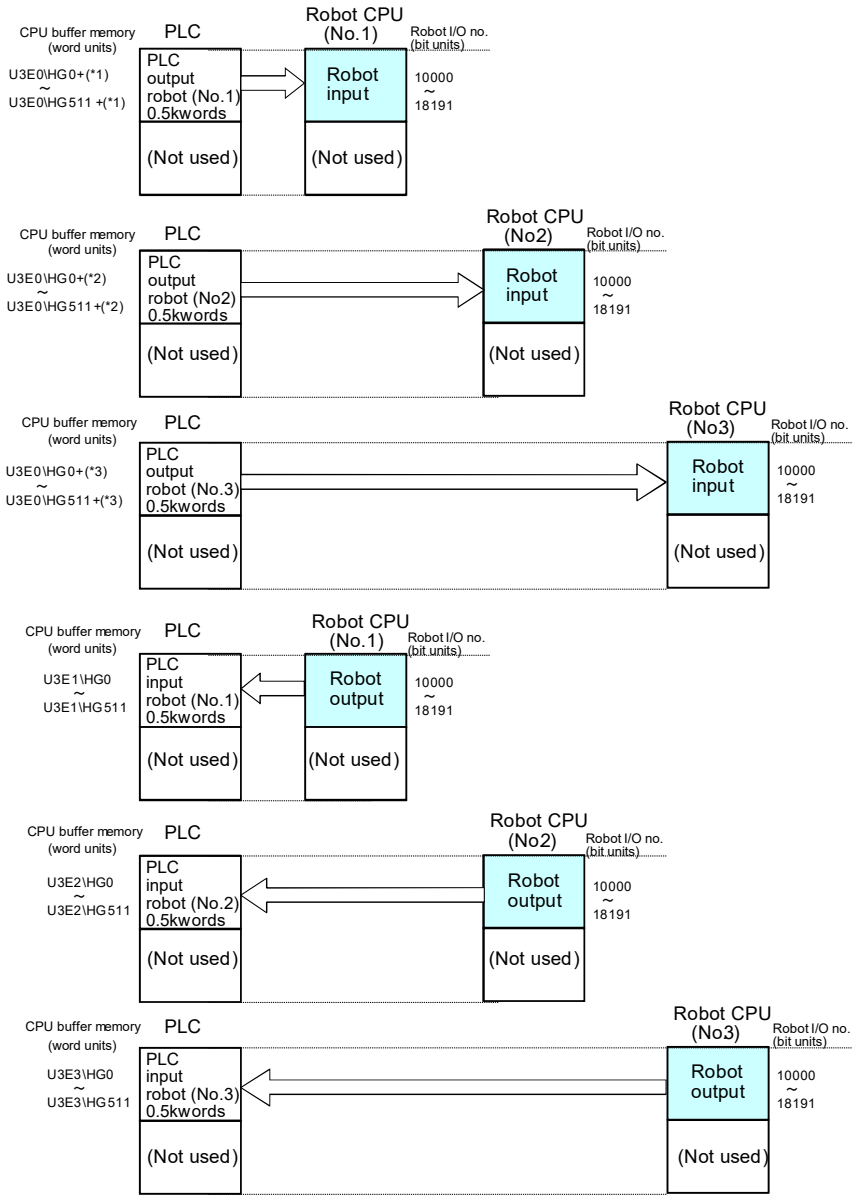


Fig.5-2:CPU buffer memory and robot I/O signal compatibility (Case (B))

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

<table>
<tr><th>Parameter</th><th>Parameter name</th><th>No. of arrays
No. of characters</th><th>Details explanation</th><th>Factory setting</th></tr>
<tr><td>Buzzer ON/OFF</td><td>BZR</td><td>Integer 1</td><td>Specifies the on/off of the buzzer sound that can be heard when an error occurs in the robot controller.
(OFF/ON=0/ÇP)</td><td>1(ON)</td></tr>
<tr><td>Program reset operation rights</td><td>PRSTENA</td><td>Integer 1</td><td>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.</td><td>0(Required)</td></tr>
<tr><td>Program reset when key switch is switched</td><td>MDRST</td><td>Integer 1</td><td>Program paused status is canceled when mode of the controller is changed.
(Invalid/Valid=0/1)</td><td>0(Invalid)</td></tr>
<tr><td>TB override operation rights</td><td>OVRDTB</td><td>Integer 1</td><td>Specifies whether the operation rights are required when changing override from T/B.
(Not required/Required = 0/1)

Note) When OVRDTB = 1 (Required), whether the operation rights are required (T/B enable) or not (T/B disable) depends on the OVRDENA setting and the controller mode. Refer to the following table.
<table>
<tr><th>Mode of the controller</th><th>OVRDENA=0</th><th>OVRDENA=1</th></tr>
<tr><td>AUTOMATIC (T/B disable)</td><td>Cannot changes the override.</td><td>Override is changeable.</td></tr>
<tr><td>MANUAL (T/B enable)</td><td>Override is changeable.</td><td>Override is changeable.</td></tr>
</table></td><td>1(Required)</td></tr>
<tr><td>Speed setting during mode change</td><td>OVRDMD</td><td>Integer 2</td><td>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 AUTOMATIC to MANUAL.
Current status is maintained if changed to 0.</td><td>0,0</td></tr>
<tr><td>Override change operation rights</td><td>OVRDENA</td><td>Integer 1</td><td>Specifies whether the operation rights are required when changing override from the external device.
(Required/Not required = 0/1)

Note) Even when OVRDENA = 1 (Not required), override cannot be changed if the controller mode is "MANUAL"</td><td>0(Required)</td></tr>
<tr><td>Access target of a program.</td><td>ROMDRV</td><td>Integer 1</td><td>The access target of a program.

2: High-speed RAM mode (Using DRAM memory)
* Cannot be changed.</td><td>2 (DRAM mode.)</td></tr>
<tr><td>Maintenance forecast setting</td><td>MFENA</td><td>Integer 1</td><td>This sets enable/disable of the maintenance forecast function.
0: Disable
1: Enable

* When the function is disabled, the information related to the maintenance forecast function is not gathered. Therefore, the maintenance timing is not calculated precisely.</td><td>1</td></tr>
<tr><td>Maintenance forecast warning day</td><td>MFWRNDAY</td><td>Integer 1</td><td>This sets the number of remaining days until the maintenance time. When the number of remaining days calculated by the maintenance forecast function becomes equal to or falls below the number of days specified by the parameter, an alarm is generated.
Setting range: 1 to 365
Unit: days</td><td>14</td></tr>
</table>

Parameter	Parameter name	No. of arrays No. of characters	Details explanation	Factory setting
Operating hours per day	MFOPTIME	Real value 1	This sets the estimated robot operating hours per day to be used for the maintenance forecast function. Setting range: 1 to 24 Unit: hours	16.0
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	1(Generates a warning), 0 (Does not output a signal)
Resetting Maintenance Forecast Note) When reading this parameter form 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 the belt in the maintenance forecast function. * When axes generated a warning (numbered in 7540's) that prompts the replacement of the 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.
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.	The setting varies depending on the model.
	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 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

Parameter	Parameter name	No. of arrays No. of characters	Details explanation	Factory setting
Program operation by T/B	TBOP	Integer 1	Specifies whether the program operation by T/B is able or not. 0: disable/1: enable	1
T/B buzzer sound	TBBZR	Integer 1	Specifies the on/off of the T/B buzzer sound that can be heard when an error occurs. However, the key touch sound of R32TB is not muted. 0: Buzzer OFF 1: Buzzer ON	1
Hand open/close operation delay time	HNDT	Integer 8	Specifies the hand response delay time when the open/close operation of the hands is performed by R32TB. Setting this prevents the hand from responding immediately to unintentional pressing of the key because continuously pressing the key is required for the specified duration. Unit: [0.1 sec]	(+C,-C,+B,-B,+A,-A,+Z,-Z)= 0,0,0,0,0,0,0,0
Behavior selection of the [RESET] key input at the error occurrence on dual system	CATEGORY	Integer 1	Set the behavior of the [RESET] key input at the error occurrence on dual system. 3: Enable the error reset 4: Disable the error reset Target errors are following. H0039 (Door Switch Signal line is faulty.) H0046 (Faulty wiring (Enabling Device).) H0051 (Wiring of the external emergency stop is abnormal.)	3
Controller software version (CR800-R/Q only)	RCDUVER	Character string 1	The software versions of CR800-R and CR800-Q controllers can be checked. Use this function to check the controller software version when the software version error "C0099" occurs.	The setting varies depending on the software version.

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 multitasks	TASKMAX	Integer 1	Designate the number of programs to be executed simultaneously.	8
Slot table (Set during multitask operation.)	Operation conditions for each task slot is set during multitask operations. These are set when the program is reset.			
	SLT* * is 1 to 32	Character string 4	Designate the [program name], [operation mode], [starting conditions], [order of priority]. Program name: Selected program name. Use uppercase letters when using alphabet. Lowercase characters are not recognized. 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.) Starting conditions: Normal/Error/Always =START/Error/ALWAYS START: This is executed by the start key 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. 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 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.	"",REP,START,1

Parameter	Parameter name	No. of arrays No. of characters	Details explanation	Factory setting
Program selection save	SLOTON	Integer 1	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. (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. 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. (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. Do not maintain: The status of program selection is not maintained at the end of cycle operation. The parameter value becomes P.0000. Setting values and operations 0: Disable storage, do not maintain 1: Enable storage, do not maintain (initial value) 2: Disable storage, maintain 3: Enable storage, maintain	1(Enable storage, do not maintain)
Setting that allows the execution of X** instructions and Servo instruction in an ALWAYS program.	ALWENA	Integer 1	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). Enable/Disable = 1/0	0(Disable)
User base program Refer to "4.3.24User-defined external variables"	PRGUSR	Character string 1	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. 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.	""(Non)
Continue function	CTN	Integer 1	Reserved	0(Invalid)
JRC command (Multiple rotation function of axes)	Set the execution status of the JRC instruction.			
	JRCEXE	Integer 1	Set the validity of the JRC command execution. Execution valid/invalid = (1/0)	0(Execution invalid)
	JRCQTT	Real value 8	Set 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. The unit relies on the parameter AXUNT.	JRC execution valid robot 0,0,0,0,0,360,0,0 or 0,0,0,360,0,0,0,0 JRC execution invalid robot 0,0,0,0,0,0,0,0
	JRCORG	Real value 8	Set the origin coordinate value for executing the JRC O command and setting the origin. This setting is valid only for the user-defined axis. The unit relies on the parameter AXUNT.	0,0,0,0,0,0,0,0

Parameter	Parameter name	No. of arrays No. of characters	Details explanation	Factory setting
Setting of additional axis	AXUNT	Integer 16	Set the unit system for the additional axis. 0: Angle (degree) 1: Length (mm) 2: Length (mm) when using a linear servo motor When using a linear servo motor, set this parameter to 2.	0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0
User error setting	UER1 to UER20	Integer 1, Character string 3	Sets the message, cause, and method of recovery for errors from the Error instruction. Maximum of 20 user errors can be set. 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. Second element: Error message Third element: Cause Fourth element: Method of recovery If a space character is included in the message, enclose the entire message in double quotation marks (""). When changing the message, enclose the entire message in double quotation marks ("") too. Example)9000,"Time Out","No Signal","Check Button" •The message cannot include a comma (,), semicolon (;), and backslash (\). •The value of the first digit can function in an auxiliary manner. For example, enter a message in No. 9000, and the message will be displayed when any of the error of Nos. 9001 to 9009 occurs.	9900,"message","cause","treat"
Unit setting for the rotational element of position data	PRGMDEG	Integer 1	Specifies the unit used for describing the rotational element of position data in the robot program. 0:Rad 1:Deg Example)M1=P1.A (Unit for this case is specified.) (Default unit for referencing data components is radian.) The default rotational element for the position constant (P1=(100, 0, 300, 0, 180, 0, 180) (7, 0)) is Deg. This parameter is irrelevant.	0(Rad)
Decimal place	PRGDPNTM	Integer 1	Specify the number of decimal digits for the position-type variable and the joint-type variable. Specify two or three places after the decimal point.	3: RH-FR, RH-CR, RH-3FRHR 2: Other than above
Robot language setting	RLNG	Integer 1	Select the robot language 3:MELFA-BASIC VI 2:MELFA-BASIC V	3
Display language Note1)	LNG	Character string 1	Set up the display language. "JPN":Japanese "ENG":English The following language is changed. (1) Personal computer support software. *alarm message of the robot. *Parameter explanation list. (2)Alarm message that read from the robot with external communication. (Standard RS232C, Extended serial I/F, Ethernet I/F)	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.	1
Compile mode	PRGMODE	Integer 1	Program processing speed is improved when compile execution mode is valid. 0: compile mode is invalid 1: compile mode is valid	1

Parameter	Parameter name	No. of arrays No. of characters	Details explanation	Factory setting
Duplication function of program	PRGDUP	Integer 1	Specifies whether or not to duplicate the program. Performing the duplication enables automatic restoration when a program error occurs due to noise, etc. 0: Disable 1: Enable Note) By duplicating the program, the capacity of the program save area is reduced to 1/2.	0
Program autosave at the time of automatic operation	AUTOSAVE	Integer 1	Specifies whether or not to save the local variables automatically after the program operation. The autosave is performed at the following timings. • When the End command called up by the CallP command is executed in a sub program • When the program is switched • When the XClr command is executed • When the program is cleared after the cycle operation 0: Not saved 1: Saved	1
Combining other program read commands with Function procedure	FUNCSPEC	Integer 1	Set whether to combine with other program read commands (CallP, XLoad, XRun, #Include). Changing the setting to "1" will increase tact time slightly when using read commands of other programs. 0: Do not combine 1: Combine	0 (Do not combine)

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 necessity.
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
RT ToolBox3 Communication method setting	COMSPEC	Integer 1	Specify the communication method of the robot controller and RT ToolBox3. (The conventional communication method / high reliability communication method) 0: Conventional communication method 1: High reliability communication method Compared with the conventional communication method, speed is slow. However, in periphery environment where many noises exist, it is the high reliability communication method.	1
Communication setting	COMDEV	Character string 8	This configures which lines will be assigned to COM1 and COM2 when using communication lines in the Open instruction. This parameter must be set if data link (used by the Open instruction) is to be performed. 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. 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: . 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.	"RS232", , , , , , ,
For ethernet	NETIP	Character string 1	IP address of robot controller. In CR800-D series controller, the address 192.168.100.xxx cannot be set. When the parameter is restored using RT ToolBox3, the parameter NETIP goes to valid after resetting power supply twice.	192.168.0.20
	NETMSK	Character string 1	Sub-net-mask	255.255.255.0
	NETPORT	Numerical value 10	Port No. Range 0 to 32767 For real-time external control functions, Correspond to OPT11 to 19 of COMDEV (OPT11), (OPT12), (OPT13), (OPT14), (OPT15), (OPT16), (OPT17), (OPT18), (OPT19)	10000, 10001, 10002, 10003, 10004, 10005, 10006, 10007, 10008, 10009
	CPRCE11 CPRCE12 CPRCE13 CPRCE14 CPRCE15 CPRCE16 CPRCE17 CPRCE18 CPRCE19	Numerical value 9	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)	0, 0, 0, 0, 0, 0, 0, 0, 0

Parameter	Parameter name	No. of arrays No. of characters	Details explanation	Factory setting
End code		Integer 9	Specifies the end code of the communication message. 0: CR, 1: CR+LF Correspond to OPT11 to 19 of COMDEV (OPT11), (OPT12), (OPT13), (OPT14), (OPT15), (OPT16), (OPT17), (OPT18), (OPT19)	 0, 0, 0, 0, 0, 0, 0, 0, 0
	CTERME11 CTERME12 CTERME13 CTERME14 CTERME15 CTERME16 CTERME17 CTERME18 CTERME19			
Ethernet packet specification for datalink	NETPSPEC	Integer 1	Specifies the specifications for packet reception of the datalink communication via Ethernet. 0: Old For reception without use of the end code. The Input command waits until one packet of data is received. The data cannot be received correctly when the packet is divided. 1: New For reception with use of the end code. The Input command waits until the end code is received even if the packet is divided.	0
Ethernet communication protocol termination code	NETTERM	Integer 9	Specifies the use of the termination code for the datalink communication via Ethernet. The elements 1 to 9 correspond to COM1 to COM9. 0: Disable 1: Enable	0,0,0,0,0,0,0,0,0
GOT port number	GOTPORT	Integer 1	Specifies the GOT port number for GOT direct connection. Match the setting with the GOT port number. When the GOT default setting is not changed, leave it as is.	5001
For ethernet	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	Character string 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 CR800-Q: A multiple of 7.1 ms CR800-D/R: A multiple of 3.5 ms (7.1 ms when user mechanism is set) Set -1 to disable timeout.	-1
Timeout time for loading a visionprogram	NVJBTOUT	Integer 1	Specify the timeout time for loading a program of a network vision sensor. (Unit: seconds, Setting range: 1 to 32767)	90
Initial tag name of EBRead command	EBRDTAG	Character string 1	Set an initial symbolic tag name (128 characters or less) for the vision sensor designated in the EBRead command. This parameter setting value is specified when the tag name is omitted in the description of the EBRead command.	"Job.Robot. FormatString"

<table>
<tr><th>Parameter</th><th>Parameter name</th><th>No. of arrays
No. of characters</th><th>Details explanation</th><th>Factory setting</th></tr>
<tr><td>Initial tag name of EBWrite command</td><td>EBWRTAG</td><td>Character string 1</td><td>Set an initial symbolic tag name (128 characters or less) for the vision sensor designated in the EBWrite command.
This parameter setting value is specified when the tag name is omitted in description of the EBWrite command.</td><td>""(None)</td></tr>
<tr><td>User name</td><td>NVUSER</td><td>Character string 1</td><td>The user name to log on the vision sensor is set.
(no more than 15 characters)</td><td>"admin"</td></tr>
<tr><td>Password</td><td>NVPSWD</td><td>Character string 1</td><td>The password to log on the vision sensor is set.
(no more than 15 characters)</td><td>""</td></tr>
<tr><td>Timing of trigger</td><td>NVTRGTMG</td><td>Integer1</td><td>Defines about processing of NVRun command and NVTrg command.
The content of each set value is as follows.

<table><tr><th>value</th><th>NVRun</th><th>NVTrg</th></tr><tr><td>0</td><td>Trigger</td><td>Trigger</td></tr><tr><td>1</td><td>Trigger+
Image processing</td><td>Trigger+
Image processing</td></tr><tr><td>2</td><td>Trigger</td><td>Trigger+
Image processing</td></tr></table>
- Trigger : The next command is executed after the communication of the image processing instruction (taking picture demand) and the vision sensor is completed.
It is possible to shorten the tact time by setting this value when another process of the robot is executed while the vision sensor is processing the image.

-Trigger + Image processing : The robot controller request the image processing (capture request) to the vision sensor. And, the next command of the robot is executed after the vision image processing is completed.
When the robot requests the image processing to the vision sensor and gets the recognition results (EBRead command is executed) with next step, please set this value.

Please set 1 to parameter NVTRGTMG when EBRead command is executed immediately after NVRun command. When parameter NVTRGTMG is factory shipment setting (NVTRGTMG=2), the next command is executed without waiting the completion of the vision recognition processing. Therefore, there is a possibility being gotten the last recognition result when EBRead command is continuously executed.</td><td>2</td></tr>
<tr><td>Initial value of tag name specified by EBRead command</td><td>EBRDTAG</td><td>Character string 1</td><td>Sets up an initial value of the "symbolic tag name" used with EBRead command (it is 128 characters or less)
When the tag name of EBRead command is omitted, the value of this parameter is specified.</td><td>"Job.Robot.
FormatString"</td></tr>
</table>

5.6 Standard Tool Coordinates

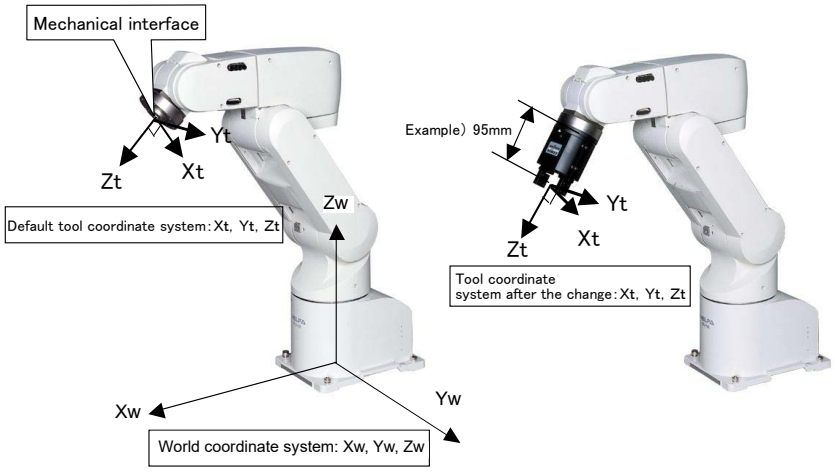
Tool conversion data must be set if the robot's control point is to be set at the hand tip when the hand is installed on the robot. The setting can be done in the following three manners.

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 16 parameters are used as tool conversion data.
   Refer to Page 408, " 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 tool conversion 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

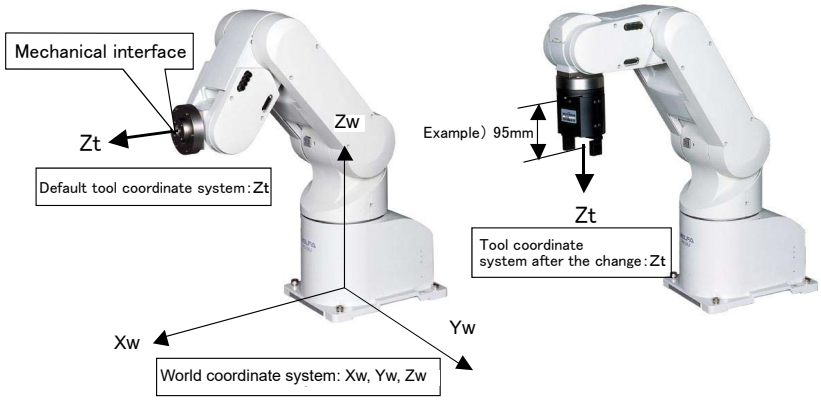
A case for a vertical 6-axis robot



<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.

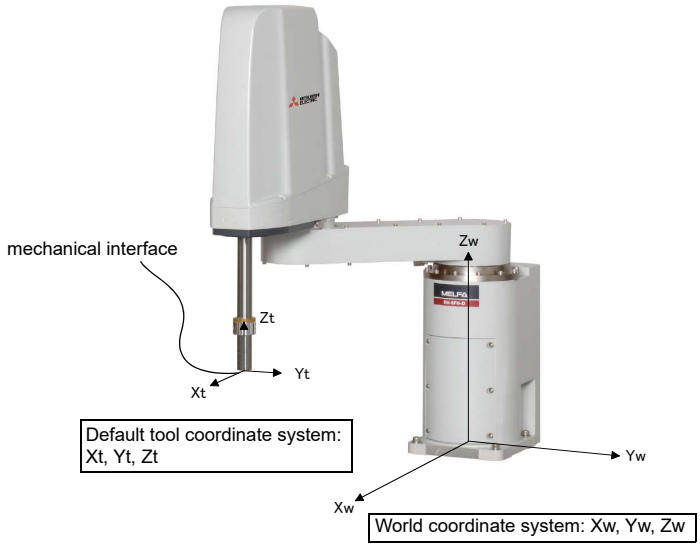
A case for a vertical 5-axis robot



<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 Note1)					
		X	Y	Z	A	B	C
RV-2FR, RV-4FR/4FRL, RV-7FR series, RV-13FR series, RV-20FR	6	○	○	○	○	○	○
RV-4FRJL	5	△	△	○	△	△	×
RH-3FRH/6FRH/12FRH/20FRH series, RH-3FRHR series	4	○	○	○	△	△	○

Note1) ○ : Valid, △ : Unused. This is meaningless and ignored if set., × : The setting value is fixed to 0.
If a value other than 0 is set, operation may be adversely affected.

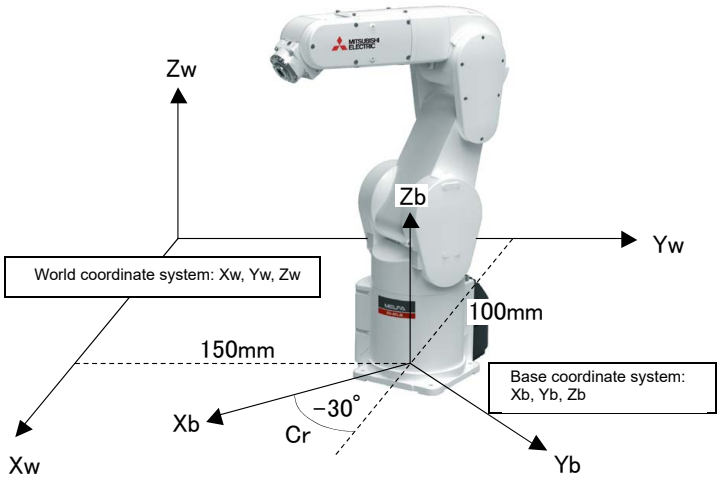
5.7 About Standard Base Coordinates

The position of the world coordinate system is set to zero (0) before leaving the factory, and therefore, the base coordinate system (robot's installation position) is in agreement with the world coordinate system (coordinate system which is the basis for robot's current position).
By utilizing the base conversion function, you can set the origin point of the world coordinate system at a location other than the center of the J1 axis.
Executing the base conversion function causes a change in the positional relation between the world coordinate system and the base coordinate system, and the robot, if allowed to move to a position to which it has been taught to move, will move to other than the position it used to. Therefore, you should maintain positive control over relation between the base conversion and the position which the robot is taught to take so that an effective use of the base conversion function is insured. Four methods are available for setting the world coordinate system:

1) Specifying parameter MEXBS directly with base conversion data
2) Specifying parameter MEXBSNO with a base coordinate number
3) Specifying the J1axis offset angle using parameter J1OFFSET (vertical 5-axis type robot only)
4) Executing a relevant base command under the robot program

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 world coordinate system
B axis: Y-axis rotation in the world coordinate system
C axis: Z-axis rotation in the world coordinate system



Note) The figure is an example of a vertical 6-axis robot.
In horizontal 4-axis robot is the same.

(Example)
1) Sample parameter setting
Parameter name: MEXBS
Value: 100,150,0,0,0,-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-2FR, RV-4FR/4FRL, RV-7FR series, RV-13FR series, RV-20FR	6	○	○	○	○	○	○
RV-4FRJL	5	○	○	○	△	△	○ Note2)
RH-3FRH/6FRH/12FRH/20FRH series, RH-3FRHR series	4	○	○	○	△	△	○

Note1) ○ : Valid, △ : Unused. This is meaningless and ignored if set., × : The setting value is fixed to 0.
Note2) Set a value in parameter J1OFFSET.

⚠CAUTION

Since the performance of the base conversion causes the reference for the robot's current position to change, data taught till then becomes unusable as it is.
If the robot is inadvertently allowed to move to a position taught before the performance of the base conversion, it can stray to an unexpected position, possibly resulting in property damage or personal injury.
When using the base conversion function, be sure to maintain positive control over relation between the base coordinate system subject to conversion and the position which the robot is taught to take so that a proper robot operation and an effective use of the base conversion function are insured.

5.8 About user-defined area

The user-defined area has the function of continuously monitoring whether or not the robot control point falls within any position area which is specified by parameter settings. The user can choose between the option to output the state of the robot control point being within or outside that area and the option to effect an error-stop when the robot control point is within that area, by using dedicated input/output or state variables. This function is instrumental in letting the robot operate in coordination of its peripherals or in avoiding interference between the robot and the peripherals, where the robot shares work space with the peripherals. Besides position area, this function can be used for making judgment on the robot in relation to posture area or additional axis area, as well.

This function can be used by the following parameter setting.

1) Selecting a coordinate system which serves as a reference system (parameter: AREAnCS)
2) Specifying a user-defined area (parameter: AREAnP1 and AREAnP2)
3) Specifying a mechanism to be checked (parameter: AREAnME)
4) Specifying a desired behavior when the robot enters user-defined area (parameter: AREAnAT)

The following is a detail description of the respective parameter settings.

5.8.1 Selecting a coordinate system

This function, when the user proceeds with operation after changing the base coordinate system by a Base command or the like, permits the user to choose between the option to move user-defined area concurrently or the option to keep it fixed. This choice is accomplished by specifying by the parameter AREAnCS that the reference coordinate system is "world coordinate system" (for moving user-defined area concurrently) or that the same is "base coordinate system" (keeping user-defined area fixed).
If the user does not make any change to the base coordinate system, the user-defined area remains unchanged regardless of the choice he makes.

Table 5-10:Explanation of coordinate system

Coordinate system	Description (feature)
World coordinate system	When the base coordinate system is changed, the user-defined area moves concurrently. A change occurs in the relative positional relation between the robot arm and the user-defined area.
Base coordinate system	Change of base coordinate system does not cause the user-defined area to move. Relative positional relation between the robot arm and the user-defined area is stationary. This provision helps when the user proceeds with operation after making a change to the base coordinate system but wants to keep stationary relative positional relation between the robot arm and the user-defined area.

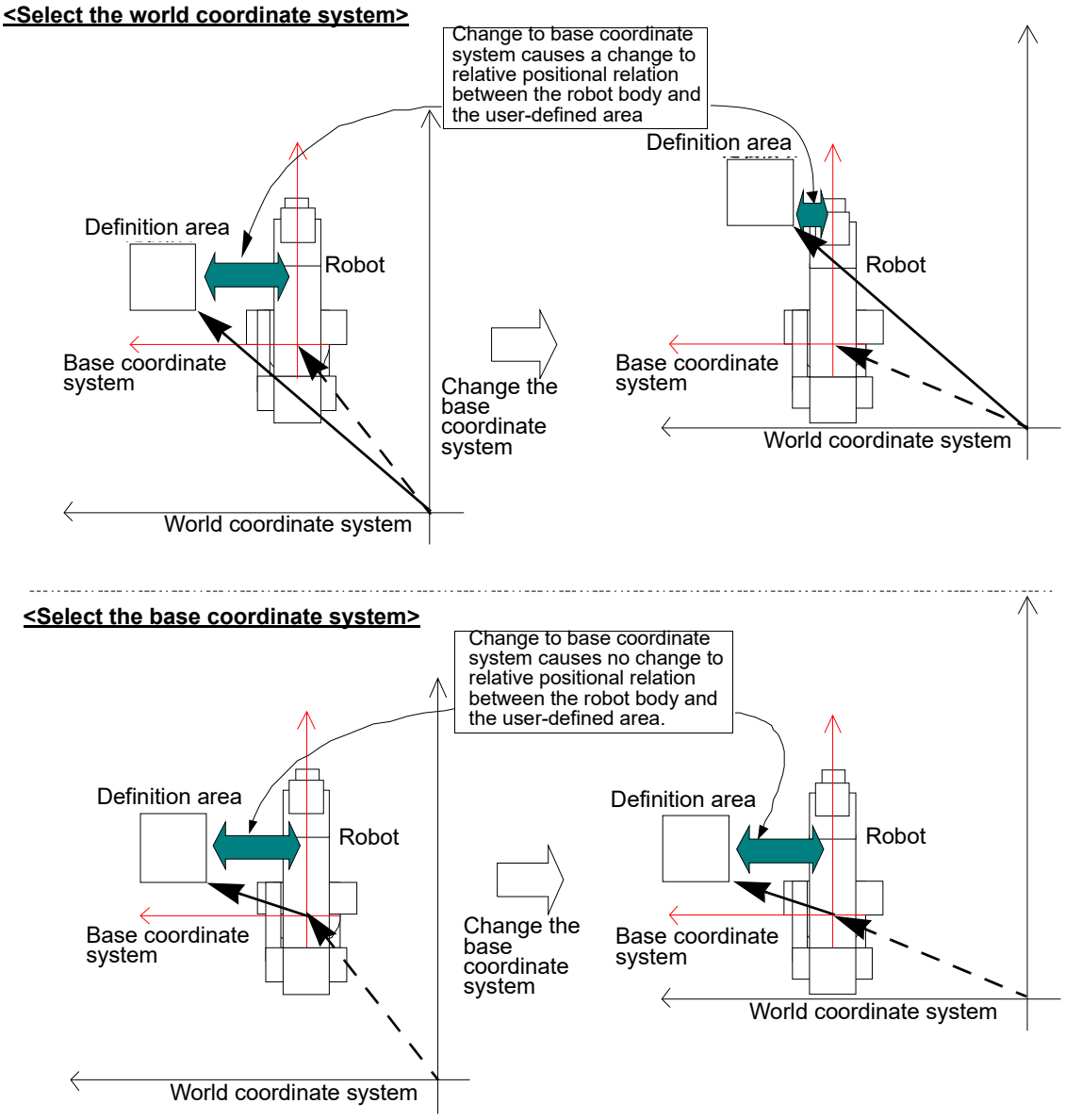


Fig.5-3:Difference between "World coordinate system" and "Base coordinate system"

5.8.2 Setting Areas

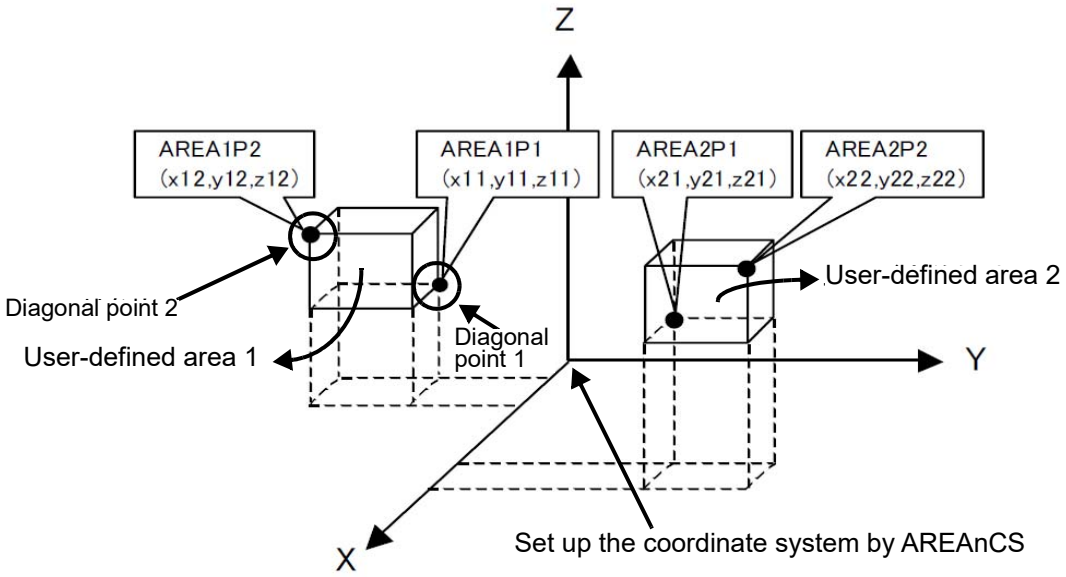
Areas to be set include a position area, posture area, and additional axis area.
The following is a description of the steps that are followed to set these areas.

(1) Position Area

A position area for user-defined area is defined by the coordinates of a diagonal point which is determined by the elements X, Y and Z in the parameters AREA*P1 and AREA*P2(* is 1 to 32).
The coordinate values thus determined are those which refer to the coordinate system selected by the parameter AREA*CS(* is 1 to 32).

<NOTES>

1) If you proceed with operation after making a change to a world coordinate system by the Base command or the like and, in addition, select a "base coordinate system" from the coordinate system options for the user-defined area, make area settings, taking note of the following points:
   Coordinate values to be specified for elements X, Y and Z in the parameters AREAnP1 and AREAnP2 must be those that were specified for the coordinate system selected in the parameter AREAnCS.
   XYZ coordinate values displayed on the T/B, RT ToolBox or the like are those that refer to the world coordinate system. Thus, when "base coordinate system" is selected in the parameter AREAnCS, coordinate values appearing on the display differ from those to be specified.
   In this case, it is necessary to make settings either by converting the displayed coordinate values into those for the base coordinate system or by temporarily returning the world coordinate system to its initial state. (The base coordinate system and world coordinate system are in agreement at factory shipping)
2) The judgments of inside or outside of the user definition area are 0.001mm and 0.001 degree unit. Therefore, if the boundary line of the area, the judgment result may get unfixed.
3) If elements X, Y and Z in the parameter AREAnP1 are interchanged with those in the parameter AREAnP2, user-defined area remains the same.



(2) Posture Area

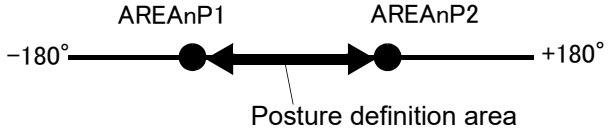
A posture area for the user-defined area is defined by specifying elements A, B and C in the parameters AREAnP1 and AREAnP2. Set up the value based on the coordinate system selected by AREAnCS.

⚠CAUTION In the 6-axis type robot, if the current coordinate value of B axis is near the +/-90 degrees, the coordinate value of A and C axes are changed a lot by even the posture movement slightly. Sign is reversed etc.
This originates in control of the robot. Therefore, if the robot is near B axis = +/-90 degrees, since the posture area judging of A and C axis may change regardless of the robot movements, it is not suitable. Please use this posture area judging function in robot operation which the current coordinate value of B axis does not consist of near the +/-90 degrees.

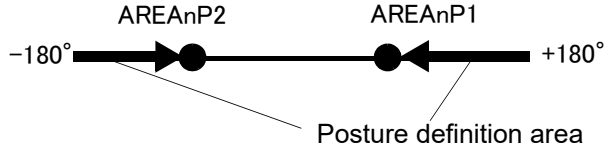
<NOTES>

1) If you proceed with operation after making a change to a world coordinate system by the Base command or the like and, in addition, select a "base coordinate system" from the coordinate system options for the user-defined area, make area settings, taking note of the following points:
Coordinate values to be specified for elements A, B and C in the parameters AREAnP1 and AREAnP2 must be those that were specified for the coordinate system selected in the parameter AREAnCS.
XYZ coordinate values displayed on the T/B, RT ToolBox or the like are those that refer to the world coordinate system. Thus, when "base coordinate system" is selected in the parameter AREAnCS, coordinate values appearing on the display differ from those to be specified.
In this case, it is necessary to make settings either by converting the displayed coordinate values into those for the base coordinate system or by temporarily returning the world coordinate system to its initial state. (The base coordinate system and world coordinate system are in agreement at factory shipping)
2) Defined area differs depending on relative locations assigned to elements A, B and C in the parameters AREAnP1 and AREAnP2. (See the figure below.)
3) When the posture area is not checked, A, B, and C element of AREAnP1 will be set as -360 degree, and A, B, and C element of AREAnP2 will be set as +360 degrees.
4) The judgments of inside or outside of the user definition area are 0.001mm and 0.001 degree unit. Therefore, if the boundary line of the area, the judgment result may get unfixed.

[If the relative locations of posture elements are set for AREAnP2 > AREAnP1]



[If the relative locations of posture elements are set for AREAnP1 > AREAnP2]



(3) Additional Axis Area

The additional axis area for the user-defined area is defined by specifying elements L1 and L2 in the parameters AREAnP1 and AREAnP2.
When the additional axis area is defined, it is judged contained in the user-defined area when all of position area, posture area and additional axis area are within the area.

<NOTES>

1) The elements of L1 and L2 in the parameter AREAnP1 and AREAnP2 are not affected by the coordinate system that defined by the parameter AREAnCS.

2) If elements L1 and L2 in the parameter AREAnP1 are interchanged with those in the parameter AREAnP2, user-defined area remains the same.
3) When the additional axis area is defined, it is judged contained in the user-defined area when all of position area, posture area and additional axis area are within the area.
4) The judgments of inside or outside of the user definition area are 0.001mm and 0.001 degree unit. Therefore, if the boundary line of the area, the judgment result may get unfixed.
5) If no additional axes (axes J7 and J8) are in use, the additional axis area need not be defined.

5.8.3 Selecting mechanism to be checked

Specify the mechanism to check the user-defined area with parameter AREA*ME. Normally, specify Mechanism 1 (1). When using the multi-mechanism etc, set up the corresponding mechanism number.

5.8.4 Specifying behavior within user-defined area

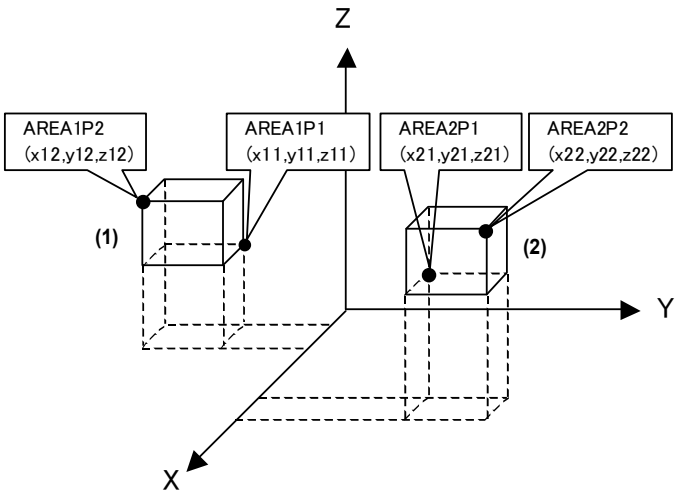
Specify the behavior of whether the robot is in the user-defined area area by setting of parameter AREAnAT. The behavior prepared is shown in Table 5-11.

Table 5-11:Specifying behavior within user-defined area

Settings	Within user-defined area	Outside user-defined area
0: Invalid	System's behavior is not specified.	System's behavior is not specified.
1: Signal output and status variable setting	Dedicated output signal USRAREA is turned on. Corresponding bit of system status variable (M_Uar32,M_Uar) is turned on.	Dedicated output signal USRAREA is turned off. Corresponding bit of system status variable (M_Uar32,M_Uar) is turned off.
2. Error output	Error H2090 occur and the robot stops. * In this case, checks the position area only, ignoring posture area and additional axis area. * To move the robot out of area, use the jog operation by "Temporarily Reset an Error that Cannot Be Canceled"	-

5.8.5 Example of settings

For instance, in the following diagram, the following parameter setting will output the signal 10 when operating in area (1) and output the signal 11 when operating in area (2).



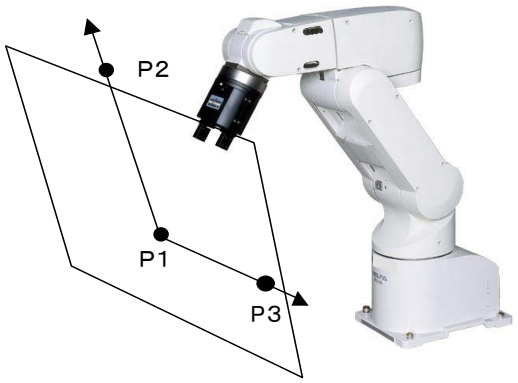
<Area (1)>
Coordinate system: World coordinate system
Posture check is unnecessary
Mechanism 1 usage
Additional axis is unused
<Area (2)>
Coordinate system: World coordinate system
Posture check is unnecessary
Mechanism 1 usage
Additional axis is unused

Parameter name	Meaning of the value	Value
AREA1CS	Selects coordinate system for Area (1).	0
AREA1P1	Position data of diagonal point 1 in Area (1): X, Y, Z, A, B, C, L1, L2	x11, y11, z11, -360, -360, -360,0,0
AREA1P2	Position data of diagonal point 2 in Area (1): X, Y, Z, A, B, C, L1, L2	x12, y12, z12, 360, 360, 360,0,0
AREA1ME	Target mechanism number: Usually 1	1
AREA1AT	Area (1) (disable/signal output/error): 0/1/2	1
AREA2CS	Selects coordinate system for Area (2).	0
AREA2P1	Position data of diagonal point 1 in Area (2): X, Y, Z, A, B, C, L1, L2	x21, y21, z21, -360, -360, -360,0,0
AREA2P2	Position data of diagonal point 2 in Area (2): X, Y, Z, A, B, C, L1, L2	x22, y22, z22, 360, 360, 360,0,0
AREA2ME	Target mechanism number: Usually 1	1
AREA2AT	Area (2) (disable/signal output/error): 0/1/2	1
USRAREA	Output signal: starting number, end number	10, 11

5.9 Free plane limit

Defines any plane in the world coordinate system, determines the front or back of the plane, and generates a free plane limit error.

5.9.1 The definition of a free plane limit



Note) The figure is an example of a vertical 6-axis robot.
In horizontal 4-axis robot is the same.

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.)

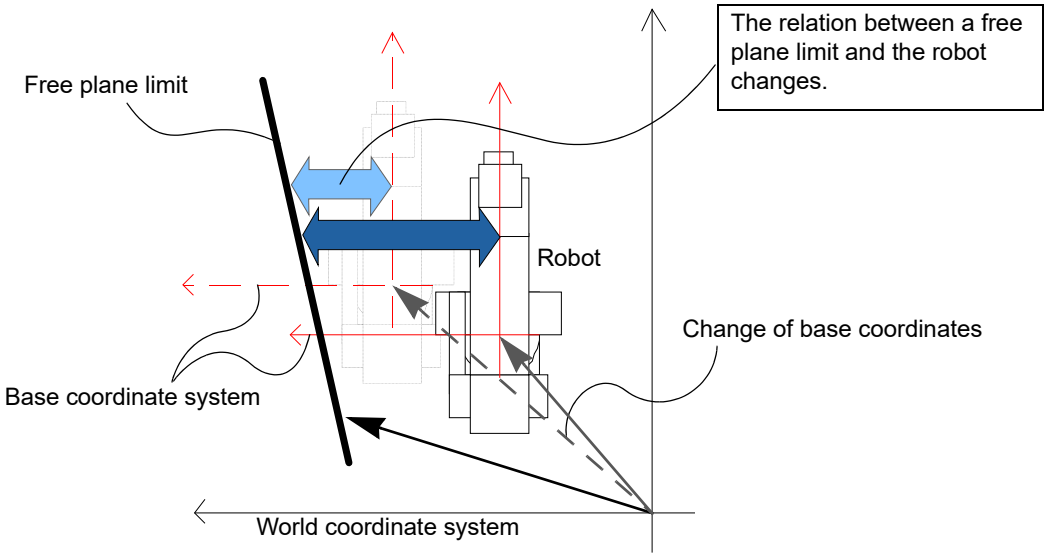
After setting the parameters above, turn the controller's power ON again. This will allow the generation of free plane limit error when it crosses the plane.

5.9.2 Selection of a coordinates system for a free plane limit

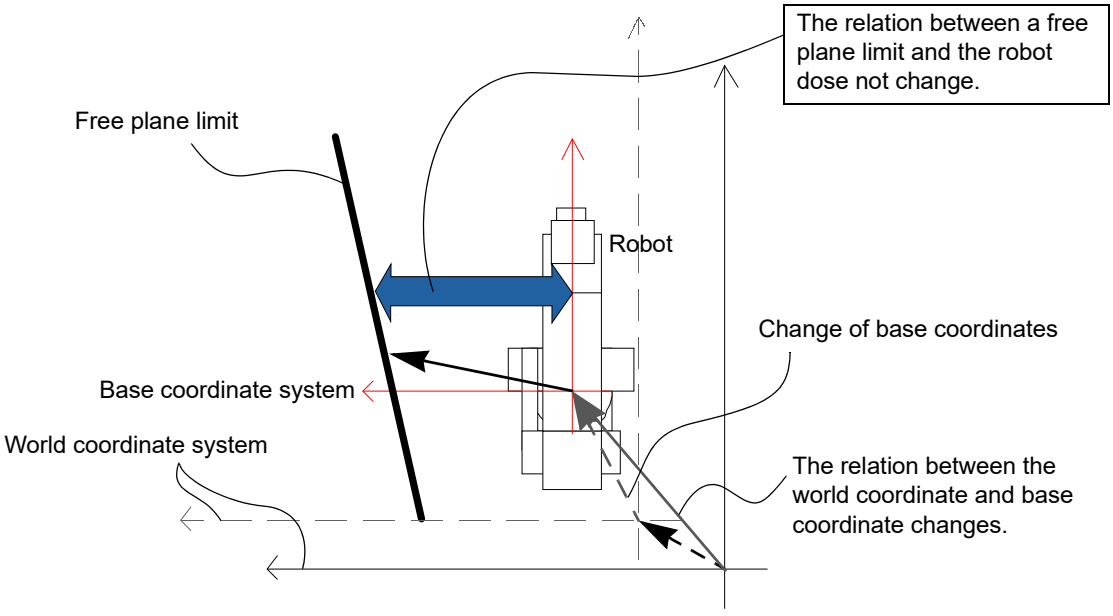
This function, when the user proceeds with operation after changing the base coordinates system by a Base command or the like, permits the user to choose between the option to move a free plane limit concurrently or the option to keep it fixed. This choice is accomplished by specifying by the parameter SFCnCS that the reference coordinate system is "world coordinate system" (for moving a free plane limit concurrently) or that the same is "base coordinate system" (keeping a free plane limit fixed).
If the user does not make any change to the base coordinate system, the free plane limit remains unchanged regardless of the choice he makes.
Refer to Page 149, "4.4 Coordinate system description of the robot" for details of the coordinate system.

Coordinate system	Explanation
World coordinate system (SFCnCS=0)	When the base coordinate system is changed, the free plane limit moves concurrently. A change occurs in the relative positional relation between the robot arm and the free plane limit.
Base coordinate system (SFCnCS=1)	Change of base coordinate system does not cause the free plane limit to move. Relative positional relation between the robot arm and the free plane limit is stationary. This provision helps when the user proceeds with operation after making a change to the base coordinate system but wants to keep stationary relative positional relation between the robot arm and the free plane limit.

<Select the World coordinate system (SFCnCS=0)>



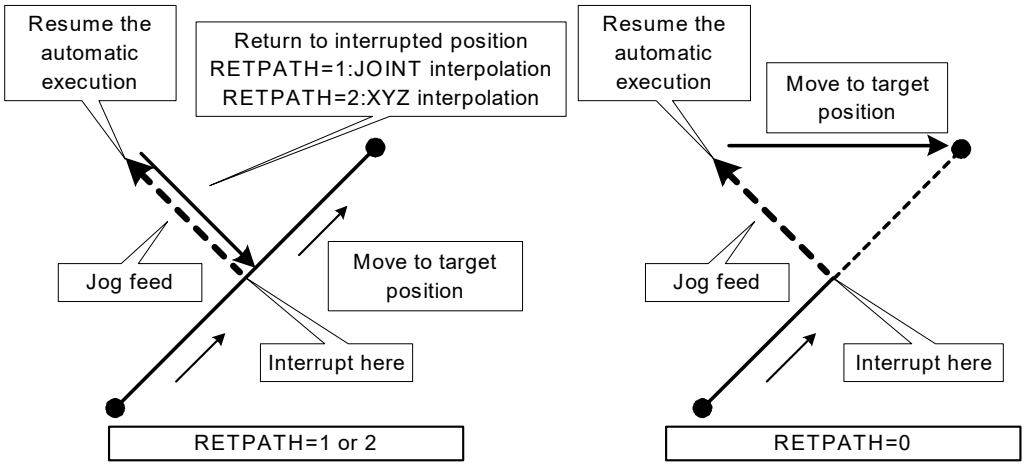
<Select the Base coordinate system (SFCnCS=1)>



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.

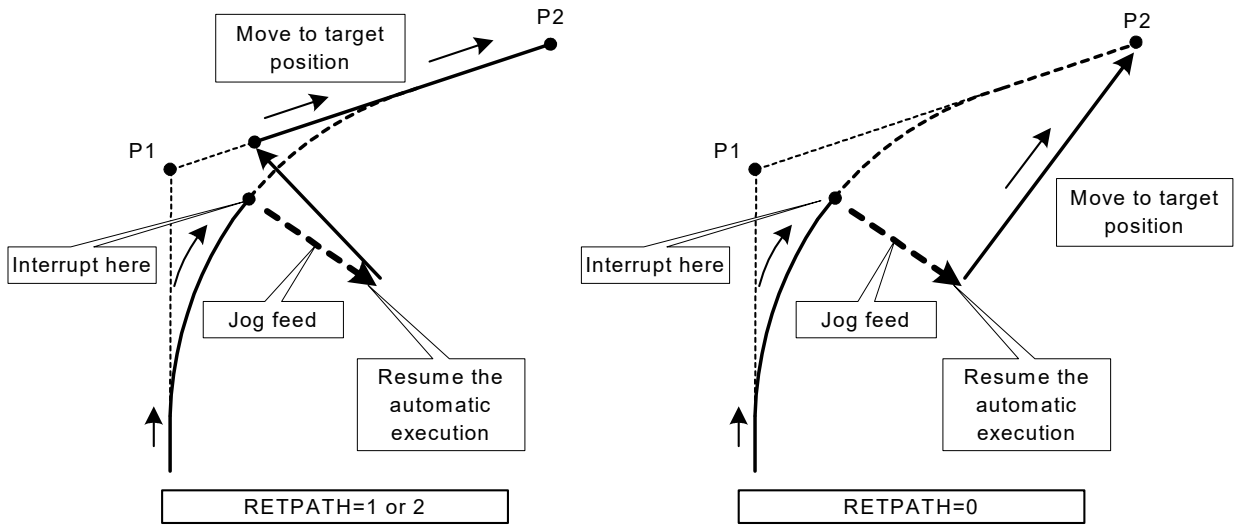
Parameter and value	Description of the operation
RETPATH=1 (Default)	1) Returns to the original position where the pause took place using joint interpolation. 2) Resumes from the line that was paused.
RETPATH=0	Resumes from the line that was paused from the position resulting after the jog operation. 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*	Exmple) SLT2=2,ALWAYS,REP Specifies the program name, start condition, and operation status. The point here is the start condition.
ALWENA	0→1 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 AUTOMATIC.
4 ' Stop the program and return the execution line to the beginning of the program if the key switch is not
    AUTOMATIC.
5 '
6 If M_Mode<>2 And (M_Run(1)=1 Or  M_Wai(1)=1) Then GoSub *MTSTOP
7 If M_Mode=2  And  M_Run(1)=0 And M_Wai(1)=0  Then GoSub *MTSTART
8 If M_Mode=1 Then Hlt ' for DEBUG
9 End
10 '
11 *MTSTART
12 XRun 1,"1"
13 Rerurn
14 '
15 *MTSTOP
16 XStp 1
17 Wait M_Run(1)=0
18 XRst 1
19 Rerurn
```

< 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→1 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 is powered on, when the controller mode is set to AUTOMATIC, program #1 is executed and the robot starts its operation.

5.12 About the hand type

(1) Solenoid valve types and signal numbers

Set the parameters according to the type of solenoid valve being used, and the output signal being connected.
The following details can be set.

a) Solenoid valve sink type/source type setting..... Parameter: HIOTYPE
Note) When this parameter is set, the hand input signal's logic will be set to the sink type or source type at the same time.
b) Solenoid valve single solenoid type/double solenoid type setting..... Parameter: HANDTYPE
c) Output signal number that drives solenoid valve..... Parameter: HANDTYPE

The default settings assume that the sink double solenoid type solenoid valve is being used.
When using a different type of solenoid valve or when using the general-purpose output signals, change the parameters shown in Table 5-12.

Table 5-12:Factory default parameter settings

Parameter name	Value and explanation
HIOTYPE	Setting value at shipment: -1 Set the sink type/source type for the solenoid valve and sink type/source type for the hand input signal logic. -1: Not set 1: Sink type 0: Source type
HANDTYPE	Setting value at shipment: D900,D902,D904,D906, , , , Set the single solenoid type/double solenoid type for the solenoid valve, and the output signal number for driving the solenoid valve. 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> * Each following is the example which connected the valve of customer preparation to the general-purpose output signal with external wiring. 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
Double-solenoid is assumed	Hand 1 = Open	900=1 901=0
	Hand 2 =Open	902=1 903=0
	Hand 3 =Open	904=1 905=0
	Hand 4 =Open	906=1 907=0

The default parameters are set as shown below so that all hands start as "Open" immediately after power up.

Parameter name	Value 900, 901, 902, 903, 904, 905, 906, 907
HANDINIT	1, 0, 1, 0, 1, 0, 1, 0

The above describes the situation for standard configuration (one unit is connected).

If for instance hand 1 alone needs to be closed when the power is turned ON, the following should be set.

Parameter name	Value 900, 901, 902, 903, 904, 905, 906, 907
HANDINIT	0, 1, 1, 0, 1, 0, 1, 0

[Caution] If you set the initial hand status to "Open," note that the workpiece may be dropped when the power is turned ON.

[Caution] 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) (specifying a signal other than one in 900's by the HANDTYPE parameter), do not use this HANDINIT parameter, but use the ORS* parameter.
The value set by the ORS* parameter becomes the initial value of signals at power 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
	:	:
	:	:
	:	:
	ORST5008	00000000, 00000000, 00000000, 00000000
	ORST5040	00000000, 00000000, 00000000, 00000000
CC-Link option	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	Signal number 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 CR800-R/Q 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、 00111000、 00000000、 00000000.....Setting value
ORST10160	10160---10167 10168---10175 10176---10183 10184---10191 00000000、 00000000、 00000000、 00000000.....At the factory default setting 11000000、 10000000、 00000000、 00000000.....Setting value

In addition to the above, to make 10148, 10143 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***0、00000000
ORST10160	11000000、10000000、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 (Ethernet)

The port number used for communication is shown in Table 5-13.

Table 5-13: Port number to use

Port number	Application	Parameter Note1)
20	Reserved	
21	Reserved	
23	COGNEX vision sensor	Cannot be changed
111	Reserved	
502	Reserved	
1024	Reserved	
5001	GOT communication	GOTPORT
10000	Mxt command	NETPORT
10001	RT ToolBox3 (This port can also be used for Data link)	NETPORT
10002 to 10009	Data link (This port can also be used for RT ToolBox3)	NETPORT
11000	R56TB	Cannot be changed (TBPORT)
12000	Oscillograph function	MONPORT
43815	Reserved	
43816	Reserved	

Note1) With a few exceptions, the port number can be changed by parameter. The setting range is 0 to 65535.

5.15.1 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 NETOPORT (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 one control cycle. The control time is approx. 7.1 ms with the CR800-Q and approx. 3.5 ms with the CR800-D/R (approx. 7.1 ms when user mechanism is set).
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 2000 (CR800-D/R) or 1000 (CR800-Q or CR800-D/R when user mechanism is set). This setting is set to -1 (timeout disabled) as the default.

5.15.2 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.

Item	Setting value
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

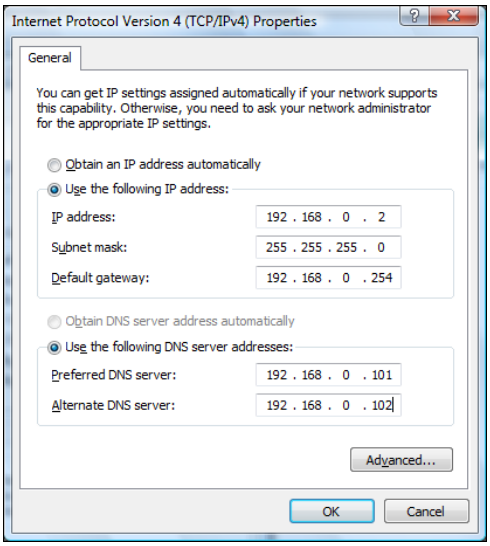
Set the robot controller parameters as shown below.

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 Vista (lower 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.

5.15.3 Example of setting of parameter 2-1

(When the data link function is used: When the controller is the server)

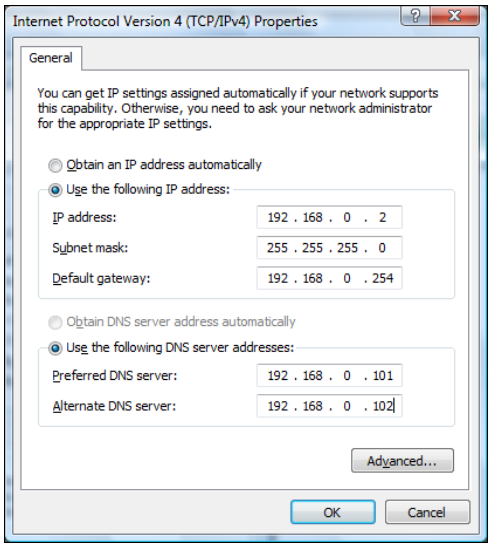
Shows the example of the setting, when the controller is server by the data link function.

Item	Setting value
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:

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	10000,10001,10002,10003,10004,10005,10006,10007,10008,10009 (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 Vista (lower 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.

5.15.4 Example of setting parameters 2-2

(When the data link function is used: When the controller is the client)

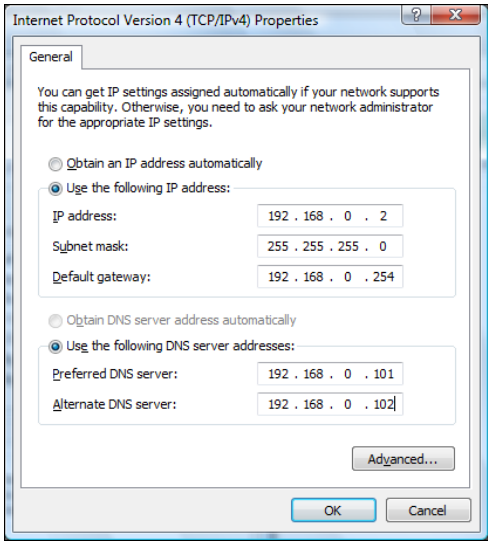
Shows the example of the setting, when the controller is client by the data link function.

Item	Setting value
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:

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	10000,10001,10002,10003,10004,10005,10006,10007,10008,10009 (Default value)
CPRCE13	Before	0
	After	2
COMDEV	Before	RS232, , , , , ,
	After	RS232, , OPT13, , , ,
NETMODE	Before	1,1,1,1,1,1,1,1,1,1
	After	1,1,0,1,1,1,1,1,1,1
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 Vista (lower 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.

5.15.5 Example of setting parameters 3

(for using the real-time external control function)

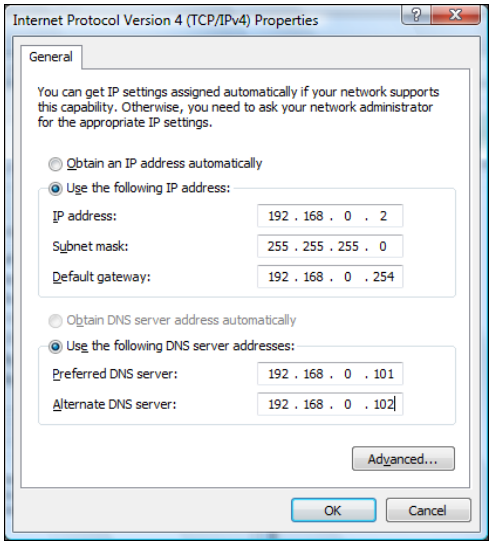
An example of the settings for using the real-time external control function is shown below.

Item	Setting value
Robot controller IP address	192.168.0.20
Personal computer IP address	192.168.0.2
Robot controller port No.	10000

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	10000,10001,10002,10003,10004,10005,10006,10007,10008,10009 (Default value)
MXTTOUT	Before	-1
	After	-1 (Default value)
MXTCOM1	Before	192.168.0.2
	After	192.168.0.2 (Default value)

Next, set the personal computer IP address to 192.168.0.2. Set this value on the Network Properties screen.

Windows Vista (lower 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.

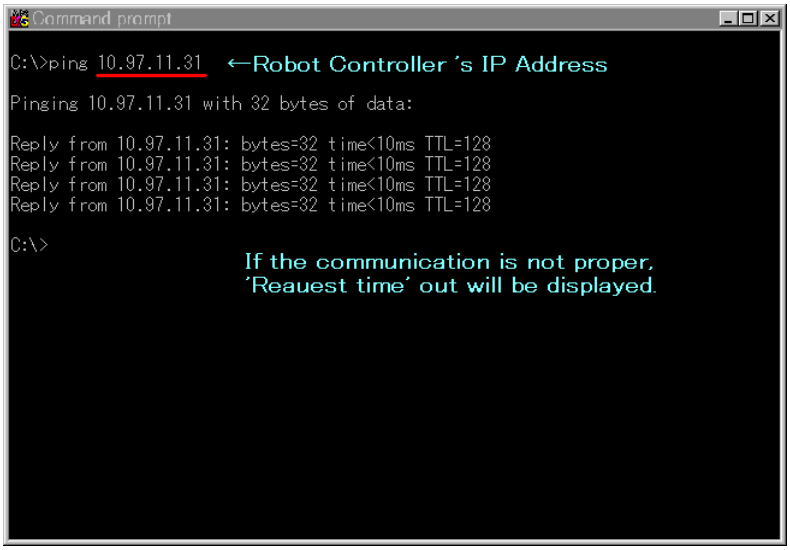
5.15.6 Connection confirmation

Before use, confirm the following items again.

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?	

5.15.7 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.



5.16 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
	HNDDAT2	Maximum load, 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
	HNDDAT4	Maximum load, 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
	HNDDAT6	Maximum load, 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
	HNDDAT8	Maximum load, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0
setting the workpiece conditions	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

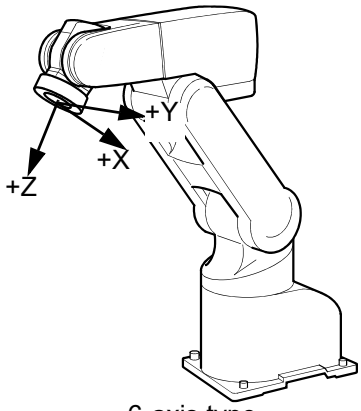
Parameter values define, from the left in order, weight, size X, Y, and Z, and center of gravity X, Y, and Z. Up to eight hand conditions and eight workpiece conditions can be set. For the size of a hand, enter the length of a rectangular solid that can contain a hand. Optimal acceleration/deceleration 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



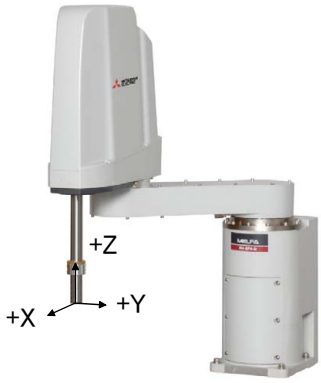
6-axis type

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.

5.17 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).

(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 command, 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

5.18 High-speed RAM operation function

The programs are executed as DRAM operation (high-speed RAM operation) to increase processing speed. When using the high-speed RAM operation function, some restrictions apply to the program operation and data saving. Thoroughly understand the specifications before using this function.

(1) Overview

The robot programs are saved in the flash ROM.
The programs are executed as DRAM operation to increase processing speed. As such, some variables used in the programs cannot be saved when the power is shut off.
Restrictions

DRAM operation: Program external variables and user-defined external variables are saved. Local variables are discarded.

(2) Precautions on saving variables at power off

Variable values used in running programs can be changed with high-speed RAM operation. Note that these changed values are discarded when the power to the controller is shut off. Use the following method to retain the variables at power off.

* When variables must be retained at power off:

Use program external variables and user-defined external variables, select a program before shutting off the power, or execute the CallP instruction to end the program (run until the End instruction of the CallP destination).
Note that the user-defined external variables are excluded.

Table 5-14:Saving variables with high-speed RAM operation (DRAM operation)

Variable Note1)	During high-speed RAM operation (DRAM operation)
Local variable	The program variables being executed at power off are discarded. The variables are saved when the program is selected or when the CallP instruction ends. By using the Save command, a robot program including local variables can be saved.
Program external variable	Variable values are retained even after the power is shut off.
User-defined external variable	Variable values are retained even after the power is shut off.

Note1) The variables include numeric variables, character string variables, position variables and joint variables.

5.19 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.

*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.

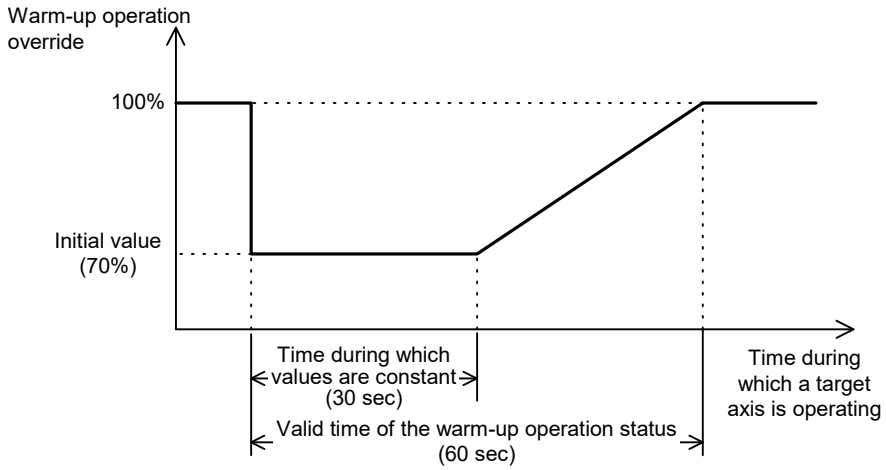


Fig.5-4:Changes in Warm-Up Operation Override

⚠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.

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 484, "5.1 Movement parameter", Page 637, "6.3 Dedicated input/output" and Page 97, "4 MELFA-BASIC VI" for details.

Table 5-15: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, ... 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 5-16:Dedicated I/O Signal List of Warm-Up Operation Mode

Parameter name	Class	Function
MnWUPENA (n=1t o 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 5-17: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_Wupst	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.

*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 two methods:

- 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. 5-5 shows an example of a timing chart for switching from the normal status to the warm-up operation status.

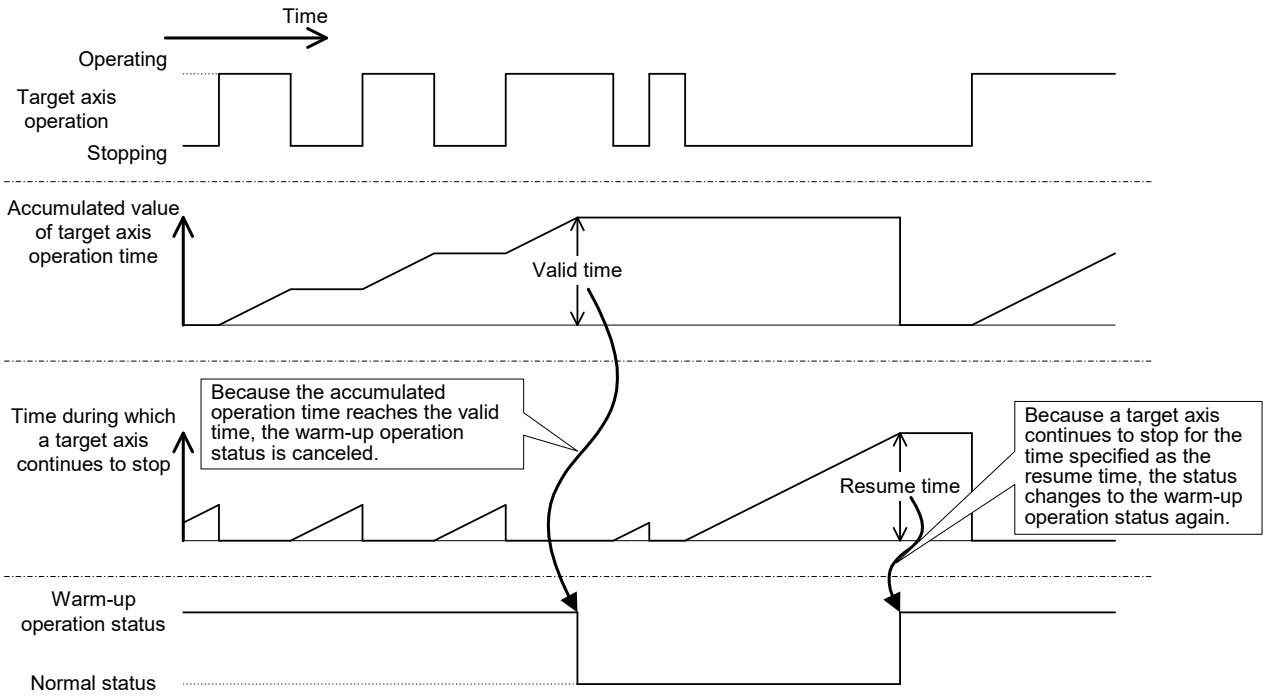


Fig.5-5:Example of Switching Between the Normal Status and the Warm-Up Operation Status

*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.

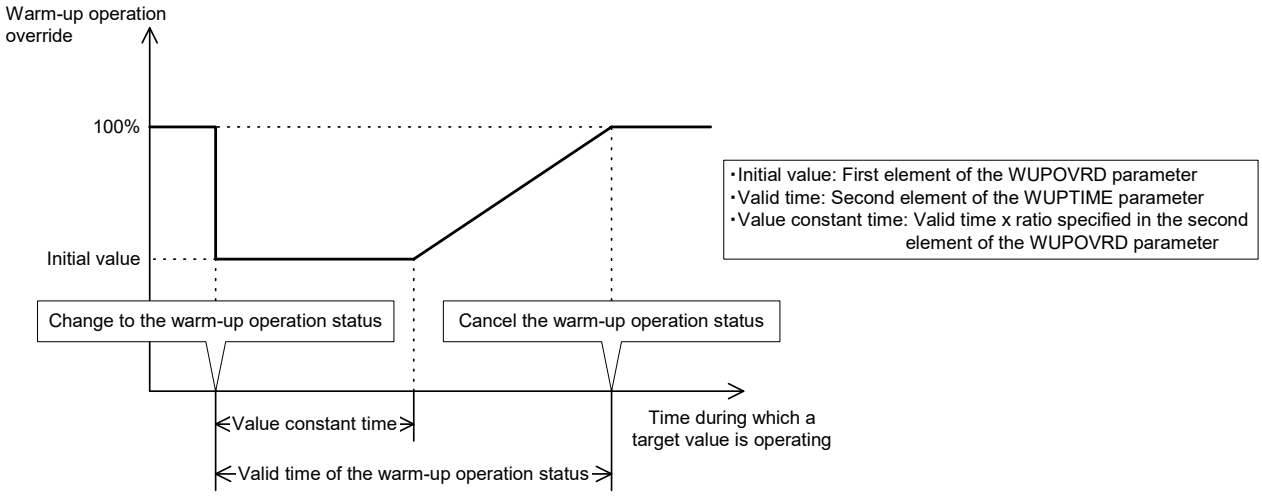


Fig.5-6:Changes in Warm-Up Operation Override

Note that the actual override in the warm-up operation status is as follows:

- During joint interpolation operation = (T/B override setting value) x (program override (Ovrd instruction)) x (joint override (JOvrd instruction)) x warm-up operation override
- During linear interpolation operation = (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.

5.20 About singular point passage function

(1) Overview of the function

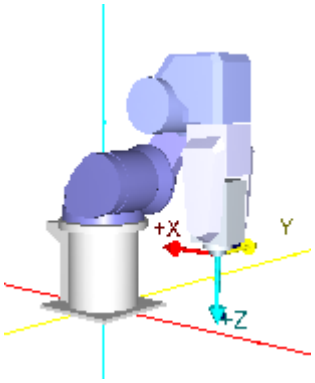
Mitsubishi's robots calculate linear interpolation movement and store teaching positions as position data in the XYZ coordinates system. In the case of a vertical 6-axis robot, for example, the position data is expressed using coordinate values of the robot's X, Y, Z, A, B and C axes, but the robot can 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.

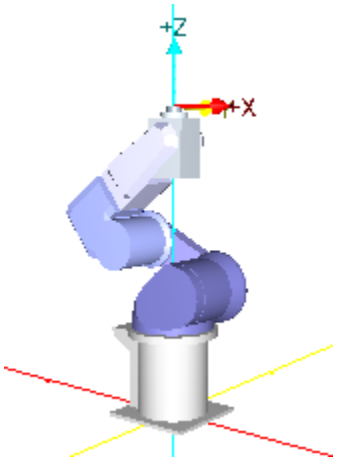
*Positions of singular points that can be passed

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

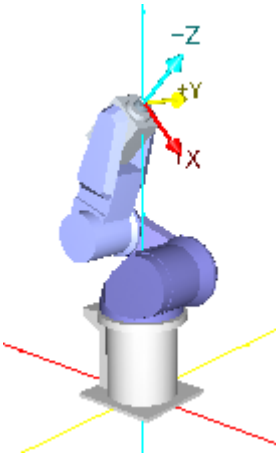


<1> Positions where axis J5 = 0 degree
In these positions, the structure flag switches between NonFlip and Flip.



<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.

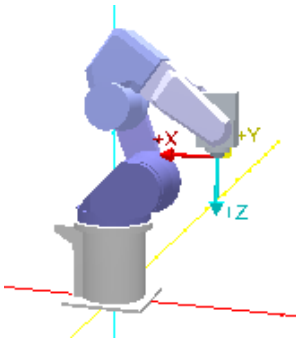
In the case of vertical 5-axis robots



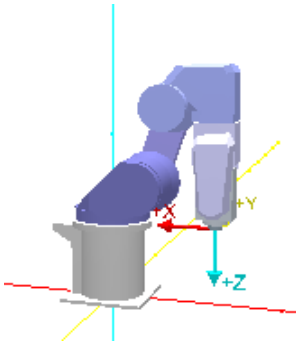
<1> Positions where the center of mechanical interface coincides with the rotation axis of axis J1
In these positions, the structure flag switches between Right and Left.

*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.

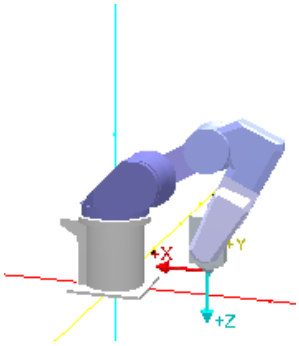
Position A



Position B

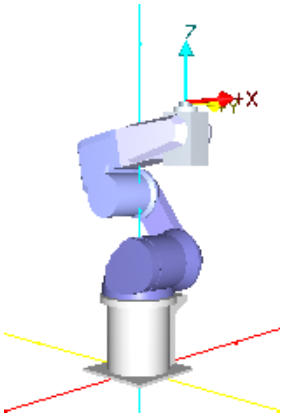


Position C

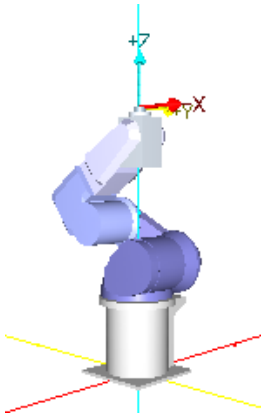


The robot can pass a singular point when the robot's motion path passes through the singular point. If the motion path does not go through the singular point (passes near the singular point), the robot continues operation without switching the value of the structure flag.
•Positions D -> E -> F: The robot's motion path passes through a singular point
(the structure flag switches before and after position E).

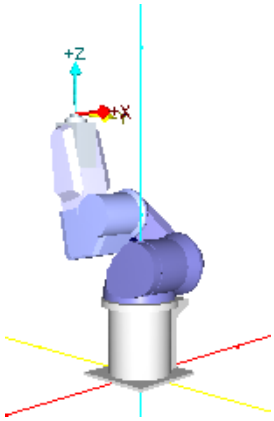
Position D



Position E

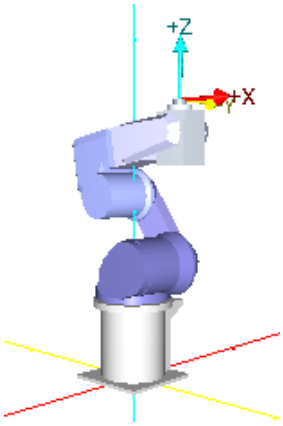


Position F

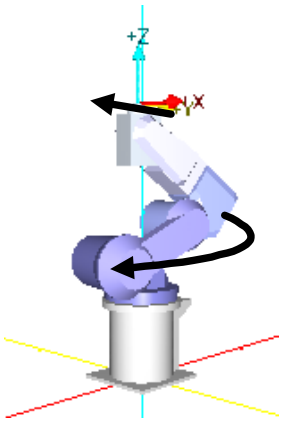


•Positions G -> H -> I: The robot's motion path passes near a singular point
(the structure flag is not switched).

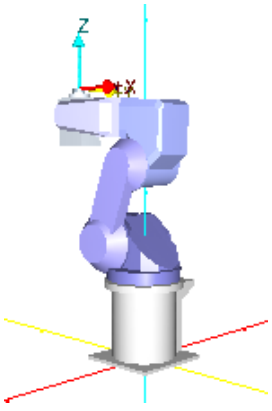
Position G



Position H



Position I



⚠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.

* Available robot models
Following models are available for the singular point passage function.

Available robot models
RV-2FR, RV-4FR series, RV-4FRJL, RV-7FR series, RV-7FRLL, RV-13FR series, RV-20FR

When the singular point passage function is enabled for the models other than the target models, normal operation (in which the posture is maintained and the structure flags are not switched) is performed for jog operation and an error occurs in automatic operation.

*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	Work jog	3-axis XYZ jog	CYLINDER jog	JOINT jog
0 (Factory setting)	Normal XYZ jog operation	Normal Tool jog operation	Normal Work jog operation	The setting value does not affect. (For the X, Y, and Z axes, the posture is maintained and the structure flags are not switched.)	The setting value does not affect. (The posture is maintained and the structure flags are not switched.)	The setting value does not affect.
1	Singular point passage XYZ jog	Singular point passage Tool jog	Singular point passage Work jog			

1) The normal XYZ/TOOL/WORK jog operation refers to the operation in which the posture is maintained and the structure flags are not switched.
2) For robots that cannot use the singular point passage function, changing the setting value of parameter FSPJOGMD has no effect; robots perform the normal operation (The posture is maintained and the structure flags are not switched.)
The models supporting the singular point passage function are RV-FR series robots. (Refer to Available robot models in details.)
3) 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.
4) 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 552, "5.20 About singular point passage function".
5) 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)	The setting value does not affect. The position moves in the joint mode.	Normal position movement (The posture is maintained and the structure flags are not switched.)
1		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]

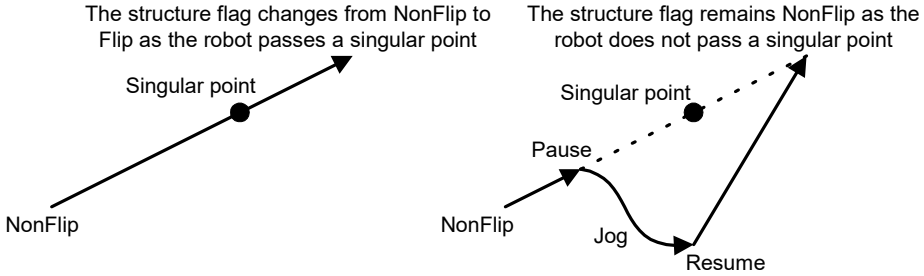
<Constant 1>	0/1	: Short cut/detour
<Constant 2>	0/1/2	: Equivalent rotation/3-axis XYZ/singular point passage

[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.

5.21 About the collision 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 collision 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 collision 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 collision detection function is enabled or disabled.

⚠CAUTION Even if the collision 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 collision 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.

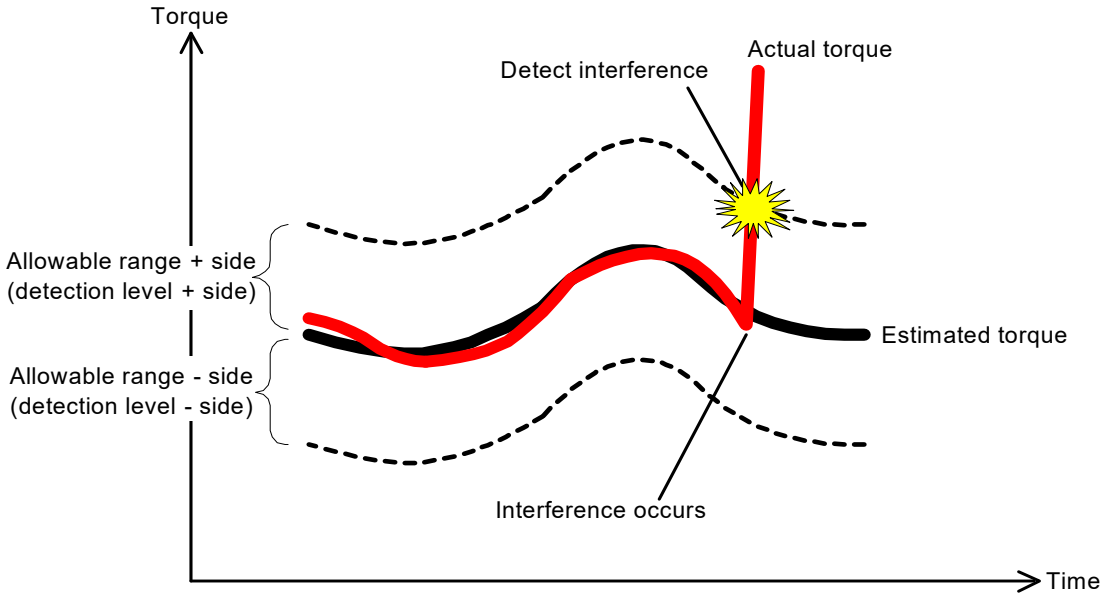


Fig.5-7:Interference detection principle

(2) Related parameters

The following parameters are related to the collision detection function. Refer to Page 484, "5.1 Movement parameter" and Page 541, "5.16 Hand and Workpiece Conditions (optimum acceleration/deceleration settings)" for the detailed explanation of these parameters.

Table 5-18:Parameters related to the collision detection function

Parameter name	Description and value	Setting value at shipment
COL	Define whether to enable or disable the collision 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 collision 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)	RH-3FRH/6FRH/ 12FRH/20FRH series: 1,0,1 RV-FR series: 0,0,1 RH-3FRHR series: 1,1,1
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 collision 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 collision 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

(3) How to use the collision detection function

To use the collision 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 collision detection function (specify to enable/disable the function and the detection level) for jog operation and program operation, respectively. (Refer to Page 558, "Table 5-18: Parameters related to the collision detection function" as well.)

1) How to use the function during jog operation

During jog operation, all the settings for the collision 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 collision detection function for jog operation.

Table 5-19:Parameters set for the collision detection function used during jog operation

Prameter name	Description and value	Setting value at shipment
COL	Define whether to enable or disable the collision detection function as well as whether it is valid or invalid immediately after turning the power supply on. Element 1: Enables (1) the collision 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)	RH-3FRH/6FRH/ 12FRH/20FRH series: 1,0,1 RV-FR series: 0,0,1 RH-3FRHR series: 1,1,1
COLLVLJG	Set the detection level (sensitivity) of each joint axis at jog operation (including pause status).	The setting varies depending on the model.
HNDDAT0	Set the hand conditions (via tool coordinates).	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 collision 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.

It is possible to choose "Ref. value of COL level" (reference value of the collision detection level) for the oscillograph function of RT ToolBox3. This function can be easy to adjust the detection level of COLLVLJG parameter. Refer to Page 563, " 3) Supplement" for details.

*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 collision 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 61, "3.11 Operation to Temporarily Reset an Error that Cannot Be Canceled" to ensure that there is no interference.

*Method for disabling collision detection temporarily during jog operation
Perform servo-on and jog operation while holding down the [RESET] key on the TB. Collision detection is disabled as long as the key is pressed.

2)How to use the function at program operation
The initial state of the collision 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 VI instruction. The parameters for setting the initial state and instructions related to the collision detection function are shown in the table below. Refer to Page 170, "4.12 Detailed explanation of command words" and Page 346, "4.13 Detailed explanation of Robot Status Variable" for the detailed explanation of the instructions.

Table 5-20:Parameters to be set for the collision detection function at program operation.

Prameter name	Description and value	Setting value at shipment
COL	Define whether to enable or disable the collision detection function as well as whether it is valid or invalid immediately after turning the power supply on. Element 1: Enables (1) the collision detection function (enable (1)/disable (0)) Element 2: Set enable (1) as the initial state of the collision detection function at program operation (enable (1)/disable (0)).	RH-3FRH/6FRH/ 12FRH/20FRH series: 1,0,1 RV-FR series: 0,0,1 RH-3FRHR series: 1,1,1
COLLVL	Set the detection level (sensitivity) of each joint axis at jog operation (including pause status).	The setting varies depending on the model.
HNDDAT* * is 1 to 8	Set the hand conditions (via tool coordinates).	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 VI commands and status variables used in the collision detection function at program operation

Command/ Status variable	Description
ColChk	Enables or disables the collision detection function or specifies the NOERR mode. Example: ColChk ON 'Enable the collision detection function.
ColLvl	Specifies the detection level (sensitivity) of the collision detection function for each joint axis. This value is a scaling factor that amplifies the detection level standard value prescribed in the collision 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 command (unit: %).
M_ColSts	Returns 1 when an interference is detected. It is used as interrupt condition in the NOERR mode.

Command/ Status variable	Description
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 collision 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 collision detection is required. Thus, it is a good idea to use the collision detection function only for operations that may cause interference, so that the detection sensitivity may be kept high when in use.

Point

If the collision 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 collision detection function only for operations that may cause interference, rather than enabling the function for the entire program.

*Adjustment of collision 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.

Table 5-22:Example of detection level adjustment procedure at program operation

Step	Description
1	Add the ColLvl and ColChk commands before and after operations for which the collision detection function is used.
2	Set the detection level low (the argument of the ColLvl command 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 command.
5	Set the value obtained in step 4 to the ColLvl command and run the program to check that no erroneous detection occurs at the operation for which the collision detection function is used. If an interference is erroneously detected, gradually increase the value of the argument of the ColLvl command 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 collision 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.

It is possible to choose "Ref. value of COL level" (reference value of the collision detection level) for the oscillograph function of RT ToolBox3. This function can be easy to adjust the detection level of ColLvl command. Refer to Page 563, " 3) Supplement" for details.

*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).

*Program example
This program moves the robot to a retreat position by interrupt processing if an interference is detected.

```
1 Def Act 1,M_ColSts(1)=1 GoTo *HOME,S    ' Define processing to be executed if an interference is detected
                                           by interruption.
2 Act 1=1
3 ColLvl 80,120,120,100,80,80,,           ' Set the detection level.
4 ColChk ON,NOErr                         ' Enable the collision detection function in the NOERR mode.
5 Mov P1
6 Mov P2                                  ' Jump to the interrupt processing if an interference is detected
                                           while executing step 5 to 8.
7 Mov P3
8 Mov P4
9 ColChk OFF                              ' Disable the collision detection function.
10 Act 1=0
     :
1000 *HOME                                ' Interrupt processing when an interference is detected
1001 ColChk OFF                           'Disable the collision detection function.
1002 Servo On                             ' Turn the servo on.
1003 PESC=P_ColDir(1)*(-5)                ' Calculate the retreat amount (reverse operation of
                                           approximately 5 mm).
1004 PDst=P_Fbc(1)+PESC                   ' Create a retreat position.
1005 Mvs PDst                             ' Move to the retreat position.
1006 Error 9100                           ' Pause the operation by generating user-defined L-level error.
     :
```

3) Supplement
*Collision detection function predicts an imminent collision by estimating the amounts of torque required at respective articulated arm axes on the basis of an prevailing position command, load settings, etc. and comparing the values thus obtained with the torques which are actually developing.
This function, even if a real collision does not happen, will identify a collision when the robot arm receives an external force during normal operation. For example, the robot hand may experience a drag from interference with a piping or cabling. Depending on the amount of resultant external force, the collision detection function judges that a collision has occurred. Check to see if the robot is not subjected to any force other than those originating from a collision while the collision detection function is enabled.

*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 collision 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. 5-8.

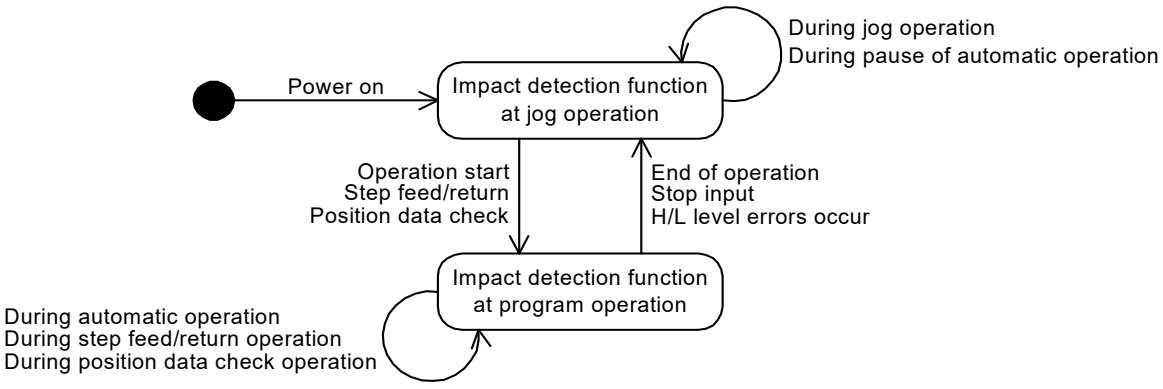


Fig.5-8:State transition diagram illustrating switch between program operation and jog operation

Thus, if the collision detection function at jog operation is enabled, for example, then even if the collision 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 collision detection is enabled.

*Collision detection function while servo off
The collision detection function is temporarily disabled while the servo is turned off at both jog operation and program operation.

*About “Ref. value of COL level” of the oscillograph function
It is possible to choose "Ref. value of COL level" (reference value of the collision detection level) for the oscillograph function of RT ToolBox3.
“Ref. value of COL level” expressed the difference between the estimated torque and the actual torque in the collision detection level (set value of ColLvl command and COLLVL, COLLVLJG parameters), it will be an indication when deciding the set value.

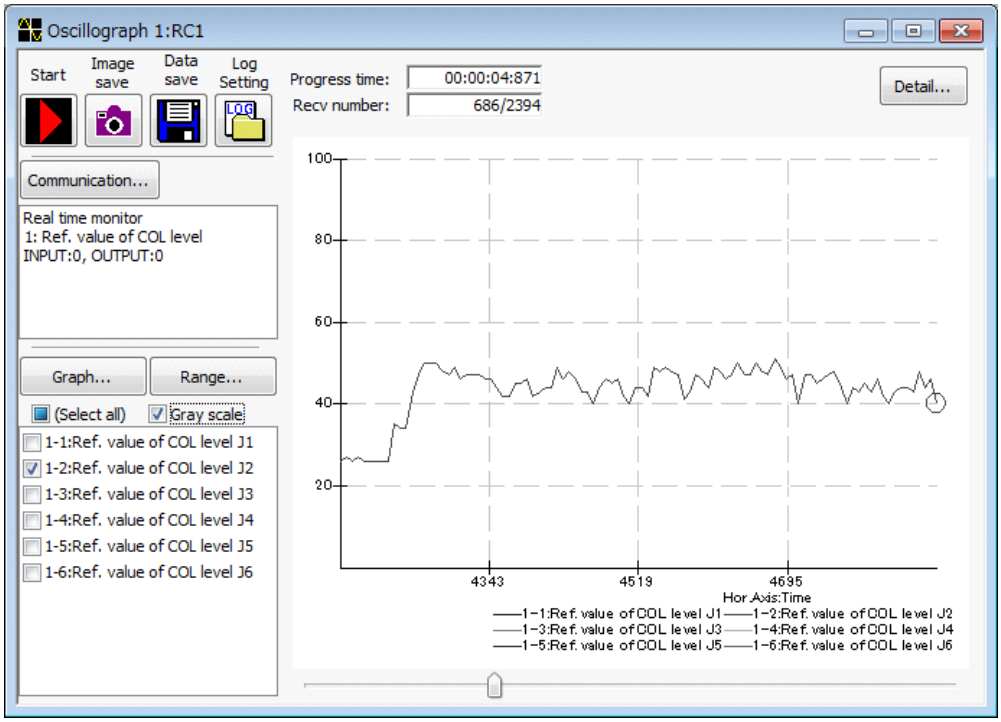


Fig.5-9:Oscillograph screen of “Ref. value of COL level”

As shown in Fig. 5-9, when the collision detection function is effective and servo power is on, the collision detection level is output by real time to the oscillograph. This value is consulted and the collision detection level is set to the movement which makes the collision detection effective.
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.

Table 5-23:Example of the collision detection level adjustment procedure at program operation

Step	Description
1	Add the ColLvl and ColChk commands before and after operations for which the collision detection function is used.
2	Set the detection level low (the argument of the ColLvl command 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 “Ref. value of COL level” of the oscillograph function in the target operation. Note that the value may fluctuate; repeat the target operation several times and record the maximum value for each joint axis.
4	Add some margin (e.g. 20%) to the maximum value for each joint axis obtained in step 3. Then set this value as the argument of the ColLvl command.

Step	Description
5	Set the value obtained in step 4 to the ColLvl command and run the program to check that no erroneous detection occurs at the operation for which the collision detection function is used. If interference is erroneously detected, gradually increase the value of the argument of the ColLvl command to lower the detection level until no erroneous detection occurs.

5.22 Optimizing the overload level

When the actual ambient temperature of the robot in use is 40°C or less, the overload error detection level (function which protects the motor from overheating), which activates during robot movement, can be optimized to match the working environment. Set the actual ambient temperature in parameter: OLTMX. The continuous operability improve.

[Available robot type] RH-FR series/RH-CR series/RV-FR series

Parameter: OLTMX is explained in Table 5-24.
If the robot's ambient temperature is controlled to 40°C or less, set the actual ambient temperature in this parameter to utilize the robot effectively.

Table 5-24:Related parameters

Parameter	Parameter name	No. of arrays No. of characters	Details explanation	Factory setting
Optimization of overload detection level	OLTMX	Integer 1	Set the upper limit of the ambient temperature for the robot's working environment. Note1) The overload detection level for robot movement is optimized based on this setting value. (Unit: °C) Setting range: 0-40	RH-3FRH/6FRH/ 12FRH/20FRH: 40 RV-2FR series: 40 RV-4FR/7FR/ 13FR/20FR series, RH-3FRHR: 30

Note1) Caution for setting ambient temperature

If the robot's ambient temperature varies through the year, set the maximum temperature. If a temperature lower than the actual temperature is set, a motor overheat error could occur before the overload error occurs.

The effect of the ambient temperature parameter change on the continuous operability differs by the model and moving axis.

The continuous operability may not change even if the parameter is changed.

5.23 Multi-rotational restrictions for the pallet definition instruction

When the pallet definition instruction: DEFPLT is used for the robot types in which the J1 axis or the J4 axis can exceed +/-180 degrees (RV-2FR, etc.), the palette that the joint angle of the J1 axis or the J4 axis straddles +/-180 degrees cannot be specified. If a position that straddles multiple rotations is specified, an error will be generated when the pallet definition instruction is executed. Change the parameter to execute the instruction.

Table 5-25:Related parameters

Parameter	Parameter name	No. of arrays No. of characters	Details explanation	Factory setting
Pallet specification	PLTSPEC	Integer 1	Enables the error check for multi-rotational positions at each point of a specified pallet. 0: Error check for multi-rotational position disabled 1: Error check for multi-rotational position enabled When 1 (Error check for multi-rotational position enabled) is set, if a position that straddles multi-rotational positions or a position to which the J1 axis or the J4 axis makes a large rotation is specified for the robot types, such as RV-2FR, in which the joint angle of the J1 axis or the J4 axis makes multiple rotations exceeding +/-180 degrees, an error will be generated when the pallet definition command is executed. When 0 (Error check for multi-rotational position disabled) is set, no error will be generated. However, it is necessary to create a program that takes into account multiple rotations for specifying a shortcut operation (TYPE) with the movement command and the multi-rotation flag (FL2). Incorrect specification may rotate the joint axis 360 degrees, interfering with peripheral devices. Be sure to confirm operation for all points on the pallet at a low speed.	1: RV-FR series 0: Other models

In addition to the hand end axis (J6 axis for RV models and J4 axis for RH models), the J1 pivot shaft and the J4 forearm twist axis of RV models can make multiple rotations exceeding +/-180 degrees.
These axes cannot rotate straddling multiple rotations with the pallet definition instruction. If a multi-rotation flag specifies a different position using the pallet definition instruction to move the robot, the J1 axis rotates in an opposite direction for joint interpolation that specifies the detour. Because the J1 axis may make unexpected movements, the following restrictions apply to the pallet definition instruction.

1) When the multi-rotation flag at the start point is different from that at end point A, end point B, or the diagonal point:
   A comparison check is performed for the multi-rotation flag of the multi-rotational axis (the J1 axis and J4 axis for RV models with six J axes, the J1 axis for RV models with five axes, and RH models) at each pallet definition point. If at least one multi-rotation flag is different, error L3750 will be generated when the pallet definition instruction is executed, resulting in operation being stopped. A comparison check is performed for both the normal pallet and the arc pallet.
2) When a joint angle is greatly different even if the multi-rotation flag from "1)" matches:
   Each point of the pallet definition is compared with the start point for the joint coordinate position of the multi-rotational axis (the J1 axis and J4 axis for RV models with six J axes, the J1 axis for RV models with five axes, and RH models). If a joint angle is different by +/-180 degrees or more, error L3760 will be generated when the pallet definition instruction is executed, resulting in operation being stopped. A comparison check is performed for both the normal pallet and the arc pallet.

A multi-rotation check is performed for robot axes (J1 pivot shaft and J4 forearm twist axis of RV models), and not performed for multiple rotations of additional axes and user-defined mechanism.

5.24 Interference avoidance function

The robot can be moved while checking for interference by this function. The target of an interference check is shown below.

1. Between robots. (Only CR800-R/Q controller)
2. Between robot and free plane limits.

The damage of robot or peripheral device can be reduced by predicting interference between robots or free plane limits and stopping the movement during jog operation or automatic operation.
When interference is predicted, the robot movement will stop. The robot can be programmed to generate an alarm or to restore operation.

[Target models]
Between robots: Only the RH-FR/RV-FR series CR800-R/Q controller. (User mechanisms not supported.)
Between robot and free plane limit: RH-FR/RV-FR series. (User mechanisms not supported.)
* Refer to Page 569, "5.24.2 Preparing and connecting the devices" about a corresponding version.
* In Q series, the interference avoidance does not function even if the setting of the function is enabled when CR700-Q controller and CR800-Q controller are mixed on the same base coordinates.



Examples of avoidable interference

By using this function, control can be executed to prevent interference caused by the following robot movements.

1) Robot collisions during jog operation caused by incorrect operations.
2) Robot collisions during automatic operation caused by program mistakes.
3) Robot or peripheral device collisions due to unintentional sequence entered during restoration work.
4) Robot collisions caused by improper interlock for an automatic robot operation performed at high-speed for the first time.
5) Robot collision between two robots moved at different overrides.

Fig.5-10:Interference check during robot operation (automatic operation, jog operation)

⚠CAUTION This interference avoidance function makes simulated robot arm, hand and workpiece. If these simulated components overlap and pose the risk of interference, the robot movement is controlled so that interference does not actually occur.
Simulated components that match the robot movement must be registered, and this function does not completely ensure that interference will be prevented.

5.24.1 Operation procedures

The outline of procedures for using the interference avoidance function is given below.

(1) Checking for interference between robots (Only for CR800-R/Q controller)

<Flow of operations>

(1)Preparing and connecting the devices ... Prepare two or three target robots and a personal computer equipped with RT ToolBox3.
Refer to Page 569, "5.24.2 Preparing and connecting the devices" for details.

↓

(2)Registering simulated components for interference check ... Using the robot arm as the reference, set the position and size with parameters for the robot arm, hand and workpiece for which the interference is to be checked and avoided.
Refer to Page 571, "5.24.3 Registering the simulated components for interference check" for details.

↓

(3)Setting the additional axis synchronization ... When using a linear axis, such as a locomotion axis, as an additional axis, set the synchronous control parameters.
Refer to Page 581, "5.24.5 Support of additional axes" for details.

↓

(4)Setting the direct communication between robot CPUs ... Set the CPU buffer memory expanded function with parameters.
(This function has a CPU buffer memory for communication between robot CPUs.)
Refer to Page 582, "5.24.6 Setting the CPU buffer memory expanded function (Checking for interference between robots)" for details on setting the function.

↓

(5)Calibrating and checking the robots ... Set and confirm the positional relation of multiple robots.
Refer to Page 585, "5.24.7 Calibration between robots (Checking for interference between robots)" for details.

↓

(6)Setting the interference avoidance function
Enable the interference avoidance function with parameters.
Refer to Page 587, "5.24.8 Enabling and disabling the interference avoidance function" for details.

↓

(7)Using the interference avoidance function ... The interference avoidance operation during jog operation and program execution are explained in the section starting on Page 587, "5.24.9 Using the interference avoidance function".
Sample programs are given on Page 589, "5.24.10 Sample programs".

(2) Checking for interference between robot and free plane limit

<Flow of operations>

(1)Preparing and connecting the devices ... Prepare two or three target robots and a personal computer equipped with RT ToolBox3.
Refer to Page 569, "5.24.2 Preparing and connecting the devices" for details.

↓

(2)Registering simulated components and free plane limits to check for interference ... Using the robot arm as a reference point, set the position and size of the robot arm, hand, workpiece, and free plane limit with parameters for the interference and avoidance check.
Refer to Page 571, "5.24.3 Registering the simulated components for interference check" for details.

↓

(3)Setting the additional axis synchronization ... When using a linear axis, such as a locomotion axis, as an additional axis, set the synchronous control parameters.
Refer to Page 581, "5.24.5 Support of additional axes" for details.

↓

(4)Setting the interference avoidance function
Enable the interference avoidance function with parameters.
Refer to Page 587, "5.24.8 Enabling and disabling the interference avoidance function" for details.

↓

(5)Using the interference avoidance function ... The interference avoidance operation during jog operation and program execution are explained in the section starting on Page 587, "5.24.9 Using the interference avoidance function".
Sample programs are given on Page 589, "5.24.10 Sample programs".

5.24.2 Preparing and connecting the devices

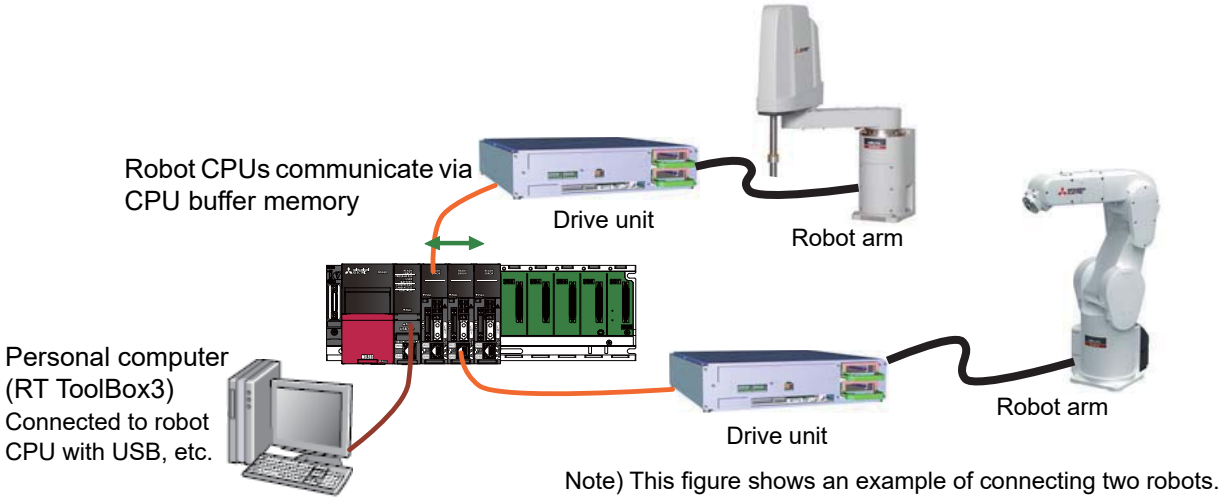
The devices required to use this function are shown in Table 5-26, and an example of the connection is given in Fig. 5-11.
Refer to the figure and connect the required devices.

(1) Checking for interference between robots (Only for CR800-R/Q controller)

*CR800-R controller

Table 5-26:Required devices (CR800-R controller)

No.	Device	Remarks
1	Up to two or three robots (CR800-R controller)	This function uses direct communication between robot CPUs via the iQPlatform's CPU buffer memory.
2	Personal computer equipped with RT ToolBox3	Connects the robot CPUs.



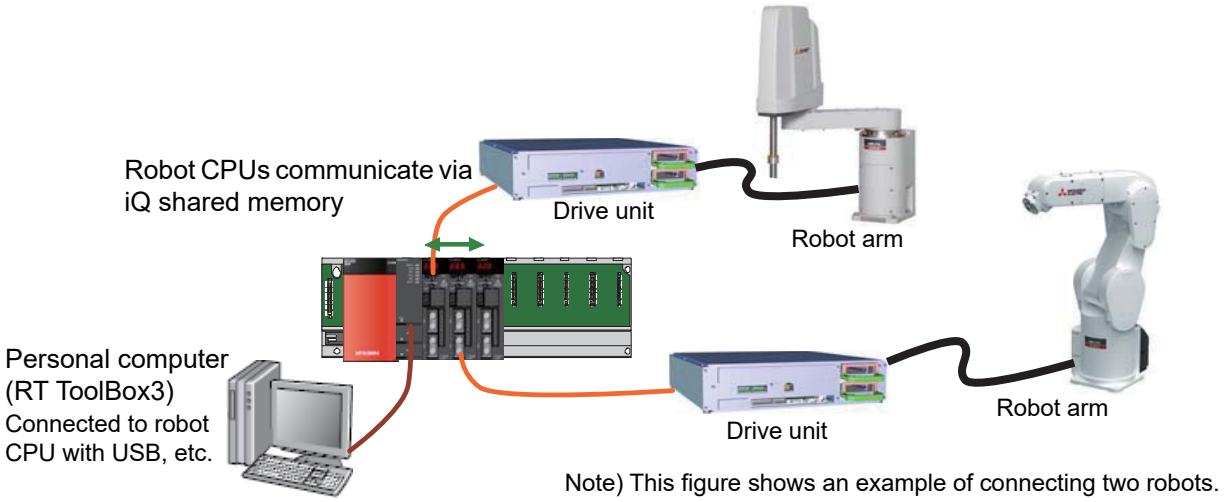
Note) This figure shows an example of connecting two robots.

Fig.5-11:Connecting the devices (CR800-R controller)

*CR800-Q controller

Table 5-27:Required devices (CR800-Q controller)

No.	Device	Remarks
1	Up to two or three robots (CR800-Q controller)	This function uses direct communication between robot CPUs via the iQPlatform's shared memory.
2	Personal computer equipped with RT ToolBox3	Connects the robot CPUs.



Note) This figure shows an example of connecting two robots.

Fig.5-12:Connecting the devices (CR800-Q controller)

(2) Checking for interference between robot and free plane limit

Table 5-28:Required devices

No.	Device	Remarks
1	One robot	Checking for interference with only free plane limit of own robot.
2	Personal computer equipped with RT ToolBox3	Connects the robot CPUs.

<CR800-D controller>

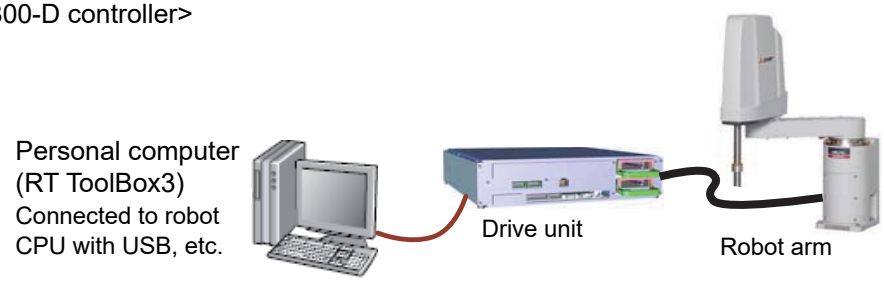
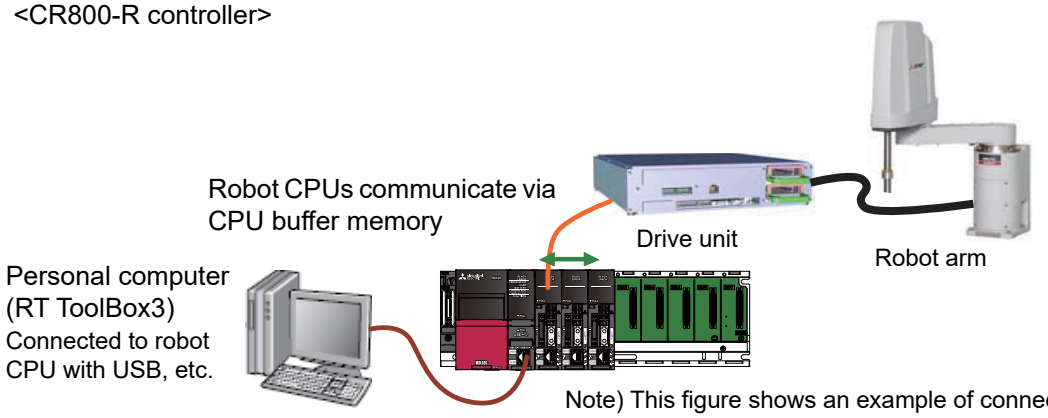


Fig.5-13:Connecting the devices (CR800-D controller)

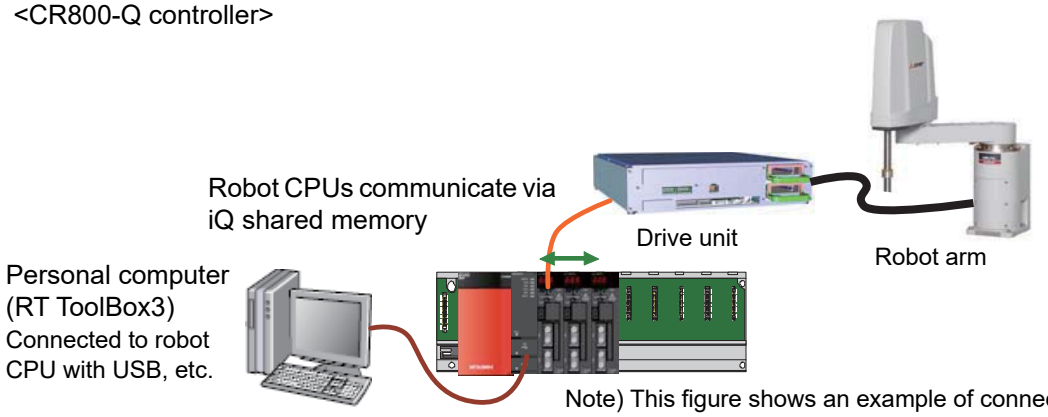
<CR800-R controller>



Note) This figure shows an example of connecting one robot.

Fig.5-14:Connecting the devices (CR800-R controller)

<CR800-Q controller>



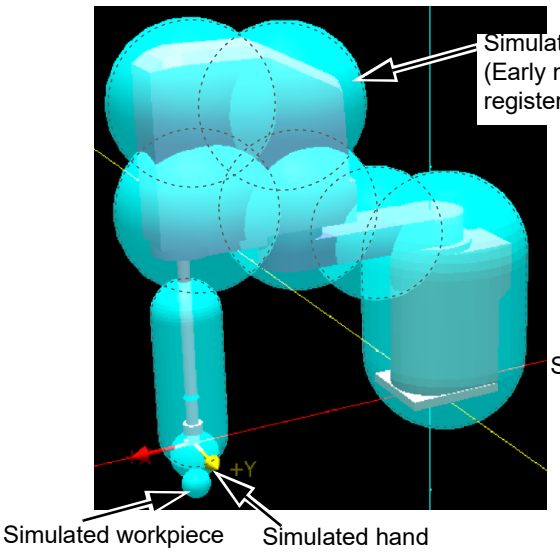
Note) This figure shows an example of connecting one robot.

Fig.5-15:Connecting the devices (CR800-Q controller)

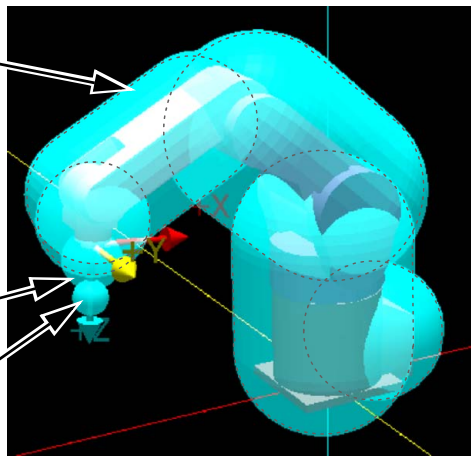
5.24.3 Registering the simulated components for interference check

Register simulated components to be checked for interference (hereinafter, "simulated components") using the robot arm as a reference point.

<RH-FR series>



<RV-FR series>



* The figure shows a screen image (setting example) from the RT ToolBox3. The shape of a simulated component is a sphere and a cylinder. The simulated components for hand and workpiece must be registered by the user.

Fig.5-16:Example of simulated component registration

The required registration details are shown in Table 5-29. Up to eight simulated component types can be registered for each of the robot arm, hand and workpiece.

Table 5-29:Details of simulated component registration

Setting items for simulated component	Simulated component type		
	Simulated robot arm Note1)	Simulated hand	Simulated workpiece
Relationship ▪ Robot arm: Set where in the robot arm a simulated component is registered to. ▪ Hand, workpiece: Set the number for each simulated component (hand No., workpiece No.).	Set how each registered simulated component constitutes the Jn axis. (Set whether a simulated component constitutes in the base section, the Jn axis section, or the flange section.) Note) The shape of a simulated component is a sphere and a cylinder. Parameters: CAVKDA1 to 8	The set hand number is interlinked to the hand condition number for the Loadset (Load Set) command. Note) The shape of a simulated component is a sphere and a cylinder. Parameters: CAVKDH1 to 8	The set workpiece number is interlinked to the workpiece condition number for the Loadset (Load Set) command. Note) The shape of a simulated component is a sphere and a cylinder. Parameters: CAVKDW1 to 8
		The simulated hand and workpiece subject to the interference check can be changed dynamically with the LoadSet command.	
Center position	Using the robot arm installation face or the rotation center of each axis as the reference point, set the simulated component's center position as a distance from that reference point. Parameters: CAVPSA1 to 8	Designate the simulated component's center as a distance from the Mechanical interface coordinate system's origin point (tip of J3 axis).	
		Parameters: CAVPSH1 to 8	Parameters: CAVPSW1 to 8
Simulated component size	Set the size of each simulated component as a radius.		
	Parameters: CAVSZA1 to 8	Parameters: CAVSZH1 to 8	Parameters: CAVSZW1 to 8
Enable/disable for each simulated component	Set whether to enable/disable a simulated component, and whether to temporarily disable a simulated component when T/B is enabled. Note) The simulated hand and workpiece must be disabled during teaching.		
	Parameters: CAVSCA1 to 8	Parameters: CAVSCH to 8	Parameters: CAVSCW1 to 8

Note1) The initial settings for each model are set at the factory.

Each parameter is explained in the following section.

(1) Simulated component registration parameter

Parameters listed in Table 5-29 are explained in detail in this section.
Up to eight simulated component types can be registered for each of the robot arm, hand and workpiece.
The last digit of a parameter name indicates the simulated component type.

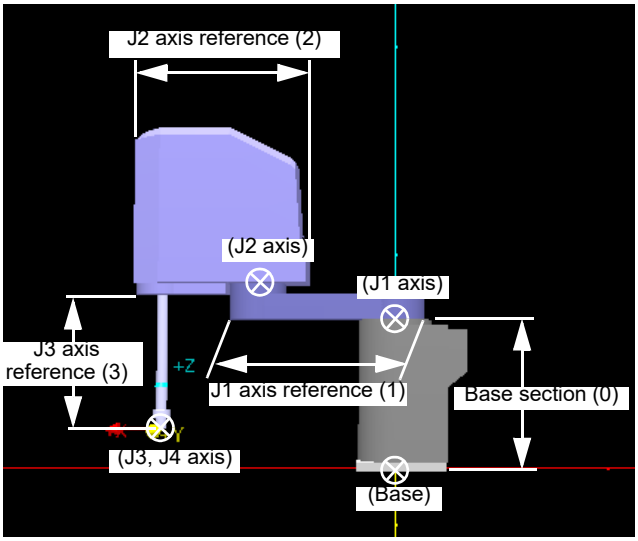
1) Simulated components for robot arm

<1> Registration section and shape of simulated components: CAVKDA1 to 8

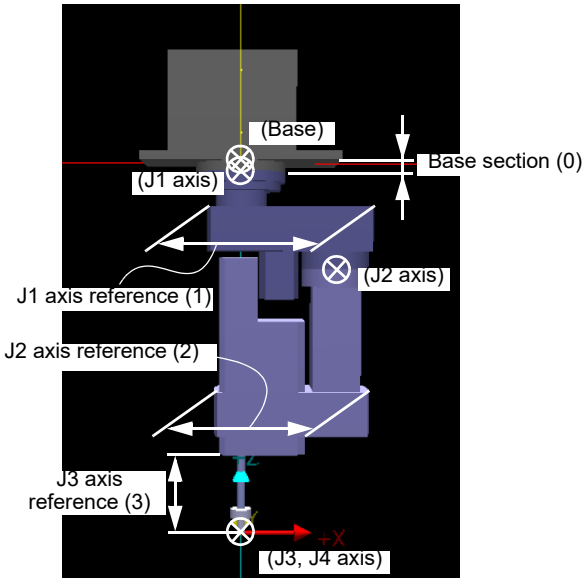
Table 5-30:Simulated component setting parameter (robot arm: CAVKDA1 to 8)

Parameter	Parameter name	No. of arrays No. of characters	Details explanation	Factory setting	
Registration section and shape of simulated component (robot arm)	CAVKDA1 to 8	Integer 2	Set the registration section (Jn axis) and shape of a simulated component. Up to eight simulated component types can be registered. (Each type corresponds to the last digit (1 to 8) of the parameter name.) 1st element: Registration section (Jn axis) 0: Base section 1 to 6: Jn axis 2nd element: Shape 0: a sphere 1: a cylinder Note) The shape of a simulated component is a sphere.	RH-3/6/12/20FRH series, RH-3/6CRH series: CAVKDA1=0, 1 CAVKDA2=0, 0 CAVKDA3=1, 0 CAVKDA4=2, 0 CAVKDA5=2, 0 CAVKDA6=2, 0 CAVKDA7=2, 0 CAVKDA8=4, 1 Note) The setting value of RH-3FRH35xx, and RH-6FRH35xx and RH-12FRH55xx is CAVKDA3=0,0. RH-3FRHR series CAVKDA1=0, 0 CAVKDA2=1, 1 CAVKDA3=1, 0 CAVKDA4=2, 1 CAVKDA5=2, 1 CAVKDA6=3, 1 CAVKDA7=0, 0 CAVKDA8=0, 0	RV-FR series, RV-8CRL: CAVKDA1=0, 0 CAVKDA2=0, 1 CAVKDA3=2, 1 CAVKDA4=4, 1 CAVKDA5=5, 0 CAVKDA6=0, 0 CAVKDA7=0, 0 CAVKDA8=0, 0 Note) The setting value of RV-2FR is CAVKDA2=0,0 CAVKDA4=3,0 CAVKDA5=4,1 CAVKDA6=5,0. RV-8CRL is CAVKDA4=3,0 CAVKDA5=4,1 CAVKDA6=5,1.

<RH-3FRH/6FRH/12FRH/20FRH series>



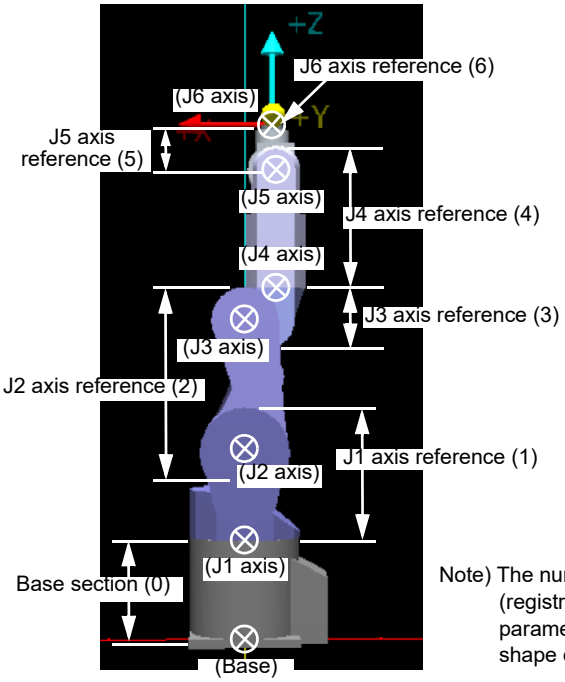
<RH-3FRHR series>



Note) The numbers in parentheses indicate the 1st element (registration section (Jn axis)) setting values of the parameters: CAVKDA1 to 8 (registration section and shape of simulated components).

Fig.5-17:Registration sections of simulated components of RH-FR series (supplement)

<RV-FR series>



Note) The numbers in parentheses indicate the 1st element (registration section (Jn axis)) setting values of the parameters: CAVKDA1 to 8 (registration section and shape of simulated components).

Fig.5-18:Registration sections of simulated components of RV-FR series (supplement)

<RH-FR series>

<RV-FR series>

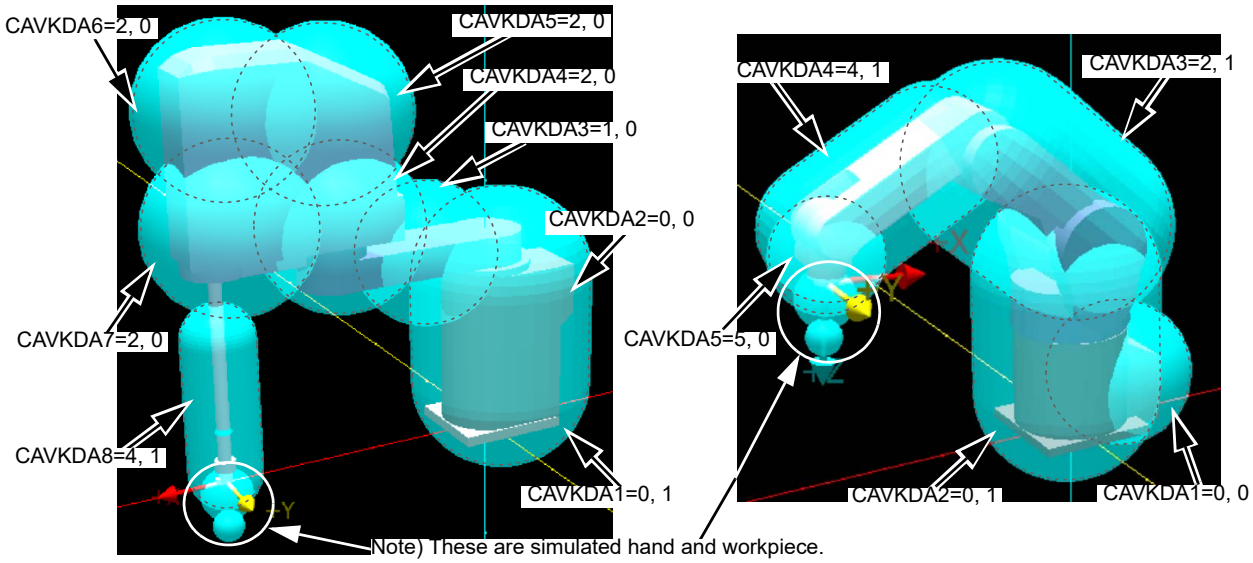
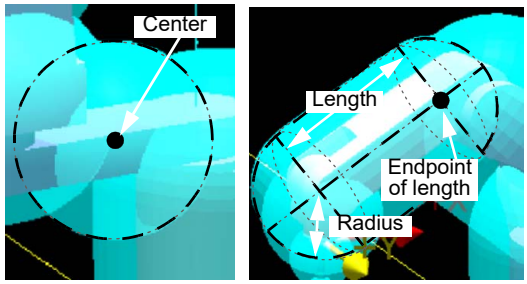


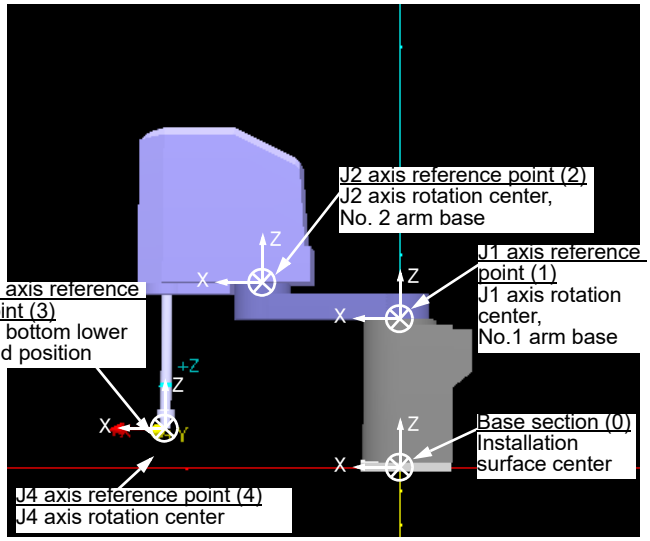
Fig.5-19:Example of registration sections and simulated component type settings

<2>Center position of simulated components: CAVPSA1 to 8

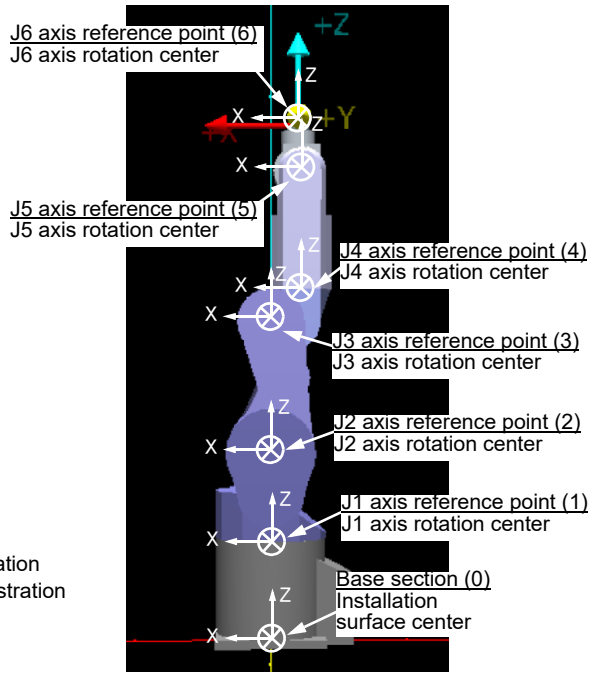
Table 5-31:Simulated component setting parameter (Robot arm: CAVPSA1 to 8)

Parameter	Parameter name	No. of arrays No. of characters	Details explanation	Factory setting
Center position of the simulated component (robot arm)	CAVPSA1 to 8	Real number 6	Set up the position of each simulated component as the distance and the rotation angle from each reference point. (Each simulated component corresponds to the last digit (1 to 8) of the parameter name.) 1st element: Distance in X axis direction (mm) 2nd element: Distance in Y axis direction (mm) 3rd element: Distance in Z axis direction (mm) 4th element: angle of rotation on X axis (degree) 5th element: angle of rotation on Y axis (degree) 6th element: angle of rotation on Z axis (degree) Note) Calculate the rotation angle in order of Z -> Y -> X axis. If shape is the sphere, setting of the rotation angle is unnecessary. <The position of the simulated component to set up>  Sphere Cylinder	

<RH-FR series>



<RV-FR series>



Note) <1> The numbers in parentheses indicate the 1st element (registration section) setting values of the parameters CAVKDA1 to 8 (registration sections and simulated component types).
<2> Top line: registration section
Bottom line: reference point of Jn axis

Fig.5-20:XYZ direction of reference point (Jn axis) for each simulated component

[Supplement]: XYZ direction of each reference point (Jn axis)

RH-FR series

When the posture is J1, J2, J4 axis = 0 degrees and the J3 axis is at the lowest end position a simulated component's XYZ direction matches the base coordinate system.

RV-FR series

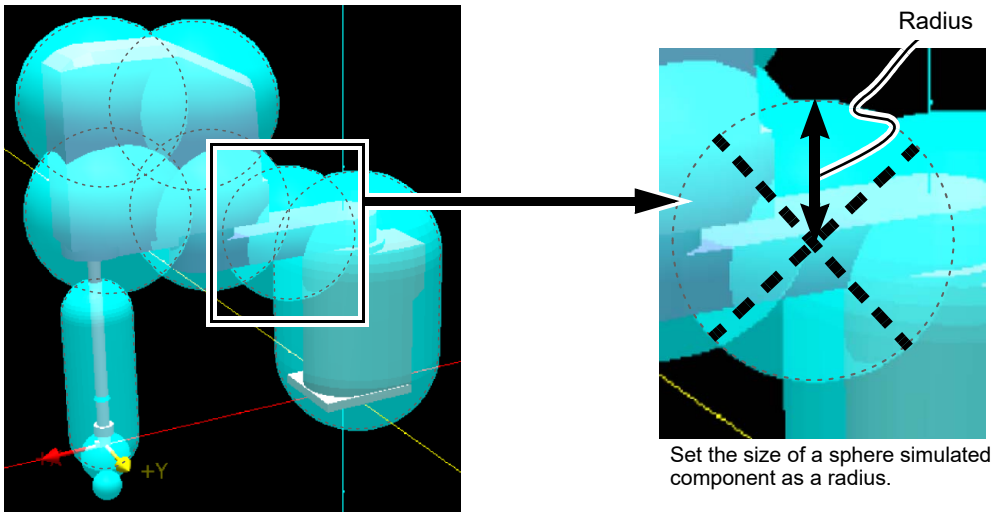
When the posture is all axis = 0 degrees a simulated component's XYZ direction matches the base coordinate system.

<3>Simulated component size: CAVSZA1 to 8

Table 5-32:Simulated component setting parameter (Robot arm: CAVSZA1 to 8)

Parameter	Parameter name	No. of arrays No. of characters	Details explanation	Factory setting
Simulated component (robot arm) size	CAVSZA1 to 8	Real number 4	Set the size of each simulated component. (Each simulated component corresponds to the last digit (1 to 8) of the parameter name.) 1st element: Radius (mm) 2nd element: Length (mm) 3rd element: Fixed to 0 4th element: Fixed to 0 Note) Setting of length is unnecessary if shape is the spheric.	

<RH-FR series>



<RV-FR series>

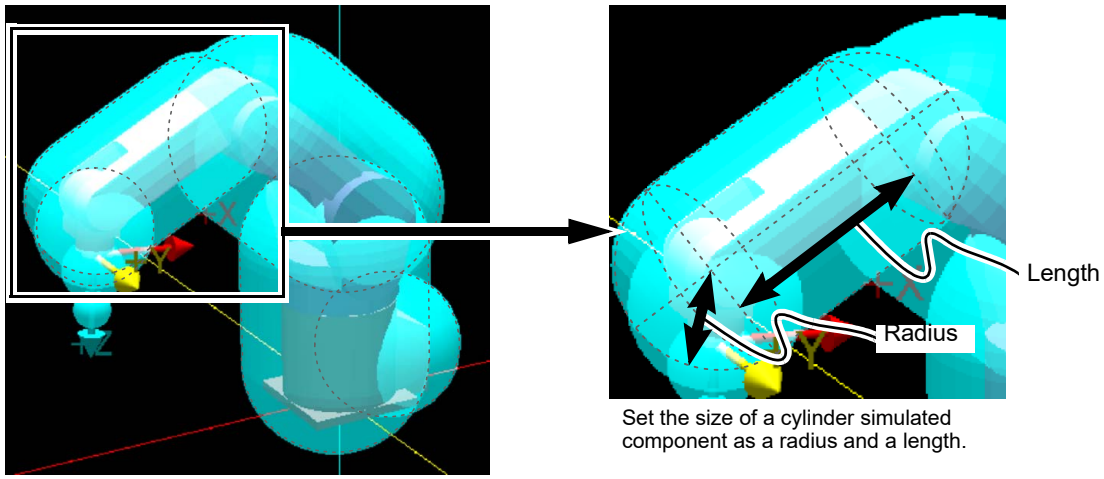


Fig.5-21:Simulated component size (supplement)

<4>Simulated component enable/disable: CAVSCA1 to 8

Table 5-33:Simulated component setting parameter (Robot arm: CAVSCA1 to 8)

Parameter	Parameter name	No. of arrays No. of characters	Details explanation	Factory setting	
Simulated component enable/disable (robot arm)	CAVSCA1 to 8	Integer 3	Set whether to check (enable or disable) interference for each simulated component. (Each simulated component corresponds to the last digit (1 to 8) of the parameter name.) 1st element: Enable/disable the setting (0: Disable, 1: Enable) 2nd element: Set whether to let the interference avoidance function to temporarily disable interference checks during jog operation. (Refer to Temporarily canceling the interference avoidance function.) (0: Disable, 1: Maintain enabled state) Note) If interference is inevitable during jog operation for teaching, setting the simulated hand or workpiece to “0: Disable” can be convenient. 3rd element: Fixed to 0	RH-3/6/12/20FRH series, RH-3/6CRH series: CAVSCA1=1, 0, 0 CAVSCA2=0, 0, 0 CAVSCA3=1, 0, 0 CAVSCA4=1, 0, 0 CAVSCA5=1, 0, 0 CAVSCA6=1, 0, 0 CAVSCA7=1, 0, 0 CAVSCA8=1, 0, 0 Note) The setting value of RH-3FRH35xx, and RH-6FRH35xx and RH-12FRH55xx is CAVSCA3=0,0,0. RH-3FRHR series CAVSCA1=1, 0, 0 CAVSCA2=1, 0, 0 CAVSCA3=1, 0, 0 CAVSCA4=1, 0, 0 CAVSCA5=1, 0, 0 CAVSCA6=1, 0, 0 CAVSCA7=0, 0, 0 CAVSCA8=0, 0, 0	RV-FR series: CAVSCA1=1, 0, 0 CAVSCA2=1, 0, 0 CAVSCA3=1, 0, 0 CAVSCA4=1, 0, 0 CAVSCA5=1, 0, 0 CAVSCA6=0, 0, 0 CAVSCA7=0, 0, 0 CAVSCA8=0, 0, 0 Note) The setting value of RV-2FR and RV-8CRL is CAVSCA6=1,0,0.

2) Simulated components for hand

The parameters required to register a simulated hand are shown in Table 5-34.

Table 5-34:Simulated component setting parameters (hand)

Parameter	Parameter name	No. of arrays No. of characters	Details explanation	Factory setting
Hand number and shape (hand)	CAVKDH1 to 8	Integer 2	Set the hand number and shape of a simulated hand to be registered. Up to eight simulated hand types can be registered. (Each type corresponds to the last digit (1 to 8) of the parameter name.) 1st element: Hand number Corresponds to the hand condition number for changing the simulated component with the Loadset (Load Set) command. 0: Simulated component type set as default. 1 to 8: Hand condition number designated with the Loadset (Load Set) command 2nd element: Shape 0: a sphere 1: a cylinder	Set all parameters (CAVKDH1 to 8) to “0, 0”.
Center position of simulated component (hand)	CAVPSH1 to 8	Real number 6	For each simulated component, designate the center position and pose of the simulated component from the origin point of the Mechanical interface coordinate system. (Each simulated component corresponds to the last digit (1 to 8) of the parameter name.) 1st element: Distance in X axis direction (mm) 2nd element: Distance in Y axis direction (mm) 3rd element: Distance in Z axis direction (mm) 4th element: angle of rotation on X axis (degree) 5th element: angle of rotation on Y axis (degree) 6th element: angle of rotation on Z axis (degree) Note) Calculate the rotation angle in order of Z → Y → X axis. If shape is the sphere, setting of the rotation angle is unnecessary.	Set all parameters (CAVPSH1 to 8) to “0, 0, 0, 0, 0, 0”.

Parameter	Parameter name	No. of arrays No. of characters	Details explanation	Factory setting
Simulated component size (hand)	CAVSZH1 to 8	Real number 4	Set the size of each simulated component. (Each simulated component corresponds to the last digit (1 to 8) of the parameter name.) 1st element: Radius (mm) 2nd element: Length (mm) 3rd element: Fixed to 0 4th element: Fixed to 0 Note) If shape is the sphere, setting of the length is unnecessary.	Set all parameters (CAVSZH1 to 8) to “0, 0, 0, 0”.
Simulated component enable/disable (hand)	CAVSCH to 8	Integer 3	Set whether to check (enable or disable) interference for each simulated component. (Each simulated component corresponds to the last digit (1 to 8) of the parameter name.) 1st element: Enable/disable setting (0: Disable, 1: Enable) 2nd element: Set whether to let the interference avoidance function to temporarily disable interference checks during jog operation. (Refer to Temporarily canceling the interference avoidance function.) (0: Disable, 1: Maintain enabled state) Note) If interference is inevitable during jog operation for teaching, setting the simulated hand or workpiece to “0: Disable” can be convenient. 3rd element: Fixed to 0	Set all parameters (CAVSCH1 to 8) to “0, 0, 0”.

Changing the simulated component to be checked at program execution (hand, workpiece)

When a program is executed, the simulated hand and workpiece targeted for the interference check with the Loadset (Load Set) command can be changed so that the interference is checked according to the hand type actually being used or the workpiece type actually being grasped.
The Loadset (Load Set) command designates a hand number or workpiece number, which is set with parameter beforehand.

3)Simulated components for workpiece
The parameters required to register a simulated workpiece are shown in Table 5-35.

Table 5-35:Simulated component setting parameters (workpiece)

Parameter	Parameter name	No. of arrays No. of characters	Details explanation	Factory setting
Workpiece number and shape (workpiece)	CAVKDW1 to 8	Integer 2	Set the workpiece number and shape of simulated workpiece to be registered. Up to eight simulated workpiece types can be registered. (Each type corresponds to the last digit (1 to 8) of the parameter name.) 1st element: Workpiece number Corresponds to the workpiece condition number for changing the simulated component with the Loadset (Load Set) command. 0: Simulated component type set as default. 1 to 8: Workpiece condition number designated with the Loadset (Load Set) command 2nd element: Shape 0: a sphere 1: a cylinder	Set all parameters (CAVKDW1 to 8) to “0, 0”.

Parameter	Parameter name	No. of arrays No. of characters	Details explanation	Factory setting
Center position of simulated component (workpiece)	CAVPSW1 to 8	Real number 6	For each simulated component, designate the center position and pose of the simulated component from the origin point of the Mechanical interface coordinate system. (Each simulated component corresponds to the last digit (1 to 8) of the parameter name.) 1st element: Distance in X axis direction (mm) 2nd element: Distance in Y axis direction (mm) 3rd element: Distance in Z axis direction (mm) 4th element: angle of rotation on X axis (degree) 5th element: angle of rotation on Y axis (degree) 6th element: angle of rotation on Z axis (degree) Note) Calculate the rotation angle in order of Z → Y → X axis. If shape is the sphere, setting of the rotation angle is unnecessary.	Set all parameters (CAVPSW1 to 8) to “0, 0, 0, 0, 0, 0”.
Simulated component size (workpiece)	CAVSZW1 to 8	Real number 4	Set the size of each simulated component. (Each simulated component corresponds to the last digit (1 to 8) of the parameter name.) 1st element: Radius (mm) 2nd element: Length (mm) 3rd element: Fixed to 0 4th element: Fixed to 0 Note) If shape is the sphere, setting of the length is unnecessary.	Set all parameters (CAVSZW1 to 8) to “0, 0, 0, 0”.
Simulated component enable/disable (workpiece)	CAVSCW1 to 8	Integer 3	Set whether to check (enable or disable) interference for each simulated component. (Each simulated component corresponds to the last digit (1 to 8) of the parameter name.) 1st element: Enable/disable setting (0: Disable, 1: Enable) 2nd element: Set whether to let the interference avoidance function to temporarily disable interference checks during jog operation. (Refer to Temporarily canceling the interference avoidance function.) (0: Disable, 1: Maintain enabled state) Note) If interference is inevitable during jog operation for teaching, setting the simulated hand or workpiece to “0: Disable” can be convenient. 3rd element: Fixed to 0	Set all parameters (CAVSCW1 to 8) to “0, 0, 0”.

Changing the simulated component to be checked at program execution (hand, workpiece)

When a program is executed, the simulated hand and workpiece targeted for the interference check with the Loadset (Load Set) command can be changed so that the interference is checked according to the hand type actually being used or the workpiece type actually being grasped.
TheLoadset (Load Set) command designates a hand number or workpiece number, which is set with parameter beforehand.

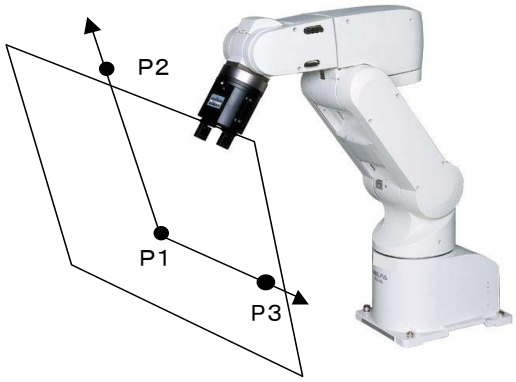
Executing interference check only when workpiece is grasped

Interference checks are performed for a simulated workpiece only while the workpiece is grasped with the parameter HNDHOLD*. (Interlinked with hand open/close)

5.24.4 Registering a free plane limit

Register a free plane limit used for checking for interference. The required registration details are shown in Table 5-36. Maximum of eight planes can be used for checking for interference.

CAUTION An interference check is executed only by registered CPU.



As can be seen in the diagram to the left, any plane can be defined by three points (P1, P2, and P3).

Note) The figure is an example of a vertical 6-axis robot.
In horizontal 4-axis robot is the same.

Table 5-36:Details of free plane limit registration

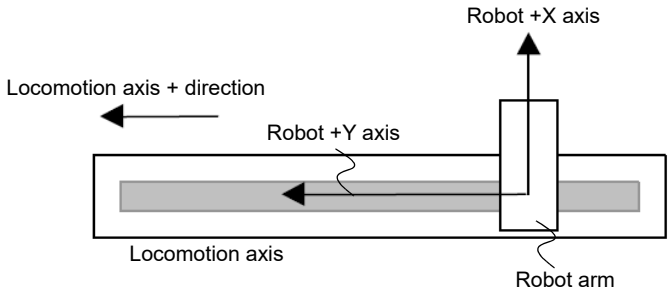
Parameter and value	Explanation
SFCnP(n=1 to 8)	Specifies the 3 points that define the plane. P1 coordinates X1, Y1, and Z1: The origin of the plane P2 coordinates X2,Y2,Z2: A position on the X axis of the plane P3 coordinates X3,Y3,Z3: A position in the positive Y direction of the X-Y plane in the plane
CAVSCFn(n=1 to 8)	Specifies the valid/Invalid of checking for interference with the set free plane limit. 0:Invalid (default) 1: Valid

5.24.5 Support of additional axes

If the robot uses additional axes, the interference avoidance function can be set to consider the movement of the additional axis by setting the additional axis synchronous control parameter. (Only linear axis such as locomotion axis.)

Refer to Table 5-37 and set the parameter according to the usage state.

Note) Set the traveling axis coordinate as “0” for the positional relation between robots, which are explained in a later section, "5.24.7Calibration between robots (Checking for interference between robots)".



In this case, parameter AXDIR is set to 0.0, 0.0, -90.0.
(-90 degree rotation around robot’s Z axis.)

Fig.5-22:Example of using locomotion axis

Table 5-37:Additional axis synchronous control parameter

Parameter	Parameter name	No. of arrays No. of characters	Details explanation	Factory setting
Interference avoidance additional axis (locomotion axis) number	CAVAXJNO	Integer 1	Set up the axial number of the additional axis (locomotion axis) made into the target of interference avoidance. The interference check is not made other than 7 and 8 axes. Setting value: 0, 7 or 8	0
Additional axis Synchronization direction	AXDIR	Real number 3	Convert the coordinate using the locomotion axis’ + direction as the X axis to the coordinate of the robot coordinate system. 1st element: Rotation angle around X axis 2nd element: Rotation angle around Y axis 3rd element: Rotation angle around Z axis Note) As the default, the robot X axis matches the locomotion axis’ + direction.	0.0, 0.0, 0.0

5.24.6 Setting the CPU buffer memory expanded function (Checking for interference between robots)

Set the CPU buffer memory expanded function with the parameters.
When the shared memory expanded function is selected, the CPU buffer memory is occupied. (Refer to Fig. 5-23.)

(1) Parameter setting

Set the parameter: IQMEM bit 4 to "1" and enable the interference avoidance function.
Set the number of CPU modules mounted on the main base unit in the multi-CPU system in parameter: QMLTCPUN, and set the parameter: QMLTCPUn element 1 to “2”.

Table 5-38:CPU buffer memory expanded function selection parameter

Parameter	Parameter name	No. of arrays No. of characters	Details explanation	Factory setting
CPU buffer memory expanded function selection	IQMEM	Real number 1	Select the CPU buffer memory expanded function. A function is assigned to each bit. 1/0 = Enable/disable 15 0 00000000 00000000 | | |...bit0: Enable expanded function Bits 2, 3, and 5 to 15 are not used. | | ....bit1: PLC direct execution function | | ... bit4: Interference avoidance function	00000000 00000000
Number of CPU modules	QMLTCPUN	Integer 1	Set the number of CPU modules mounted in the main base unit of the multi-CPU system.	2

<table>
<tr><th>Parameter</th><th>Parameter name</th><th>No. of arrays
No. of characters</th><th>Details explanation</th><th>Factory setting</th></tr>
<tr><td>CPU buffer memory periodical communication area setting
(* CR800-R only)</td><td>QMLTCPUn
n=1 to 4</td><td>Integer 4</td><td>At the multi CPU system, read the number of points transmitting and receiving data by the CPU buffer memory periodical communication area of the multi CPU No. 1 to 4 from the multi CPU No. 1, and set it automatically. It is not necessary to change the value.

Element 1: Size of the periodical communication area (K word)
Range: 0 to 12
*The total size of all multi CPUs is 24K words at maximum.

Element 2: Number of automatically refreshed points (word)
Range: 0 to 14335
The robot CPU does not support automatic refreshing, so <u>always set</u> the number of automatically refreshed points <u>to 0</u>.

Element 3: System reservation

Element 4: Multi-CPU synchronous startup (1: Enable, 0: Disable)
The robot CPU takes time to start up, so basically <u>do not change this setting</u>. Leave it set to 1 (Enable synchronization).</td><td>1,0,1,1</td></tr>
<tr><td>Multi CPU No. n high-speed communication area setting
(* CR800-Q only)</td><td></td><td></td><td>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 PLC CPU If the parameter settings do not match, and therefore care should be taken to ensure that the parameter settings for each CPU match.

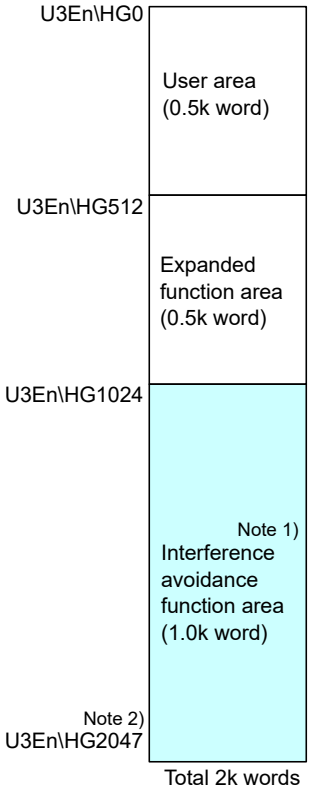
First element: User free area size (k points)
Range: 1 to 14 (Max. *) * The max. value will differ based on the number of multi CPUs as shown below.
<table><tr><th>CPU Qty</th><th>Setting Range</th></tr><tr><td>2</td><td>0 to 14K points</td></tr><tr><td>3</td><td>0 to 13K points</td></tr><tr><td>4</td><td>0 to 12K points</td></tr></table>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.
Third element: System area size (K points)
Range: 1 or 2
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.</td><td></td></tr>
</table>

(2) CPU buffer memory map

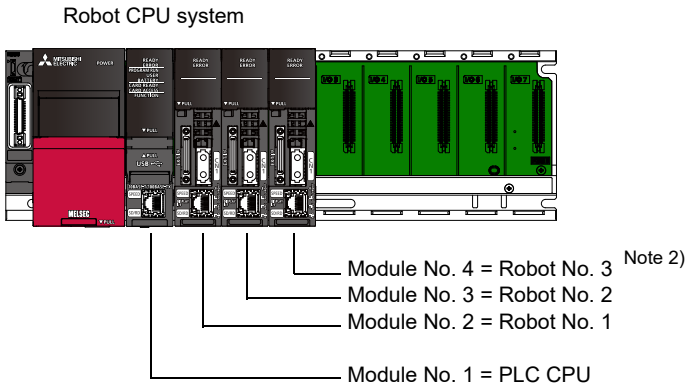
This section shows the memory map of the robot's CPU output area in the CPU buffer memory, which is allocated based on the parameter: IQMEM setting.

The following figure shows the memory map for the CR800-R series. For the CR800-Q series, refer to the Table 5-3 and replace the device numbers with the corresponding device numbers in the figure.

<CPU buffer memory map>



<Relation of slot and robot number>



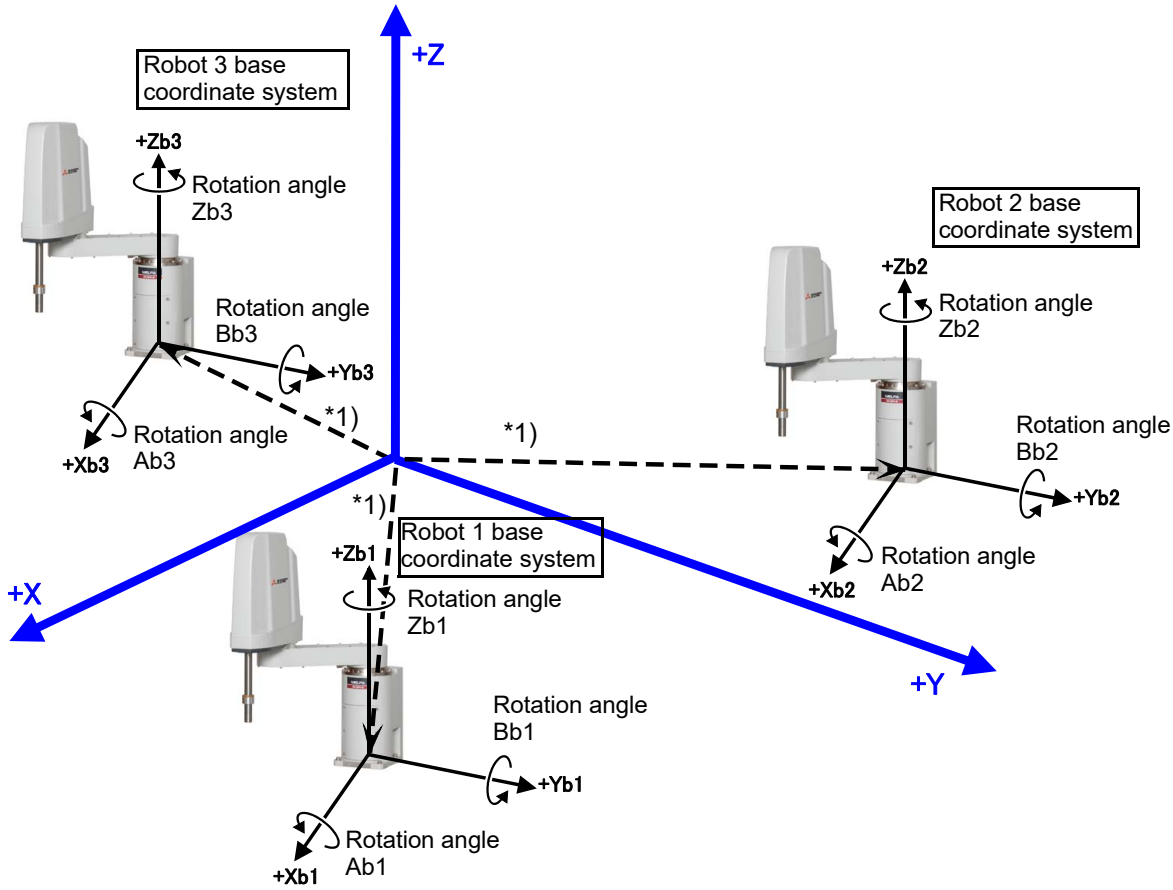
Note 1) The interference avoidance function address is fixed regardless of the expanded function enable/disable state.

Note 2) "n" in U3En\ of the CPU buffer memory address corresponds to the robot No. Each slot position in the robot CPU system (iQPlatform) has a fixed robot No.

Fig.5-23:CPU buffer memory map

5.24.7 Calibration between robots (Checking for interference between robots)

Set the positional relation for multiple robots which are using the interference avoidance function.
Set a common coordinate system origin point between the robots based on the system layout drawing, etc. Then, set the Base coordinate system origin point of each robot in parameter: RBCORD looking from that common coordinate system.



*1) The robot's base coordinate system origin point looking from the common coordinate system. Set these coordinate values in parameter: RBCORD for each robot.

Fig.5-24:Image of calibration between robots

(1) Setting the calibration

Set the Base coordinate system origin point for each robot, looking from the common coordinates between robots, in parameter: RBCORD with the X, Y, Z, A, B and C coordinate values.

Note) When using a locomotion axis, set the positional relation for when the locomotion axis coordinate value is “0”.

Table 5-39:Set parameters for calibration between robots

Parameter	Parameter name	No. of arrays No. of characters	Details explanation	Factory setting
Common coordinates for robots	RBCORD	Real number 6	The robot's Base coordinate system origin point looking from the common coordinates between robots. (Designate with X, Y, Z, A, B and C coordinate values) 1st element: X-axis coordinate value (mm) 2nd element: Y-axis coordinate value (mm) 3rd element: Z-axis coordinate value (mm) 4th element: A-axis coordinate value (deg) (Rotation angle around X axis) 5th element: B-axis coordinate value (deg) (Rotation angle around Y axis) 6th element: C-axis coordinate value (deg) (Rotation angle around Z axis) Note 1) For the A, B and C-axis coordinate values (rotation angles), set the values obtained by rotating in the order of around Z axis → around Y axis → around X axis.	0.00, 0.00, 0.00, 0.00, 0.00, 0.00,

(2) Checking the calibration setting results

Check that each robot has been correctly calibrated with the following steps.

1) Looking at the system layout drawing, etc., set one reference point for each robot. (Fig. 5-25 a) (Hereinafter, reference point)
2) Move the tip of the robot hand to the set reference point using jog operations.
3) Check the robot (system) status variable: P_CurrR (current robot position looking from common coordinates) using T/B, etc.
4) Compare the above P_CurrR value to the value on the layout drawing, etc.
   Only the XYZ values are compared. If the values match, the robot has been correctly calibrated. If the values do not match, correct the parameter: RBCORD setting value.
5) Carry out the above steps for all robots which are using the interference avoidance function.

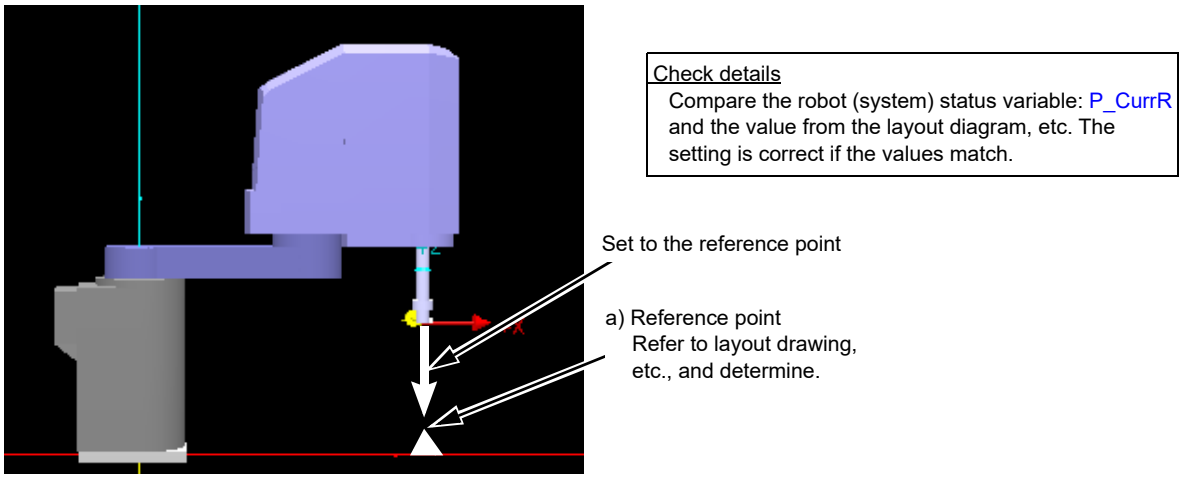


Fig.5-25:Checking the setting for calibration between robots

5.24.8 Enabling and disabling the interference avoidance function

Whether to enable/disable the interference avoidance function in general, and whether to enable/disable it during program execution and jog operation can be set by setting the parameter: CAV.
Details of the parameter are given in Table 5-40.

Table 5-40:Interference avoidance function enable/disable setting parameter

Parameter	Parameter name	No. of arrays No. of characters	Details explanation	Factory setting
Interference avoidance function	CAV	Integer 1	Set whether to enable or disable the interference avoidance function in general. When set to enable, additionally set to enable/disable the interference avoidance function during program execution, and enable (including movement during interference)/disable the function during jog operation. 1st element: Use of the interference avoidance function (0: Disabled/1: Enabled) 2nd element: Initial state at program execution (0: Disabled/1: Enabled) 3rd element: Disable/enable setting for jog operation (0: Disabled, 1: Enabled with error occurrence, 2: Enabled with only buzzer sound)	0, 1, 1

5.24.9 Using the interference avoidance function

The interference avoidance operation during jog operation and program execution is explained below. Set the 1st element of parameter: CAV to “1” (Enable). (Refer to Page 587, "5.24.8 Enabling and disabling the interference avoidance function".)

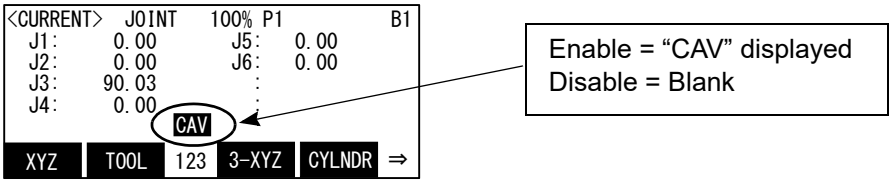
(1) Interference avoidance during jog operation

When the 3rd element of parameter: CAV is set to Enable (“1” or “2”), the interference avoidance function can be used during jog operation.
When interference is detected, the robot decelerates to a stop and performs the following operation according to the parameter’s setting value.

Setting value = 1: Alarm (L241n) occurs. The alarm must be reset to resume jog operation.
Setting value = 2: A buzzer sounds. The robot can be jogged in the direction where there is no interference.

Temporarily canceling the interference avoidance function

Press the [CLEAR] key on T/B to temporary cancel the interference avoidance function for the simulated components. It can be temporarily canceled by setting "0" (temporarily disable) to the second element of the simulated component enable/disable setting parameters: CAVSCA1 to 8 (robot arm), CAVSCH to 8 (hand) and CAVSCW1 to 8 (workpiece).
Every time the [CLEAR] key is pressed, the status switches between Enable and Disable. “CAV” is displayed at the bottom center of the T/B screen when set to Enable.



If the robot cannot be restarted from an interference state by [CLEAR] key (if the operator manually puts the robot into the interference area during servo OFF, etc.), carry out Page 61, "3.11 Operation to Temporarily Reset an Error that Cannot Be Canceled" (jog operation while [RESET] is pressed).
The interference avoidance function can be temporarily canceled and the robot can be moved.

Note) The robot will not stop at the movement range limit when using this operation. Do not move in the direction outside of the movement range.

(2) Interference avoidance during program execution

The interference avoidance function can be used during program execution (during automatic operation) by setting the 2nd element of parameter: CAV to Enable (“1”), and using the instructions shown in Table 5-41 and the external variables shown in Table 5-42.

Table 5-41:Commands (Interference avoidance function)

Command	Details	Explanation page
CavChk	Enable/disable the interference avoidance function, and set whether to generate an alarm when an interference is detected. CavChk <ON/OFF>[, <Robot CPU No.>[,NOErr]]	Page 181

Table 5-42:Robot (system) status variables (Interference avoidance function)

Variable name	Details	Explanation page
P_CordR	The robot’s base coordinate system origin point looking from common coordinate system. Data type: Position type	Page 426
P_CurrR	Local robot’s current position looking from the common coordinate system Data type: Position type	Page 428
P_CavDir	The robot’s movement direction when interference is detected Data type: Position type	Page 424
M_CavSts	Interference state 0: No interference 1 to 3: The number of the interfering robot when interference is detected Data type: Integer type	Page 360

[Supplement]

When an interference is detected, the robot decelerates to a stop. The movement is determined according to whether “NOErr” is designated with the CavChk On (CavChk On) instruction.

NOErr not designated: Alarm (L240n) occurs. The alarm must be reset to resume automatic operation.

NOErr designated: An alarm does not occur, and instead the interrupt process designated in the program is executed.

Note) To designate NOErr, an interrupt program, which is executed when an interruption is detected, must be set.

An alarm will occur if this instruction is executed without an interrupt process being set.

Overview of interrupt process: An interference is detected when the robot (system) status variable M_CavSts value changes from “0”. Make a declaration to execute the interrupt process in this state. A sample program is given on Page 589, "5.24.10 Sample programs".

⚠CAUTION If origin setting is not completed, the current joint position of all axes will be executed as 0 always.
For this reason, this function does not operate correctly. Executes this function, after making origin setting certainly complete.

5.24.10 Sample programs

(1) Starting and ending the interference avoidance function (all robots)

<Program example> Note) The step numbers are omitted.

```
  :
'--- Default state (Interference avoidance function enabled) ---;When interference check is enabled
                                                                  with parameter: CAV.
  :
MVS P1          'Movement when the interference avoidance function is enabled
  :
'--- Interference avoidance disabled for all robots ---
CavChk OFF      'Function disabled for all robots when no robot No. is specified
  :
MVS P2          'Movement when the interference avoidance function is disabled
  :
'--- Interference avoidance enabled for all robots---
CavChk ON       'Function enabled for all robots when no robot No. is specified
  :
MVS P3          'Movement with the interference avoidance function is enabled
  :
```

(2) Starting and ending interference avoidance function (designated robot)

<Program example> Note) The step numbers are omitted.

```
  :
'--- To end the interference avoidance function with the No. 2 robot ---
CavChk OFF,2    'End the interference avoidance function with the robot No. 2
  :
MVS P5          'Movement when the interference avoidance function with the robot No. 2 is
                 disabled
  :
'--- To start the interference avoidance function with the No. 2 robot ---
CavChk ON,2     'Resume the interference avoidance function with the robot No. 2
  :
MVS P6          'Movement when the interference avoidance function with the robot No. 2 is
                 enabled
  :
```

(3) Changing the simulated hand and workpiece types

<Program example> Note) The step numbers are omitted.

```
  :
'--- Change the simulated hand type ---
LoadSet 2,2     'Designate the simulated hand and workpiece type 2
  :
MVS P7          'Execute the interference avoidance function with the simulated hand and
                 workpiece type 2
  :
'--- Change the simulated component for interference check (workpiece) ---
LoadSet 1,2     'Designate the simulated hand type 1, simulated workpiece type 2
  :
MVS P8          'Execute the interference avoidance function with the simulated hand type 1 and
                 the simulated workpiece type 2
  :
'--- Change the simulated component for interference check (workpiece) ---
LoadSet 0,0     'Return the simulated hand and workpiece types to default values
  :
MVS P9          '
  :
```

(4) Executing avoidance operation after detecting interference (interrupt process)
<Program example> Note) The step numbers are omitted.

```
Def Act 1,M_CavSts<>0 GoTo *Home,S ' Define a process to be executed as an interrupt when
                                     interference is detected
Act 1=1
CavChk On,0,NOErr       ' Enable the interference avoidance function in error disabled mode
Mov P1                  ' Movement when the interference avoidance function is enabled
Mov P2                  ' Movement when the interference avoidance function is enabled
Mov P3                  ' Movement when the interference avoidance function is enabled
  :
  :
*Home                   ' Step to interrupt when interference is detected
CavChk Off
M_CavSts=0              ' Clear the interference state
MDist=Sqr(P_CavDir.X*P_CavDir.X+P_CavDir.Y*P_CavDir.Y+P_CavDir.Z*P_CavDir.Z)
                         ' Find the movement rate ratio
PESC=P_CavDir(1)*(-50)*(1/MDist)
                        ' Create movement amount for avoidance operation (move back 50mm
                          from the interference position)
PDST=P_Fbc(1)+PESC      ' Create an avoidance destination
Mvs PDST                ' Avoidance movement
Mvs PHome               ' Move to the avoidance destination
  :
```

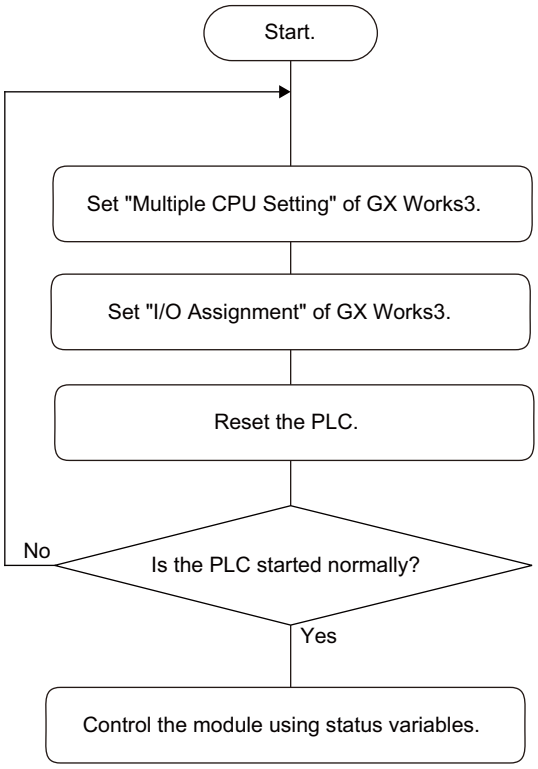
5.25 Direct control of the PLC input/output module

This section explains the function by which the robot CPU controls three types of modules for the PLC: "input module", "output module", and "I/O combined module".
The setting procedure differs between the CR800-R series and the CR800-Q series. For the CR800-R series, refer to Page 591, "5.25.1 CR800-R series". For the CR800-Q series, refer to Page 605, "5.25.2 CR800-Q series"

5.25.1 CR800-R series

(1) Operation procedure

The operation procedure for controlling the PLC I/O module is shown below.



•••●Set the parameters in "Multiple CPU Setting " of GX Works3.

•••●Set the parameters in "I/O Assignment" of GX Works3.

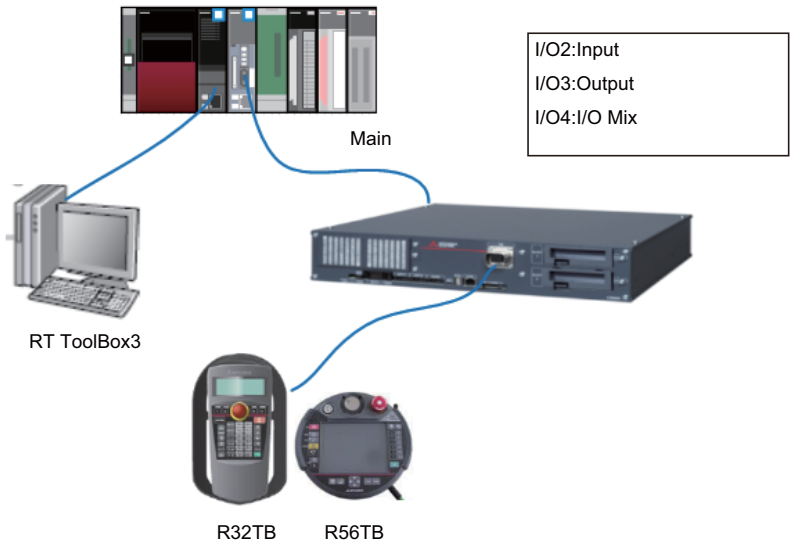
•••●Perform the reset operation of the PLC or turn off and on the power to reset the PLC.

Check the input information with the status variables "M_XDev", "M_XDevB", "M_XDevW", and "M_XDevD", and output a signal to the module controlled by the host CPU with the status variables "M_YDev", "M_YDevB", "M_YDevW", and "M_YDevD".
(Refer to "(5)Controlling with status variables".)

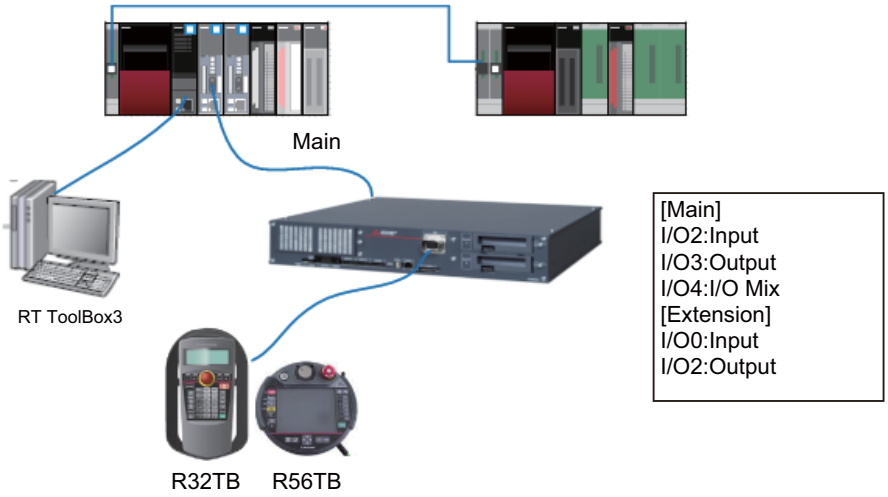
(2) Configuration example of the hardware

The following shows the system configurations "1 robot CPU + 3 types of I/O modules" as the example 1, and "2 robot CPUs + 3 types of I/O modules + extension base (added)" as the example 2.

*System configuration example 1: 1 robot CPU + 3 types of I/O modules

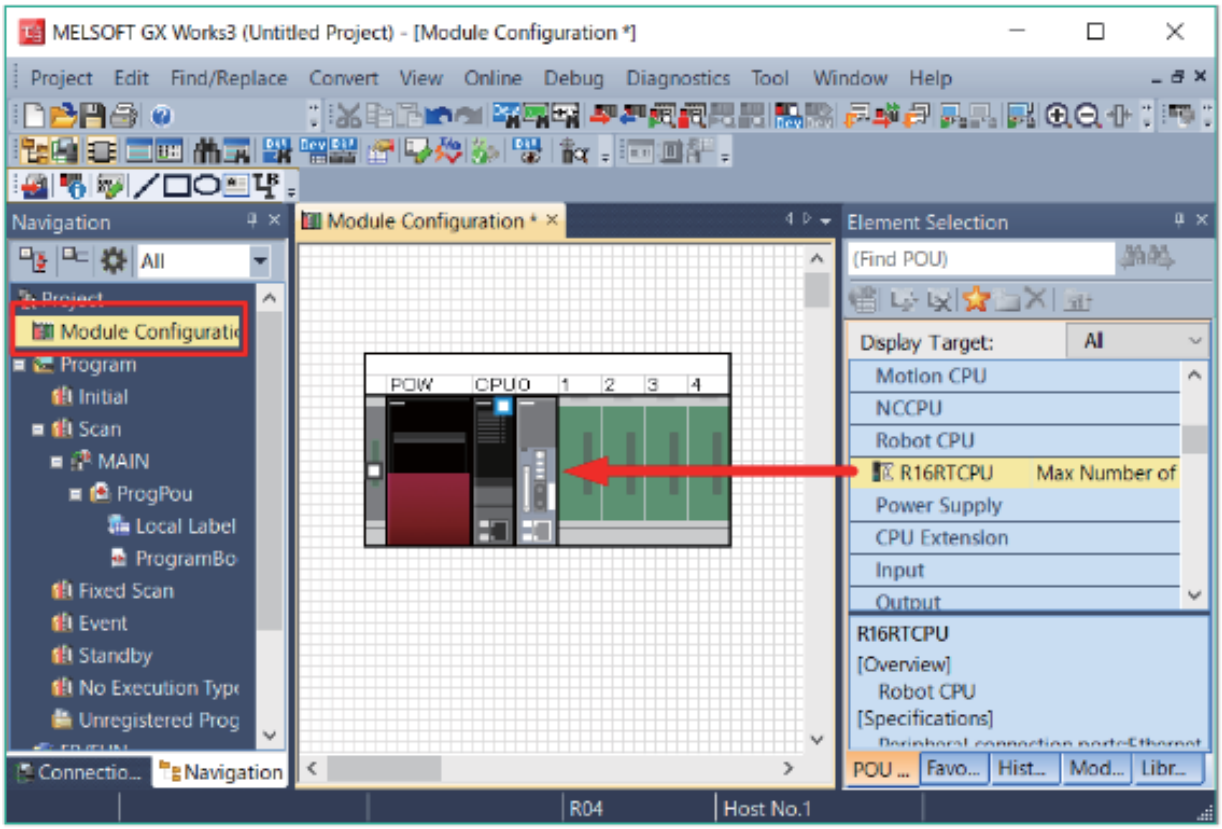


*System configuration example 2: 2 robot CPUs + 3 types of I/O modules + extension base (added)

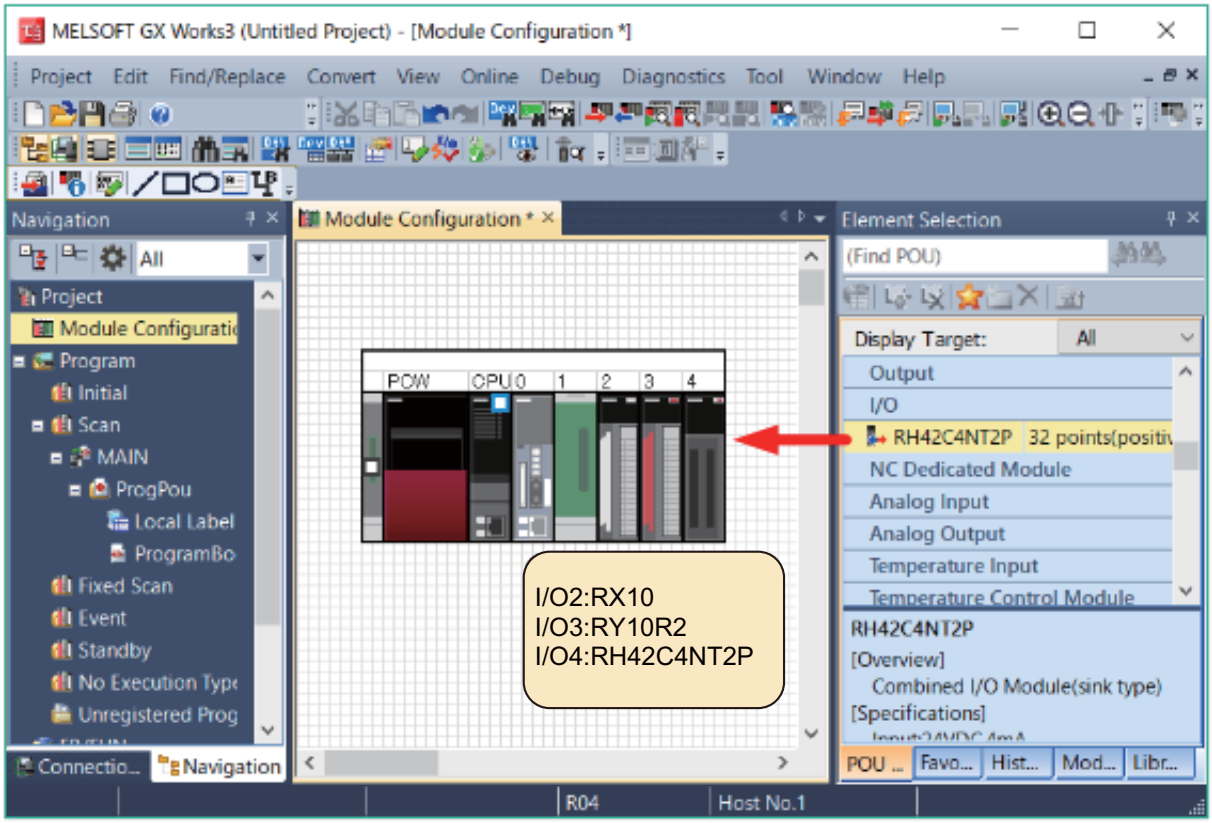


(3) Setting the parameters for the system configuration example 1

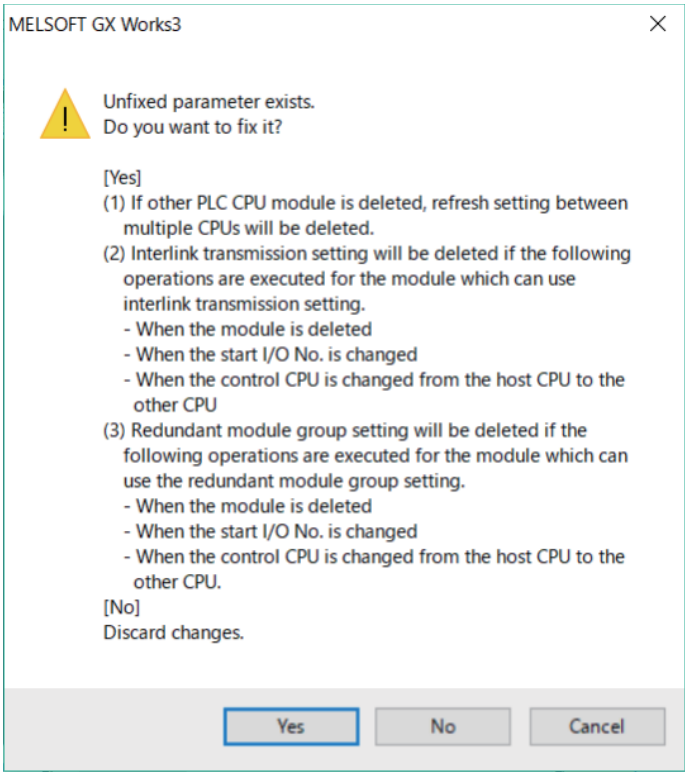
Under the condition of the system configuration example 1 (1 robot CPU + 3 types of I/O modules), arrange the base, power module, PLC CPU, and robot CPU to be used in "Module Configuration" of GX Works3.



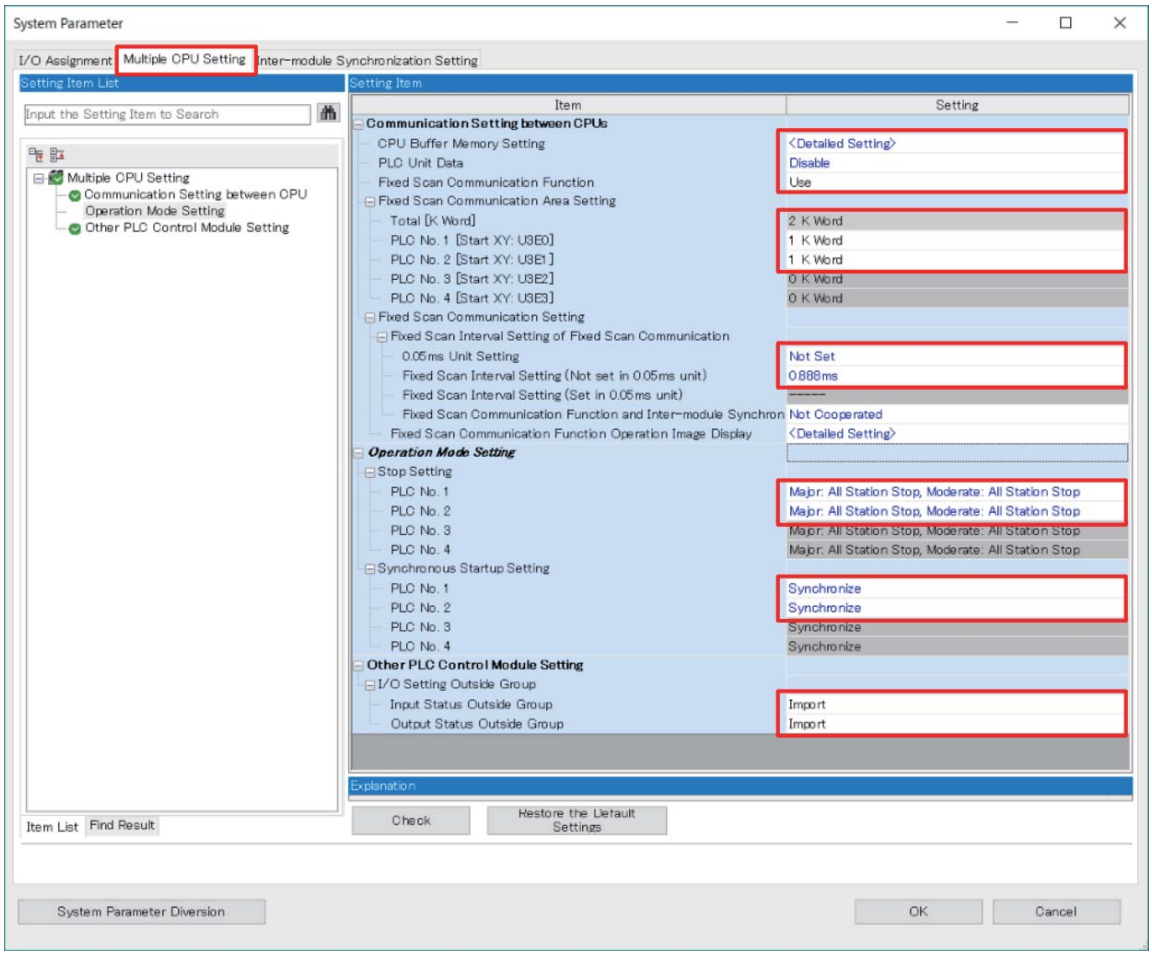
Arrange the input module, output module, and I/O combined module to be used.



When the Module Configuration screen is closed after arranging the modules, the following dialogue will appear. Click the [Yes] button.



Open the System Parameter screen to set up the multiple CPU.

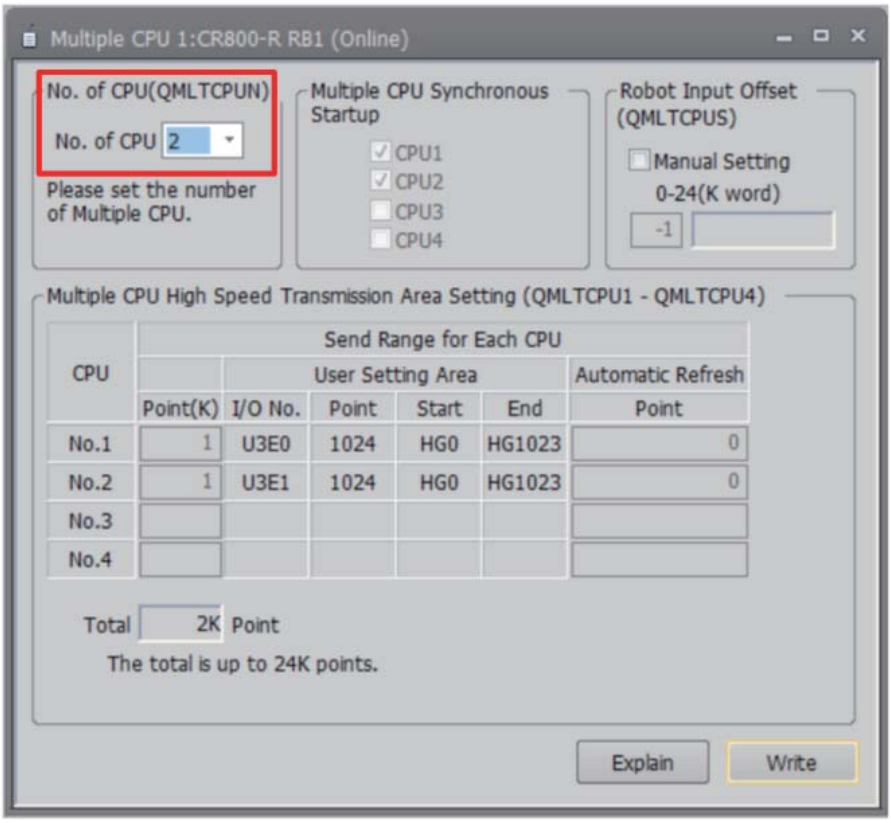


Setting Item		Description	Setting Value
Communication Setting between CPUs	PLC Unit Data	Set up this to avoid data separation at data communications by refresh. When data of more than 64 bit data is handled, data croaking may occur.	Enable Disable The update cycle of the PLC link I/O function of the robot CPU is as follows. < PLC Unit Data - Disable > It will be the specified periodic communication interval. < PLC Unit Data - Enable > Even when setting the periodic communication interval setting to "0.444 ms" or less, it will be 0.888 ms. * The update period of other data is described in each chapter.
	Fixed Scan Communication Function	Select whether the fixed scan communication function is used or not. Always select "Use" when using together with a robot CPU.	Use
	Fixed Scan Communication Area Setting	Set up the range of transmission area of each device in the fixed scan communication area. The necessary area Note1) for robot is as follows: CPU buffer memory extended functions are valid: • **Robot input area: 1.0K** • **Robot output area: 1.0K** CPU buffer memory extended functions are invalid: • **Robot input area: 0.5K** • **Robot output area: 0.5K**	<CPU buffer memory extended functions are valid:> • Device #1: Sum of the size (1K) of the data to be sent to the robot and the size of the data to be sent to other devices • Robot device: Set 1K for it • Other devices: Set its own transmission size <CPU buffer memory extended functions are invalid:> • Device #1: Sum of the size (0.5K) of the data to be sent to the robot and the size of the data to be sent to other devices • Robot device: Set 1K for it Note2) • Other devices: Set its own transmission size
	0.05ms Unit Setting	Specify whether to set the fixed scan communication cycle in increments of 0.05ms. Please be sure to select "Not Set" when combining with robot CPU.	Not Set
	Fixed Scan Interval Setting (Not set in 0.05ms unit)	Select the fixed scan communication cycle to be set from the list items. The same option should be specified only for CPU modules which will use the fixed scan communication function. Please select "0.888ms" or less regardless of the PLC Unit Data. When using cooperative control function or interference avoidance function of the robot CPU.	0.222ms 0.444ms 0.888ms 1.777ms 3.555ms 7.111ms
Operation Mode Setting	Stop Setting	If a moderate or major error occurs at any of the CPUs, set whether to stop or continue operation for all CPUs.	Major: All Station Stop, Moderate: All Station Continue. * Set for all CPUs.
	Synchronous Startup Setting	Set up this to synchronize startup time of CPU modules in multiple CPUs system. * Always select "Synchronize" due to the robot CPU takes as long as 18 seconds to startup.	Synchronize
Other PLC Control Module Setting	I/O Setting Outside Group	Set up this to import the input (X) ON/OFF data of the module outside the control and the output (Y) ON/OFF data of other PLC CPU.	Import

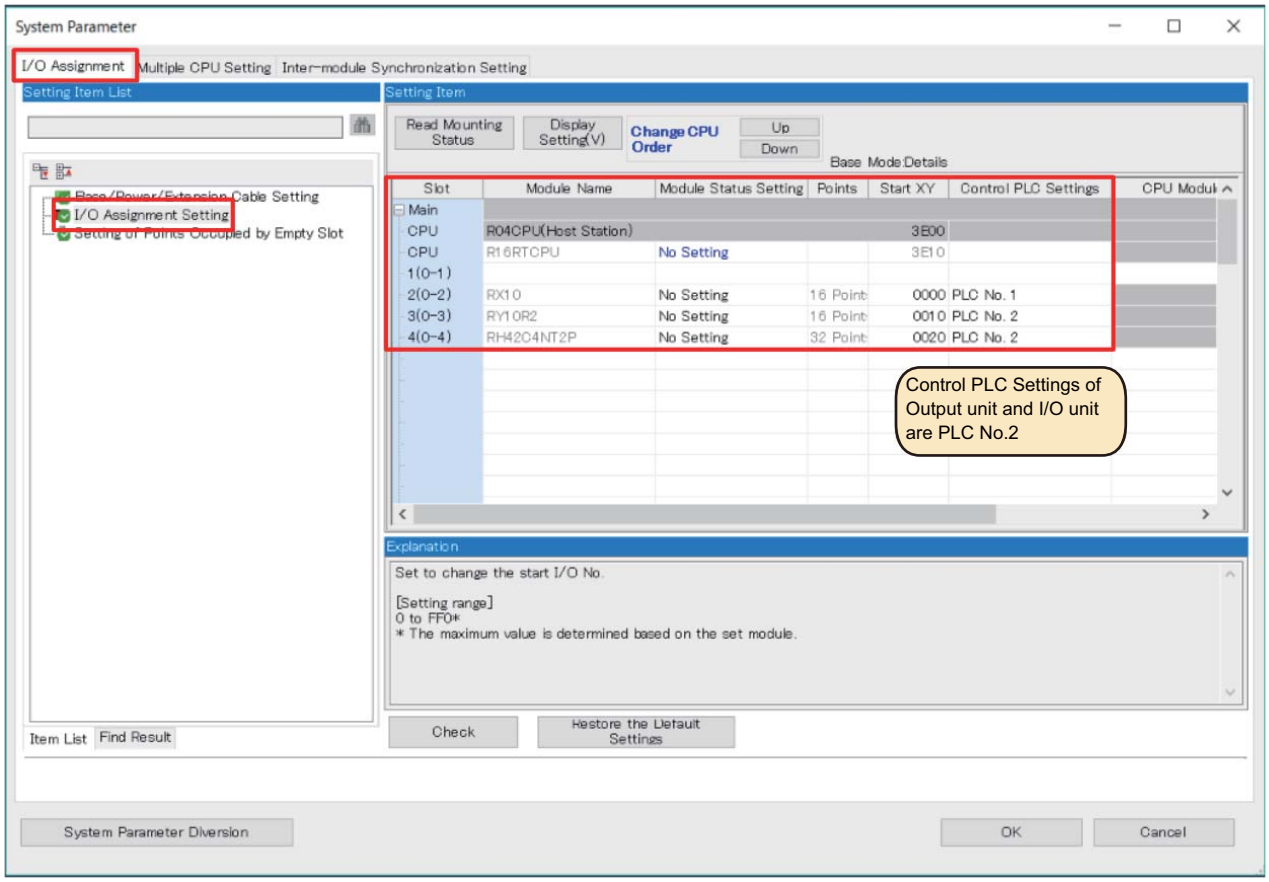
Note1) For information about multiple CPUs and fixed scan communication area, refer to the RCPU manual (MELSEC iQ-R CPU Module User's Manual (Application)).

Note2) Because the area is set up in 1K unit, allocate 1K even in case of 0.5K.

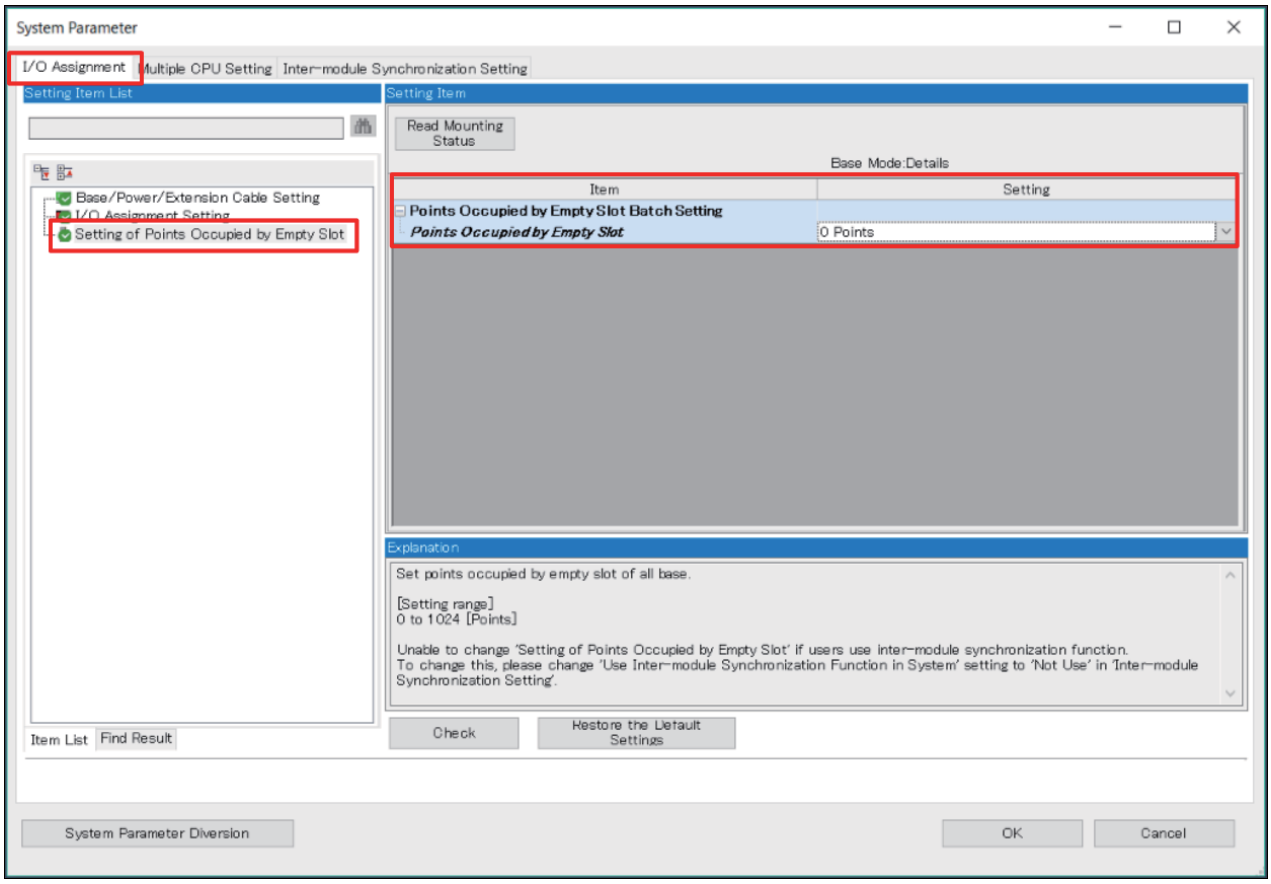
Set up "Multiple CPU Setting" in the parameter menu of RT ToolBox3 as follows.
The settings of "Multiple CPU Synchronous Startup" and "Multiple CPU High Speed Transmission Area Setting" are not required since their setting values are automatically acquired from the PLC CPU.



Set up "I/O Assignment" with the CPU parameters of GX Works3 as follows.



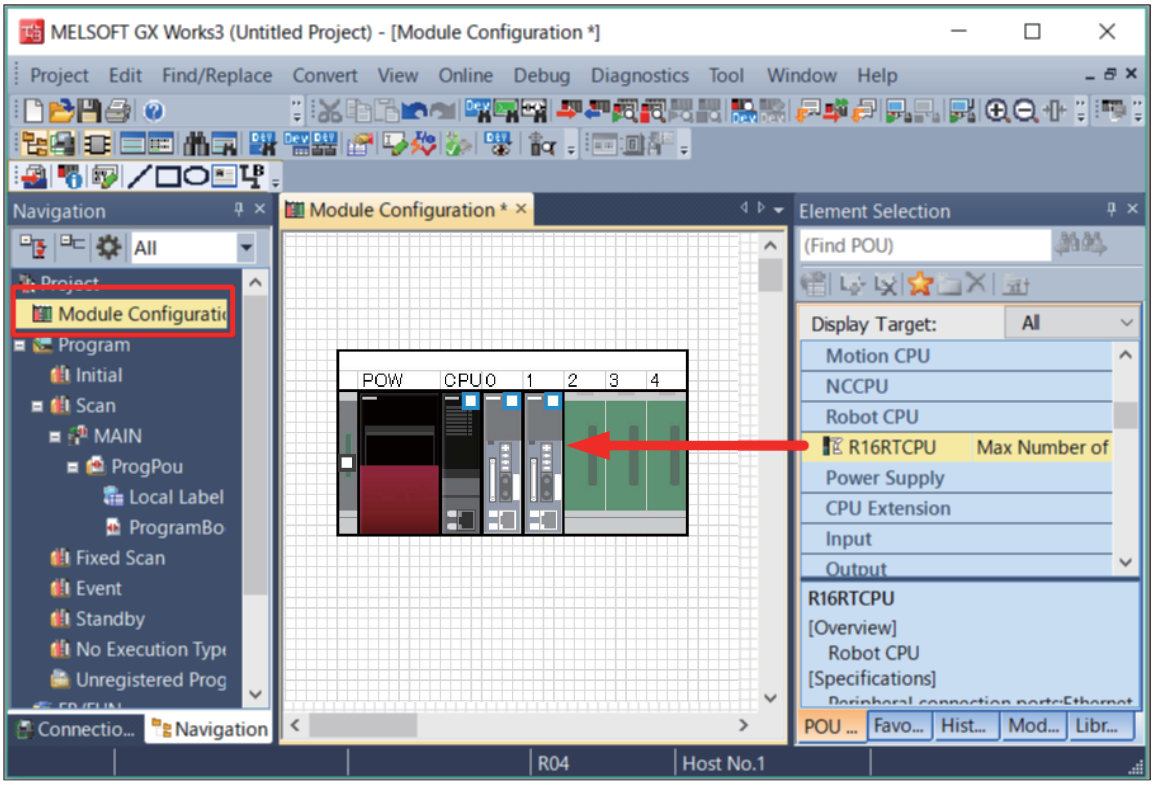
With this setting, the PLC controls the input module, and the robot CPU No. 1 controls the output module and I/O combined module.
If there is any empty slot, set "Setting of Points Occupied by Empty Slot" as follows.



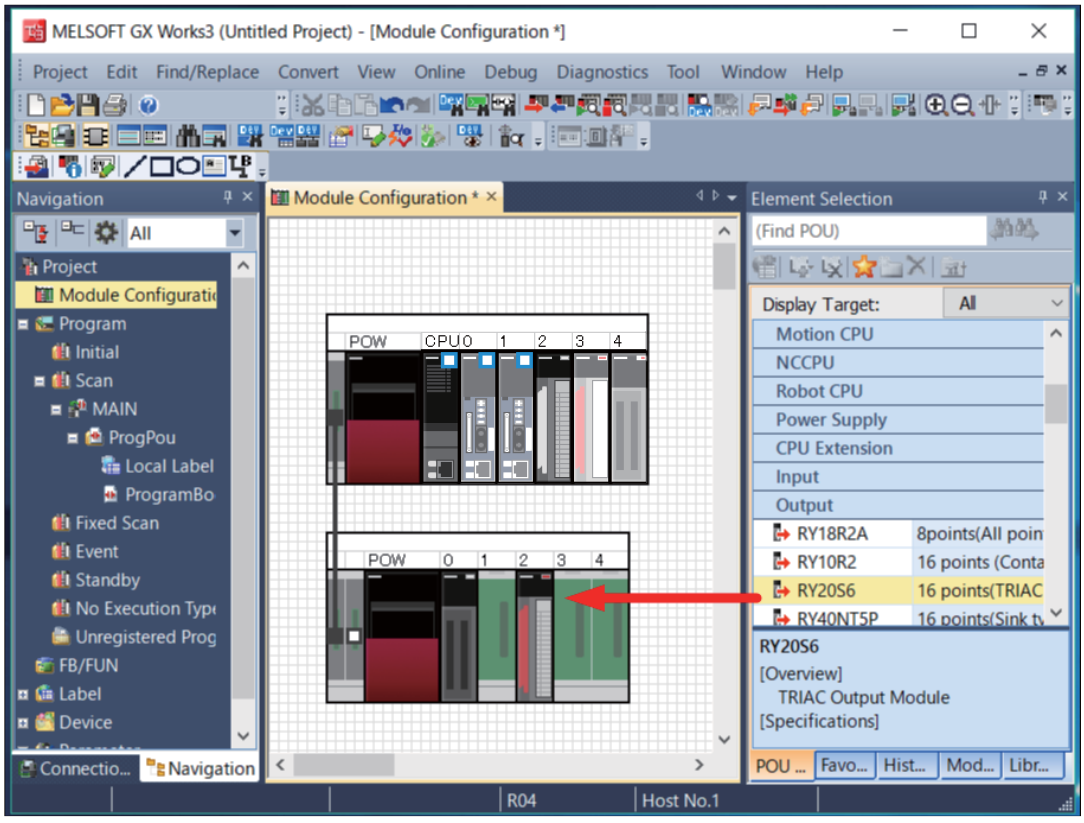
After setting the above parameters, reset the power of the PLC and robot CPU.

(4) Setting the parameters for the system configuration example 2

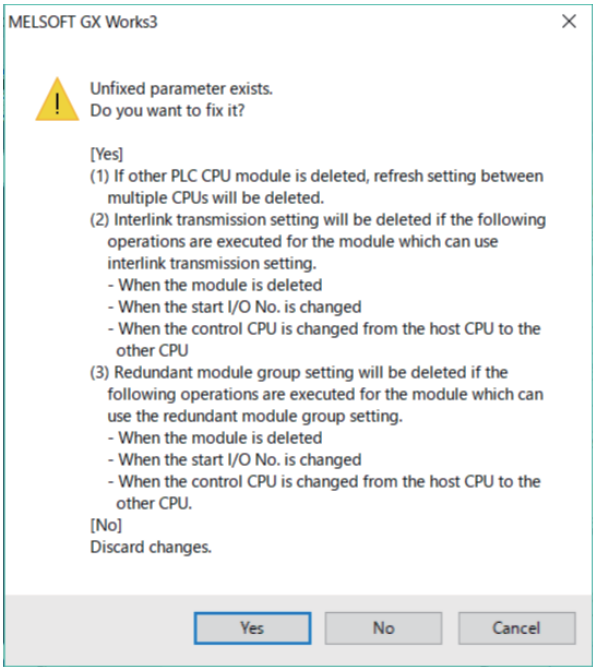
Under the condition of the system configuration example 2 (2 robot CPUs + 3 types of I/O modules + extension base), arrange the base, power module, PLC CPU, and robot CPU to be used in "Module Configuration" of GX Works3.



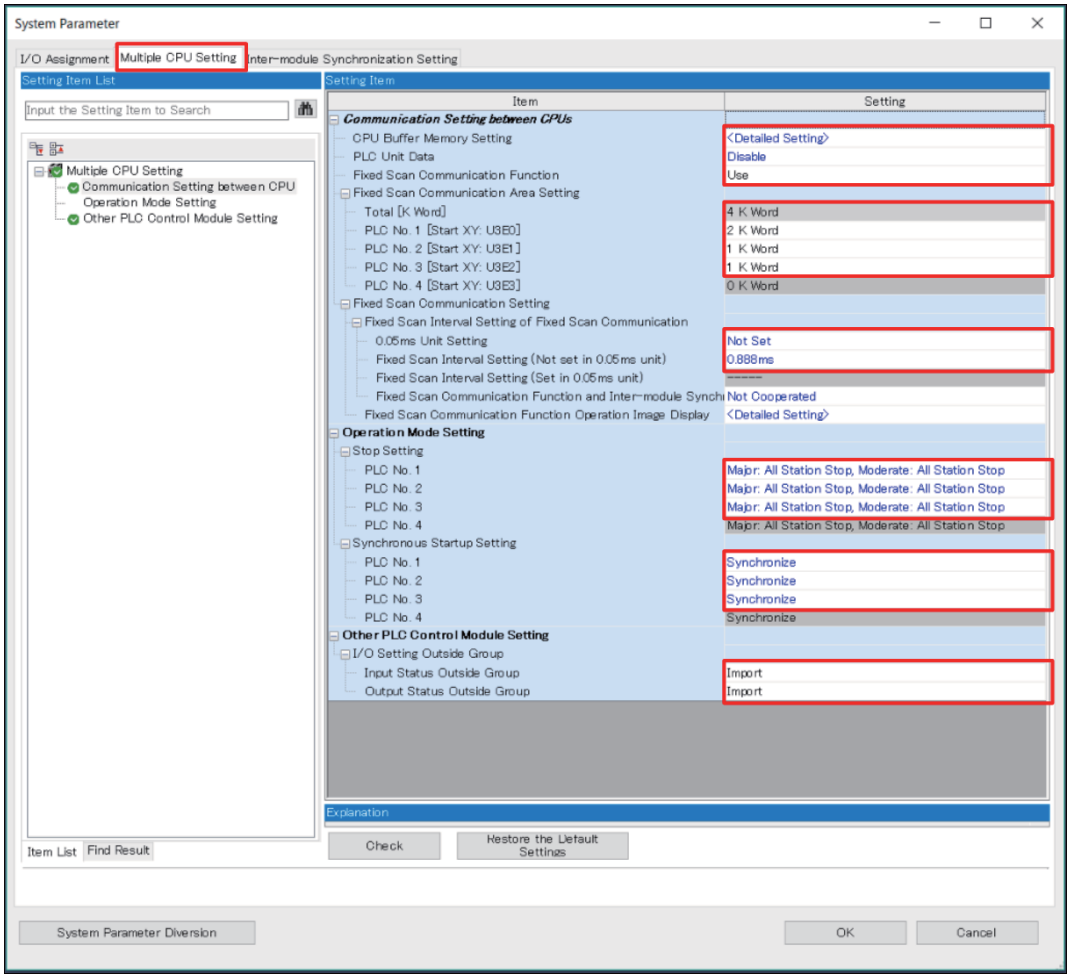
Arrange the extension base, input module, output module, and I/O combined module to be used.



When the Module Configuration screen is closed after arranging the modules, the following dialogue will appear. Click the [Yes] button.



Open the System Parameter screen to set up the multiple CPU.



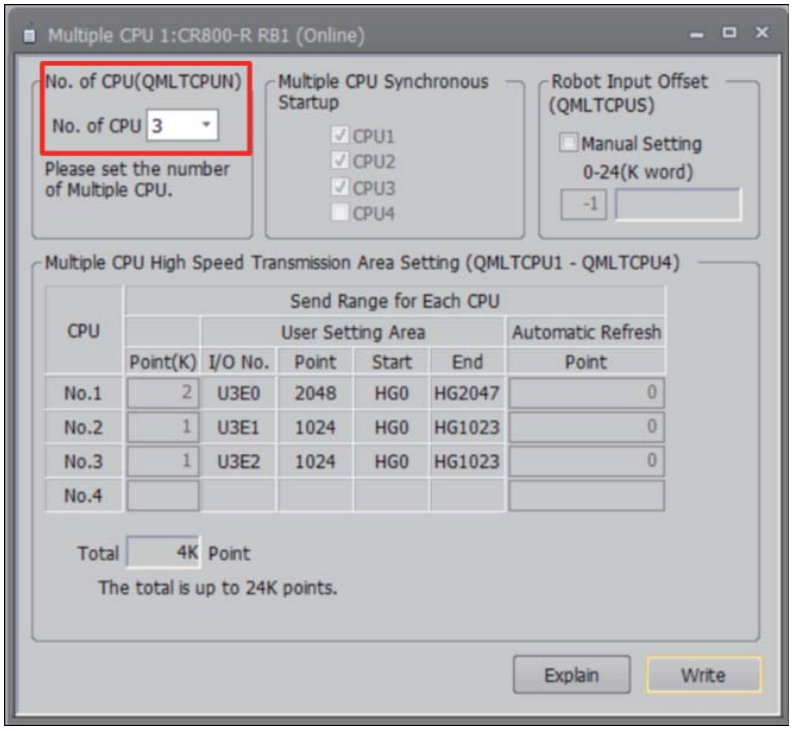
Setting Item		Description	Setting Value
Communication Setting between CPUs	PLC Unit Data	Set up this to avoid data separation at data communications by refresh. When data of more than 64 bit data is handled, data croaking may occur.	Enable Disable The update cycle of the PLC link I/O function of the robot CPU is as follows. < PLC Unit Data - Disable > It will be the specified periodic communication interval. < PLC Unit Data - Enable > Even when setting the periodic communication interval setting to "0.444 ms" or less, it will be 0.888 ms. * The update period of other data is described in each chapter.
	Fixed Scan Communication Function	Select whether the fixed scan communication function is used or not. Always select "Use" when using together with a robot CPU.	Use
	Fixed Scan Communication Area Setting	Set up the range of transmission area of each device in the fixed scan communication area. The necessary area Note1) for robot is as follows: CPU buffer memory extended functions are valid: • **Robot input area: 1.0K** • **Robot output area: 1.0K** CPU buffer memory extended functions are invalid: • **Robot input area: 0.5K** • **Robot output area: 0.5K**	<CPU buffer memory extended functions are valid:> • Device #1: Sum of the size (1K) of the data to be sent to the robot and the size of the data to be sent to other devices • Robot device: Set 1K for it • Other devices: Set its own transmission size <CPU buffer memory extended functions are invalid:> • Device #1: Sum of the size (0.5K) of the data to be sent to the robot and the size of the data to be sent to other devices • Robot device: Set 1K for it Note2) • Other devices: Set its own transmission size
	0.05ms Unit Setting	Specify whether to set the fixed scan communication cycle in increments of 0.05ms. Please be sure to select "Not Set" when combining with robot CPU.	Not Set
	Fixed Scan Interval Setting (Not set in 0.05ms unit)	Select the fixed scan communication cycle to be set from the list items. The same option should be specified only for CPU modules which will use the fixed scan communication function. Please select "0.888ms" or less regardless of the PLC Unit Data. When using cooperative control function or interference avoidance function of the robot CPU.	0.222ms 0.444ms 0.888ms 1.777ms 3.555ms 7.111ms
Operation Mode Setting	Stop Setting	If a moderate or major error occurs at any of the CPUs, set whether to stop or continue operation for all CPUs.	Major: All Station Stop, Moderate: All Station Continue. * Set for all CPUs.
	Synchronous Startup Setting	Set up this to synchronize startup time of CPU modules in multiple CPUs system. * Always select "Synchronize" due to the robot CPU takes as long as 18 seconds to startup.	Synchronize
Other PLC Control Module Setting	I/O Setting Outside Group	Set up this to import the input (X) ON/OFF data of the module outside the control and the output (Y) ON/OFF data of other PLC CPU.	Import

Note1) For information about multiple CPUs and fixed scan communication area, refer to the RCPU manual (MELSEC iQ-R CPU Module User's Manual (Application)).

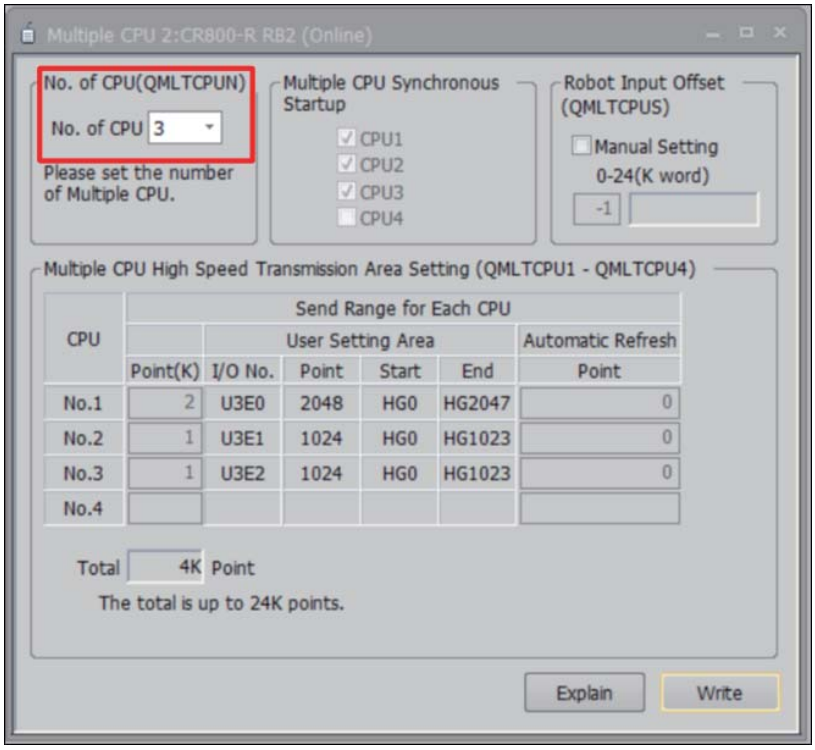
Note2) Because the area is set up in 1K unit, allocate 1K even in case of 0.5K.

Set up "Multiple CPU Setting" in the parameter menu of RT ToolBox3 as follows.
The settings of "Multiple CPU Synchronous Startup" and "Multiple CPU High Speed Transmission Area Setting" are not required since their setting values are automatically acquired from the PLC CPU.

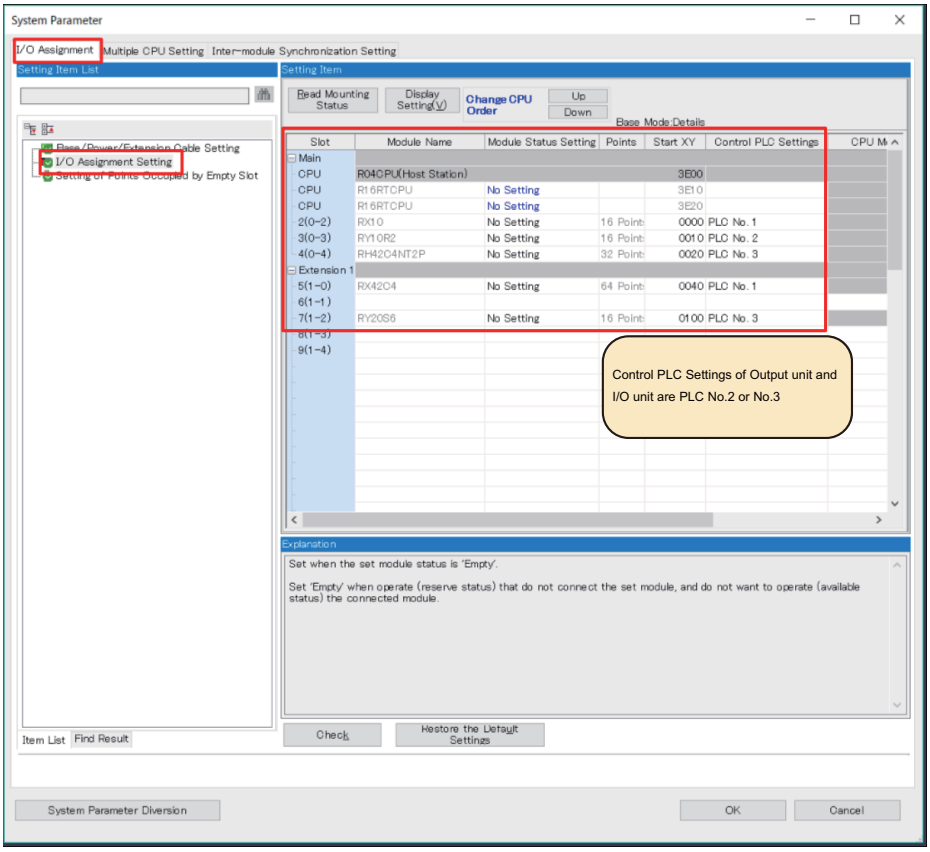
[Robot CPU No. 1 (= CPU No. 2)]



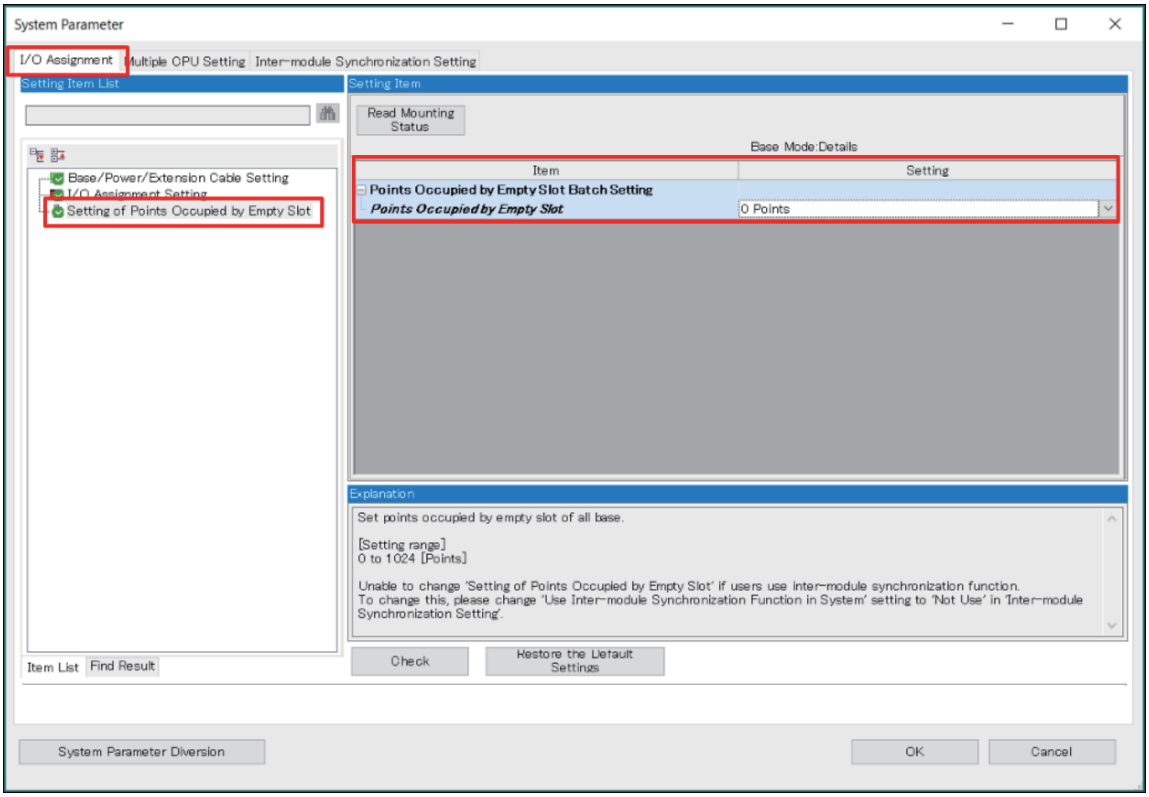
[Robot CPU No. 2 (= CPU No. 3)]



Set up "I/O Assignment" with the CPU parameters of GX Works3 as follows.



With this setting, the PLC controls the input module, and the robot CPU No. 1 and No. 2 control the output module and I/O combined module.
If there is any empty slot, set "Setting of Points Occupied by Empty Slot" as follows.



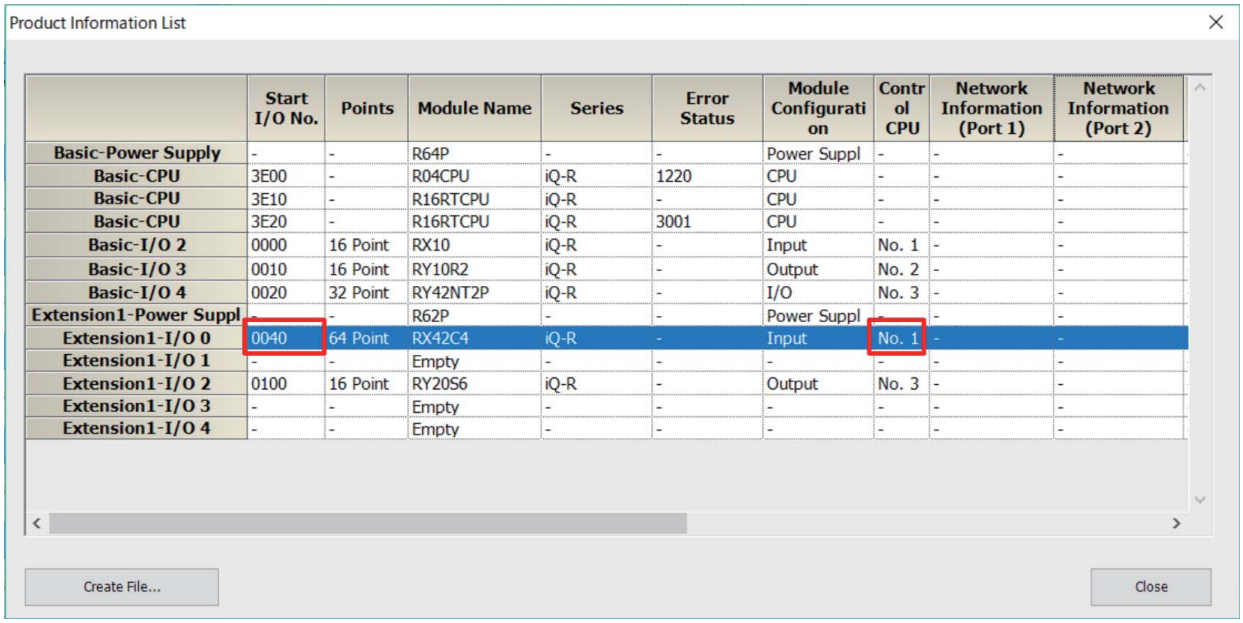
After setting the above parameters, reset the power of the PLC and robot CPU.

(5) Controlling with status variables

The following types of status variables are provided to control the PLC I/O module.

Variable name	Details
M_XDev	Reads the PLC input signal (X) per bit. ex.) 1 M1%=M_XDev (1) ' The value of the PLC input signal 1 (1 or 0) is substituted to M2.
M_XDevB	Reads the PLC input signal (X) per byte. ex.) 1 M2%=M_XDevB(&H10) ' The value of 8-bit width from 10 (hexadecimal number) of PLC input signals is substituted to M2.
M_XDevW	Reads the PLC input signal (X) per word. ex.) 1 M4%=M_XDevW(&H20) ' The value of 16-bit width from 20 (hexadecimal number) of PLC input signals is substituted to M4.
M_XDevD	Reads the PLC input signal (X) per double word. ex.) 1 M5&=M_XDevD(&H100) ' The value of 32-bit width from 100 (hexadecimal number) of PLC input signals is substituted to M5.
M_YDev	Reads/Writes the PLC output signal (Y) per bit. ex.) 1 M_YDev(2)=1 'Turns on the PLC output signal No. 2.
M_YDevB	Reads/Writes the PLC output signal (Y) per byte. ex.) 1 M_YDevB(&H10)=&HFF 'Turns on the 8-bit width from 10 (hexadecimal number) of PLC output signals.
M_YDevW	Reads/Writes the PLC output signal (Y) per word. ex.) 1 M_YDevW(&H20)=&HFFFF ' Turns on the 16-bit width from 20 (hexadecimal number) of PLC output signals.
M_YDevD	Reads/Writes the PLC output signal (Y) per double word. ex.) 1 M_YDevD(&H100)=P1.X * 1000 ' Outputs the multiplication result value of X coordinate value of the position variable P1 by 1000 to 32-bit width from 100 (hexadecimal number) of PLC output signals.

For example, the system configuration example 2 is set as follows.

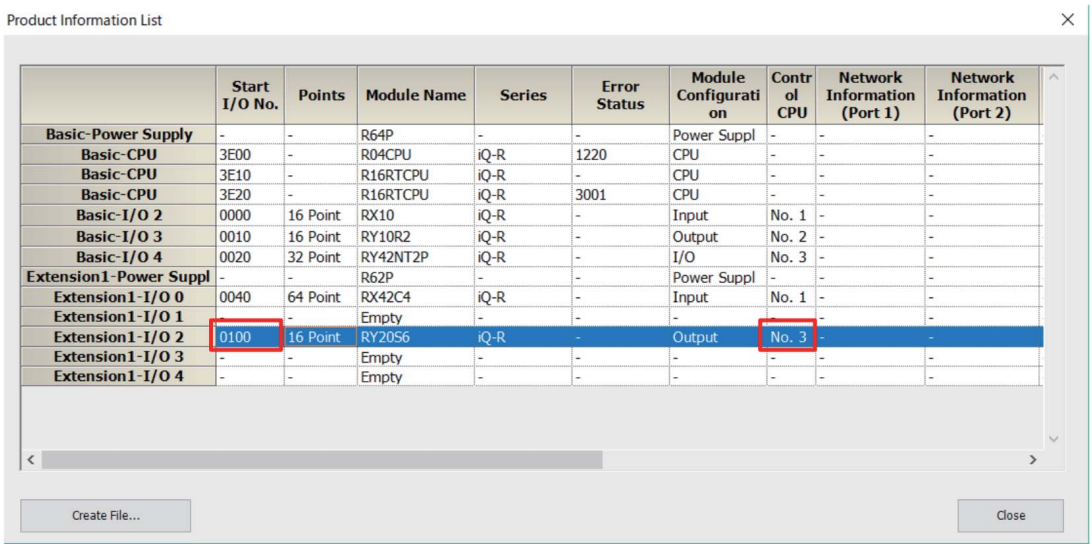


To check the input information of the input module mounted on the slot 0 of the extension base 1 that is controlled by the CPU No. 1 (= PLC), prepare the robot program in the CPU No. 2 (= robot CPU No. 1) as follows.

```
M1&=M_XDevD(&H40)
```

- Specify the "Start I/O No." of the corresponding module.
- Status variable that reads input signals for 32 bits

The following describes the settings when outputting a signal to the output module controlled by the user or the I/O module.



For example, to output a signal to the output module mounted on the slot 2 of the extension base 1 shown in the above, prepare a robot program in the CPU No. 3 (= robot CPU No. 2) as follows.

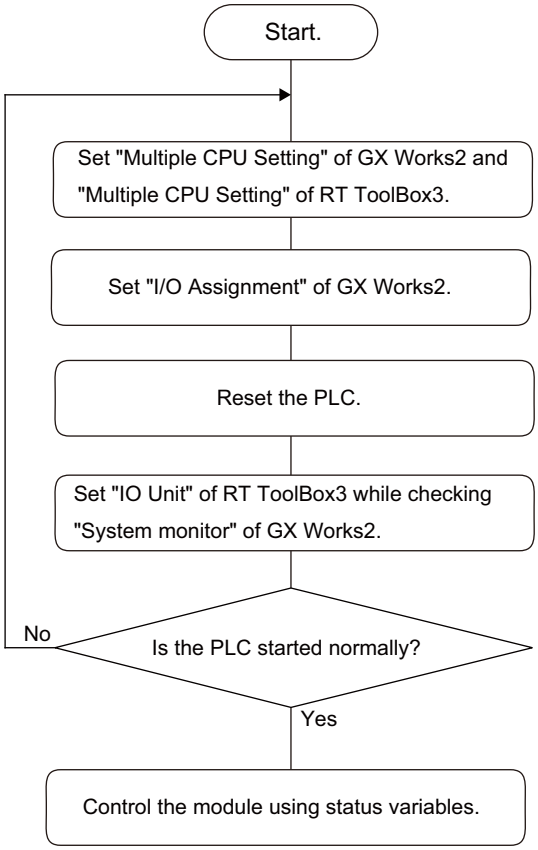
```
M_YDevD(&H100) = &H1234
```

- `&H1234` — Specify the data to be output.
- `&H100` — Specify the "Start I/O No." of the corresponding module.
- `M_YDevD` — Status variable that outputs data for 32 bits

5.25.2 CR800-Q series

(1) Operation procedure

The operation procedure for controlling the PLC I/O module is shown below.



•••●Set the parameters in "Multiple CPU Setting" of GX Works2.
This also can be set from GX Developer and GX Works3.
●Set the parameters in "Multiple CPU Setting " of RT ToolBox3.

•••●Set the parameters in "I/O Assignment" of GX Works2.

•••●Perform the reset operation of the PLC or turn off and on the power to reset the PLC.

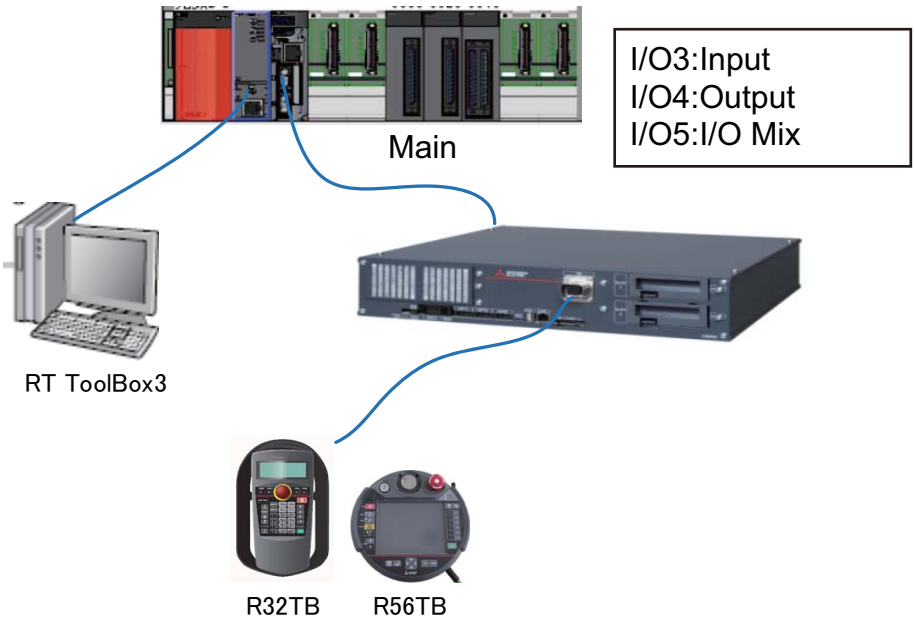
•••●Check "System monitor" of GX Works2.
●Set the parameters in "IO Unit" of RT ToolBox3.

Check the input information with the status variables "M_XDev", "M_XDevB", "M_XDevW", and "M_XDevD", and output a signal to the module controlled by the host CPU with the status variables "M_YDev", "M_YDevB", "M_YDevW", and "M_YDevD".
(Refer to "(6)Controlling with status variables".)

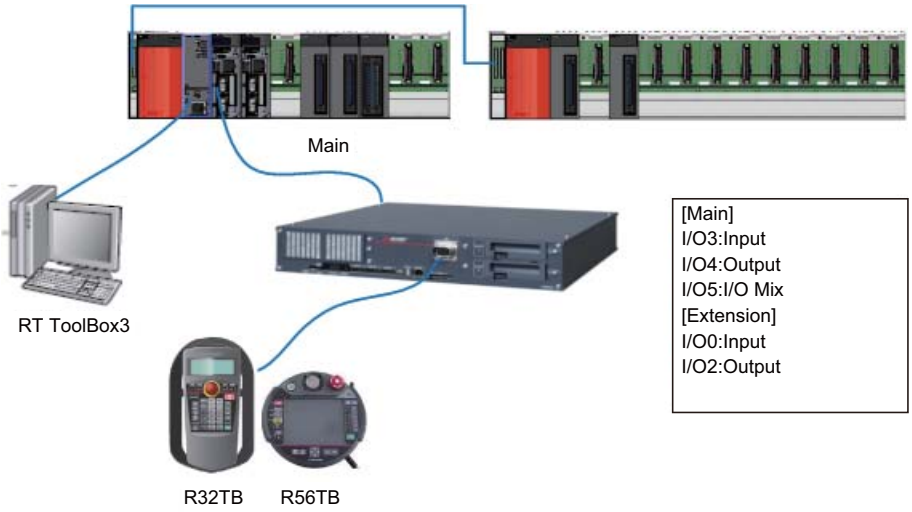
(2) Configuration example of the hardware

The following shows the system configurations "1 robot CPU + 3 types of I/O modules" as the example 1, and "2 robot CPUs + 3 types of I/O modules + extension base (added)" as the example 2.

*System configuration example 1: 1 robot CPU + 3 types of I/O modules



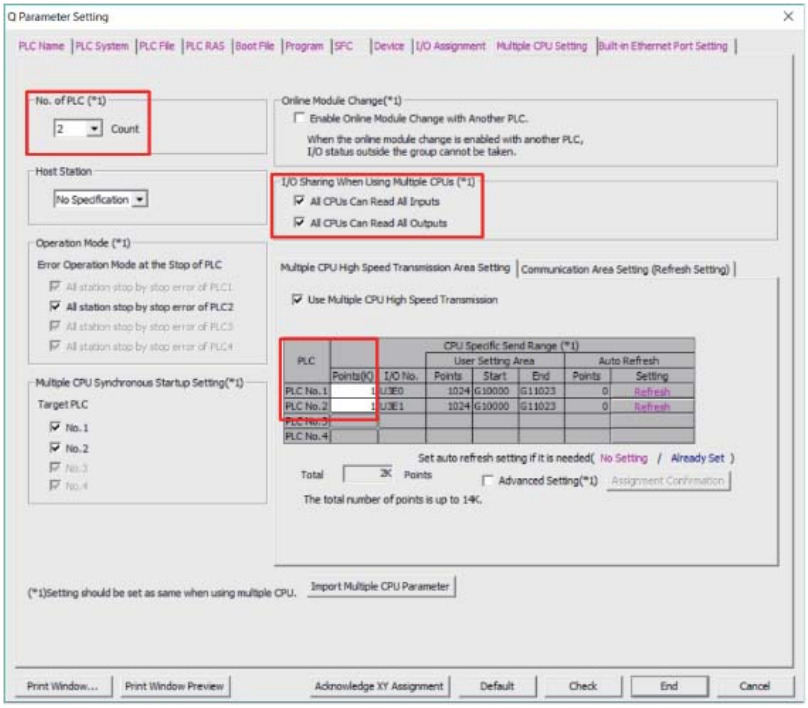
*System configuration example 2: 2 robot CPUs + 3 types of I/O modules + extension base (added)



(3) Setting the parameters for the system configuration example 1

Under the condition of the system configuration example 1 (1 robot CPU + 3 types of I/O modules), set "Multiple CPU Setting" in the PLC parameter of GX Works2 and "Multiple CPU Setting" of RT ToolBox3.

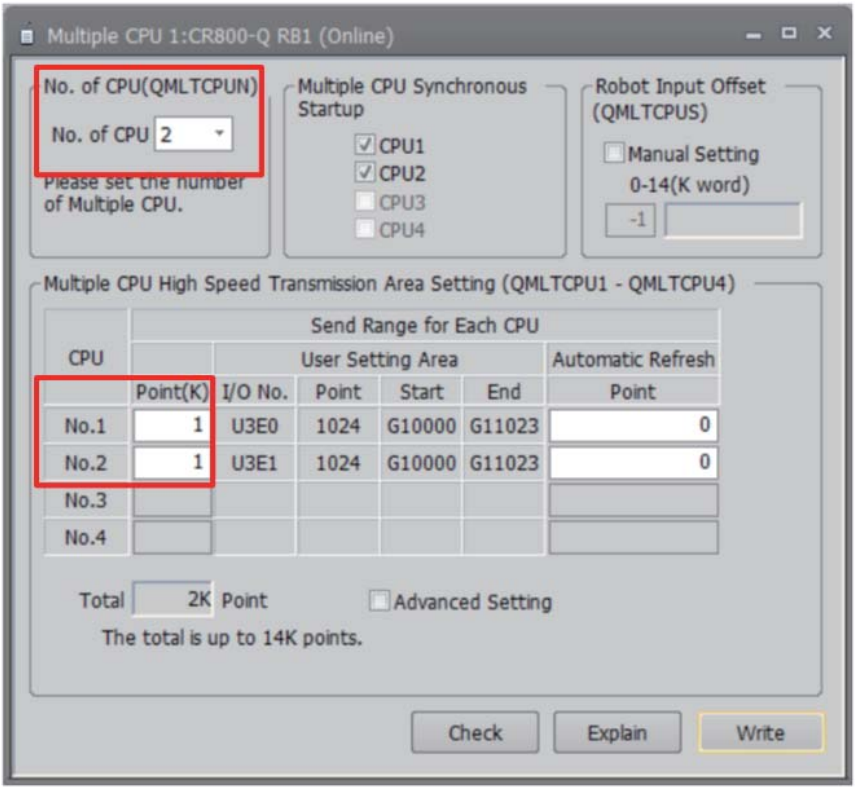
Set "Multiple CPU Setting" with the PLC parameter of GX Works2 as follows.



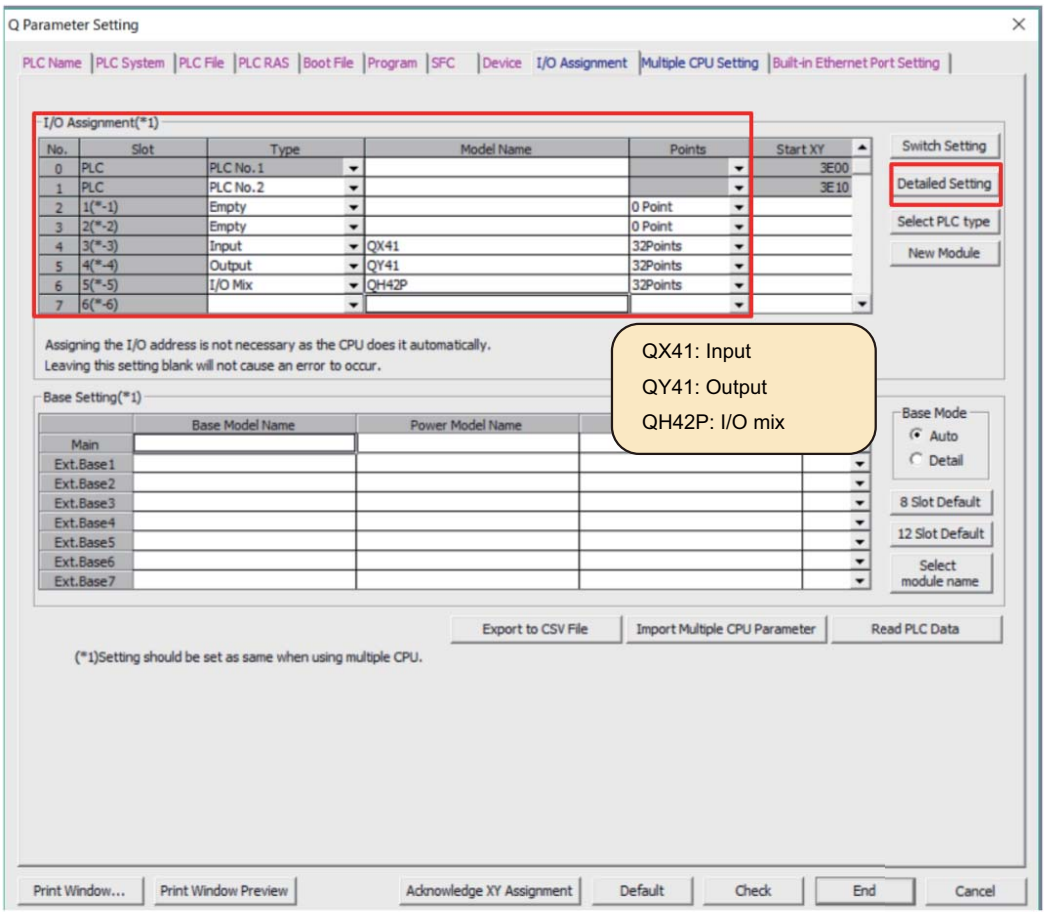
Setting Item	Description	Setting Value
No. of PLC	Set the number of CPU modules to be used in the multiple CPU system.	2
I/O Sharing When Using Multiple CPUs	Set this item to import the input (X) ON/OFF data of the module outside the control and the output (Y) ON/OFF data of other PLC CPU.	Put a check mark.
Multiple CPU High Speed Transmission Area Setting	Set this item when transferring data using the multiple CPU high speed transmission area[Note1)]	• PLC No. 1 (= PLC): Sum of the size (1K) of the data to be sent to the robot and the size of the data to be sent to other devices • PLC No. 1 (= Robot No. 1): Set 1K for it.

Note1) For information about multiple CPUs and multiple CPU high speed transmission area, refer to the QCPU manual (QCPU User's Manual (Multiple CPU System)).

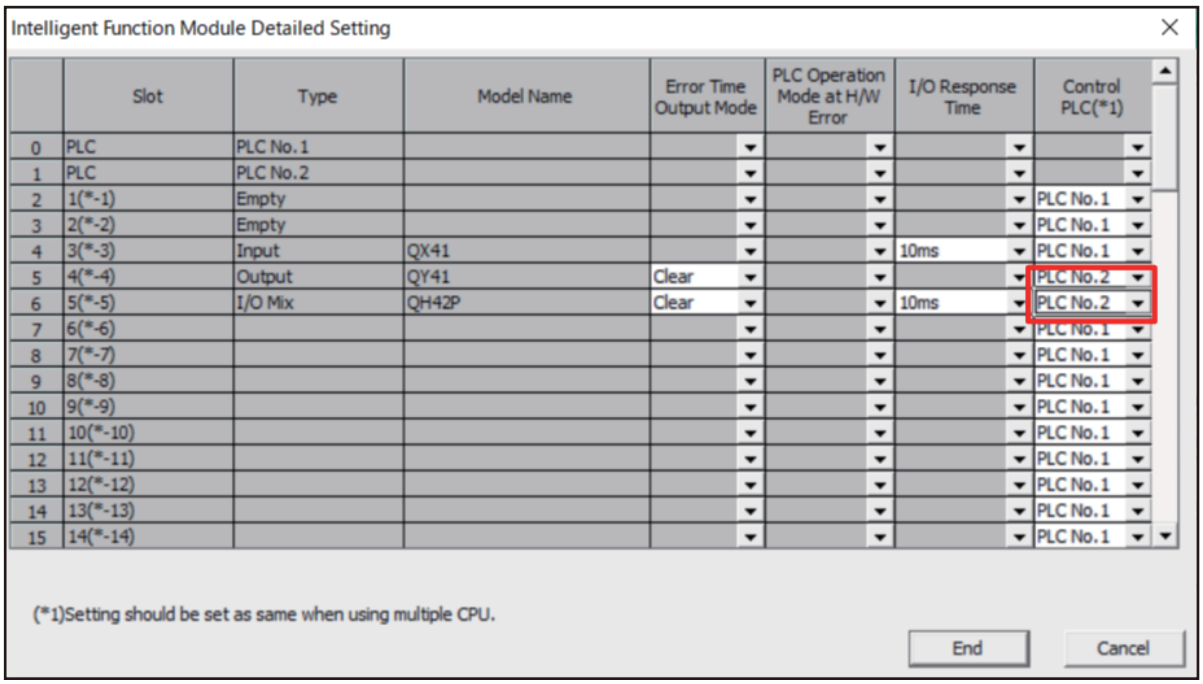
In the same way, set "Multiple CPU Setting" in the parameter menu of RT ToolBox3 as follows.



Set "I/O Assignment" with the PLC parameter of GX Works2 as follows.

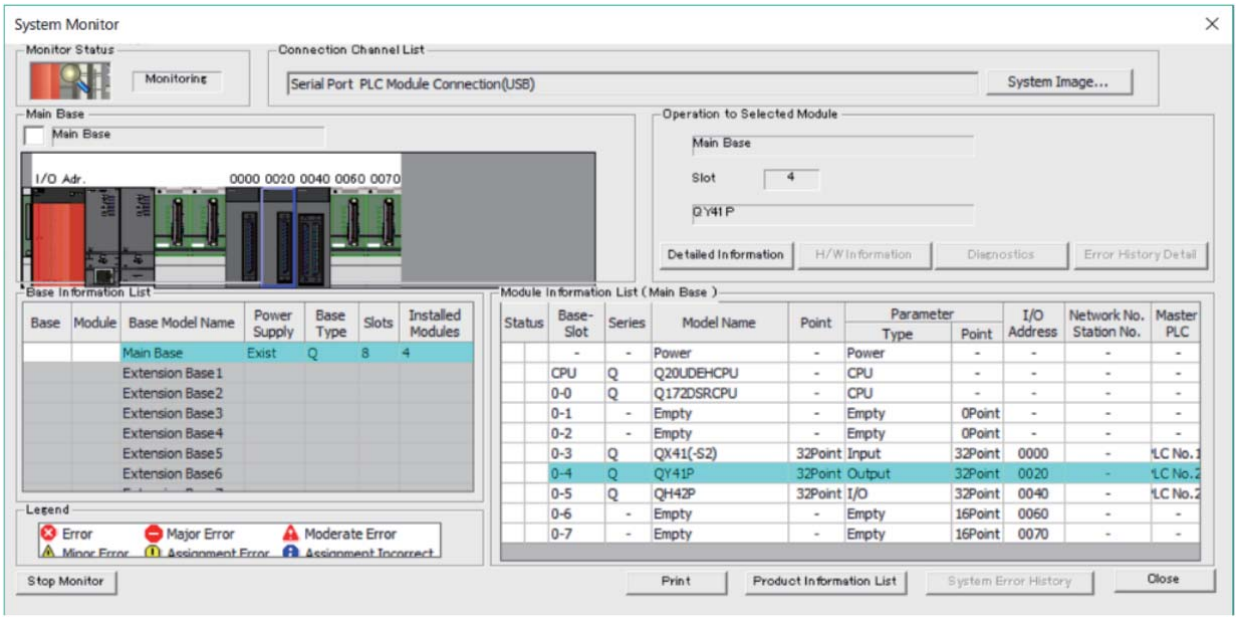


Click the "Detailed Setting" button and set the control CPU as follows.

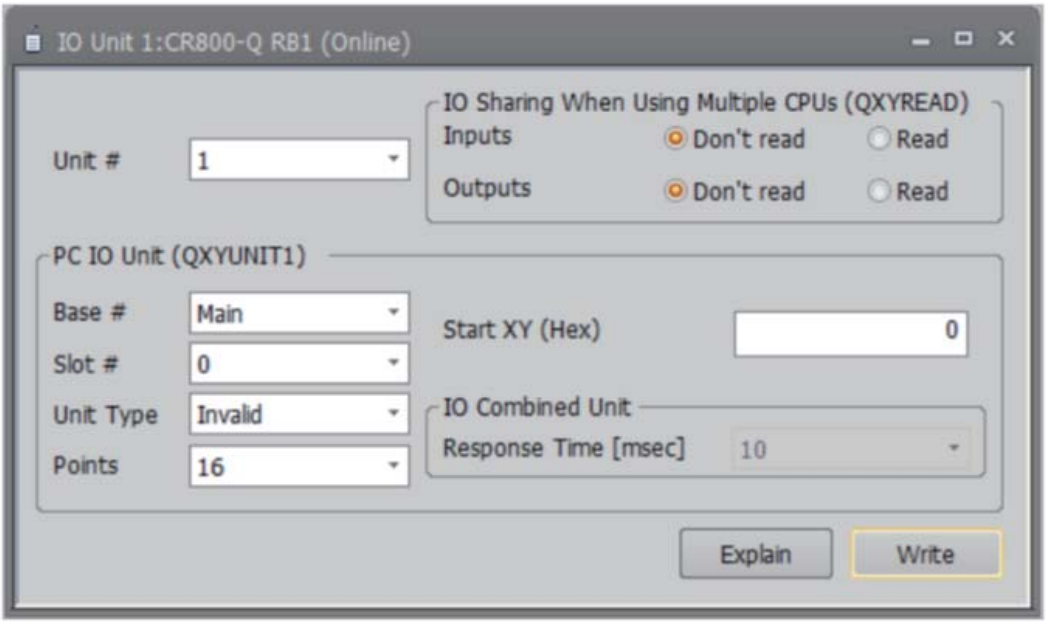


With this setting, the PLC controls the input module, and the robot CPU No. 1 controls the output module and I/O combined module.

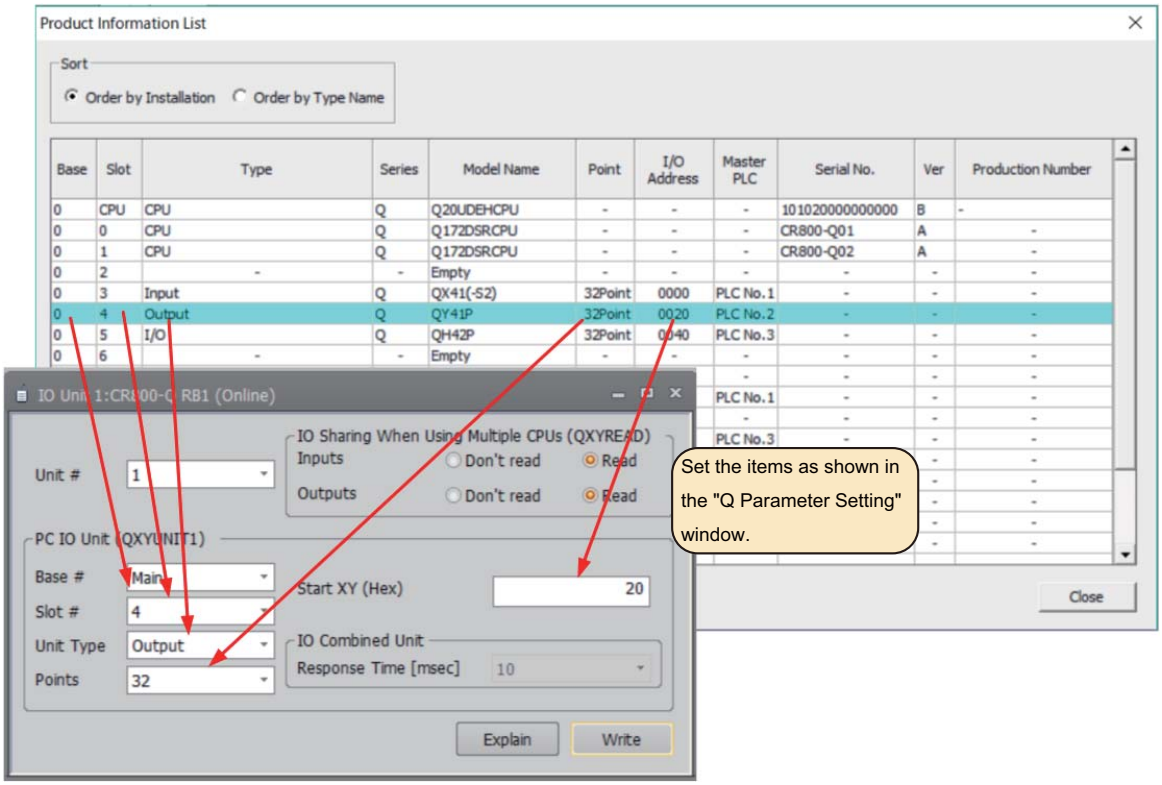
Reset the power of the PLC before setting the parameters of RT ToolBox3.
When the PLC is started, select [System Monitor] from the [Diagnostics] menu of GX Works2.



Set the parameters of RT ToolBox3 for the I/O module controlled by the robot CPU.
Double-click "IO unit" in the "Parameter" menu to open the "IO Unit" parameter window.

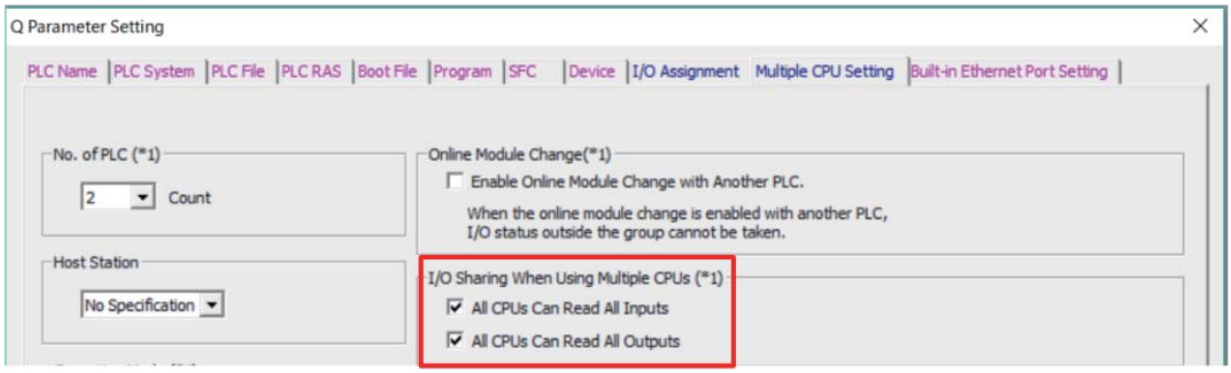


On the "System Monitor" window of GX Works2, click the corresponding module, and the [Product Information List] button, and then set the "IO Unit" parameters of RT ToolBox3 while checking the displayed "Product Information List" window.

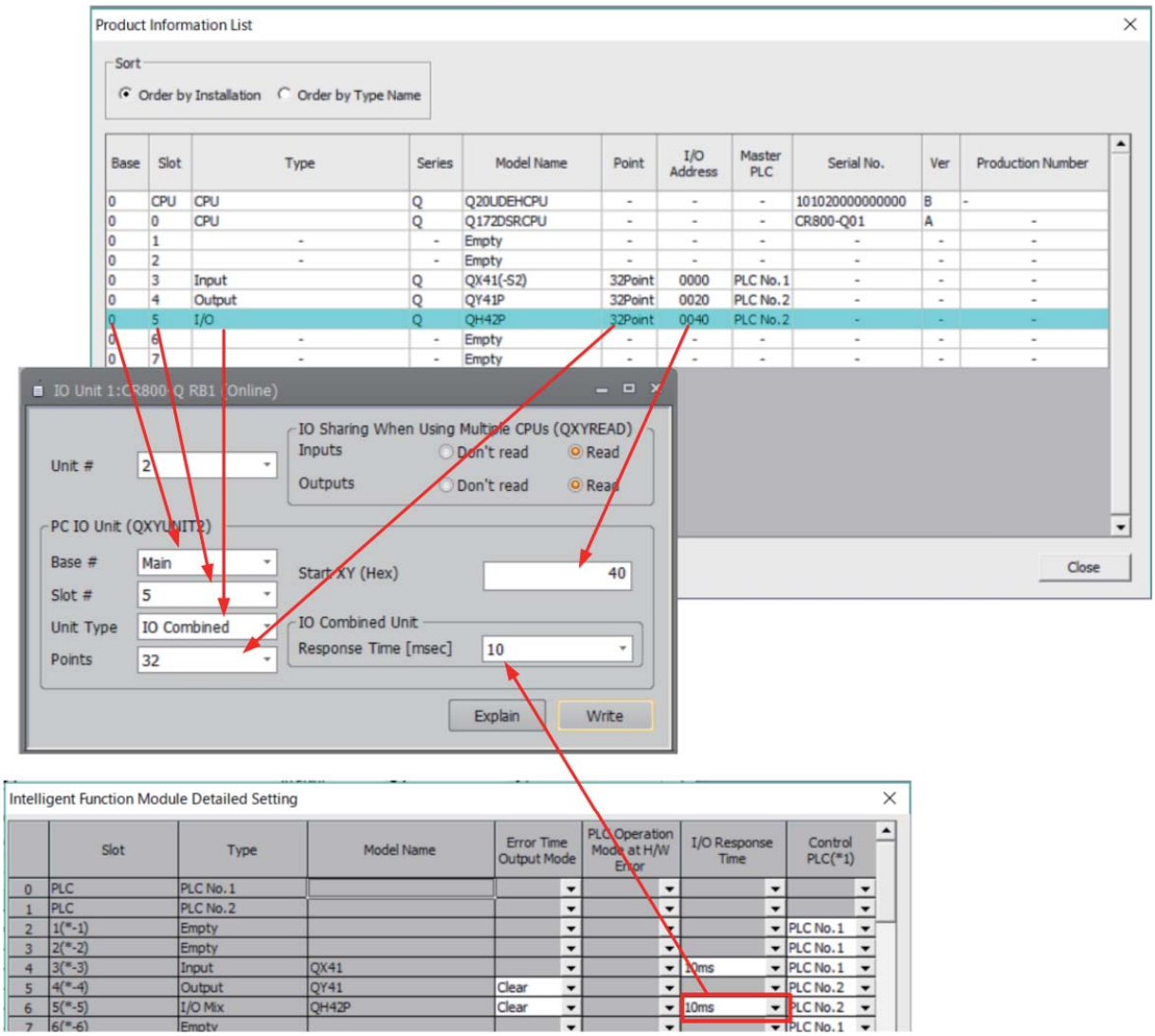


A single robot CPU can control up to four I/O modules.
"1" to "4" can be set for [Unit #]. It indicates the setting order of the module among four modules.

Set "IO Sharing When Using Multiple CPUs (QXYREAD)" according to the "Q Parameter Setting" window of GX Works2 shown below.



Set the second module.
On the "System Monitor" window of GX Works2, click the corresponding module, and the [Product Information List] button, and then set the "IO Unit" parameters of RT ToolBox3 while checking the displayed "Product Information List" window.



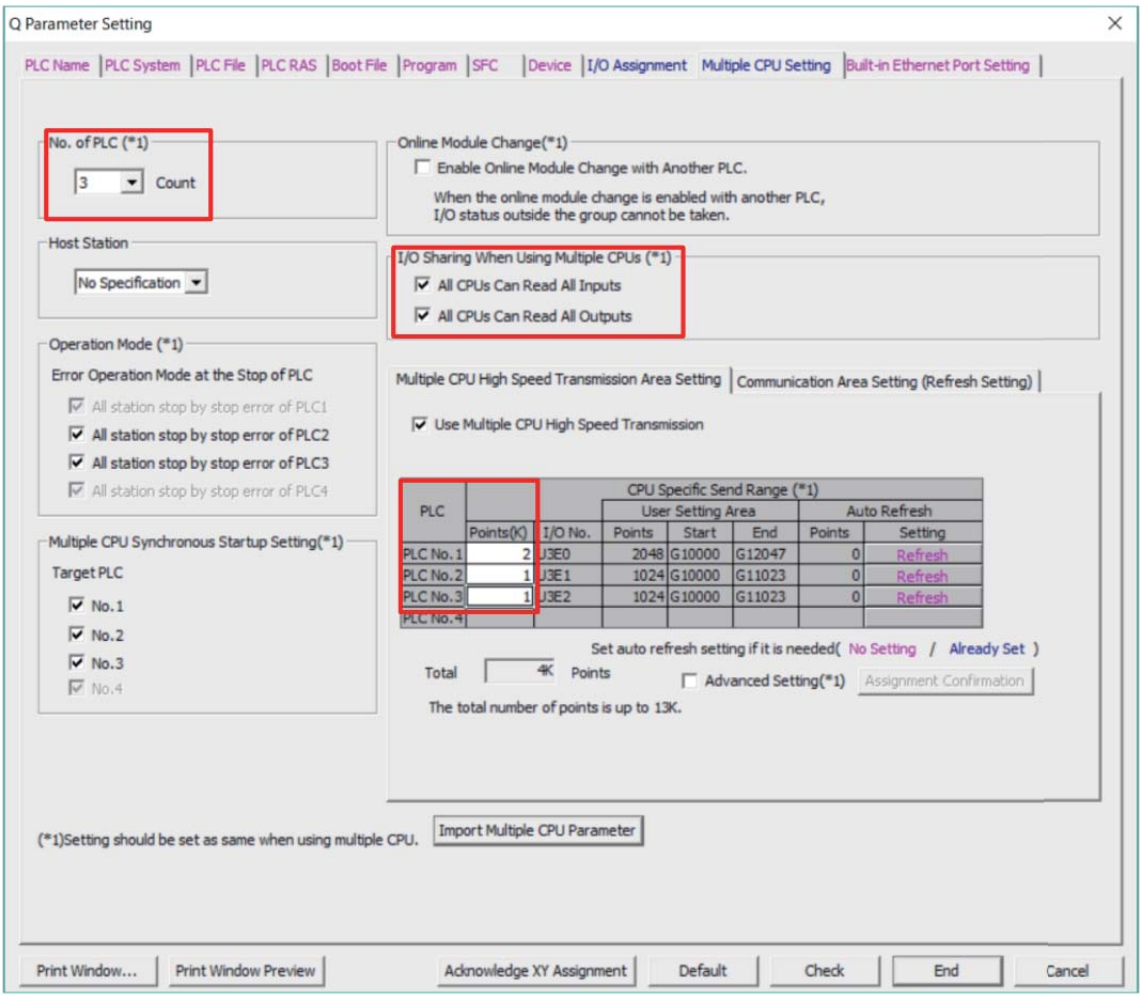
Set "Response Time" according to the value on the window that is displayed by clicking the "Detailed Setting" button in "Q Parameter Setting".

After setting the above parameters, reset the power of the PLC and robot CPU.

(4) Setting the parameters for the system configuration example 2

Under the condition of the system configuration example 2 (2 robot CPUs + 3 types of I/O modules + extension base), set up "Multiple CPU Setting" in PLC parameter of GX Works2 and "Multiple CPU Setting" of RT ToolBox3.

Set "Multiple CPU Setting" with the PC parameter of GX Works2 as follows.

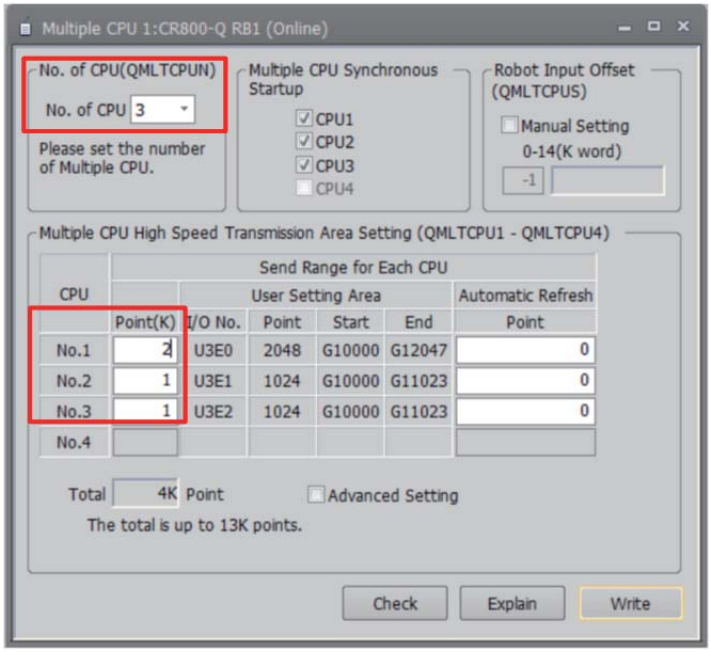


Setting Item	Description	Setting Value
No. of PLC	Set the number of CPU modules to be used in the multiple CPU system.	3
I/O Sharing When Using Multiple CPUs	Set this item to import the input (X) ON/OFF data of the module outside the control and the output (Y) ON/OFF data of other PLC CPU.	Put a check mark.
Multiple CPU High Speed Transmission Area Setting	Set this item when transferring data using the multiple CPU high speed transmission area[Note1)]	• PLC No. 1 (= PLC): Sum of the size (1K) of the data to be sent to the robot and the size of the data to be sent to other devices • PLC No. 1 (= Robot No. 1): Set 1K for it. • PLC No. 3 (= Robot No. 2): Set 1K for it.

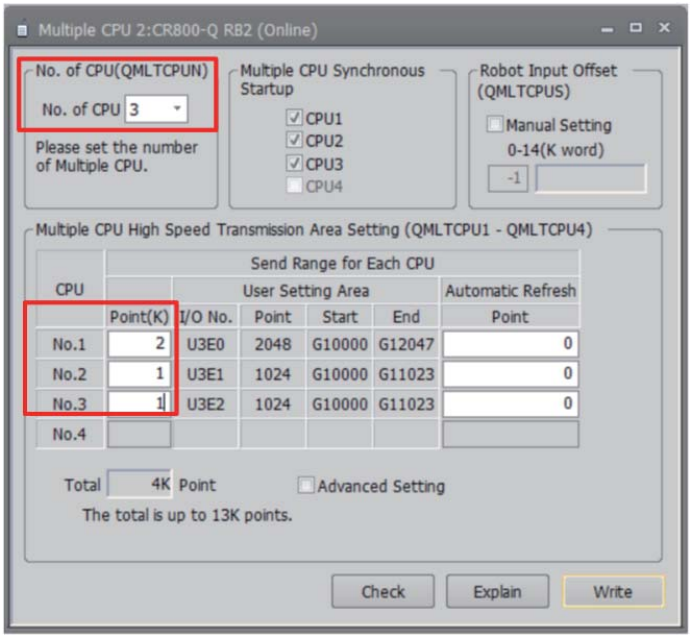
Note1) For information about multiple CPUs and multiple CPU high speed transmission area, refer to the QCPU manual (QCPU User's Manual (Multiple CPU System)).

In the same way, set "Multiple CPU Setting" in the parameter menu of the robot CPU No. 1 and No. 2 as follows with RT ToolBox3.

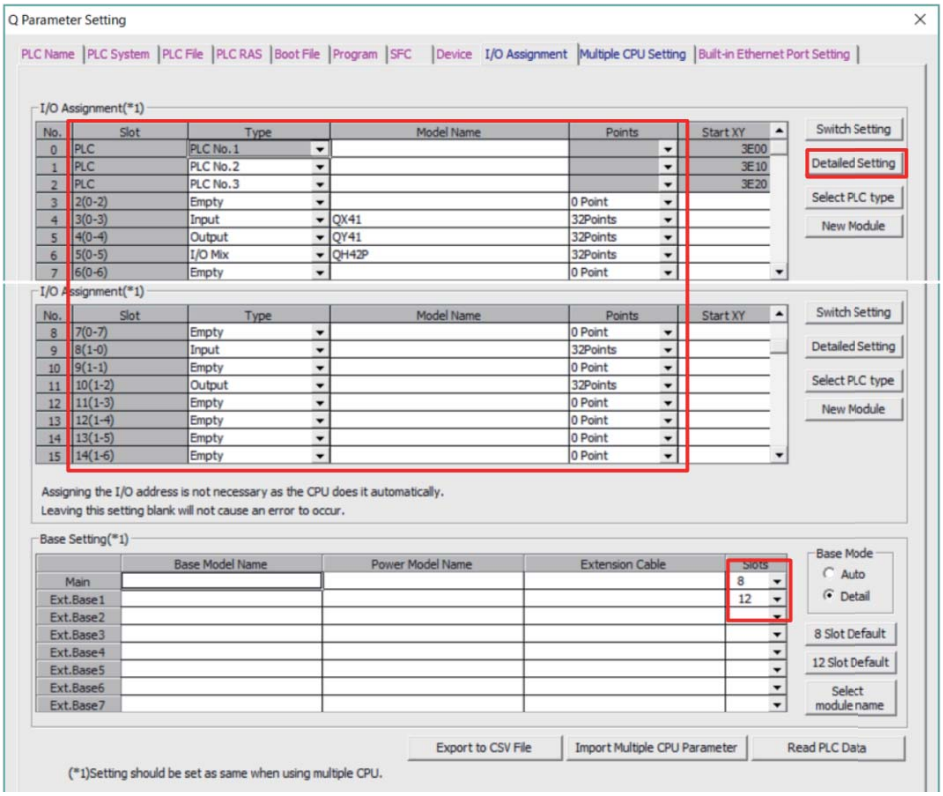
[Robot CPU No. 1 (= PLC No. 2)]



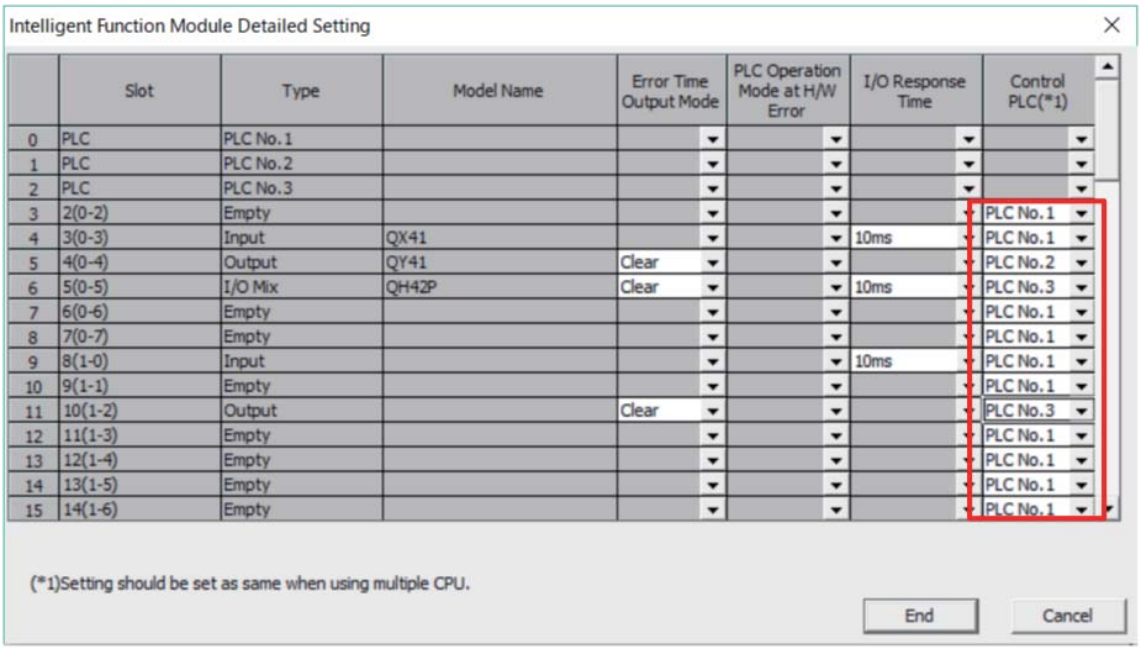
[Robot CPU No. 2 (= PLC No. 3)]



Set up "I/O Assignment" with the PC parameter of GX Works2 as follows.

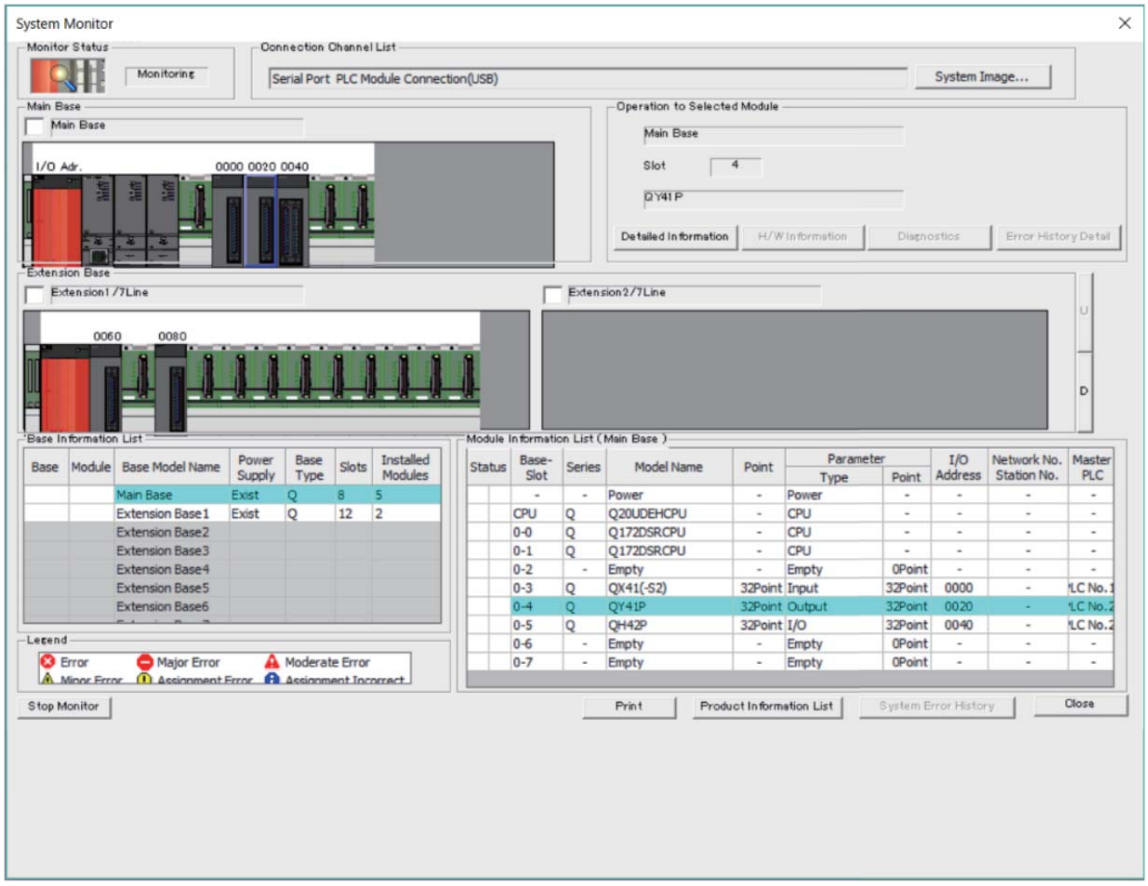


Click the "Detailed Setting" button and set the control CPU as follows.

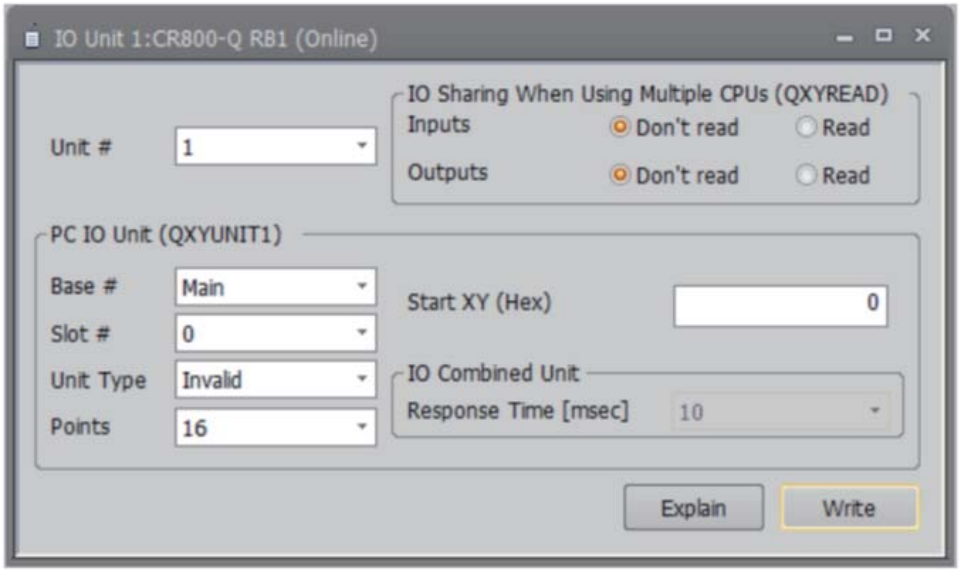


With this setting, the PLC controls the input module, and the robot CPU No. 1 and No. 2 control the output module and I/O combined module.

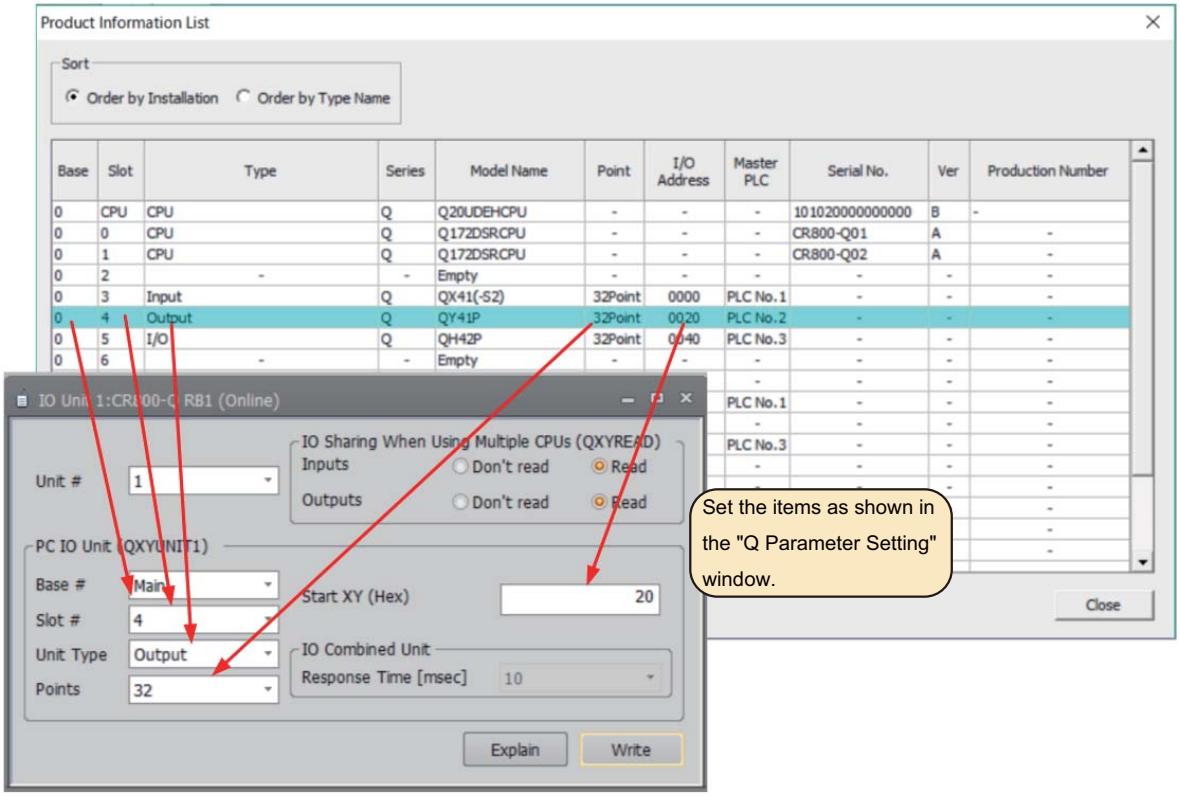
Reset the power of the PLC before setting the parameters of RT ToolBox3.
When the PLC is started, select [System Monitor] from the [Diagnostics] menu of GX Works2.



Set the parameters of RT ToolBox3 for the I/O module controlled by the robot CPU.
For both the robot CPU No. 1 and No. 2, double-click "IO unit" in the "Parameter" menu to open the "IO Unit" parameter window.

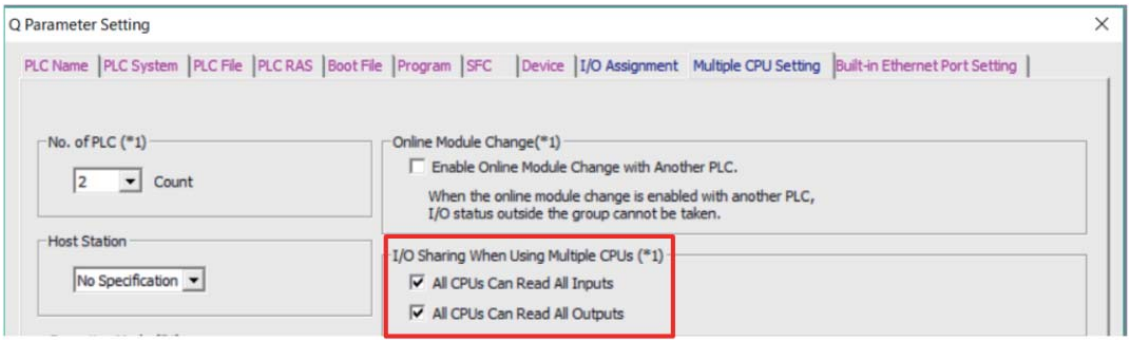


First, set the robot CPU No. 1.
On the "System Monitor" window of GX Works2, click the corresponding module, and the [Product Information List] button, and then set the "IO Unit" parameters of RT ToolBox3 while checking the displayed "Product Information List" window.

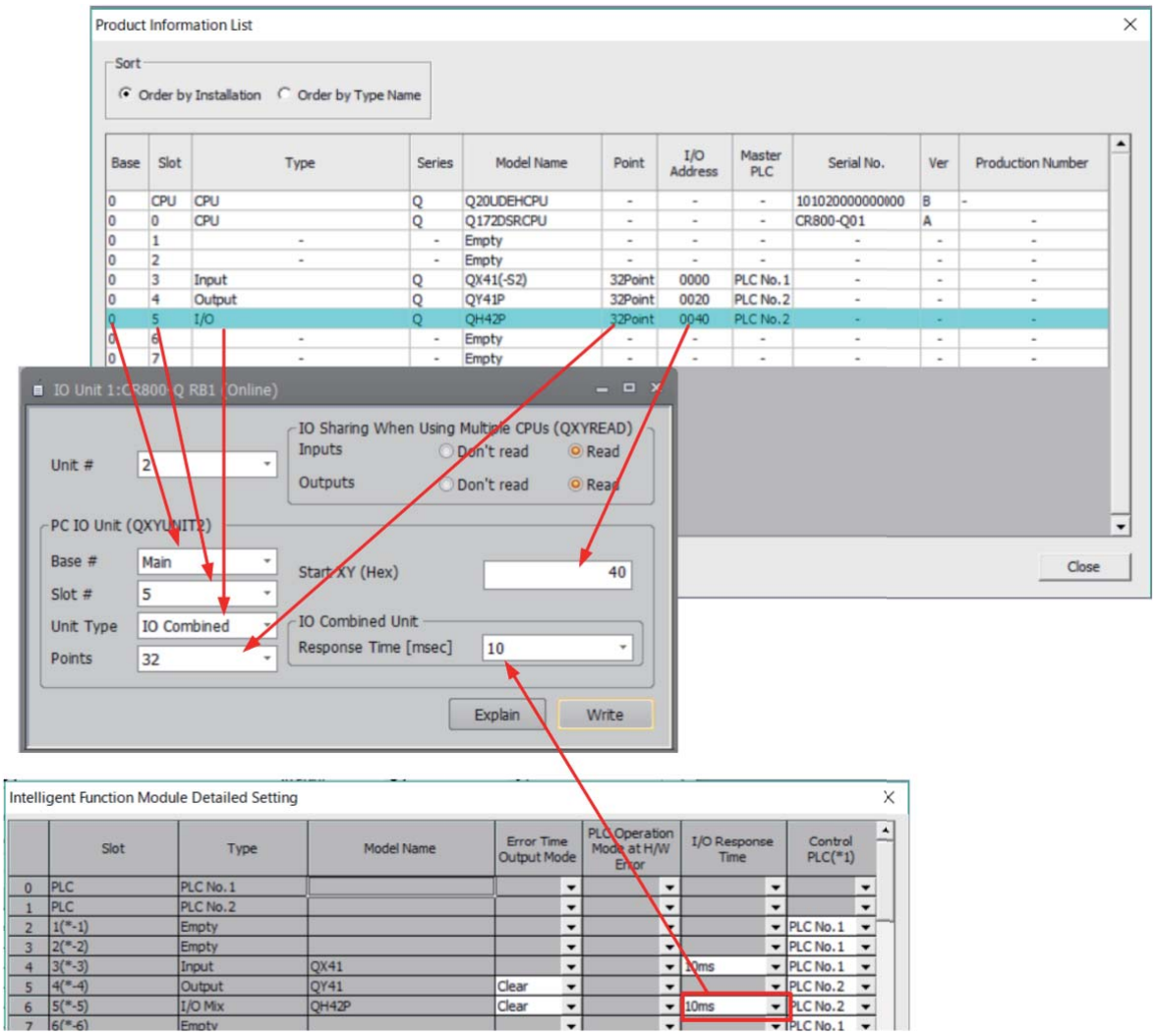


A single robot CPU can control up to four I/O modules.
"1" to "4" can be set for [Unit #]. It indicates the setting order of the module among four modules.

Set "IO Sharing When Using Multiple CPUs (QXYREAD)" according to the "Q Parameter Setting" window of GX Works2 shown below.

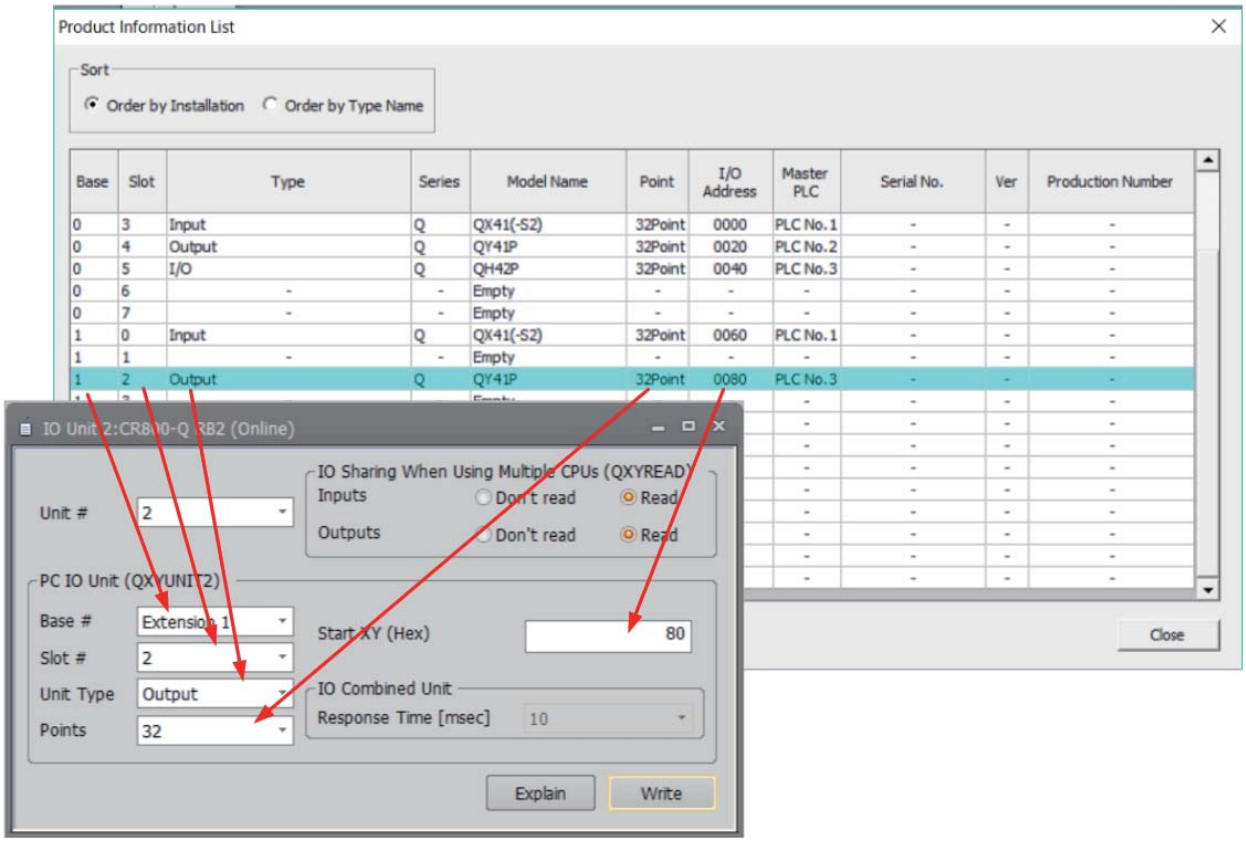


Next, set the robot CPU No. 2.
On the "System Monitor" window of GX Works2, click the corresponding module, and the [Product Information List] button, and then set the "IO Unit" parameters of RT ToolBox3 while checking the displayed "Product Information List" window.



Set "Response Time" according to the value on the window that is displayed by clicking the "Detailed Setting" button in "Q Parameter Setting".

Set the second module.
On the "System Monitor" window of GX Works2, click the corresponding module, and the [Product Information List] button, and then set the "IO Unit" parameters of RT ToolBox3 while checking the displayed "Product Information List" window.



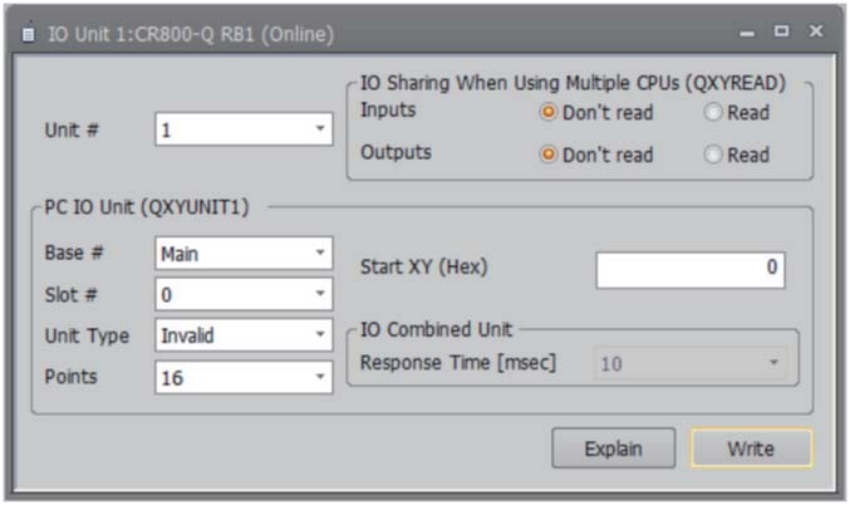
After setting the above parameters, reset the power of the PLC and robot CPU.

(5) Details of robot parameters

The following describes the details of the parameters set with RT ToolBox3.

Parameter	Parameter name	No. of arrays No. of characters	Details explanation	Factory setting
Setup of the reference to PLC input output signal (* CR800-Q only)	QXYREAD	Integer 2	Set up to import the input and output status of the I/O module (including the I/O combined module) that is controlled by other CPU. [Element 1] Reference of input signal (X) [0: Invalid / 1:valid] [Element 2] Reference of output signal (Y) [0: Invalid / 1:valid]	0,0
Setup of the PLC input output unit (* CR800-Q only)	QXYUNITn n: 1 to 4	Integer 7	Specify the input output unit/input output mixing unit which robot CPU's manage.(Invalid/Valid=0/1) [Element 1] Unit type 0: With no target unit 1: Not use (with no meaning) 2: Output unit 3: Input output mixing unit [Element 2] Top input output number 0-4080 (decimal number) [Element 3] Base number 0: Basic base unit 1-7: Extension base unit [Element 4] Slot number 0-11 (decimal number) [Element 5] Width of input output points 0: 16 points/ 1: 32 points/ 2: 48 points/ 3: 64 points/ 4: 128 points/ 5: 256 points/ 6: 512 points/ 7: 1024 points [Element 6] Output mode at error 0: Clear 1: Holding [Element 7] Response time 0: 10ms/ 4: 1.0ms/ 5: 5.0ms/ 6: 20ms/ 7: 70ms [Element 8] Unused	0,0,0,0,0,0,0,0

The above parameters are set with the following window of RT ToolBox3.

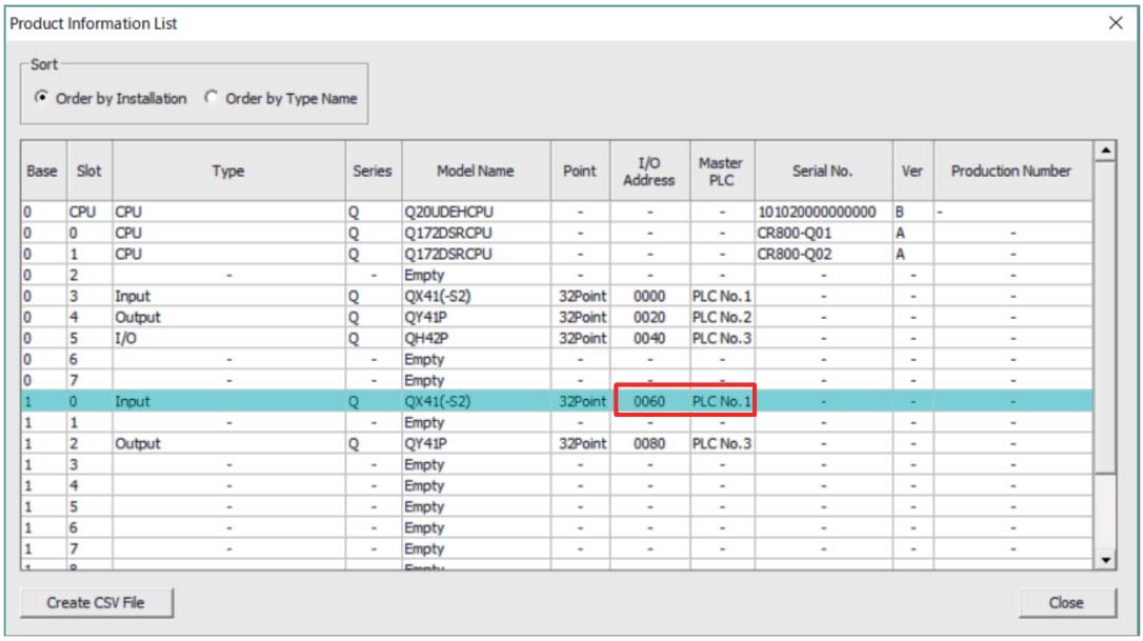


(6) Controlling with status variables

The following types of status variables are provided to control the PLC I/O module.

Variable name	Details
M_XDev	Reads the PLC input signal (X) per bit. ex.) 1 M1%=M_XDev (1) ' The value of the PLC input signal 1 (1 or 0) is substituted to M2.
M_XDevB	Reads the PLC input signal (X) per byte. ex.) 1 M2%=M_XDevB(&H10) ' The value of 8-bit width from 10 (hexadecimal number) of PLC input signals is substituted to M2.
M_XDevW	Reads the PLC input signal (X) per word. ex.) 1 M4%=M_XDevW(&H20) ' The value of 16-bit width from 20 (hexadecimal number) of PLC input signals is substituted to M4.
M_XDevD	Reads the PLC input signal (X) per double word. ex.) 1 M5&=M_XDevD(&H100) ' The value of 32-bit width from 100 (hexadecimal number) of PLC input signals is substituted to M5.
M_YDev	Reads/Writes the PLC output signal (Y) per bit. ex.) 1 M_YDev(2)=1 'Turns on the PLC output signal No. 2.
M_YDevB	Reads/Writes the PLC output signal (Y) per byte. ex.) 1 M_YDevB(&H10)=&HFF 'Turns on the 8-bit width from 10 (hexadecimal number) of PLC output signals.
M_YDevW	Reads/Writes the PLC output signal (Y) per word. ex.) 1 M_YDevW(&H20)=&HFFFF ' Turns on the 16-bit width from 20 (hexadecimal number) of PLC output signals.
M_YDevD	Reads/Writes the PLC output signal (Y) per double word. ex.) 1 M_YDevD(&H100)=P1.X * 1000 ' Outputs the multiplication result value of X coordinate value of the position variable P1 by 1000 to 32-bit width from 100 (hexadecimal number) of PLC output signals.

For example, the system configuration example 2 is set as follows.



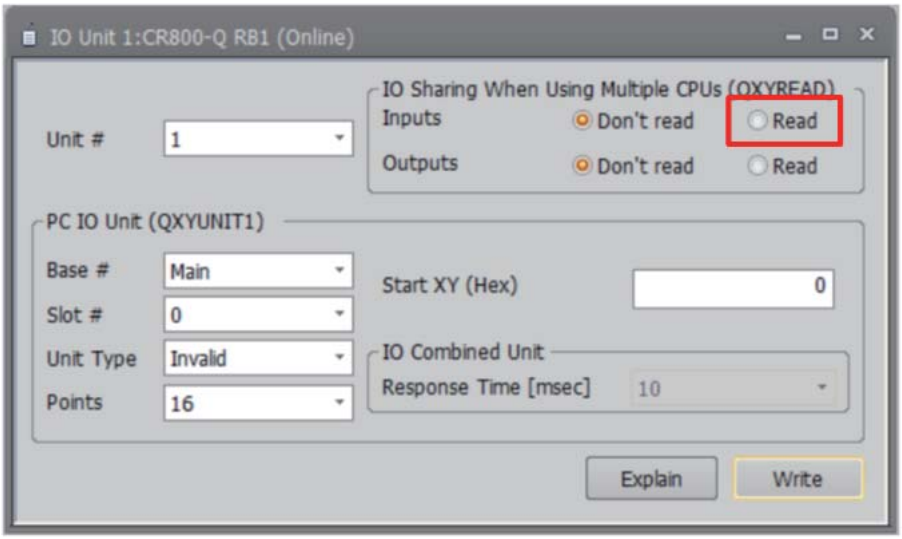
Base	Slot	Type	Series	Model Name	Point	I/O Address	Master PLC	Serial No.	Ver	Production Number
0	CPU	CPU	Q	Q20UDEHCPU	-	-	-	101020000000000	B	-
0	0	CPU	Q	Q172DSRCPU	-	-	-	CR800-Q01	A	-
0	1	CPU	Q	Q172DSRCPU	-	-	-	CR800-Q02	A	-
0	2	-	-	Empty	-	-	-	-	-	-
0	3	Input	Q	QX41(-S2)	32Point	0000	PLC No.1	-	-	-
0	4	Output	Q	QY41P	32Point	0020	PLC No.2	-	-	-
0	5	I/O	Q	QH42P	32Point	0040	PLC No.3	-	-	-
0	6	-	-	Empty	-	-	-	-	-	-
0	7	-	-	Empty	-	-	-	-	-	-
1	0	Input	Q	QX41(-S2)	32Point	0060	PLC No.1	-	-	-
1	1	-	-	Empty	-	-	-	-	-	-
1	2	Output	Q	QY41P	32Point	0080	PLC No.3	-	-	-
1	3	-	-	Empty	-	-	-	-	-	-
1	4	-	-	Empty	-	-	-	-	-	-
1	5	-	-	Empty	-	-	-	-	-	-
1	6	-	-	Empty	-	-	-	-	-	-
1	7	-	-	Empty	-	-	-	-	-	-

To check the input information of the input module mounted on the slot 0 of the extension base 1 that is controlled by the PLC No. 1 (= PLC), prepare the robot program in the PLC No. 2 (= robot CPU No. 1) as follows.

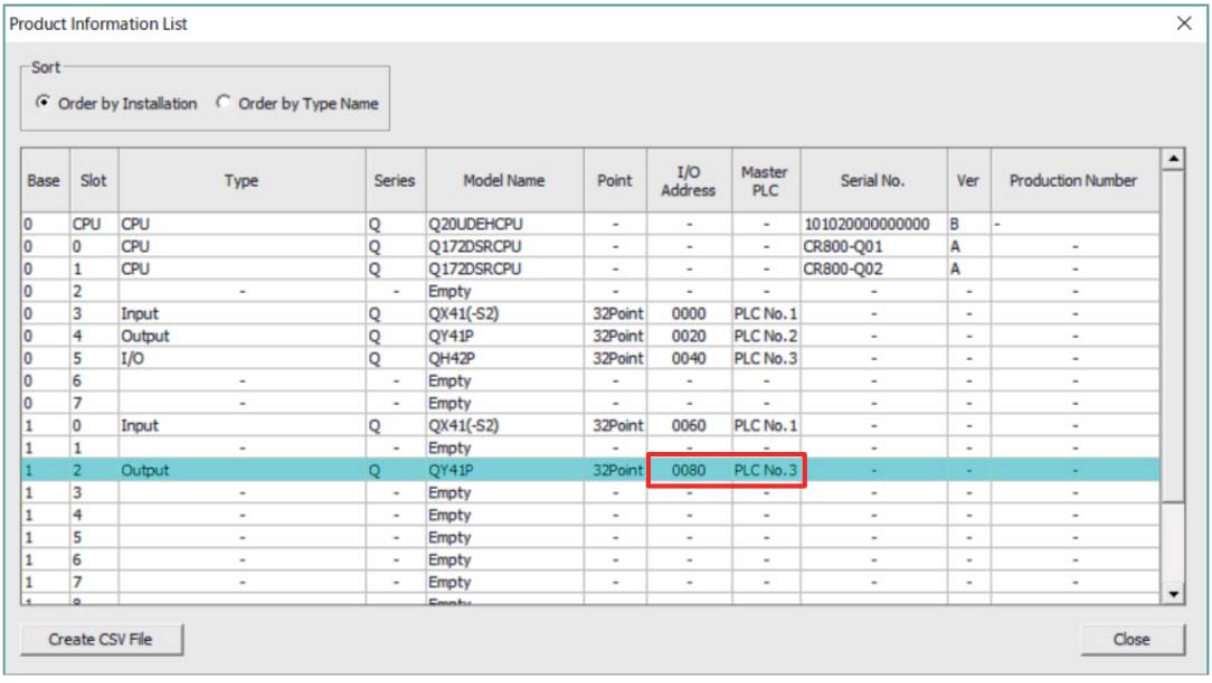
```
M1&=M_XDevD(&H60)
```

- `M_XDevD`: Status variable that reads input signals for 32 bits
- `&H60`: Specify the "Start I/O No." of the corresponding module.

The information of the input module controlled by the PLC can be checked with the CPU No. 3 (= robot CPU No. 2) using the same program.
However, the first element of the parameter "QXYREAD" must be set to "1".
Set the following window for RT ToolBox3.



The following describes the settings when outputting a signal to the output module controlled by the user or the I/O module.



Base	Slot	Type	Series	Model Name	Point	I/O Address	Master PLC	Serial No.	Ver	Production Number
0	CPU	CPU	Q	Q20UDEHCPU	-	-	-	101020000000000	B	-
0	0	CPU	Q	Q172DSRCPU	-	-	-	CR800-Q01	A	-
0	1	CPU	Q	Q172DSRCPU	-	-	-	CR800-Q02	A	-
0	2	-	-	Empty	-	-	-	-	-	-
0	3	Input	Q	QX41(-S2)	32Point	0000	PLC No.1	-	-	-
0	4	Output	Q	QY41P	32Point	0020	PLC No.2	-	-	-
0	5	I/O	Q	QH42P	32Point	0040	PLC No.3	-	-	-
0	6	-	-	Empty	-	-	-	-	-	-
0	7	-	-	Empty	-	-	-	-	-	-
1	0	Input	Q	QX41(-S2)	32Point	0060	PLC No.1	-	-	-
1	1	-	-	Empty	-	-	-	-	-	-
1	2	Output	Q	QY41P	32Point	0080	PLC No.3	-	-	-
1	3	-	-	Empty	-	-	-	-	-	-
1	4	-	-	Empty	-	-	-	-	-	-
1	5	-	-	Empty	-	-	-	-	-	-
1	6	-	-	Empty	-	-	-	-	-	-
1	7	-	-	Empty	-	-	-	-	-	-

For example, to output a signal to the output module mounted on the slot 2 of the extension base 1 shown in the above, prepare a robot program in the PLC No. 3 (= robot CPU No. 2) as follows.

```
M_YDevD(&H80) = &H1234
```

- `&H1234` — Specify the data to be output.
- `&H80` — Specify the "I/O Address" number of the corresponding module.
- `M_YDevD` — Status variable that outputs data for 32 bits

5.26 Direct communication with robot CPUs

This function is to exchange the signals directly with two or more robot CPUs in the robot controller of CR800-R/Q series.
Since the rudder program of the PLC is not needed, the exchange of the signal can be executed more speedily. And, the reference of CPU buffer memory information other than robot CPUs, such as motion CPU, is also possible.

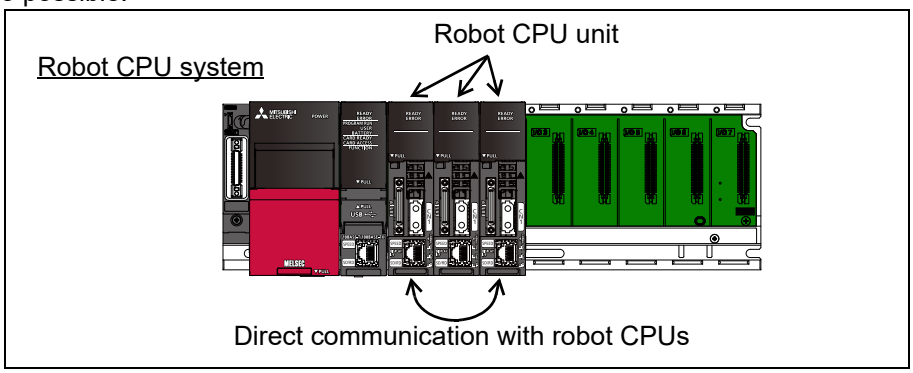


Fig.5-26:Direct communication with robot CPUs

(1) Specification

Specification is shown in Table 5-43.

Table 5-43:Specification

No	Item	Specification
1	Available device range	Range that can be specified by the CPU buffer memory access device.
2	Assignment of the dedicated input output signal	The control of the robot by dedicated signal is executed by the PLC (No. 1). The dedicated input output parameter cannot be assigned to the CPU buffer memory access device No. 2 or later. Always assign the dedicated input output signal to the CPU buffer memory access device of the PLC (No. 1).

(2) Description of the status variable

Explains the outline of the Robot Status Variable to Table 5-44. Refer to each description page of Page 346, "4.13.2 Explanation of Each Robot Status Variable" for details.

Table 5-44:Related Robot Status Variable (CR800-R series)

Variable name	Details	Reference page
M_GDev/ M_GDevW/ M_GDevD	Writing or referencing numerical values to CPU buffer memory	Page 373
M_HGDev/ M_HGDevW/ M_HGDevD	Writing or referencing numerical values to periodical communication area	Page 377
P_GDev	Writing or referencing positional data to CPU buffer memory	Page 432
P_HGDev	Writing or referencing positional data to periodical communication area	Page 435

Table 5-45:Related Robot Status Variable (CR800-Q series)

Variable name	Details	Reference page
M_UDevW	Read/Write the multi-CPU shared device per word. (U3En\G□)	Page 411
M_XDevD	Read/Write the multi-CPU shared device per double word. (U3En\G□)	Page 411
P_UDev	Read/Write the multi-CPU shared device per position data. (U3En\G□)	Page 438

5.27 Parameter for behavior selection at the error occurrence on dual system

When the parameter CATEGORY is set for the error on dual system shown in Table 5-46, the behavior of the [RESET] key input at the error occurrence on dual system can be selected.

Table 5-46:Target error list

Error number	Content
H0039	Door Switch Signal line is faulty.
H0046	Faulty wiring (Mode sel. switch).
H0051	Wiring of the external emergency stop is abnormal.

The behavior of the [RESET] key input is set as shown in Table 5-47.

Table 5-47:Action to the settings

Parameter settings	Content
3 (Enable the error reset)	Error can be reset. (Default settings)
4 (Disable the error reset)	Error cannot be reset.

To reflect the settings of this parameter, you must turn the controller's power on again.

[Note]
The setting shown in this section is to select the behavior in a state without any abnormalities on the related signal lines or wiring.
With the abnormalities, the errors shown above cannot be reset regardless of the parameter setting.

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 637, "6.3 Dedicated input/output".) or by an emergency stop output (Refer to Page 666, "6.7 Emergency stop input".)

(2) General-purpose input/output.. These signals are used for communication with the PLC 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.)

(4)Devices..... As with PLCs, devices include bit devices, such as X and Y, used to store data in bits and word devices, such as D, used to store data in words. These devices are used to exchange data with GOTs and SLMP-compatible equipment. They are used to publish robot information and provide instructions to robots from external equipment.

Table 6-1:Overall I/O signal map

Item	I/O signal no.	Usage method
Hand input/output	900 to 907	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)=1 Then M_Out(900)=1
PLC 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.
SKIP input	800 to 803	800: Exclusively for stop input 801 to 803: Reference with M_In variables

6.2 PLC link I/O function

This function is only valid on the CR800-R/Q Series drive unit. The RnCPU/QnUD(H)CPU (hereafter referred to as PLC CPU) and R16RTCPU/Q172DSRCPU (hereafter referred to as robot CPU) use CPU buffer memory between CPUs, and communication via a system ladder program. The CPU buffer memory “CPU buffer memory periodical communication area” 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 PLC CPU and robot CPU In order to use the PLC link function.

For the robot CPU, use RT ToolBox or a teaching box (R32TB, R56TB) to set the parameters, and for the PLC CPU, use GX Works. Refer to the operation manual for each setting tool for further details.

(1) PLC CPU parameter setting

* CR800-R series

Use GX Works3 to perform multi CPU parameter settings. The following figure shows a setting example for the system which has a PLC CPU and a robot CPU.

Set up a module configuration before set up PLC's multiple CPUs. Based on the actual module configuration, arrange the modules as shown below.

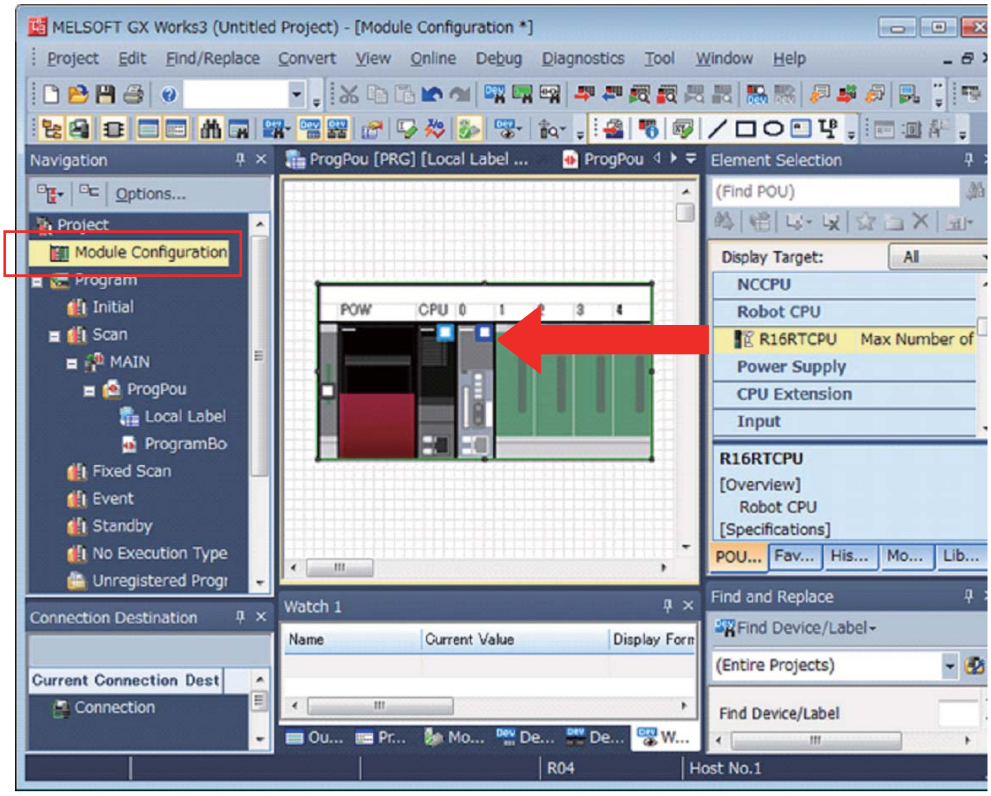
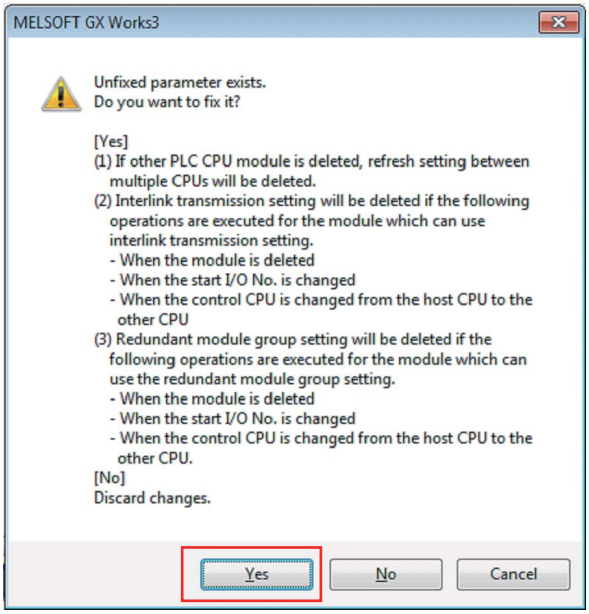
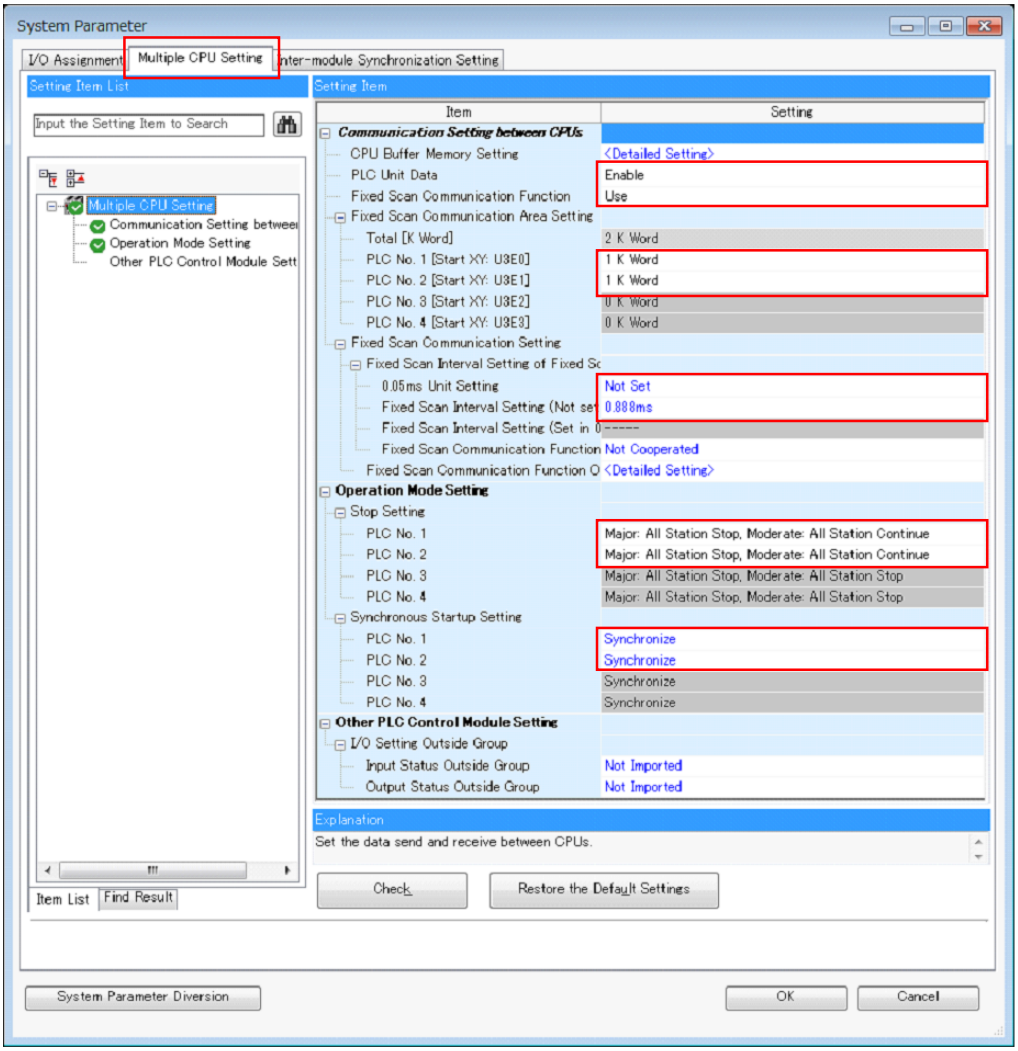


Fig.6-1:GX Works3 module configuration screen (one robot)

When the module configuration screen is closed after arranging the modules, the following dialogue will appear. Click [Yes] button.



Open the system parameter screen to set up the multiple CPUs.
The number of points is set in units of K words. Robot CPUs can use only up to 1K word, so this setting should be set to 1K word.



Setting Item		Description	Setting Value
Communication Setting between CPUs	PLC Unit Data	Set up this to avoid data separation at data communications by refresh. When data of more than 64 bit data is handled, data croaking may occur.	Enable Disable The update cycle of the PLC link I/O function of the robot CPU is as follows. < PLC Unit Data - Disable > It will be the specified periodic communication interval. < PLC Unit Data - Enable > Even when setting the periodic communication interval setting to "0.444 ms" or less, it will be 0.888 ms.
	Fixed Scan Communication Function	Select whether the fixed scan communication function is used or not. Always select "Use" when using together with a robot CPU.	Use
	Fixed Scan Communication Area Setting	Set up the range of transmission area of each device in the fixed scan communication area. The necessary area Note1) for robot is as follows: **CPU buffer memory extended functions are valid:** **• Robot input area: 1.0K** **• Robot output area: 1.0K** **CPU buffer memory extended functions are invalid:** **• Robot input area: 0.5K** **• Robot output area: 0.5K**	<CPU buffer memory extended functions are valid:> • Device #1: Sum of the size (1K) of the data to be sent to the robot and the size of the data to be sent to other devices • Robot device: Set 1K for it • Other devices: Set its own transmission size <CPU buffer memory extended functions are invalid:> • Device #1: Sum of the size (0.5K) of the data to be sent to the robot and the size of the data to be sent to other devices • Robot device: Set 1K for it Note2) • Other devices: Set its own transmission size
	0.05ms Unit Setting	Specify whether to set the fixed scan communication cycle in increments of 0.05ms. Please be sure to select "Not Set" when combining with robot CPU.	Not Set
	Fixed Scan Interval Setting (Not set in 0.05ms unit)	Select the fixed scan communication cycle to be set from the list items. The same option should be specified only for CPU modules which will use the fixed scan communication function. Please select "0.888ms" or less regardless of the PLC Unit Data. When using cooperative control function or interference avoidance function of the robot CPU.	0.222ms 0.444ms 0.888ms 1.777ms 3.555ms 7.111ms
Operation Mode Setting	Stop Setting	If a moderate or major error occurs at any of the CPUs, set whether to stop or continue operation for all CPUs.	Major: All Station Stop, Moderate: All Station Continue. * Set for all CPUs.
	Synchronous Startup Setting	Set up this to synchronize startup time of CPU modules in multiple CPUs system. * Always select "Synchronize" due to the robot CPU takes as long as 18 seconds to startup.	Synchronize

Note1) For information about multiple CPUs and fixed scan communication area, refer to the RCPU manual (MELSEC iQ-R CPU Module User's Manual (Application)).

Note2) Because the area is set up in 1K unit, allocate 1K even in case of 0.5K.

* CR800-Q series

Use GX Works 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. The following figure shows a setting example for the system which has a PLC CPU and a robot CPU.

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.

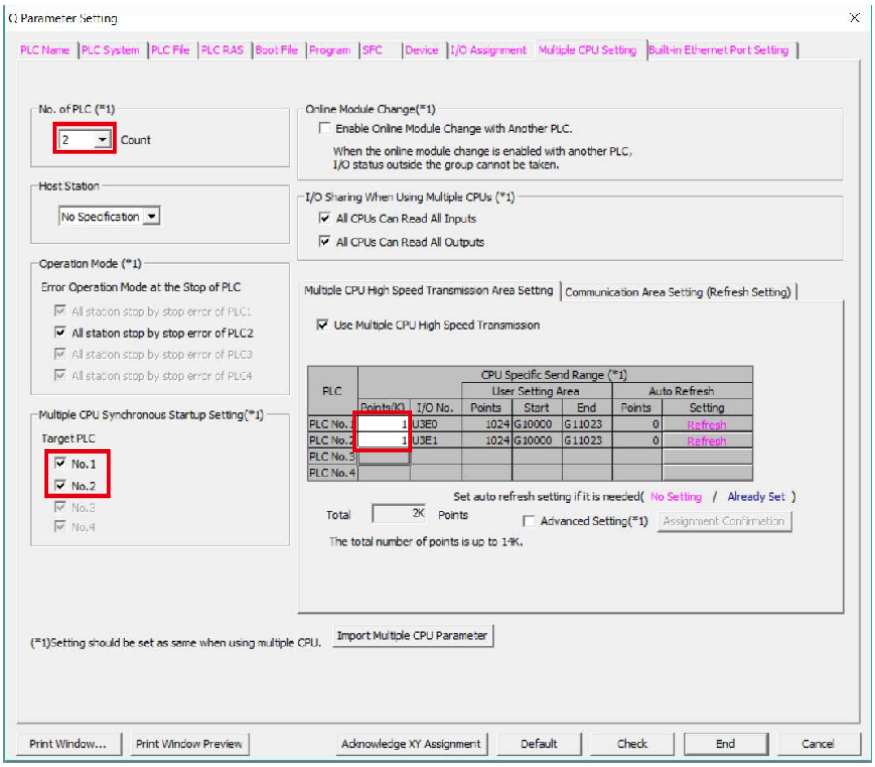


Fig.6-2:PLC CPU: Setting screen on GX Works (example)

A user free area and auto refresh area can be set for the high-speed communication area between multi CPUs, however, the robot CPU (Q172DSRCPU) 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

<table>
<tr><th>Parameter name</th><th>Details</th><th>Factory setting</th></tr>
<tr><td>QMLTCPUN</td><td>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</td><td>2</td></tr>
<tr><td rowspan="2">QMLTCPUn
n = 1 to 4</td><td>[CR800-R series]
CPU buffer memory periodical communication area setting (n = 1 to 4)
At the multi CPU system, read the number of points transmitting and receiving data by the CPU buffer memory periodical communication area of the multi CPU No. 1 to 4 from the multi CPU No. 1, and set it automatically.
It is not necessary to change the value.

First element: Size of the periodical communication area (K word)
Range: 0 to 12
*The total size of all multi CPUs is 24K words at maximum.

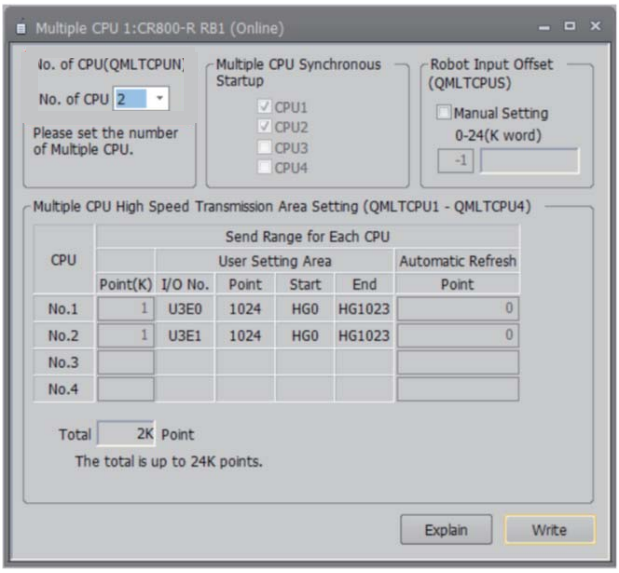
Second element: No. of auto refresh points (word)
Range: 0 to 14335
However, the robot CPU does not support auto refresh, and therefore the number of points for the robot CPU auto refresh are should always be set to 0.
Third element: System reservation
Fourth element: Multi CPU synchronous start-up (1: 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.</td><td rowspan="2">1,0,1,1</td></tr>
<tr><td>[CR800-Q series]
Multi CPUn 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 PLC CPU If the parameter settings do not match, and therefore care should be taken to ensure that the parameter settings for each CPU match.

First element: User free area size (k points)
Range: 1 to 14 (Max.)

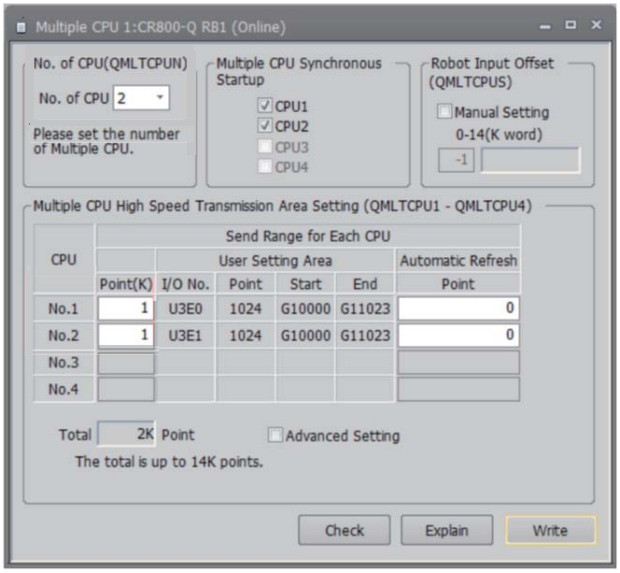
Table 6-3:Setting range by number of CPU
<table><tr><th>CPU quantity</th><th>Setting range</th></tr><tr><td>2</td><td>0 to 14K point</td></tr><tr><td>3</td><td>0 to 13K point</td></tr><tr><td>4</td><td>0 to 12K point</td></tr></table>
Second element: No. of auto refresh points (points)
Range: 0 to 14335
However, the robot CPU does not support auto refresh, and therefore the number of points for the robot CPU auto refresh are should always be set to 0.
Third element: System area size (K points)
Range: 1 or 2
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.</td></tr>
</table>

Parameter name	Details	Factory setting
IQMEM Note1)	Select the CPU buffer memory expanded function. The function is assigned for each bit. 1/0 = available/unavailable 15 0 00000000 00000000 Bits 2, 3, and 5 to 15 are not used. | | |...bit0: Enable expanded function | |.....bit1: PLC direct execution function | |.....bit4: Interference avoidance function	0000000000000000
IQSPEC Note1)	Set up CR800-R/Q series robot's function The function is assigned for each bit. 15 0 00000000 00000000 bit2-15 is unused |...bit0:The direction of CPU buffer memory write-in =0: Read-out and write-in process are both executed in order of head address to final address =1:Read-out process is executed in order of head address to final address. Write-in process is executed in order of final	0000000000000001

Note1) Refer to separate manual: "iQ Platform Supporting Extended Function Instruction Manual (BFP-A3528)" for details.



CR800-R



CR800-Q

Fig.6-3:Robot CPU: Setting screen on RT ToolBox (example)

Applicable Multi CPUs

Multi CPUs are the following iQ Platform compatible CPUs and bases. (Current as of In March, 2018)

CPU type	Model		Remarks
	CR800-R	CR800-Q	
PLC CPU	R00CPU, R01CPU, R02CPU, R04CPU, R08CPU, R16CPU, R32CPU, R120CPU	Universal model QCPU Q03UD(E)CPU, Q04UD(E)HCPU, Q06UD(E)HCPU, Q10UD(E)HCPU, Q13UD(E)HCPU, Q20UD(E)HCPU, Q26UD(E)HCPU, Q50UDEHCPU, Q100UDEHCPU	・The first CPU must be a PLC CPU.
Robot CPU	R16RTCPU	Q172DSRCPU	
Motion CPU	R16MTCPUV, R32MTCPU, R64MTCPU	Q172DCPU, Q173DCPU, Q173DSCPU	
NCCPU	R16NCCPU	Q172NCCPU	
Base	The high-speed basic base between multi-CPUs R35B, R38B, R312B	The high-speed basic base between multi-CPUs Q38DB, Q312DB	・The base which is corresponding to the high-speed communication between multi-CPUs

To set parameters, use RT ToolBox or a teaching box (R32TB, R56TB) for the robot CPU, GX Works or GX Developer for the PLC CPU, MT Works or MT Developer for the motion controller CPU, and Remote Monitor Tool for the NC CPU. Refer to the operation manual for each setting tool for further details.

6.2.2 CPU buffer memory and robot I/O signal compatibility

At the PLC CPU, the CPU buffer memory is accessed like U3E0\HG511 (U3E0\G10511 for the CR800-Q series). The robot CPU No. n CPU buffer memory accesses like U3En\HG511 (U3En\G10511 for the CR800-Q series).
(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 PLC side and bit devices are used at the robot side, and therefore caution is advised.
Please note that the CPU buffer memory and robot I/O signal compatibility is as shown in the following table and cannot be changed.

Table 6-4:CPU buffer memory and robot I/O signal compatibility

PLC (word device)			Robot (bit device)	
Controller	CR800-R	CR800-Q		
Output	U3E0\HG0 to U3E0\HG511	U3E0\G10000 to U3E0\G10511	Input	Robot CPU No.1 / 10000 to 18191
	U3E0\HG512 to U3E0\HG1023	U3E0\G10512 to U3E0\G11023		Robot CPU No.2 / 10000 to 18191
	U3E0\HG1024 to U3E0\HG1535	U3E0\G11024 to U3E0\G11535		Robot CPU No.3 / 10000 to 18191
Input	U3E1\HG0 to U3E1\HG511	U3E1\G10000 to U3E1\G10511	Output	Robot CPU No.1 / 10000 to 18191
	U3E2\HG0 to U3E2\HG511	U3E2\G10000 to U3E2\G10511		Robot CPU No.2 / 10000 to 18191
	U3E3\HG0 to U3E3\HG511	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 as follows.
•CR800-R series
A PLC RnCPU for the first multi CPU, and a robot R16RTCPU for the second multi CPU
•CR800-Q series
A PLC QnUD(H)CPU for the first multi CPU, and a robot Q172DSRCPU for the second multi CPU

[Explanation]
The following explanation is for the CR800-R series. For the CR800-Q series, referring to the Table 6-4 and replace the device numbers.
<0 to 16th row>
M100 to M131 is written to the U3E0\HG0 and U3E0\HG1 shared device memory, and this represents the input from the PLC to the robot. The U3E1\HG0 and U3E1\HG1 shared device memory is read to the bit devices for M200 to M231, and this represents the output from the robot to the PLC.
<17 to 22nd row>
By turning X0 ON, M105 turns ON and the PLC U3E0\HG0 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\HG0 bit 5 turns ON. Consequently, the PLC M205 corresponding to U3E1\HG0 bit 5 turns ON, and Y20 turns ON.

Please note that bit device M201 (U3E0\HG0 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.)

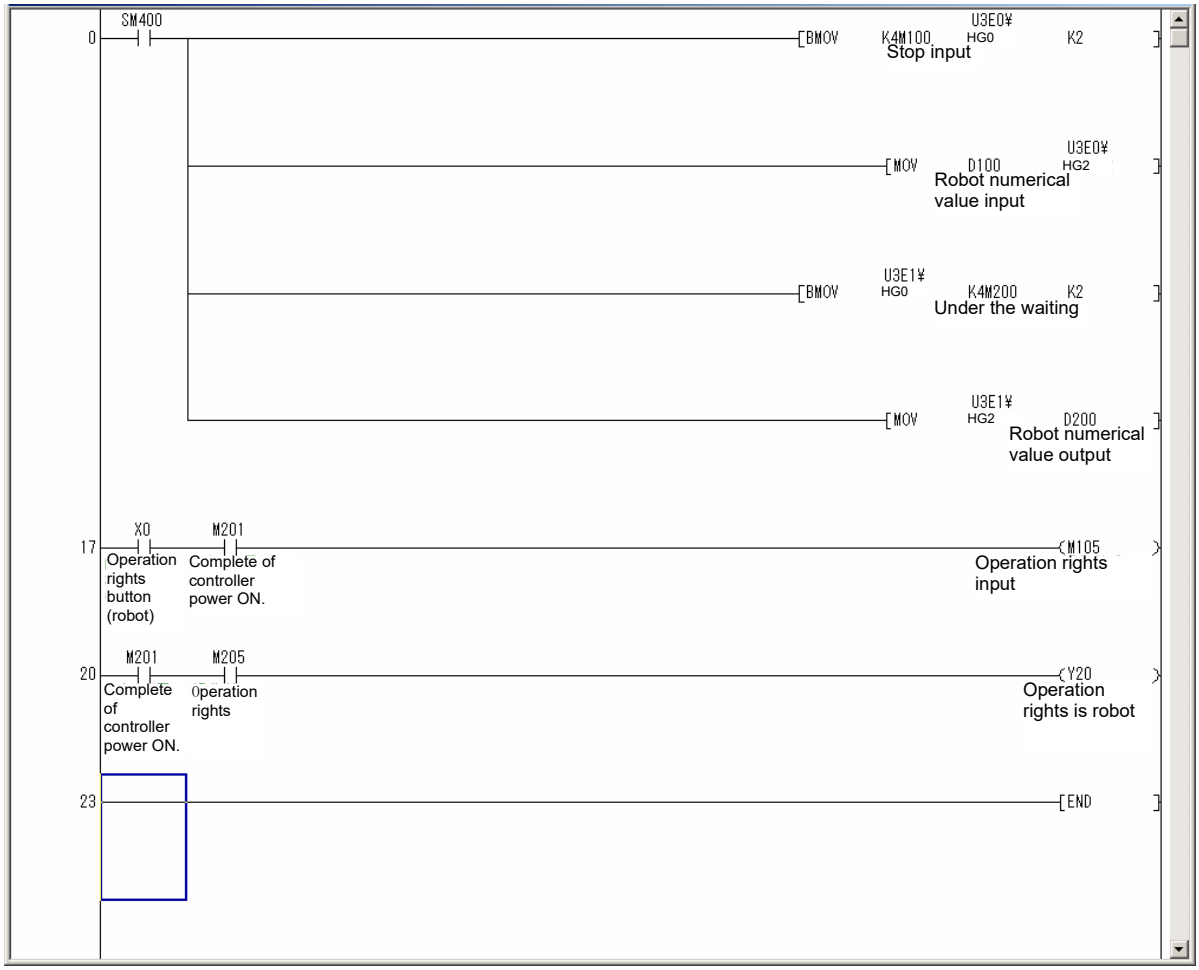


Fig.6-4:Sequence ladder example

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)]	
					CR800-R	CR800-Q
STOP	Stop input (assignment change is impossible)	Pausing output	10000	10000	HG0	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 reset (*)	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	HG1	G10001
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		
-	-	-	-	-		
-	-	-	-	-		
-	-	-	-	-		
-	-	-	-	-		
-	-	-	-	-		
-	-	-	-	-		

Parameter name	Input signal name (*: Operation rights is necessity)	Output signal name	Input	Output	G device[Note1)]	
					CR800-R	CR800-Q
IODATA	Numeric value input 0	Numeric value output 0	10032	10032	HG2	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		
	Numeric value input 4	Numeric value output 4	10036	10036		
	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	HG3	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		
	-	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		

Parameter name	Input signal name (*: Operation rights is necessity)	Output signal name	Input	Output	G device[Note1)]	
					CR800-R	CR800-Q
USRAREA	-	User-designated area 8-points 1	-	10064	HG4	G10004
	-	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 PLC CPU side)

6.3 Dedicated input/output

The functions shown in Table 6-6 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 84, "3.15 Operation of parameter 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). And refer to Page 658, "6.5.2 Timing chart example" for time chart.
To use the dedicated I/O signals, set the controller mode to AUTOMATIC, and turn on the operation rights input signal (IOENA) beforehand.

Table 6-6:Table of dedicated input/output

Parameter name	Class	Name	Function	Signal level Note1)	Factory shipment signal number. Input, output	
					CR800-R/Q	CR800-D
RCREADY	Input	-	-		-1(No meaning),	-1(No meaning),
	Output	Controller power ON ready	Outputs that the power has been turned ON and that the external input signal can be received.		10001	-1
ATEXTMD	Input	-	-		-1(No meaning),	-1(No meaning),
	Output	Remote mode output	This output indicates that the controller mode is set to AUTOMATIC and remote operation mode. This signal must be turned ON before any control tasks using I/O signals can be performed.		10002	-1
TEACHMD	Input	-	-		-1(No meaning),	-1(No meaning),
	Output	Teaching mode output	This output indicates that the controller mode is set to Teaching mode.		10003	-1
ATTOPMD	Input	-	-		-1(No meaning),	-1(No meaning),
	Output	Automatic mode output	This output indicates that the controller mode is set to AUTOMATIC.		10004	-1
IOENA	Input	Operation rights input signal	Sets the validity of the operation rights for the external signal control.	Level	10005,	5,
	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 controller mode is set to AUTOMATIC, and there is no other device that currently has the operation right.		10005	3
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	
					CR800-R/Q	CR800-D
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. Normal open and normal close may be changed using the parameter INB.	Level	10000 (Cannot change),	0(Cannot change),
	Output	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
	Output	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,
	Output	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

Parameter name	Class	Name	Function	Signal level Note1)	Factory shipment signal number. Input, output	
					CR800-R/Q	CR800-D
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,
	Output	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
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)		10011	-1
AUTOENA	Input	Automatic operation enabled input	Disables automatic operation when inactive. If this signal is inactive, and the AUTOMATIC mode is entered, a low level error L5010 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.	Level	-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.		10012	-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.	Level	-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.		10013	-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

Parameter name	Class	Name	Function	Signal level Note1)	Factory shipment signal number. Input, output	
					CR800-R/Q	CR800-D
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 ORS18160.	Edge	10015,	-1,
	Output	-	-		-1(No meaning)	-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. [EMGERR output conditions] • External emergency stop error: H0050, H0051 (Dual line is faulty) • Operation panel emergency stop error: H0060, H0061 (Dual line is faulty) • T/B emergency stop error: H0070, H0071 (Dual line is faulty) • Door switch signal faulty: H0039, H0040 (Dual line is faulty) • Wiring error of EMG connector: H0141		10019	-1
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	-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	-1
BRKLOCK	Input	Brake lock input	This input signal is turned on to lock the brake during servo on. This function is available for all of the axes with brakes.	Level	-1,	-1,
	Output	Output during brake lock	This output signal is turned on when the brake lock input signal is turned on.		-1	-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	-1

Parameter name	Class	Name	Function	Signal level Note1)	Factory shipment signal number. Input, output	
					CR800-R/Q	CR800-D
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 15 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 15 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 15 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 15 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 OVDSEL) 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 15 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 OVRDOUT), *Outputs the line number (output by the LINEOUT) *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 15 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)
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 15 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.		10022	-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 15 ms before reading the numerical output (IODATA) signal.	Edge	10023,	-1,
	Output	Line No. output signal	The "line number output in progress" status is output to the numerical output.		10023	-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 15 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.		10024	-1

Parameter name	Class	Name	Function	Signal level Note1)	Factory shipment signal number. Input, output	
					CR800-R/Q	CR800-D
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 15 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.		10025	-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. 0/1/2/3/4/5 = Joint/ XYZ/ Cylindrical/ 3-axis XYZ/ tool/Work (Ex-T) Note) For Ex-T control or Ex-T jog, refer to Page 750, "7.3 Ex-T control".	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)
JOGMENO	Input	Jog mechanism number input (start No., end No.)	Designates the mechanism number. If this parameter is not specified, mechanism number is fixed to machine 1.	Level	-1(Start bit), -1(End bit),	-1(Start bit), -1(End bit),
	Output	Jog mechanism number output (start No., end No.)	Outputs the current mechanism number.		-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, θ, 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. WORK jog mode (Ex-T jog mode): X, Y, Z, A, B and C axes from the start number. Note) For Ex-T control or Ex-T jog, refer to Page 750, "7.3 Ex-T control".	Level	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, θ, 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. WORK jog mode (Ex-T jog mode): X, Y, Z, A, B and C axes from the start number. Note) For Ex-T control or Ex-T jog, refer to Page 750, "7.3 Ex-T control".	Level	Note4) -1, -1	Note4) -1, -1
	Output	-	-			

Parameter name	Class	Name	Function	Signal level Note1)	Factory shipment signal number. Input, output	
					CR800-R/Q	CR800-D
JOGWKNO	Input	Work coordinates number	Specify the work coordinates number (Ex-T coordinates number) for the standard of work jog operation with numerical value 1 to 8. Note) Specify the work coordinates number for the standard of work jog operation with numerical value 1 to 8. This input signal is read with the edge (change from off to on) of Jog valid input signal: JOGENA. When you change the work coordinates number, please once change Jog valid input signal: JOGENA from off to on. Note) For Ex-T control or Ex-T coordinates number, refer to Page 750, "7.3 Ex-T control".	Level	Note3) -1(Start bit), -1(End bit), -1(Start bit), -1(End bit)	Note3) -1(Start bit), -1(End bit), -1(Start bit), -1(End bit)
	Output		The current value inputted to the Work coordinates number (Ex-T coordinates number) is outputted. Note) For Ex-T control or Ex-T coordinates number, refer to Page 750, "7.3 Ex-T control".			
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	-	-		HNDCNTL1 10048(Start bit), 10055(End bit)	-1(Start bit), -1(End bit)
	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).			
HNDSTSn (n=1 to 3)	Input	-	-		HNDSTS1 10056(Start bit), 10063(End bit)	-1(Start bit), -1(End bit)
	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).			

Parameter name	Class	Name	Function	Signal level Note1)	Factory shipment signal number. Input, output	
					CR800-R/Q	CR800-D
HANDENA	Input	Hand control permission input	Permit or prohibit control of the robot hand by the external signal. 1/0 = permission / prohibition Notes) The control of the robot's hand is available during automatic execution. The interlocking of the robot and external equipment, such as the PLC, is necessary sure because of the safety. When the control of robot's hand by an external signal is permitted, the commands: HOpen/HClose, of a program becomes invalid.	Level	-1, -1	-1, -1
	Output	Hand control permission output	The permission condition of control of robot's hand by the external signal is outputted. 1/0 = permission / prohibition When the hand control permission input signal is turned on and T/B is not available, this signal turns on.			
HANDOUT	Input	Hand output control signal	Set up the external input-signal range for controlling the robot hand. The input signal set up here is matched in order with the hand signal set up by parameter: HANDTYPE Element 1: Hand output control signal start number Element 2: Hand output control signal finish number	Edge	-1, -1	-1, -1
	Output	-	-			
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 520, "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

Parameter name	Class	Name	Function	Signal level Note1)	Factory shipment signal number. Input, output	
					CR800-R/Q	CR800-D
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
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
PSSLOT	Input	Slot number specification	Slot number into which program storing the position data the user wants to be outputted is loaded is specified. (1 to 32) *Change is available in the state input signal of parameter: PSOUT is OFF.	Level	-1 (input/starting value), -1 (input/ending value) * 6 bits in width maximum -1 (output/starting value) -1 (output/ending value) * 6 bits in width maximum	
	Output	Specified slot number output	Slot number specified on the input side is outputted.			
PSTYPE	Input	Position data type specification	Type of the position data which the user wants to be outputted is specified. [Conditions for specification] 0(OFF): Position-type variable (P1, P10 or the like) 1(ON): Joint-type variable (J1, J10 or the like) *Change is available in the state input signal of parameter:PSOUT is OFF.	Level	-1,-1	
	Output	Specified position data type output	Type of the position data specified on the input side is outputted. [Output information] 0(OFF): Position-type variable (P1, P10 or the like) 1(ON): Joint-type variable (J1, J10 or the like)			
PSNUM	Input	Position number specification	Position number (number of "P" or "J" variable) for the position data the user wants to be outputted is specified. 0 ~ 65535 (P0 ~ P65535 or J0 ~ J65535) Example: If you need a position data for P100, specify the value "100," using the signal number you specified between an input/starting number and an input/ending number. The width which can be specified for a signal number is 16 bits maximum, which allows you to specify position data for up to "P66535." However, it should be taken note that the position variable for "P001" is not accepted. *Change is available in the state input signal of parameter: PSOUT is OFF.	Level	-1 (input/starting value) -1 (input/ending value) * 16 bits in width maximum -1 (output/starting value) -1 (output/ending value) * 16 bits in width maximum	
	Output	Specified position number output	Position number specified on the input side is outputted.			

Parameter name	Class	Name	Function	Signal level Note1)	Factory shipment signal number. Input, output	
					CR800-R/Q	CR800-D
PSOUT	Input	Position data output specification	Specifications are made so that specified position number data for specified slot number is outputted. Position data is updated when specified signal is turned ON. Input signal level is "level," but position data remains un-updated whenever the signal stays ON. Information is updated upon the signal being turned ON. 0(OFF): Position data is not required to be outputted or position number is being specified. 1(ON): Position data output demanded.	Level	-1,-1	
	Output	Position data being outputted	Output is made to indicate that specified position data has been outputted. [Output information] 0(OFF): Position data not yet outputted 1(ON): Position data being outputted			
PSPOS	Input	-	-	-	-1 (Not significant)	
	Output	Specified position data	Specified position data is outputted by using signals consisting of 32 bits for 8 axes plus 32 bits for 2 elements (structural flags) derived from signal numbers specified under this parameter (320 bits are used). The range of the setting value: (1)CR800-R/Q series 10000 ~ 17872: CPU buffer memory (2)CR800-D series 2000 ~ 3632: Profibus 6000 ~ 7728: CC-Link Unit of each component value for position data expressed by a 32-bit signal is micrometer (um) = 10^{-3} mm or 10^{-3} degree. Take note that outputted data comes as a signed integer (-2^{31} ~ 2^{31}-1). However, structural flags are outputted in the form of values they carry without being converted in terms of micrometer. Position type variable: X, Y, Z, A, B, C, L1, L2, FL1, FL2 Joint type variable: J1, J2, J3, J4, J5, J6, J7, J8 If an error occurs in slot number, "0x7FFFFFFF" is outputted. If an error occurs in position number, "0x80000000" is outputted. Since the time chart and precautions are shown in Page 662, "(5) Example of external operation timing chart (Part 5)", refer to it.		-1	
TMPOUT	Input	Temperature output request	The temperature inside the robot controller is output to the numerical output (IODATA). After the start of inputting this signal to the robot, wait at least 15 ms before reading the numerical output (IODATA) signal	Edge	-1,-1	
	Output	Temperature output signal	The "temperature output in progress" status is output to the numerical output.	-		
RSTBAT	Input	Battery cumulative time reset	Reset the battery cumulative tine.	-	-1,	
	Output	Reset is completed	Output that the reset has been completed.		-1	

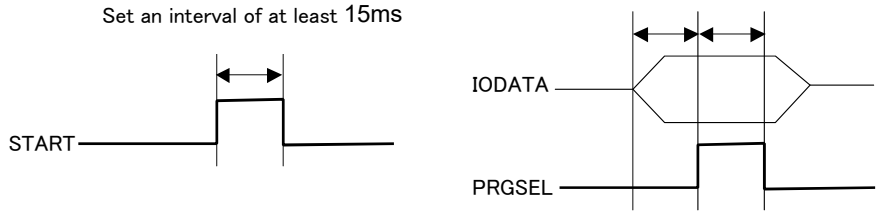
Parameter name	Class	Name	Function	Signal level Note1)	Factory shipment signal number. Input, output	
					CR800-R/Q	CR800-D
RSTGRS	Input	Maintenance forecast reset (grease)	Reset the grease information of the maintenance forecast. * The axis bit pattern is specified by the parameters IODATA or DIODATA.	-	-1,	
	Output	Reset is completed	Output that the reset has been completed.		-1	
RSTBLT	Input	Maintenance forecast reset (belt)	Reset the belt information of the maintenance forecast. * The axis bit pattern is specified by the parameters IODATA or DIODATA.	-	-1,	
	Output	Reset is completed	Output that the reset has been completed.		-1	
SVDATA	Input	-	-	-	-1 (Not significant)	-1, -1,
	Output	The load factor of the servo	The maximum load factor (%) of J1 axis to J8 axis is outputted from the register number specified as the third element of this parameter to the number specified as the 4th element. The outputted load factor is updated with every 2 seconds. (CC-Link register supported)			-1 (Start register), -1 (End register)
DOORSTS1	Input	-	-	-	-1,	-1,
	Output	States of the door switch 1	Output that the status of the door switch 1 system.		-1	-1
DOORSTS2	Input	-	-	-	-1,	-1,
	Output	States of the door switch 2	Output that the status of the door switch 2 system.		-1	-1
DOORSTS	Input	-	-	-	-1,	-1,
	Output	States of the door switches	Output that the logical ADD of the door switch 1 and 2 systems. Both systems are ON, this signal is also ON.		-1	-1

Note1) 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 15 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)



Note2) 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}

Note3) Set in the order by input start No., input end No., output start No. and output end No.
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 at Jog mode input, the expression can be created by setting 1 bits.

Note4) 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.

Note5) 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 6-7: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).
SAFEPOS	Safe point return input	
JOGENA	Jog enable input	
SRVON	Servo ON input	This does not function in the servo OFF input state.
MELOCK	Machine lock input	This functions only in the program selection state (when SLOTINIT output is ON).
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 the "AUTOMATIC" and the IOENA is on)

(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)

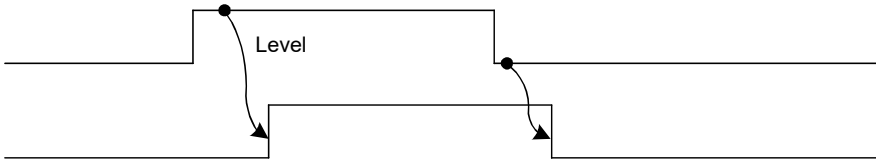
<Output>
Auto mode output
(ATTOPMD)

(Indicates when the key switch on the operation panel is the "AUTOMATIC" and the IOENA is off)

(5) IOENA (Operation right input signal/operation right output signal)

<Intput>
Operation right input (IOENA)

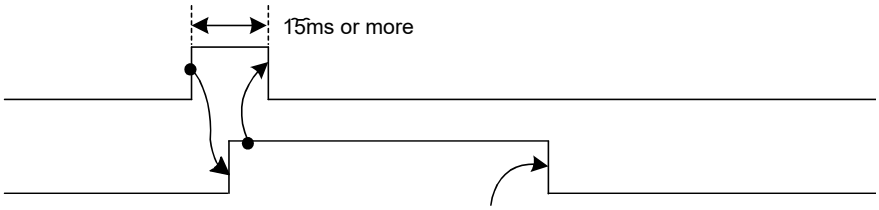
<Output>
Operation right output (IOENA)



(6) START (Start input/operating output)

<Intput>
Start input (START)

<Output>
Operating output (START)

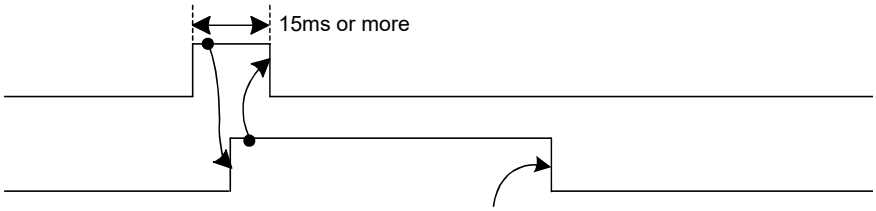


When the STOP signal, or the emergency stop or other signal was input, or after the completion of the CYCLE signal

(7) STOP (Stop input/aborting output)

<Intput>
Stop input (STOP)

<Output>
Aborting output (STOP)



When the START, SnSTART or SLOTINIT signal was input

(8) STOPSTS (Output during stop signal input)

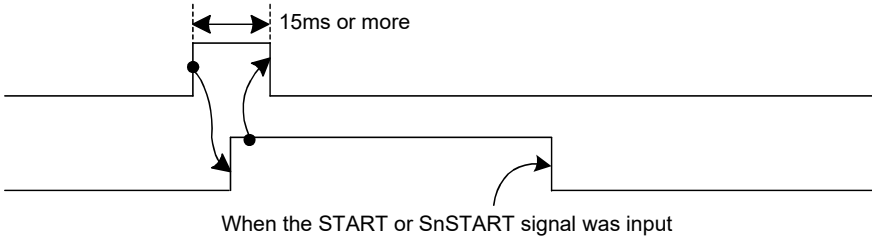
<Output>
During stop signal input
(STOPSTS)

(Indicates that the STOP is being input.)

(9) SLOTINIT (Program reset input/program selectable output)

<Intput>
Program reset (SLOTINIT)

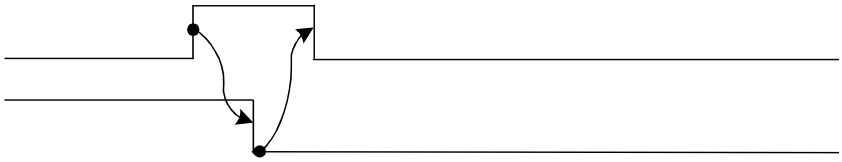
<Output>
Program selectable output
(SLOTINIT)



(10) ERRRESET (Error reset input/output during error occurrence)

<Intput>
Error reset input (ERRRESET)

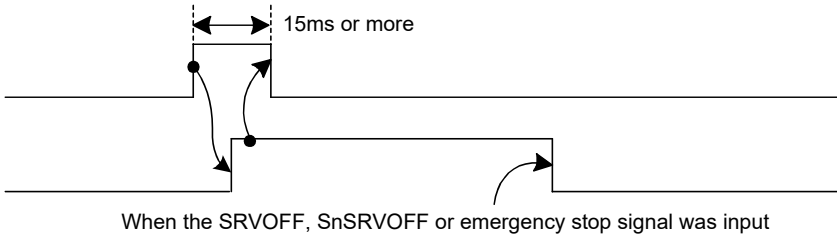
<Output>
Output during error occurrence
(ERRRESET)



(11) SRVON (Servo ON input/output during servo ON))

<Intput>
Servo ON input (SRVON)

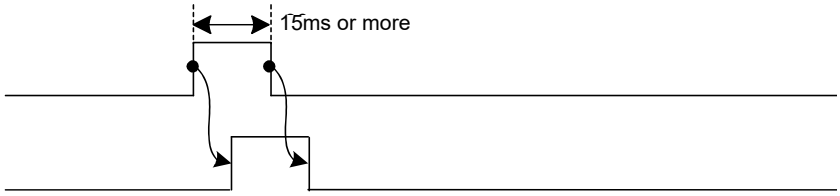
<Output>
Output during servo ON (SRVON)



(12) SRVOFF (Servo OFF input/servo ON disable output)

<Intput>
Servo OFF input (SRVOFF)

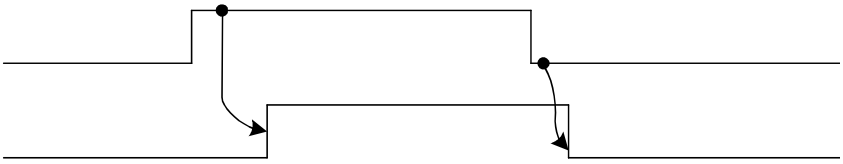
<Output>
Servo ON disable output (SRVOFF)



(13) AUTOENA (Auto operation input/auto operation enable output)

<Intput>
Auto operation enable input
(AUTOENA)

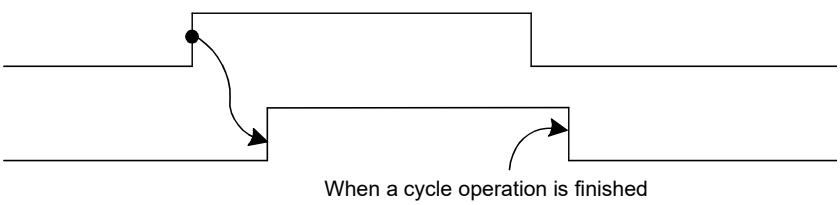
<Output>
Auto operation enable output
(AUTOENA)



(14) CYCLE (Cycle stop input/output during cycle stop operation)

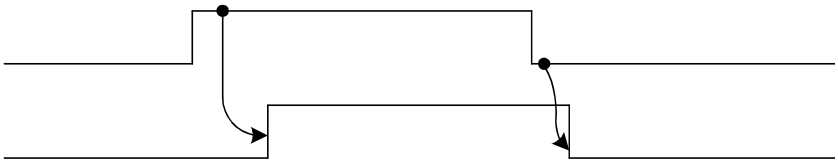
<Intput>
Cycle stop input (CYCLE)

<Output>
Output during cycle stop operation
(CYCLE)



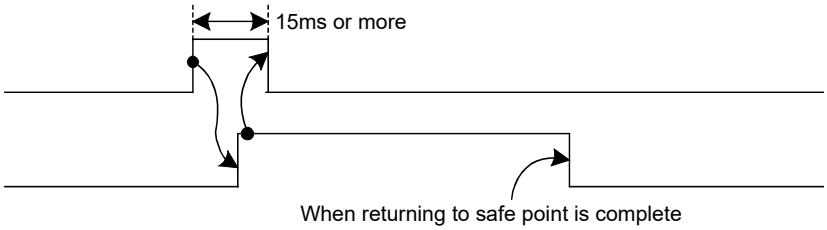
(15) MELOCK (Machine lock input/output during machine lock)

<Intput>
Machine lock input (MELOCK)
<Output>
Output during machine lock (MELOCK)



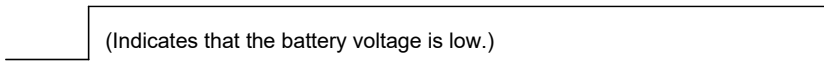
(16) SAFEPOS (Return to safe point input/output during return to safe point)

<Intput>
Return to safe point input (SAFEPOS)
<Output>
Output during return to safe point (SAFEPOS)



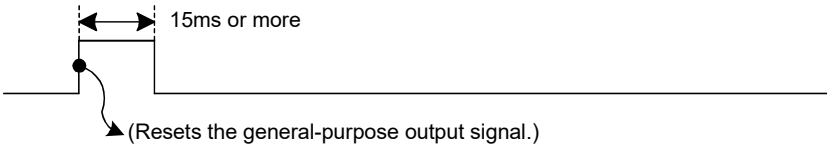
(17) BATERR (Low battery voltage output)

<Output>
Low battery voltage (BATERR)



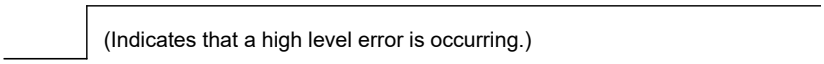
(18) OUTRESET (General-purpose output signal reset request input)

<Intput>
General-purpose output signal reset (OUTRESET)



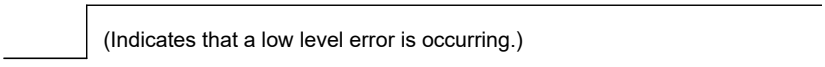
(19) HLVLERR (Output during high level error occurrence)

<Output>
High level error output (HLVLERR)



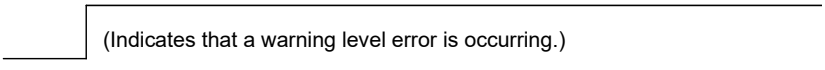
(20) LLVLERR (Output during low level error occurrence)

<Output>
Low level error output (LLVLERR)



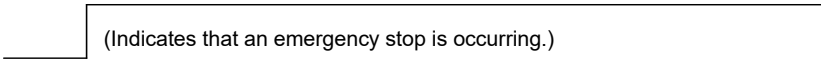
(21) CLVLERR (Output during warning level error occurrence)

<Output>
Warning level error output (CLVLERR)



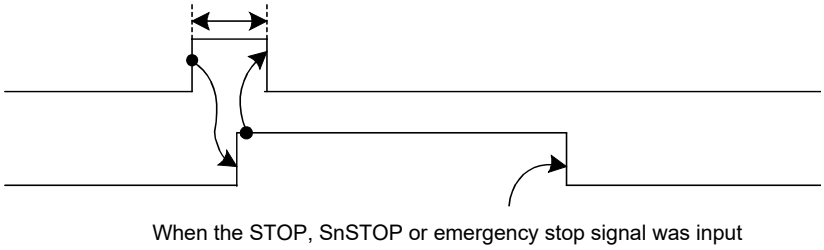
(22) EMGERR (Output during emergency stop)

<Output>
Emergency stop output (EMGERR)



(23) SnSTART (Slot n start input/output during slot n operation)

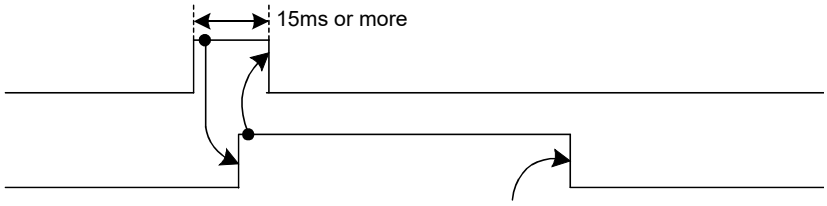
<Intput>
Slot n start input (SnSTART)
<Output>
Output during slot n operation (SnSTART)



(24) SnSTOP (Slot n stop input/output during slot n aborting)

<Intput>
Slot n stop input (SnSTOP)

<Output>
Output during slot n aborting (SnSTOP)

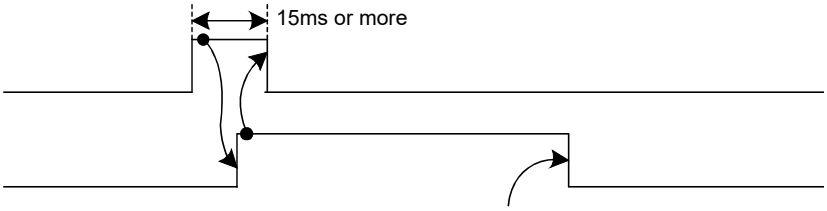


When the START, SnSTART or SLOTINIT signal was input

(25) MnSRVOFF (Mechanical n servo OFF input/mechanical n servo ON disable output)

<Intput>
Mechanical n servo OFF input (MnSRVOFF)

<Output>
Mechanical n servo ON disable output (MnSRVOFF)

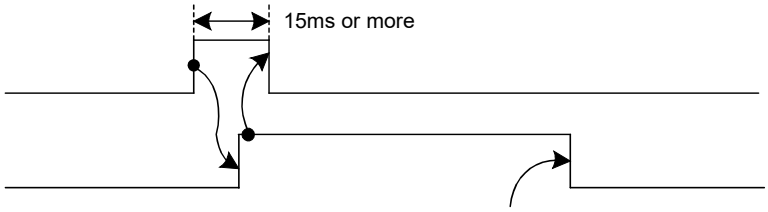


When the SRVON, SnSRVON or SRVON signal was input

(26) MnSRVON (Mechanical n servo ON input/output during mechanical n servo ON)

<Intput>
Mechanical n servo ON input (MnSRVON)

<Output>
Output during mechanical n servo ON (MnSRVON)

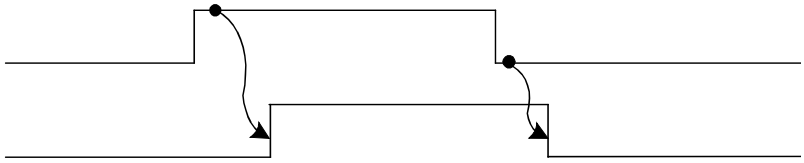


When the SRVOFF, SnSRVOFF or emergency stop signal was input

(27) MnMELOCK (Mechanical n machine lock input/output during mechanical n machine lock)

<Intput>
Mechanical n machine lock input (MnMELOCK)

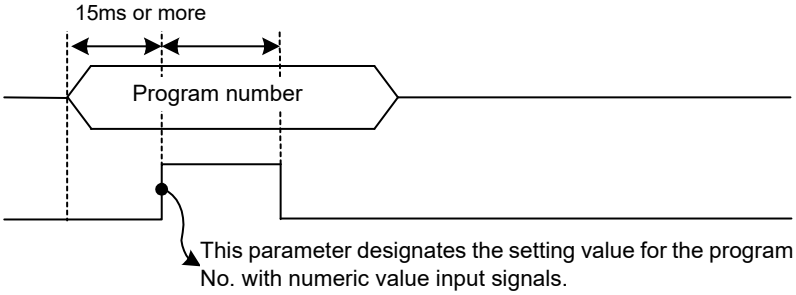
<Output>
Output during mechanical n machine lock (MnMELOCK)



(28) PRGSEL (Program selection input)
* This is used together with the numeric value input (IODATA).

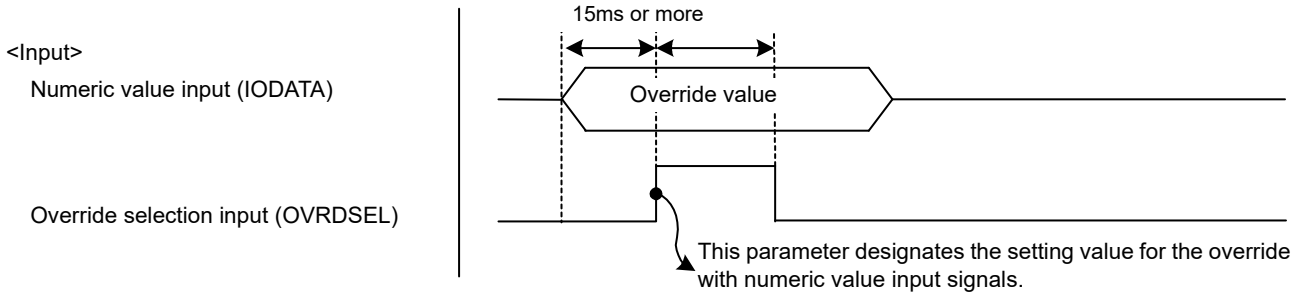
<Input>
Numeric value input (IODATA)

Program selection input (PRGSEL)



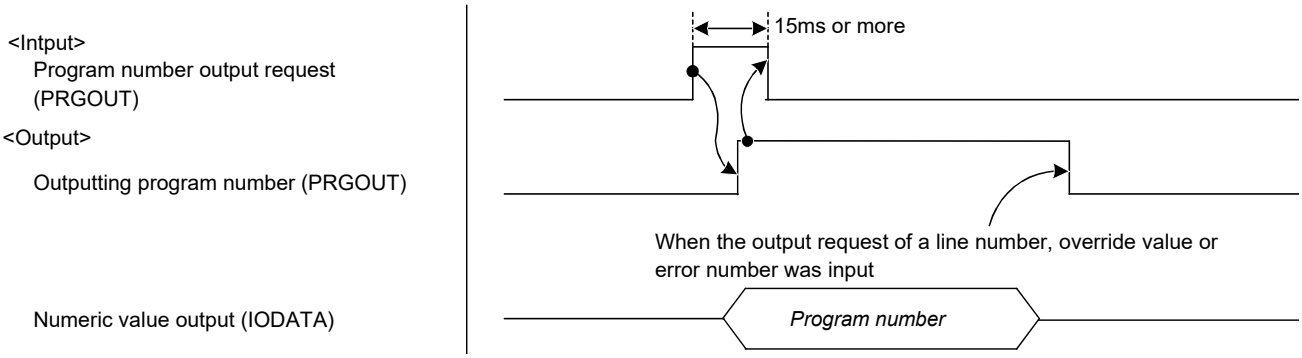
This parameter designates the setting value for the program No. with numeric value input signals.

(29) OVRDSEL (Override selection input)
* This is used together with the numeric value input (IODATA).

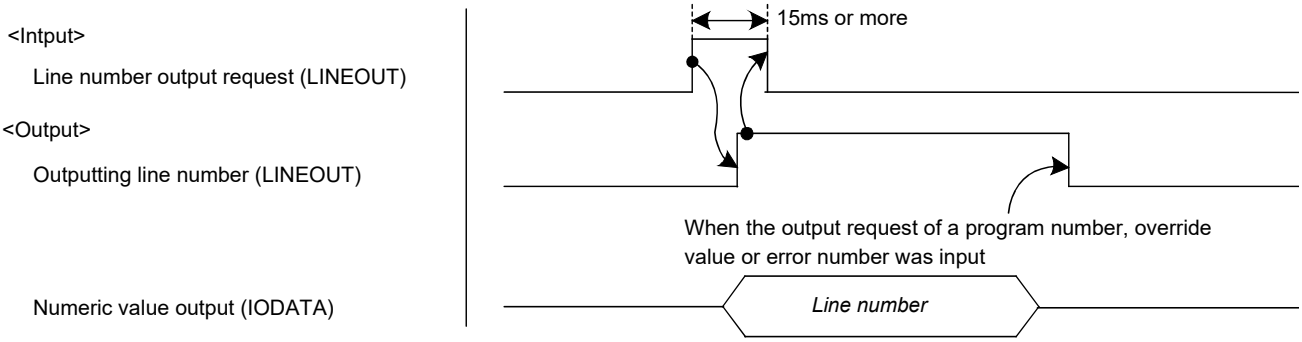


(30) IODATA (Numeric value input/numeric value output)
* This is used together with PRGSEL, OVRDSEL, PRGOUT, LINEOUT, OVRDOUT or ERROUT.

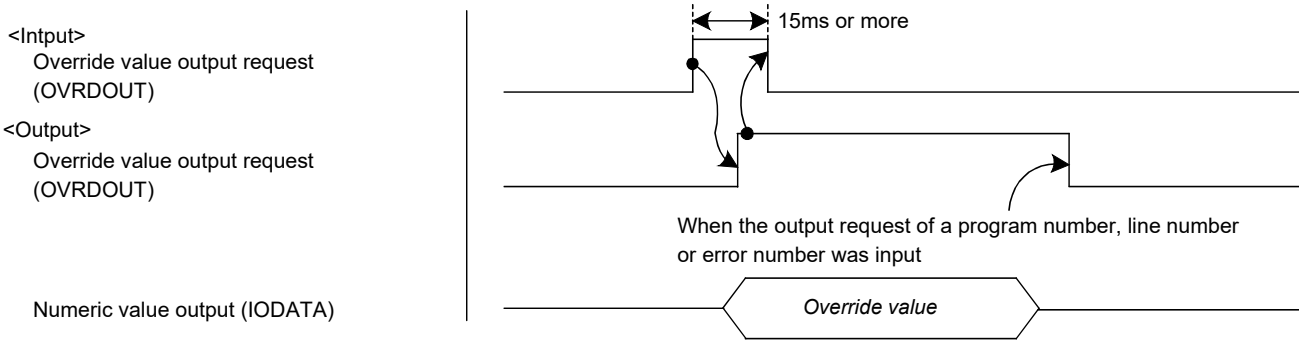
(31) PRGOUT (Program number output request input/outputting program number)
* This is used together with the numeric value output (IODATA).



(32) LINEOUT (Line number output request input/outputting line number)
* This is used together with the numeric value output (IODATA).



(33) OVRDOUT (Override value output request/outputting override value)
* This is used together with the numeric value output (IODATA).



(34) ERROUT (Error number output request/outputting error number)

* This is used together with the numeric value input (IODATA).

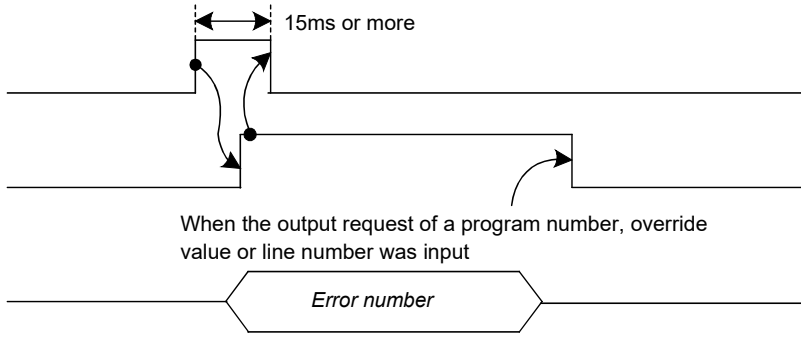
<Intput>

Error number output request (ERROUT)

<Output>

Outputting error number (ERROUT)

Numeric value output (IODATA)



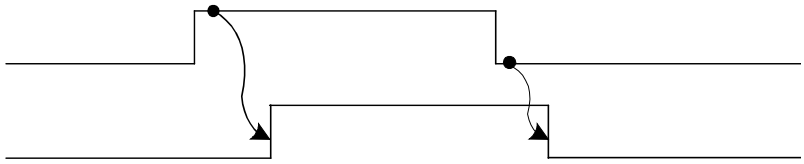
(35) JOGENA (Jog enable input/output during jog enabled)

<Intput>

Jog enable input (JOGENA)

<Output>

Output during jog enabled (JOGENA)



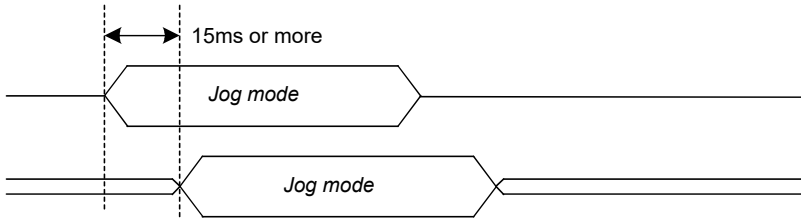
(36) JOGM (Jog mode input/jog mode output)

<Intput>

Jog mode input (JOGM)

<Output>

Jog mode output (JOGM)

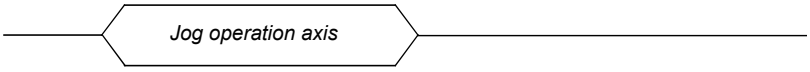


(Replies the setting value of the jog mode input signal with jog mode output.)

(37) JOG+ (Input for 8 axes on jog feed plus side)

<Intput>

8 axes on jog feed plus side (JOG+)

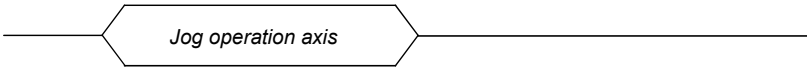


(Specify the axis that will perform jog operation in the plus direction.)

(38) JOG- (Input for 8 axes on jog feed minus side)

<Intput>

8 axes on jog feed minus side (JOG-)

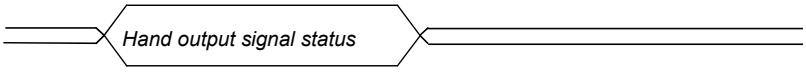


(Specify the axis that will perform jog operation in the minus direction.)

(39) HNDCNTLn (Mechanical n hand output signal status)

<Output>

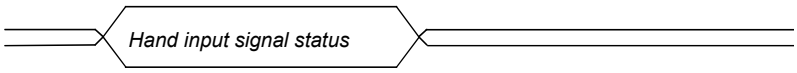
Mechanical n hand output signal status (HNDCNTLn)



(Indicates the output signal status of the hand.)

(40) HNDSTSn (Mechanical n hand input signal status)

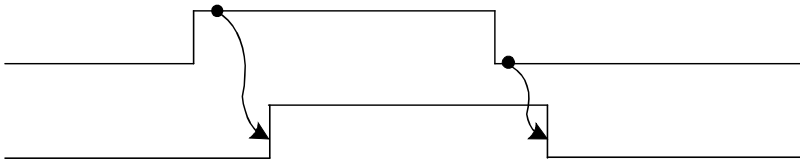
<Output>
Mechanical n hand input signal status (HNDSTSn)



(Indicates the input signal status of the hand.)

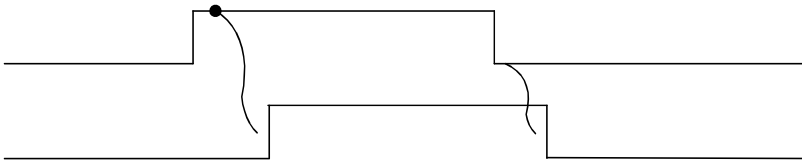
(41) HNDERRn (Mechanical n hand error input signal/output during mechanical n hand error occurrence)

<Intput>
Mechanical n hand error input (HNDERRn)
<Output>
Output during mechanical n hand error occurrence (HNDERRn)



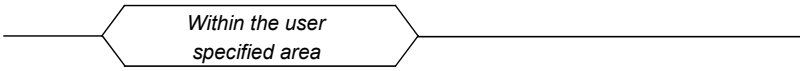
(42) AIRERRn (Mechanical n pneumatic error input signal/outputting mechanical n pneumatic error)

<Intput>
Mechanical n pneumatic error input (AIRERRn)
<Output>
Outputting mechanical n pneumatic error (AIRERRn)



(43) USRAREA (User-specified area 8 points output)

<Output>
User-specified area 8 points (USRAREA)

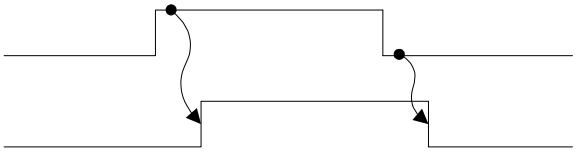


(Indicates that it is within the area specified by areas 1 though 8.)

(44) MnWUPENA (Mechanism n warm-up operation mode enable input signal/ Mechanism n warm-up operation mode output signal)

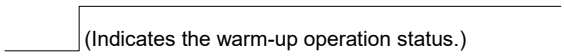
<Input>
Mechanism n warm-up operation mode enable input signal (M*n*WUPENA)

<Output>
Mechanism n warm-up operation mode output signal (M*n*WUPENA)



(45) MnWUPMD (Mechanism n warm-up operation status output signal)

<Output>
Mechanism n warm-up operation status output signal (M*n*WUPMD)

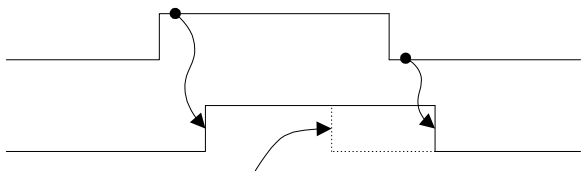


(Indicates the warm-up operation status.)

* 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)

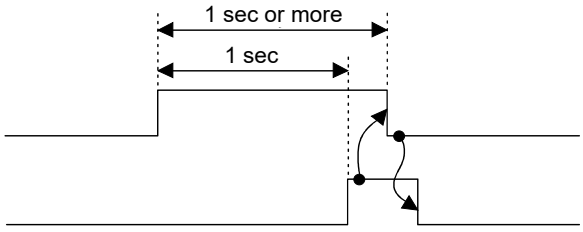


When the warm-up operation status is canceled while the warm-up operation mode is enabled

(46) RSTBAT (Battery cumulative time reset)

<Input>
Battery cumulative time reset input (RSTBAT)

<Output>
Battery cumulative time reset completed output (RSTBAT)

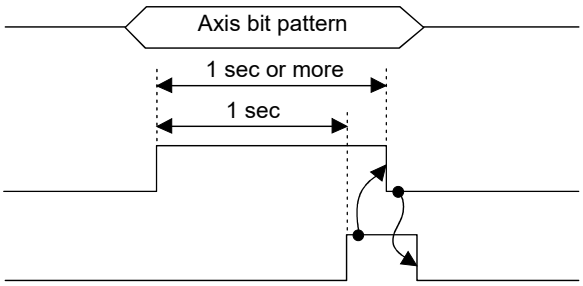


(47) RSTGRS (Maintenance forecast reset (grease))

<Input>
Numeric value input (IODATA)

Maintenance forecast reset (grease) input (RSTGRS)

<Output>
Maintenance forecast reset (grease) completed output (RSTBAT)

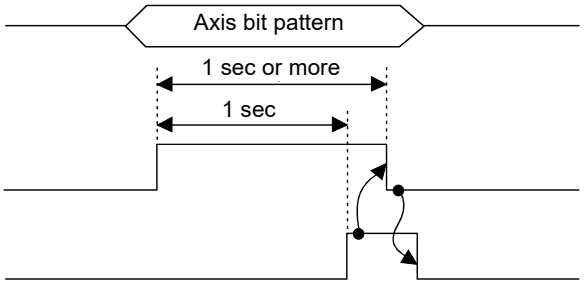


(48) RSTBLT (Maintenance forecast reset (belt))

<Input>
Numeric value input (IODATA)

Maintenance forecast reset (belt) input (RSTGRS)

<Output>
Maintenance forecast reset (belt) completed output (RSTBAT)



6.5.2 Timing chart example

(1) External signal operation timing chart (Part 1)

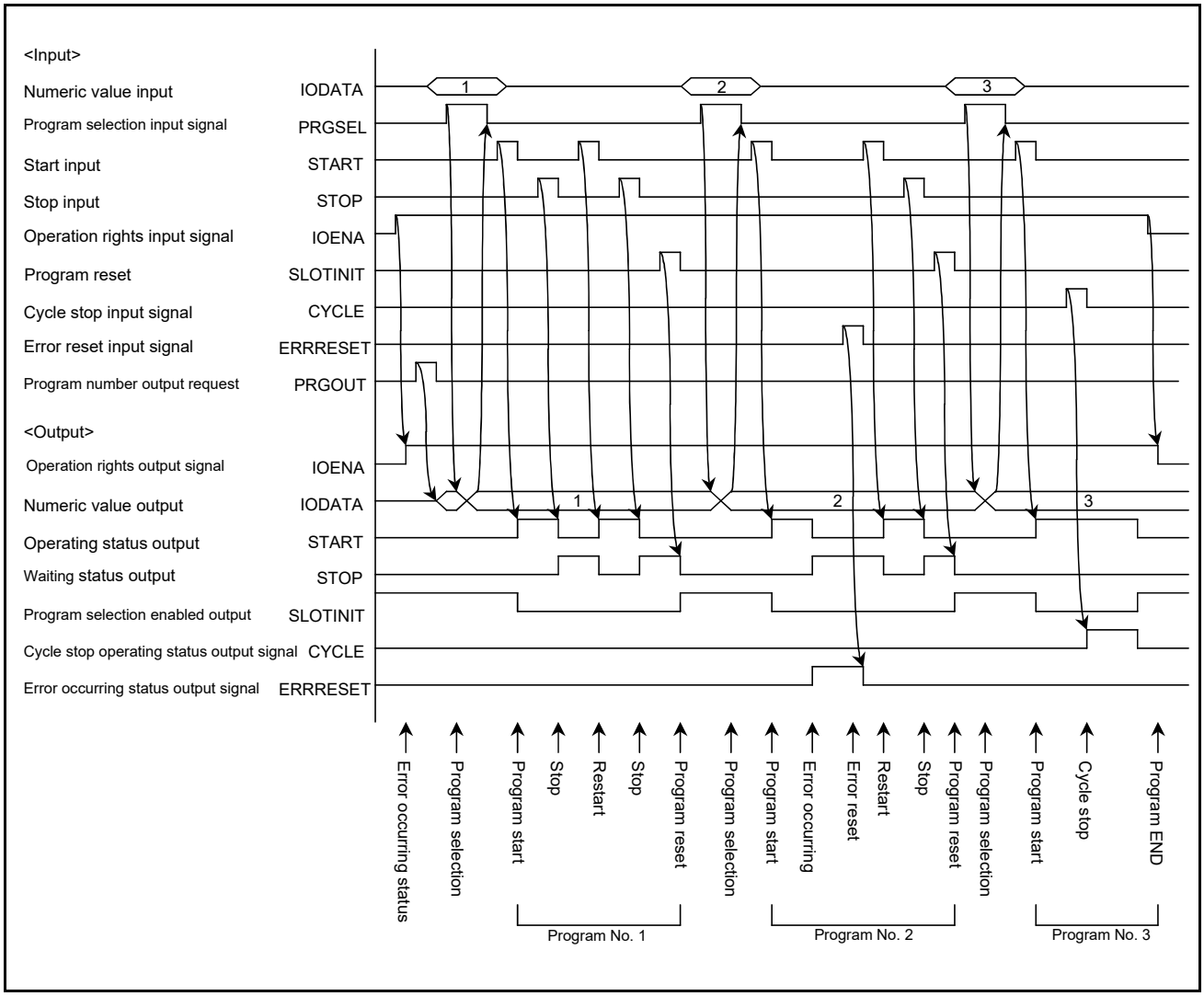


Fig.6-5:Example of external operation timing chart (Part 1)

(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. 6-6.

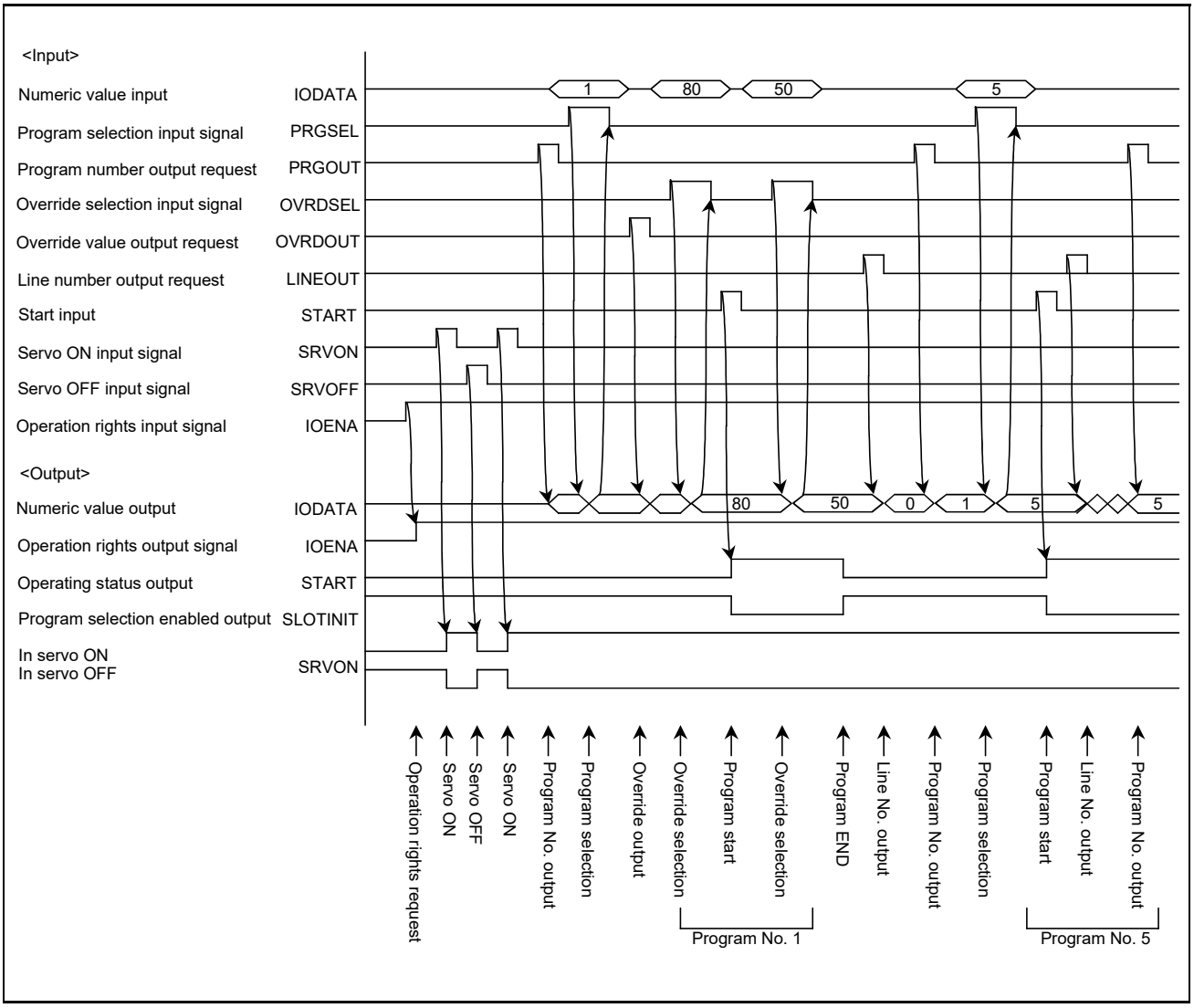


Fig.6-6:Example of external operation timing chart (Part 2)

(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. 6-7.

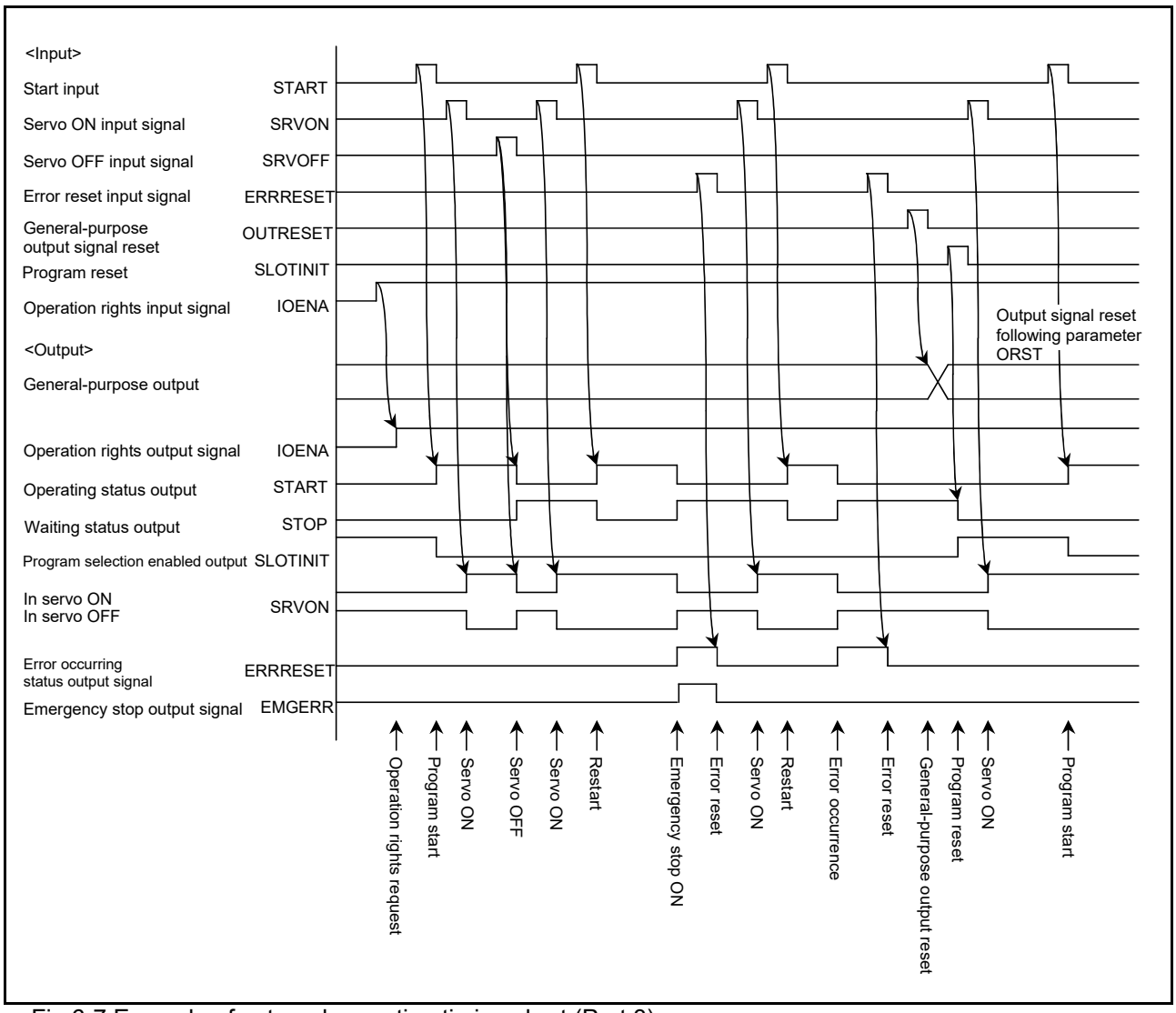


Fig.6-7:Example of external operation timing chart (Part 3)

(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. 6-8.

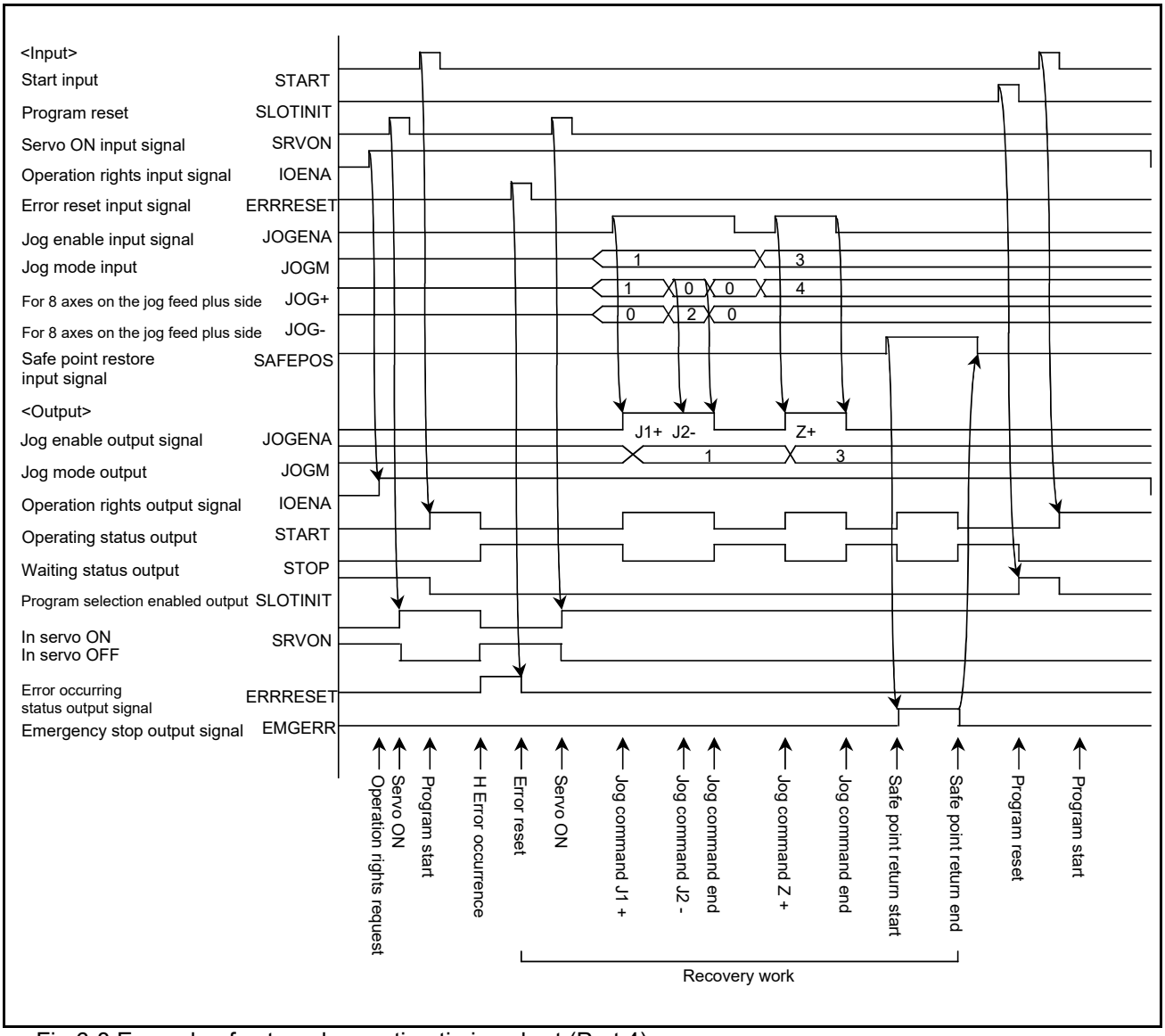


Fig.6-8:Example of external operation timing chart (Part 4)

(5) Example of external operation timing chart (Part 5)

Given below is a timing chart for the dedicated input/output signals.

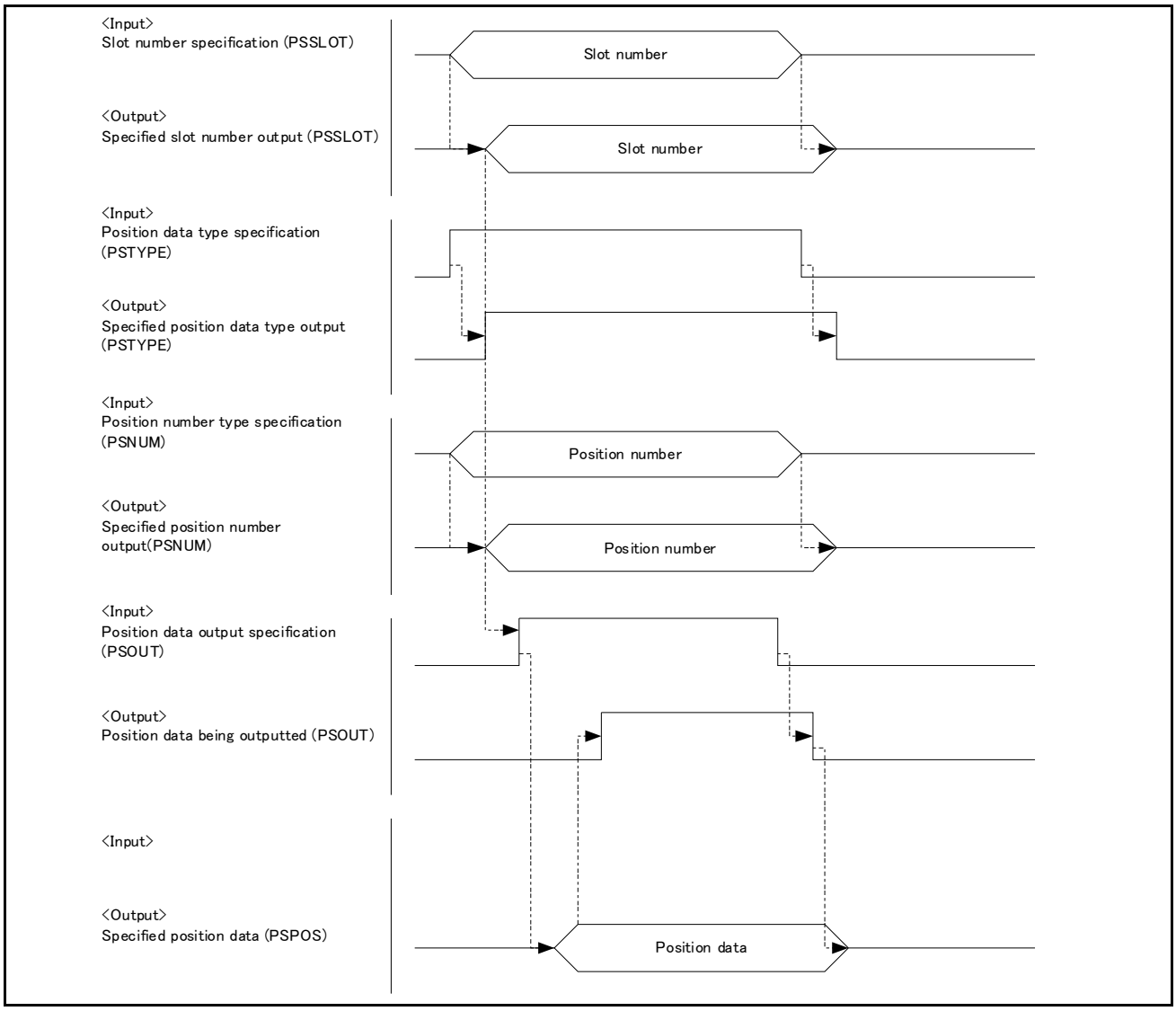


Fig.6-9:Example of external operation timing chart (Part 5)

[Notes]

(*1) If 320 points' worth of signals, from the signal number specified under the Parameter "PSPOS", do not exist, Error 7081 (unwritable as the parameter value falls outside the prescribed range) occurs.

(*2) If the range of signal number specified under the Parameter "PSSLOT" is greater than 6 bits, Error 7081 (unwritable as the parameter value falls outside the prescribed range) occurs.

(*3) If the range of position number specified under the Parameter "PSNUM" is greater than 16 bits, Error 7081 (unwritable as the parameter value falls outside the prescribed range) occurs.

(*4) If slot number, position data type or position number is changed in the processing of inputting position data output specification (PSOUT), relevant command is not accepted. Turn the position data output specification (PSOUT) input off and then back on. To determine which position is subject to data output, check slot number output (PSSLOT), position data type output (PSTYPE), and position number output (PSNUM).

(*5) If required program has not been loaded into the specified slot, "0x7FFFFFFF" is outputted for each of axes associated with specified position data output (PSPOS).

(*6) If a specified position does not exist, "0x80000000" is outputted for each of axes associated with specified position data output (PSPOS).

(*7) If, in the process of outputting position data, switching takes place in regard to the program being executed in the specified slot (CallP command, XRun command, or Parameter "PRGSEL"), "0x80000000" is outputted for each of axes associated with specified position data output (PSPOS).

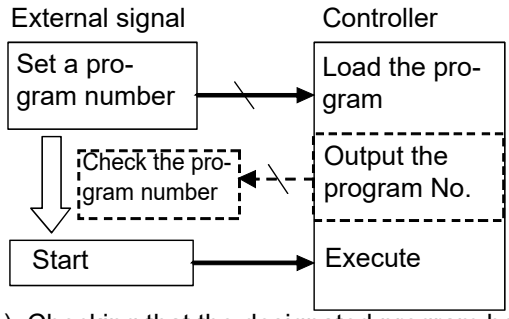
6.6 How to select and run a program using external signals

6.6.1 Methods

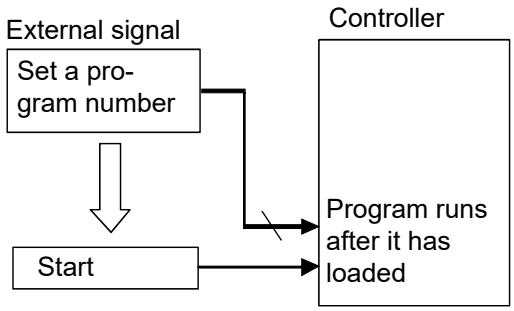
The following two methods are available for selecting and running a program using external signals.

(a) Running a designated program after checking that the program has been loaded

(b) Running a designated program upon input of the start signal



(a): Checking that the designated program has been loaded



(b): Designating a program upon input of the start signal

6.6.2 Selecting a method to run a program

To run a program using external signals, select either of the above methods (a) or (b) by setting the parameter described in Table 6-8.

Table 6-8:Parameter for selecting a method to run a program

Parameter name	Outline of functions
PST	This parameter switches program selection method. 0: Run a designated program after checking that the program has been loaded 1: Run a designated program upon input of the start signal

6.6.3 Related I/O parameters

Table 6-9:Parameters of I/O signals

Parameter name	Class	Outline of functions
IOENA	Input	This parameter enables/disables operation via the control of external signals.
PRGOUT	Input	This parameter outputs the program number for task slot 1 to the numerical output (IODATA).
IODATA	Input/Output	This parameter sets numerical inputs and outputs. This parameter configures settings so that the bit status in the set range can be read as a binary value.
PRGSEL	Input	This parameter designates the setting value for the program No. with numeric value input signals.
START	Input	This parameter runs the program.

6.6.4 Operation procedure

This section provides an example of how I/O signals are assigned based on the information in Table 6-10.

Table 6-10:I/O signal assignment

Parameter name	Input bit	Output bit
IOENA	5	3
PRGOUT	7	-
IODATA	8 to 11	8 to 11
PRGSEL	6	-
START	3	-

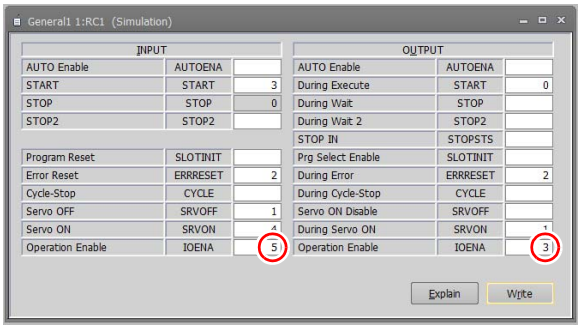
(1) Running a designated program after checking that the program has been loaded

1) Set "0" in parameter PST. (Initial setting)

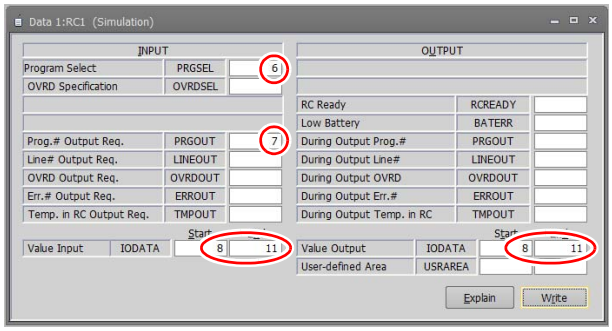
2) Assign I/O bits (values in Table 6-10) to parameters.

The following images are the setting windows for "Dedicated Input/Output Signals Assignment" in "Signal Parameter" of RT ToolBox3. Enter values in the fields marked with circles.

Dedicated Input/Output Signals Assignment: General 1 window

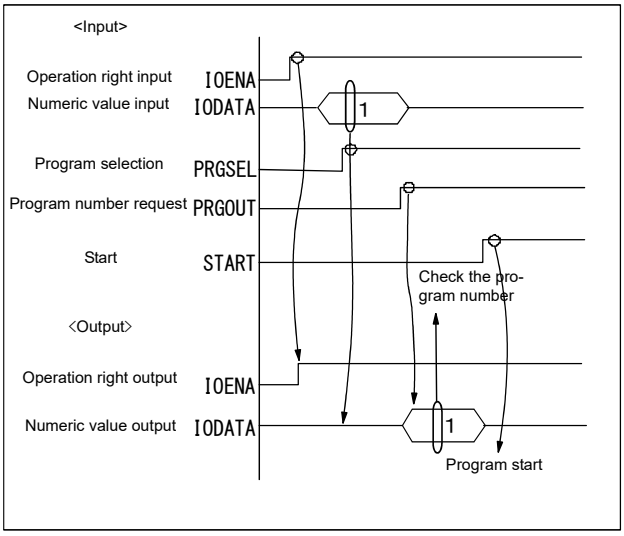


Dedicated Input/Output Signals Assignment: Data window



3) Program selection and run

1. Turns on input signal "IOENA" (bit 5) to obtain the operation right.
2. Checks that output signal "IOENA" (bit 3). If the signal is on, this means that the operation right has been acquired.
3. Sets the program number in binary values for IODATA of the input signal (bits 8 to 11).
4. Turns on PRGSEL (bit 6).
   The programs with the numbers set in IODATA of the input signal (bits 8 to 11) will be read.
5. Turns on PRGOUT (bit 7).
   The read program numbers will be output to IODATA of the output signal (bits 8 to 11). You can check whether the program number is correct.
6. Inputs START (bit 3) to run the program.



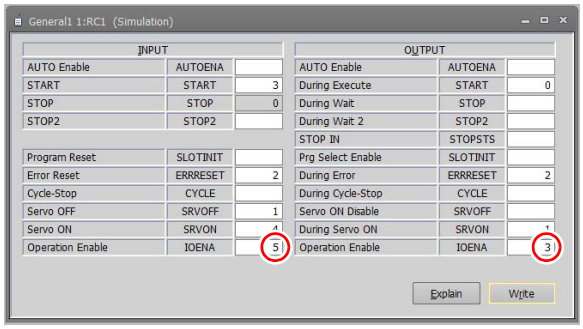
(2) Running a designated program upon input of the start signal

1) Set "1" in parameter PST. (Initial value: 0)

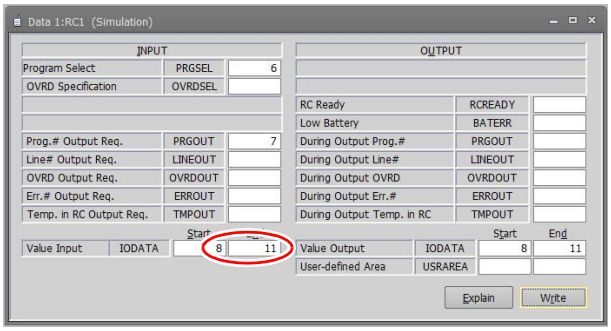
2) Assign I/O bits (values in Table 6-10) to parameters.

The following images are the setting windows for "Dedicated Input/Output Signals Assignment" in "Signal Parameter" of RT ToolBox3. Enter values in the fields marked with circles.

Dedicated Input/Output Signals Assignment: General 1 window

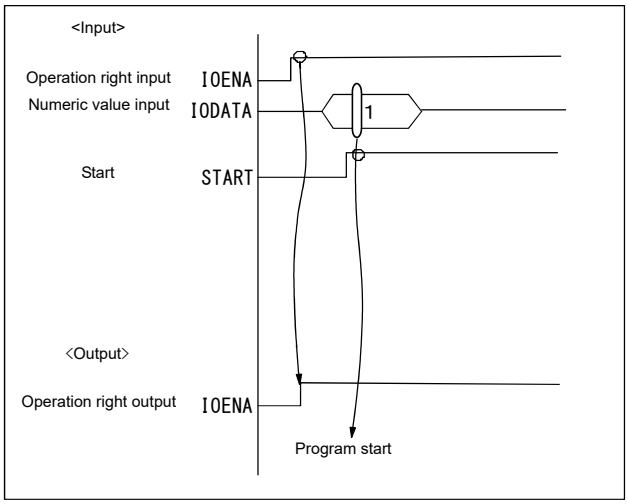


Dedicated Input/Output Signals Assignment: Data window



3) Program selection and run

1. Turns on input signal "IOENA" (bit 5) to obtain the operation right.
2. Checks that output signal "IOENA" (bit 3). If the signal is on, this means that the operation right has been acquired.
3. Sets the program number in binary values for IODATA of the input signal (bits 8 to 11).
4. Inputs START (bit 3) to run the program.



Note 1) Cycle the power of the controller after parameters have been changed.

Note 2) Operations such as program execution, servo on, and program initialization have been omitted.

Refer to Page 637, "6.3 Dedicated input/output" to Page 649, "6.4 Enable/disable status of signals".

6.7 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."

6.7.1 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.

6.8 Devices

6.8.1 Device list

The following tables list the robot devices accessible from GOTs and SLMP-compatible equipment. Device ranges are fixed and it cannot be changed.

(1) CR800-R series

Table 6-11:Compatible device list

Classification	Type	Device name	Symbol	Device range			Remarks
				Number of points	Setting range	Notation	
User device	Bit device	Input	X	4096 points (4k)	X0 to XFFF	Hexadecimal	
		Output	Y	4096 points (4k)	Y0 to YFFF	Hexadecimal	
		Internal relay	M	18432 points (18k)	M0 to M18431	Decimal	
	Word device	Data register	D	5120 points (5k)	D0 to D5119	Decimal	
System device	Bit device	Special relay	SM	4096 points	SM0 to SM4095	Decimal	
	Word device	Special register	SD	4096 points	SD0 to SD4095	Decimal	
CPU buffer memory access device	Word device	CPU buffer memory access device	U3En\G	524288 points	U3En\G0 to U3En\G524287	Decimal	
		CPU buffer memory access device (Periodical communication area)	U3En\HG	Maximum 12288 points (12k)	U3En\HG0 to U3En\HG12287	Decimal	

(2) CR800-D series

Table 6-12:Compatible device list

Classification	Type	Device name	Symbol	Device range			Remarks
				Number of points	Setting range	Notation	
User device	Bit device	Input	X	8192 points (8k)	X0 to X1FFF	Hexadecimal	
		Output	Y	8192 points (8k)	Y0 to Y1FFF	Hexadecimal	
	Word device	Data register	D	5120 points (5k)	D0 to D5119	Decimal	
System device	Bit device	Special relay	SM	4096 points	SM0 to SM4095	Decimal	
	Word device	Special register	SD	4096 points	SD0 to SD4095	Decimal	
CPU buffer memory access device	Word device	CPU buffer memory access device (Periodical communication area)	U3En\HG (n=0, 1) Note1)	2048 points (2k)	U3En\HG0 to U3En\HG2047 (n=0, 1)	Decimal	

Note1) U3E0/U3E1 can perform reads and writes. Only the compatible U3E1 device ranges have an R attribute when functions represented in "Table 6-15Device assignment" are enabled. Performing writes in such cases result in the written content being ignored.

(3) CR800-Q series

Table 6-13:Compatible device list

Classification	Type	Device name	Symbol	Device range			Remarks
				Number of points	Setting range	Notation	
User device	Bit device	Input	X	4096 points (4k)	X0 to XFFF	Hexadecimal	
		Output	Y	4096 points (4k)	Y0 to YFFF	Hexadecimal	
		Internal relay	M	18432 points (18k)	M0 to M18431	Decimal	
	Word device	Data register	D	5120 points (5k)	D0 to D5119	Decimal	
System device	Bit device	Special relay	SM	2048 points	SM0 to SM2047	Decimal	
	Word device	Special register	SD	2048 points	SD0 to SD2047	Decimal	
CPU shared memory device	Word device	Multiple CPU high-speed communication area	U3En\G	Maximum 14336 points (14k)	U3En\G10000 to U3En\G24335	Decimal	

6.8.2 Device assignment

The following table shows the robot functions and corresponding assignment to I/O signal devices. Device ranges not assigned to functions can be read and written freely.

(1) CR800-R series

Table 6-14:Device assignment

Function		Devices seen from external equipment Note1)	R/W attribute Note2)	Robot I/O signals	Number of points	Remarks
Direct control of the PLC input/output unit		X0 to XFFF	R	Input of 0 to 255	Maximum 256 points	Functions are enabled when an input/output unit is mounted and corresponding functions are enabled with the QXYREAD parameter.
		Y0 to YFFF	R	Output of 0 to 255	Maximum 256 points	
Hand		X384 to X38B	R	Input of 900 to 907	8 points/ mechanism	Always enabled.
PLC link I/O function		U3E0\HG0 to U3E0\HG511	R/W	Input of 10000 to 18191	8192 points	Always enabled.
		U3En\HG0 to U3En\HG511 (n=1, 2, 3)	R	Output of 10000 to 18191	8192 points	
CPU buffer memory extension function		U3E0\HG512 to U3E0\HG1023	R/W	-	-	This function is enabled with the IQMEM parameter.
		U3En\HG512 to U3En\HG1023 (n=1, 2, 3)	R	-	-	
PLC device assignment function		D4096 to D5119	R/W or R Note3)	-	-	This function is enabled with the parameter DDEVVL*.
Special	Relay	SM0 to SM4095	R/W	-	-	Always enabled.
	Register	SD0 to SD4095	R/W	-	-	

Note1) These ranges of devices can be used freely in that write and read operations can be performed even on ranges of devices not assigned specific functions or when the extension function is disabled.

Note2) Indicates whether attributes are readable/writable when functions are enabled. Attributes are readable/writable when functions are disabled.

Note3) Depending on the assigned variable, attributes are either readable and writable or read-only. (When program external variables are used, attributes are readable and writable. When status variables are used, attributes are read-only.)

(2) CR800-D series

Table 6-15:Device assignment

Function	Devices seen from external equipment Note1)	R/W attribute Note2)	Robot I/O signals	Number of points	Remarks
Parallel input/output unit	X0 to XFF	R	Input of 0 to 255	256 points	The function is enabled when corresponding equipment is connected.
	Y0 to YFF	R	Output of 0 to 255	256 points	
Parallel input/output interface	X0 to X3F	R	Input of 0 to 63	64 points	
	Y0 to Y3F	R	Output of 0 to 63	64 points	
GOT link	X0 to XFF	R/W Note3)	Input of 0 to 255	256 points	
	Y0 to YFF	R	Output of 0 to 255	256 points	

Function		Devices seen from external equipment Note1)	R/W attribute Note2)	Robot I/O signals	Number of points	Remarks
STOP (SKIP0)		X320	R	Input of 800	1 point	Always enabled.
SKIP (1-7)		X321 to X327	R	Input of 801 to 807	7 points (2 points for hardware)	
Hand		X384 to X3B3	R	Input of 9*0-9*7 (*=0-4)	8 points/ mechanism	
PROFIBUS		X7D0 to X13CF	R	Input of 2000 to 5071	Maximum 3072 points	The function is enabled when an option card is connected.
		Y7D0 to Y13CF	R	Output of 2000 to 5071	Maximum 3072 points	
CC-Link		Y1770 to Y1F6F	R	Remote input of 6000 to 8047	Maximum 2048 points	
		X1770 to X1F6F	R	Remote output of 6000 to 8047	Maximum 2048 points	
		D0 to D255	R	Remote register input of 6000 to 6255	Maximum 128 words	
		D256 to D511	R	Remote register output of 6000 to 6255	Maximum 128 words	
CC-Link IEF		Y1770 to Y1F6F	R	Remote input of 6000 to 8047	Maximum 2048 points	
		X1770 to X1F6F	R	Remote output of 6000 to 8047	Maximum 2048 points	
		D0 to D1023	R	Remote register input of 6000 to 7023	Maximum 768 words	
		D1024 to D2047	R	Remote register output of 6000 to 7023	Maximum 768 words	
PLC link I/O function		U3E0\HG0 to U3E0\HG511	R/W	Input of 10000 to 18191	8192 points	Always enabled.
		U3E1\HG0 to U3E1\HG511	R	Output of 10000 to 18191	8192 points	
CPU buffer memory extension function		U3E0\HG512 to U3E0\HG1023	R/W	-	-	This function is enabled with the IQMEM parameter.
		U3E1\HG512 to U3E1\HG1023	R	-	-	
PLC device assignment function		D4096 to D5119	R/W or R Note4)	-	-	This function is enabled with the parameter DDEVVL*.
Special	Relay	SM0 to SM4095	R/W	-	-	Always enabled.
	Register	SD0 to SD4095	R/W	-	-	

Note1) These ranges of devices can be used freely in that write and read operations can be performed even on ranges of devices not assigned specific functions or when the extension function is disabled.

Note2) Indicates the R/W attribute when each function is enabled. When the function is disabled, both of reading and writing are possible.

Note3) Devices must not be in use by the parallel input/output interface and parallel input/output unit to be writable with robot instructions by external equipment.

Note4) Depending on the assigned variable, attributes are either readable and writable or read-only. (When program external variables are used, attributes are readable and writable. When status variables are used, attributes are read-only.)

(3) CR800-Q series

Table 6-16:Device assignment

Function		Devices seen from external equipment Note1)	R/W attribute Note2)	Robot I/O signals	Number of points	Remarks
Direct control of the PLC input/output unit		X0 to XFFF	R	Input of 0 to 255	Maximum 256 points	Functions are enabled when an input/output unit is mounted and corresponding functions are enabled with the QXYREAD parameter.
		Y0 to YFFF	R	Output of 0 to 255	Maximum 256 points	
Hand		X384 to X38B	R	Input of 900 to 907	8 points/ mechanism	Always enabled.
PLC link I/O function		U3E0¥G10000 to U3E0¥G10511 (For PLC No. 2)	R/W	Input of 10000 to 18191	8192 points	Always enabled.
		U3En\G10000 to U3En\G10511 (n=1, 2, 3)	R	Output of 10000 to 18191	8192 points	
Shared memory extension function		U3E0\HG10512 to U3E0\HG11023 (For PLC No. 2)	R/W	-	-	This function is enabled with the parameter IQMEM.
		U3En\HG10512 to U3En\HG11023 (n=1, 2, 3)	R	-	-	
PLC device assignment function		D4096 to D5119	R/W or R Note3)	-	-	This function is enabled with the parameter DDEVVL*.
Special	Relay	SM0 to SM2047	R/W	-	-	Always enabled.
	Register	SD0 to SD2047	R/W	-	-	

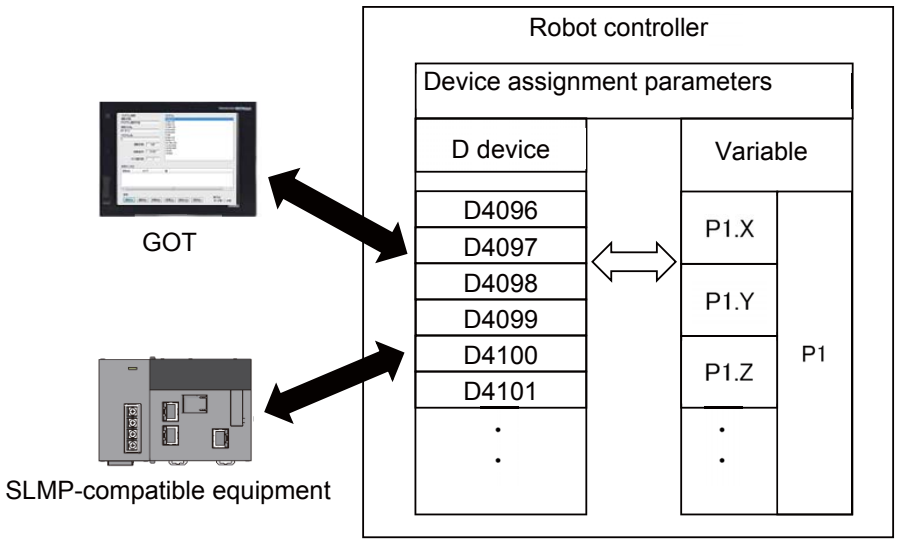
Note1) These ranges of devices can be used freely in that write and read operations can be performed even on ranges of devices not assigned specific functions or when the extension function is disabled.

Note2) Indicates whether attributes are readable/writable when functions are enabled.

Note3) Depending on the assigned variable, attributes are either readable and writable or read-only. (When program external variables are used, attributes are readable and writable. When status variables are used, attributes are read-only.)

6.8.3 PLC device assignment function

Directly assigning program external variables and state variables to devices enables robot information to be collected from GOTs and other FA equipment through such devices.



(1) Compatible device

Assignable devices are described as follows.

Table 6-17:Compatible device

Type	Device name	Symbol	Device range			Remarks
			Number of points	Setting range	Notation	
Word device	Data register	D	1024 points (1k)	D4096 to D5119	Decimal	2 bytes*1024 points = 2048 bytes

(2) Assignment to a robot variable device

Assignable robot variables are described as follows.

Table 6-18:Assignable variable

Classification	Type	Data type	Remarks
Variable	Program external variables	Position type	Each data type supports one-dimensional arrays. * For details on program external variables, refer to Page 132, "4.3 Detailed specifications of MELFA-BASIC VI".
		Joint type	
		Numeric value type (double-precision real number type only)	
		Character string type	
	State variables	Position type	* For details on status variables, refer to Page 157, "4.5 Robot status variables" and Page 346, "4.13 Detailed explanation of Robot Status Variable". Assignment of status variables with signal input/output functions (such as M_In() and M_Out()) is unavailable.
		Joint type	
		Numeric value type	
		Character string type	

Variables and devices are assigned with the following parameters.

Parameter	Parameter name	Number of arrays Character string	Description	Default setting
Assignment of robot variables to D device	DDEVVL1 to DDEVVL32	Integer 4, Character string 1, Integer 5	Assigns robot variables to D device. Up to 32 variables can be assigned. Element 1: Variable type 0: Not set 1: Program external variables 2: State variables Element 2: Data type 0:Integer type 1: Long-precision integer type 2: Single-precision real number type 3: Double-precision real number type 4: Position data type 5: Joint data type 6: Workpiece coordinate type 7: Character string type Element 3: (Reserved) Element 4: Used to assign data types that include real numbers. (Ignores all other data types stated in Element 2 except 2 to 6.) 0: Variables of single-precision real number type (including position, joint, and workpiece coordinate types) are multiplied by 10^4 and then assigned as long-precision integer type. Variables of double-precision real number type are multiplied by 10^8 and then assigned as 64-bit integers (twice the length of long-precision integers). 1: Variables are assigned as they are. Element 5: Variable name (Ex.) M_100, J_Fbc, or similar Element 6: Argument1 * For external variables: Array elements/For status variables: Argument 1 Setting range: Blank (when omitted) or from 0 Element 7: Argument 2(Only for state variables) Setting range: Blank (when omitted) or from 0 Element 8: (Reserved) Element 9: Number of bytes of a character string variable (Ignores all other data types stated in Element 2 except 7.) * Specify the sum of 1 (area where a 1-byte character string is saved) and the maximum number of bytes of a character string variable to be assigned. Setting range: 1 to 240 Element 10: Start D device number Setting range: 4096 to 5119 (Ex.) Specify 4096 for D4096	0,0,0,0, , , , ,1,4096

*Example of assignment to numeric variable
(1)Integer
[Example]M_OPOvrd(Integers)
When assigned to D4096

Device	Description	Updating cycle
D4096	M_OPOvrd value [Integer (signed)]	Control cycle

[Example]M_ErCode(Long-precision integer)
When assigned to D4096

Device	Description	Updating cycle
D4096	M_ErCode value [Long-precision integer (signed)]	Control cycle
D4097		

(2)Real number
[Example]M_Timer(1)(Single-precision real number)
When assigned to D4500

Device	Description	Updating cycle
D4500	M_Timer(1) value [Long-precision integer (signed) multiplied by 10^4] or [Single-precision real number]	Control cycle
D4501		

[Example]M_100(1)(Double-precision real number)
When assigned to D4500

Device	Description	Updating cycle
D4500	M_100(1) value [64-bit integer (signed) multiplied by 10^8] or [Double-precision real number]	Control cycle
D4501		
D4502		
D4503		

*Example of assignment to position variable/workpiece coordinate
When position type/workpiece coordinate type variable is assigned to D5000

Device	Description	Updating cycle
D5000	X coordinate value [Long-precision integer (signed) multiplied by 10^4]	Control cycle
D5001		
D5002	Y coordinate value [Long-precision integer (signed) multiplied by 10^4]	
D5003		
D5004	Z coordinate value [Long-precision integer (signed) multiplied by 10^4]	
D5005		
D5006	A coordinate value [Long-precision integer (signed) multiplied by 10^4]	
D5007		
D5008	B coordinate value [Long-precision integer (signed) multiplied by 10^4]	
D5009		
D5010	C coordinate value [Long-precision integer (signed) multiplied by 10^4]	
D5011		
D5012	L1 coordinate value [Long-precision integer (signed) multiplied by 10^4]	
D5013		
D5014	L2 coordinate value [Long-precision integer (signed) multiplied by 10^4]	
D5015		
D5016	Structure flag Note1) [Long-precision integer (signed)]	
D5017		
D5018	Multiple rotation data [Long-precision integer (signed)]	
D5019		
D5020	Mechanism number(MELFA-BASIC VI only) [Long-precision integer (signed)]	
D5021		

Note1) Only the lower one word is used for the structure flag/mechanism number. The upper one word is reserved.

*Example of assignment to joint variable
When a joint type variable is assigned to D5000

Device	Description	Updating cycle
D5000	J1 coordinate value [Long-precision integer multiplied by 10^4]	Control cycle
D5001		
D5002	J2 coordinate value [Long-precision integer multiplied by 10^4]	
D5003		
D5004	J3 coordinate value [Long-precision integer multiplied by 10^4]	
D5005		
D5006	J4 coordinate value [Long-precision integer multiplied by 10^4]	
D5007		
D5008	J5 coordinate value [Long-precision integer multiplied by 10^4]	
D5009		
D5010	J6 coordinate value [Long-precision integer multiplied by 10^4]	
D5011		
D5012	J7 coordinate value [Long-precision integer multiplied by 10^4]	
D5013		
D5014	J8 coordinate value [Long-precision integer multiplied by 10^4]	
D5015		
D5016	Mechanism number(MELFA-BASIC VI only) [Long-precision integer (signed)] Note1)	
D5017		

Note1) Only the lower one word is used for the mechanism number. The upper one word is reserved.

*Example of assignment to character string
When a 7-byte character string type variable is assigned to D5100

Device	Description	Updating cycle
D5100	Character string length [Lower 8 bits], 1st byte of character code [Higher 8 bits]	Control cycle
D5101	2nd byte of character code [Lower 8 bits], 3rd byte of character code [Higher 8 bits]	
D5102	4th byte of character code [Lower 8 bits], 5th byte of character code [Higher 8 bits]	
D5103	6th byte of character code [Lower 8 bits]	

7 Appendix

7.1 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

R is center of flange in comparison with the plane through the J1 axis vertical to the ground. (5-axis type robot)
P is center of J5 axis rotation in comparison with the plane through the J1 axis vertical to the ground. (6-axis type robot)

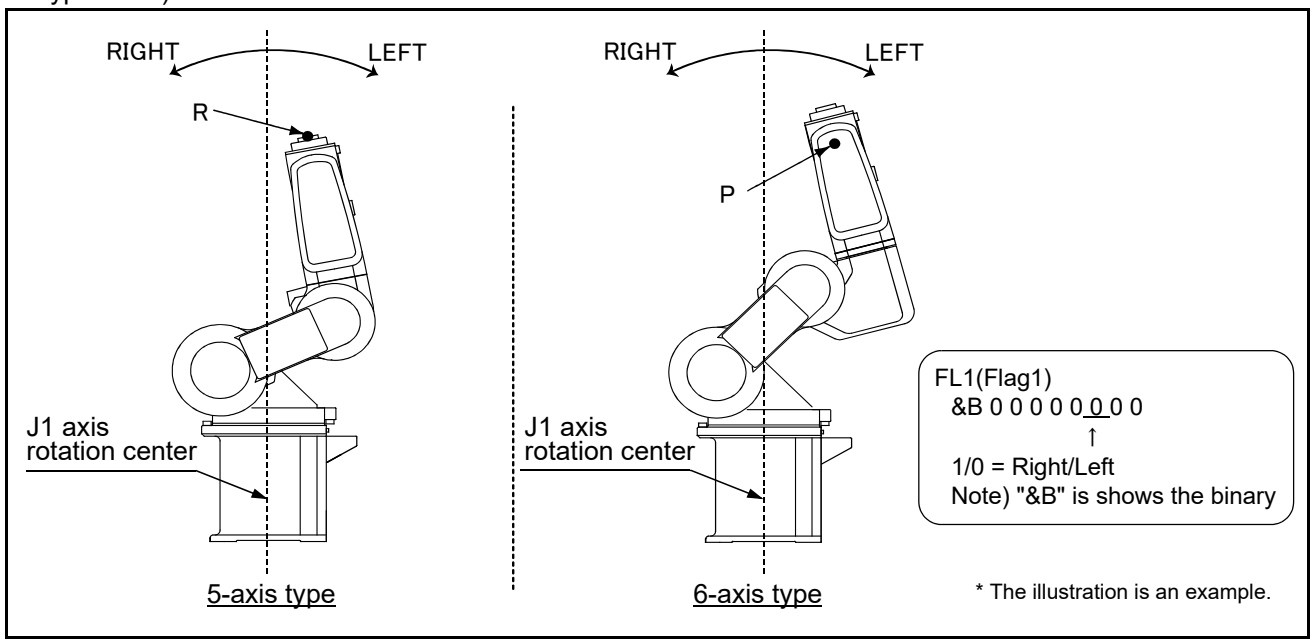


Fig.7-1:Configuration flag (Right/Left)

(2) ABOVE/BELOW

P is center of J5 axis rotation in comparison with the plane through both the J3 and the J2 axis.

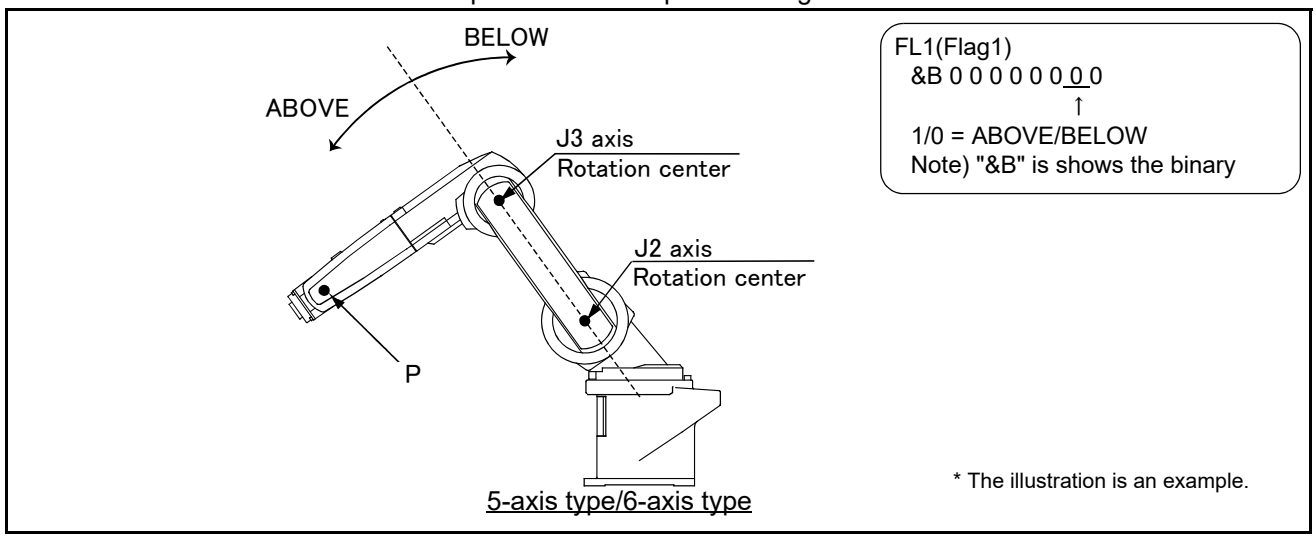


Fig.7-2:Configuration flag (ABOVE/BELOW)

(3) NONFLIP/FLIP (6-axis type robot only.)

This means in which side the J6 axis is in comparison with the plane through both the J4 and the J5 axis.

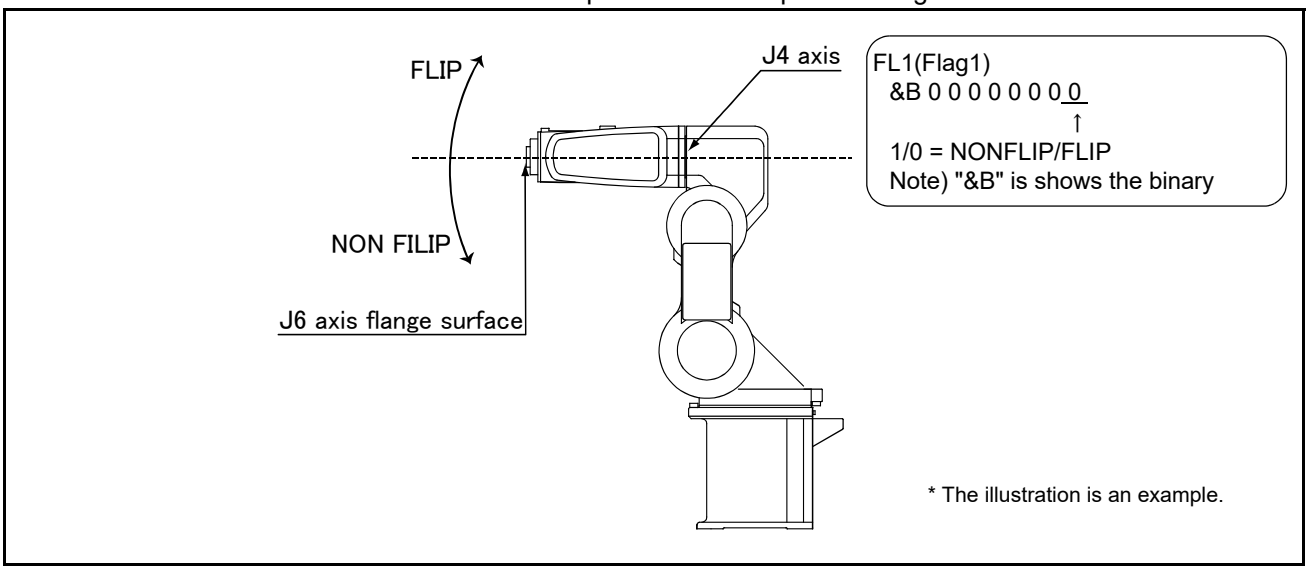


Fig.7-3 : Configuration flag (NONFLIP/FLIP)

*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.



FL1(Flag1)
&B 0 0 0 0 0 0 0 0
↑
1/0 = Right/Left
Note) "&B" is shows the binary

* The illustration is an example.

Fig.7-4:Configuration flag (Right/Left)

7.2 Spline interpolation

Spline interpolation, one of the robot movement commands, is explained in this section.

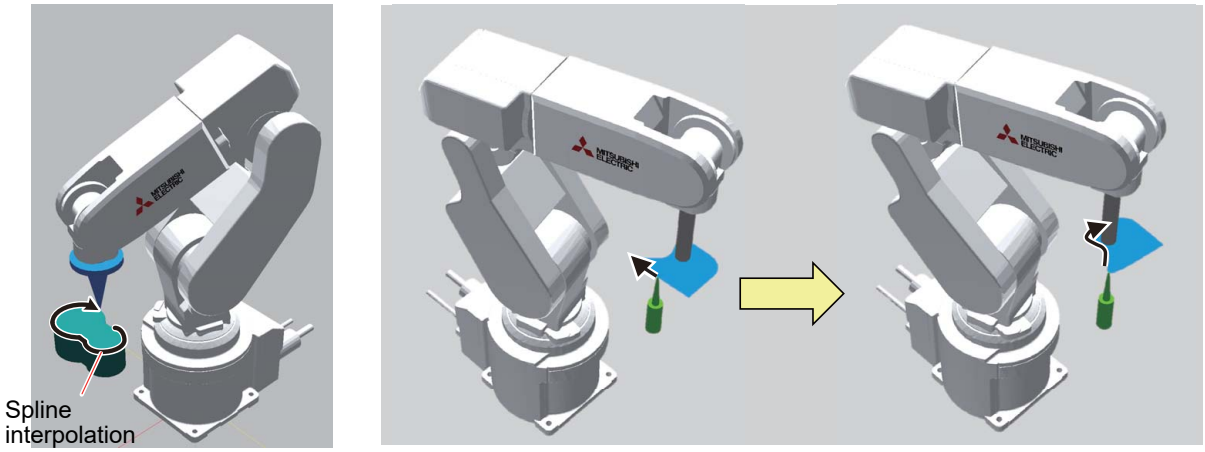
7.2.1 Outline

(1) Outline

Spline interpolation is a function that moves the robot at the designated speed along a spline curve that smoothly connects designated path points.

Ex-T spline interpolation is a function used to move the spline curve that smoothly links path points specified on the workpiece grasped by the robot along an arbitrary coordinate system origin (Ex-T coordinate system origin) at a specified speed (Ex-T coordinate system origin viewed from workpiece moves relatively at specified speed).

The robot can be moved along a curved path that was not possible with conventional linear or circular interpolation. This interpolation can be used for sealing, polishing and chamfering processes, etc.



(a) Spline interpolation (b) Ex-T spline interpolation

Fig.7-5: Outline of spline interpolation

(2) Features

- A smooth spline curve is generated between each path point so that the robot positions and posture designated as path points are passed. The robot moves along that curve at the speed designated with linear speed.

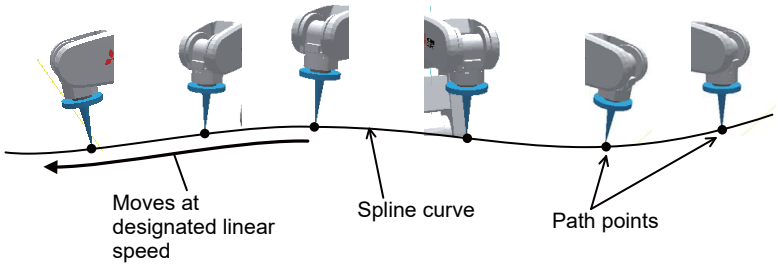


Fig.7-6:Spline interpolation

• A file dedicated for spline interpolation (spline file) in which the path point data is registered is used separately from the robot program. This file is created and edited in the RT ToolBox3 dedicated edit screen (Spline File Edit screen), and written to the controller.

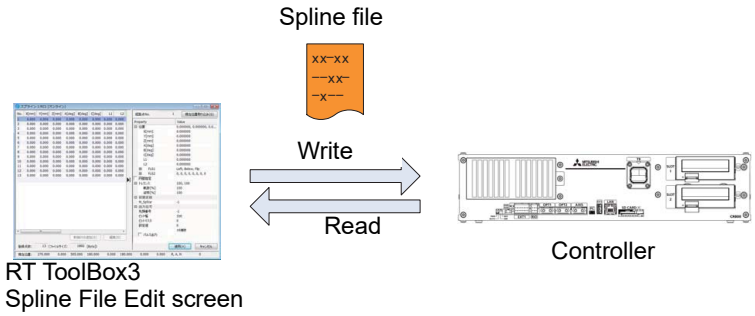


Fig.7-7:Creating dedicated file for path point data

• By creating an independently dedicated file, the spline interpolation can be shared by multiple robot programs, and the spline interpolation can be switched easily.

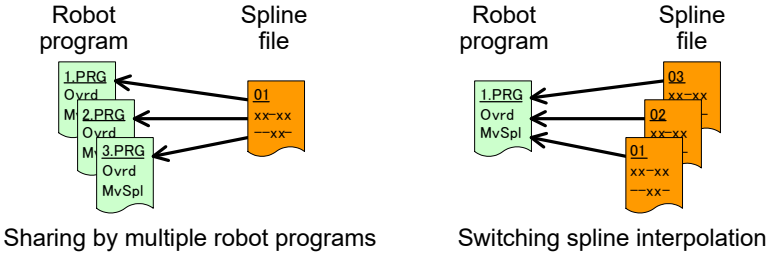


Fig.7-8:Features of dedicated file

• The swell of the spline curve can be partially compensated, and changed to a straight or arc path.
• In addition to the robot positions, output signals or random values can be registered as the path point data.

(3) Required devices and software version

RT ToolBox3 is required to use spline interpolation. Spline interpolation can be used with devices having the software version shown in Table 7-1.

Table 7-1:Software versions compatible with spline interpolation

Device	Function	Controller	RT ToolBox3
Compatible software version	Spline interpolation	Ver. A1 or later	Ver. 1.00 or later
	Ex-T spline interpolation	Ver. A1 or later	Ver. 1.00 or later

There is no designated software version for the teaching pendant (R32TB/R56TB).

(4) Terminology

The terminology used when explaining spline interpolation is shown in Table 7-2.

Table 7-2:Explanation of terminology

Terminology	Explanation
Path point	This is the robot position data (Cartesian coordinate value) used to generate the spline curve.
Block	Refers to the curve line or segment generated between two adjacent path points.
Path	Refers to the spline curve generated by the spline interpolation command and passes through path points.
Start position	Refers to the path point where spline interpolation movement starts.

Terminology	Explanation
End position	Refers to the path point where spline interpolation movement ends.
Spline file	File containing the path points and setting values required for execution, etc. One file corresponds to one spline interpolation.
Spline File Edit screen	RT ToolBox3 screen dedicated for creating, editing and saving spline file.
DXF File Import screen Note1)	RT ToolBox3 screen dedicated for creating, editing and saving spline file from a DXF file.
Ex-T coordinate system origin	This is the coordinate system origin that is the subject of arbitrarily defined control outside the robot.

Note1) Not supported with the RT ToolBox3 mini version.

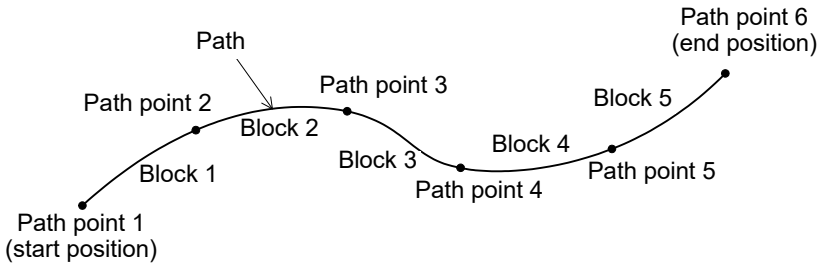


Fig.7-9:Spline interpolation terminology

7.2.2 Specifications

(1) Basic specifications

Table 7-3:Basic specifications

Item		Specifications
Compatible robot		Vertical articulated (6-axis) robot, horizontal articulated (4-axis) robot * Not supported with vertical articulated (5-axis) robot or user mechanism.
Compatible robot language		Spline interpolation commands, functions, and status variables have been added to MELFA-BASIC VI. (MvSpl command, EMvSpl command, SetCalFrm command, SplPos function, SplSpd function, SplECord function, status variables M_SplPno, M_SplVar)
Moving method		The spline file is registered into the controller, and the spline interpolation command is executed. The spline curve is generated to pass through the path points registered in the spline file. The robot moves along this curve.
Spline file	No. of registered files	Maximum 99 files * The number of files that can actually be registered will depend on the file size and amount of open storage area.
	No. of registered path points	Maximum 5000 points per file * The file size is approx. 470 KB with 5000 points.
	Create/edit/save	The RT ToolBox3 Spline File Edit screen is used to create, edit and save the file. SplWrt command or SplFWrt command are used to create, edit and save the file.
	Path point registration method	The path points are registered with the following operations on the RT ToolBox3 Spline File Edit screen. Teaching operation, import of MDI and CSV files, import of MXT files, import of DXF files, robot program (SplWrt command)
Path adjustment	Tolerance designation	The degree of curve swelling can be designated in block units. Random blocks can be changed from spline curves to linear paths.
	Arc designation	Three consecutive points are designated as the path point to create an arc path instead of a spline curve. * Arc designation method is different depending on the version of the spline file.
	Spline cancel	The spline curve is split at a path point where change in moving direction is large.
	Block length ratio	If the segment length is larger than the specified ratio compared to the previous and next blocks, that block is automatically handled as a linear block.

Item		Specifications
Operation mode	Constant linear speed	The interpolation command is generated so the robot moves uniformly at the designated speed.
	Variable linear speed	If the speed may be exceeded during movement because the designated speed is large or the change in posture is large, etc., the speed is automatically lowered.
Signal output		A general-purpose signal can be output when passing through the path points.
Numerical setting		A random value can be set for each path point, and that value referred to with the status variable.

(2) Restrictions

Table 7-4:Restrictions

Item	Details of restrictions
Additional axis	Spline interpolation does not support additional axes. Even if coordinate values for the additional axis are set at each path point, the robot will not start from the start position coordinate values during spline interpolation.
Cnt command	Even if the interpolation continuous movement is designated with the Cnt command, this does not function for the spline interpolation start position and end position.
Oadl command	Even if optimum acceleration/deceleration control is enabled with the Oadl command, it does not function for spline interpolation.
MvTune command	Operation mode designation with the MvTune command does not function for spline interpolation.
FsGChg command	Spline interpolation does not support the <Changeover start position> target designated with the FsGChg command (designates control characteristics changeover for force sensor control).
Tool/base changeover	The data cannot be switched between tool conversion data and base conversion data while spline interpolation execution is halted.
Jrc command	The Jrc command cannot be executed while the spline interpolation execution is halted.
Interpolation movement related status variables	The following status variables related to interpolation movement will not return the spline interpolation status. Status variable / Function M_Acl: Returns the current acceleration time ratio. M_DAcl: Returns the current deceleration time ratio. M_Ratio: Returns the attainment ratio to the target position. M_RDst: Returns the remaining distance to the target position. M_Spd: Returns the currently set speed.
Interrupt process	Even if "Stop type 1" is designated as the interrupt stop type, the same stopping as "Stop type 2" will take when interrupting during spline interpolation movement.
Tracking function	The tracking function and spline interpolation cannot be used simultaneously.

(3) Robot behavior during spline interpolation

• If the robot's current position and spline interpolation start position differ and spline interpolation is executed, the robot will move with linear interpolation to the start position and then will start spline interpolation. During linear interpolation, the robot moves with the speed designated with Ovrd command, Spd command and override.
Even with the additional axis, if the current position and start position differ, the robot will move to the start position.

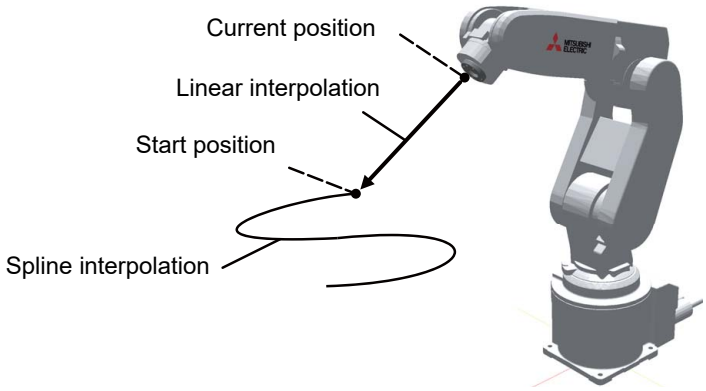


Fig.7-10:When current position and start position differ (Spline interpolation)

• If the robot's current position and Ex-T spline interpolation start position differ and Ex-T spline interpolation is executed, the robot will move with Ex-T linear interpolation to the start position and then will start Ex-T spline interpolation. During Ex-T linear interpolation, the robot moves with the speed designated with Ovrd command, Spd command and override.
Even with the additional axis, if the current position and start position differ, the robot will move to the start position.

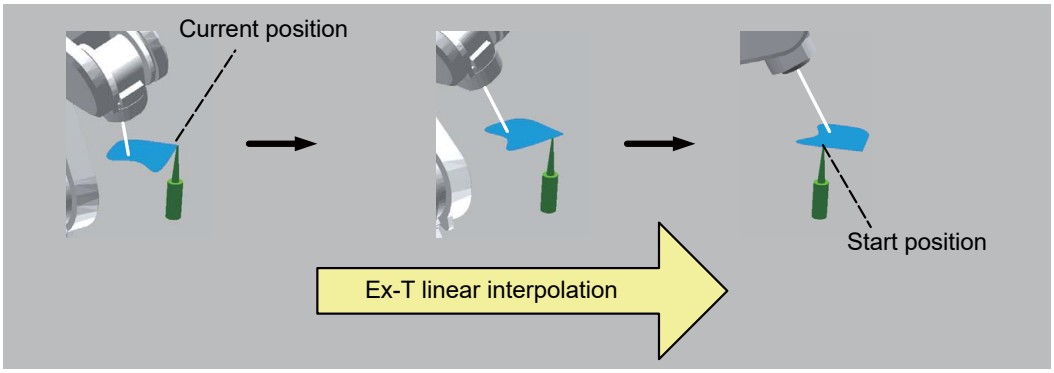


Fig.7-11:When current position and start position differ (Ex-T spline interpolation)

• Operation during spline interpolation can be halted with a stop input, etc. When operation is resumed, the remaining spline interpolation is continued from the halted position.
• If the robot or additional axis is moved with jog operation after the spline interpolation has been halted with a stop input, etc., the robot will return to the position where spline interpolation was halted before continuing

operation. The robot will return to the halted position with joint interpolation. The parameter RETPATH (setting for automatic return after jog feed during halt) setting is not applied.

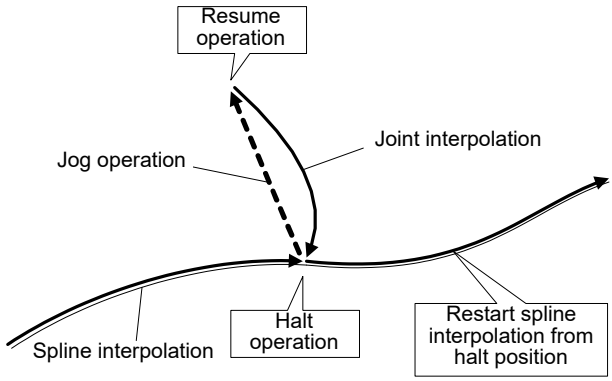


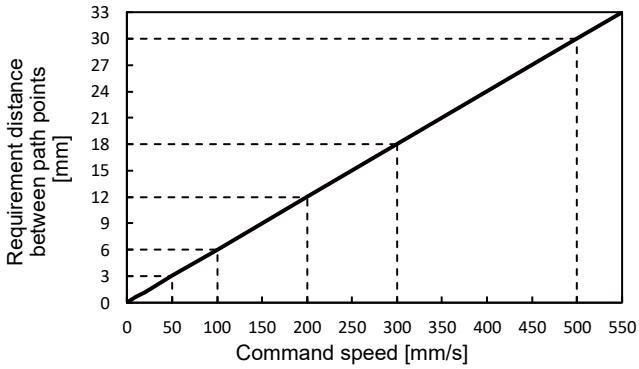
Fig.7-12:Behavior during halt and resume

• After operation is halted during spline interpolation with a stop input, etc., if the halted robot program is read into the teaching pendant or RT ToolBox3 program edit screen, the spline interpolation halt position information will be reset. Thus, if operation is resumed from the halted line, the robot will move to the spline interpolation start position before resuming operation.

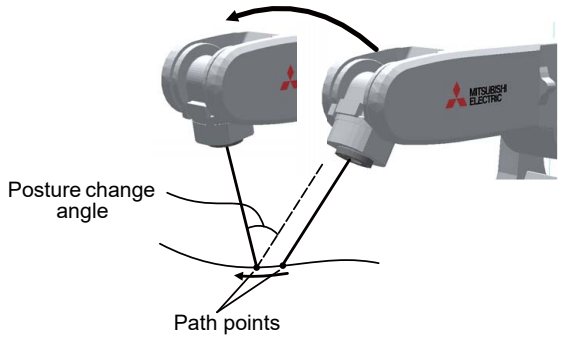
(4) Check related to path points

The check listed in Table 7-5 is preformed on each path point.

Table 7-5:Check related to path points

<table>
<tr><th>Type</th><th>Explanation</th></tr>
<tr><td>Distance between path points and command speed</td><td>The program is checked to confirm there is adequate distance between two adjacent path points. The appropriate distance differs according to the designated speed, and if two path points are too close for the speed, error L2611 (path points too close) will occur.
Refer to the following table and graph, and determine the command speed and distance between path points.
<table>
<tr><th>Command speed [mm/s]</th><th>Required distance between path points</th></tr>
<tr><td>10</td><td>0.6 mm or more</td></tr>
<tr><td>20</td><td>1.2 mm or more</td></tr>
<tr><td>50</td><td>3.0 mm or more</td></tr>
<tr><td>100</td><td>6.0 mm or more</td></tr>
<tr><td>200</td><td>12.0 mm or more</td></tr>
<tr><td>500</td><td>30.0 mm or more</td></tr>
</table>
The command speed in the table can be obtained with the following formula.
Command speed = (<Speed> of MvSpl command) × (Ovrd command)
 

* The high speed spline interpolation command has been added to the software version A3a. With this function enabled, a command speed higher than what was previously possible can be specified for the distance between the respective path points. Refer to Page 747, "7.2.10 High speed spline interpolation command" for further information.</td></tr>
</table>

Type	Explanation
Posture variation amount	The program is checked to confirm that the posture change between two adjacent path points is not too large. If the posture change angle is larger than 150 degrees, error L2611 (posture change too large) will occur. * In Ex-T spline interpolation, the program is checked to confirm that the posture change of path points on the workpiece is not too large. 
Configuration flag value	The program is checked to confirm that the configuration flag values for two adjacent path points are the same. If the configuration flag values are different, error L2611 (path point configuration flags differ) will occur.

7.2.3 Explanation of functions

(1) Path adjustment

The spline curve's shape can be compensated by using the path adjustment function. "Tolerance designation", "arc designation", "spline cancel" and "block length ratio" are available for compensating the path.

■Tolerance designation

Designate the tolerance[*1)] to adjust the degree of spline curve swelling in block units. The setting range is 0 to 100%, and the default value is 100% (No adjustment). The smaller the setting value is, the smaller the degree of curve swelling is. When 0% is set, the target block will be linear.
The tolerance can also be designated for the posture. The smaller the setting value is, the smaller the degree of swelling in posture change is. However, the smoothness in posture change will drop when passing through the path points.

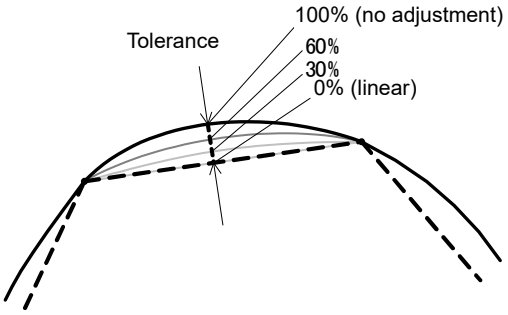
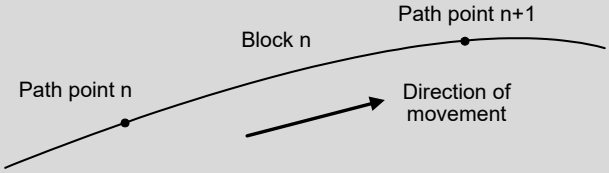


Fig.7-13:Tolerance

Blocks subject to tolerance

The tolerance is set as the path point data. The tolerance set for the path point on the start position side of each block is reflected onto that block. In the following example, the tolerance set for the path point n is applied on block n.



■Arc designation

Arc designation method is different depending on the version of the spline file.

1) Spline file version 02

When two consecutive blocks are designated as an arc (arc designation at start point of a block), the curve that consist of those two blocks will be an arc instead of a spline. If this is applied at a section requiring an arc-shaped curve, such as at a corner, an accurate arc can be drawn with fewer path points than the spline curve.

*1) Tolerance typically refers to the tolerable error and accuracy. However, in this function it refers to the degree of spline curve swelling in respect to the lines (segments) that connect the path points.

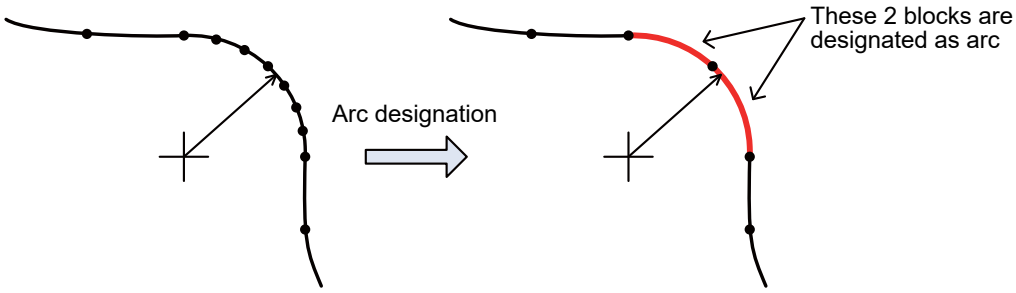


Fig.7-14:Arc designation (Spline file version 02)

With arc designation, two consecutive blocks must always be designated. If only one block is designated, the error L2613 (not enough points designated for arc) will occur.
Furthermore, if the three points used to form an arc by specifying two blocks are aligned in a straight line and arc generation is not possible, error L2613 (block data calculation error (Cir.Arc)) will occur.

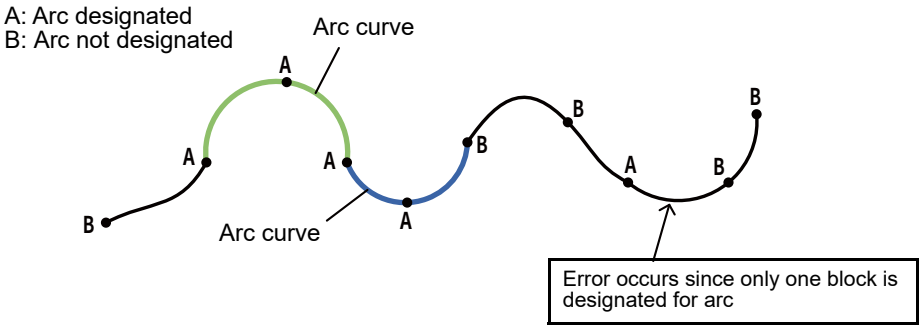


Fig.7-15:Example of error in arc designation (Spline file version 02)

Spline file version 02

The spline file version 02 is supported with the following software version.
Controller: Ver. A1 or later
RT ToolBox3: Ver. 1.00 or later

2) Spline file version 01

When three consecutive path points are designated as an arc, the curve that connects those three points will be an arc instead of a spline. If this is applied at a section requiring an arc-shaped curve, such as at a corner, an accurate arc can be drawn with fewer path points than the spline curve.

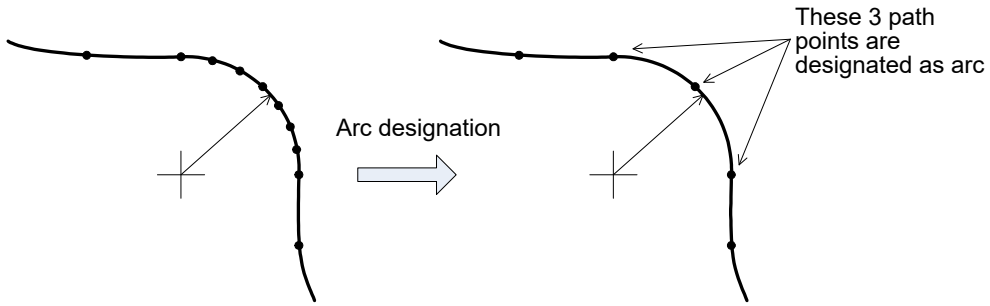


Fig.7-16:Arc designation (Spline file version 01)

With arc designation, three consecutive path points must always be designated. If only one point is designated, the designation will be ignored, and a spline curve will be drawn. If only two points are designated, the error L2613 (not enough points designated for arc) will occur.
If the three designated points are on a straight line and the arc cannot be generated, the error L2613 (block data calculation error (Cir. Arc)) will occur.

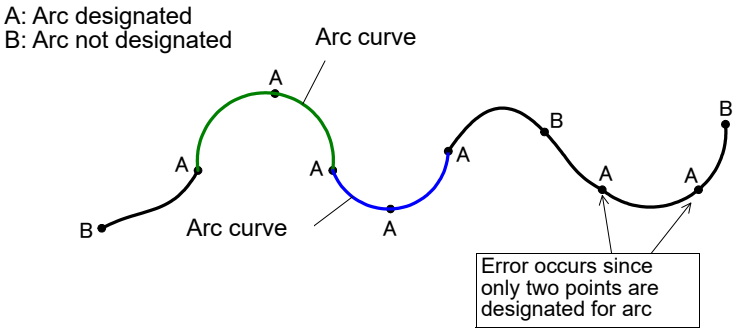
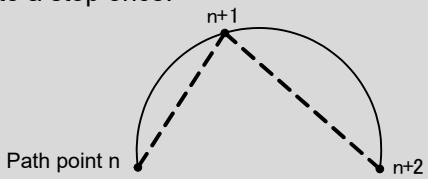


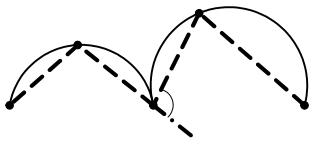
Fig.7-17:Example of error in arc designation (Spline file version 01)

Relation of arc designation and other path point adjustment functions

Tolerance and block length ratio do not function on a block that draws an arc. Spline cancel does not function on the arc's transit points (path point n+1). For the start point (path point n) and end point (path point n+2), if the angle created by the previous and next block is larger than the cancel angle, the movement will decelerate to a stop once.



CAUTION When a continuous arc is designated, depending on the path points the arc may have a sharp return point as shown in the following drawing. If the spline cancel does not function and decelerate to stop does to take place between the connections of adjacent arcs, a sharp reversal will take place. Take special care to the shape of the generated arc curve.



■Spline cancel

To create a corner edge at path points having a large change in movement direction instead of passing through them with a smooth spline curve, designate the cancel angle to split the spline curve at that path point.

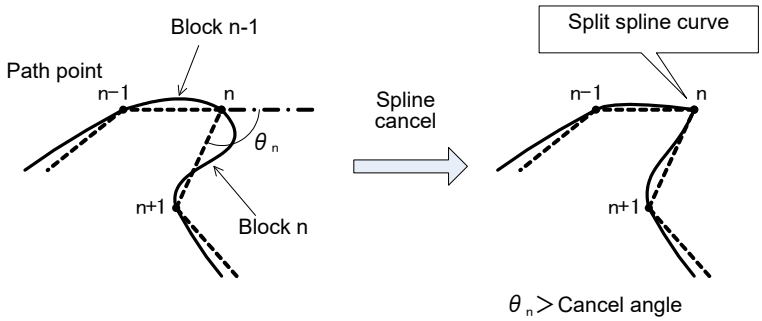


Fig.7-18:Spline cancel

For example, to create a corner edge at the path point n shown in Fig. 7-18, if a smaller angle than the angle θ n created by path point n's previous and next blocks (block n-1 and block n) is set as the cancel angle, the

spline curve will be split at path point n. The robot will decelerate to a stop once at the path point n where the spline curve is split, and then will accelerate and execute the remaining interpolation.
The cancel angle setting range is shown below. This angle can be set individually for each spline interpolation.

Setting range: 0 to 180 degrees (1 degree increments)
* Spline cancel does not function when 0 degrees is set.
* The default value is 120 degrees.

⚠CAUTION If spline cancel is disabled or if the cancel angle is set to a large value, the spline curve will not be split where the movement direction changes greatly, and the robot will not decelerate. This can cause the robot to vibrate or a servo error to occur.
If the spline curve bends and breaks (zigzag), the interpolation position may not be calculated and error L2163 (spline interpolation command calculation error) may occur.
Check the path point positions and set the appropriate cancel angle.

■Block length ratio
If the block length is extremely long in respect to an adjacent block, that block can automatically be moved as a linear block instead of a spline curve.

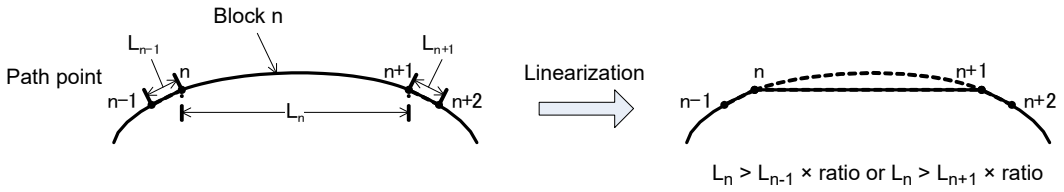


Fig.7-19:Linearization by block length ratio

If the block n length L_n is longer than the designated length compared to the adjacent block length L_{n-1} or L_{n+1}, block n will be linear.
The block length ratio setting range is as follows. This can be set individually for each spline interpolation.

Setting range: 0 to 100 fold (1-fold unit)
* The block length ratio does not function when 0-fold is designated.
* The default value is 8-fold.

Even if tolerance is set for a block that is linear because of the block length ratio, the block length ratio function will have priority.

Precautions for path adjustment

The curve onto which path adjustment has been applied and the connection of the previous and next curve are not as smooth as the connections of spline curves that have not been adjusted. Thus, depending on the state of the connection and the speed command, the speed may fluctuate at the curve connections.

(2) Operation mode

With spline interpolation, the robot is operated to move through control points while maintaining the designated speed. If the designated speed is high, or if the posture variation amount in respect to the distance between path points is large, the speed may be exceeded during the movement.
An operation mode is available so that in such cases, the designated speed is not maintained but the robot moves while not generating an excessive speed when possible. There are two operation modes as shown in Table 7-6.

Table 7-6:Operation mode

Operation mode	Explanation
Constant linear speed	The robot moves at the designated speed and maintains the constant speed. The speed may be exceeded during the movement depending on the designated speed or posture variation amount.
Variable linear speed	The robot moves at the designated speed and maintains the constant speed. However, if it is judged that the speed may be exceeded during the movement, control is applied so that the movement speed is automatically lowered and the speed is not exceeded. The speed returns to the originally designated speed when the possibility of over speed is eliminated. * It may not be possible to suppress the over speed depending on the movement state.

The operation mode can be set individually for each spline interpolation. The default state is constant linear speed.
If the speed will be exceeded during spline interpolation movement, try using "variable linear speed" if the movement does not need to be performed at a constant speed.

Step feed

If the specified speed may be exceeded when executing spline interpolation with step feed, the movement speed will drop even if the operation mode is "constant linear speed".

(3) Signal output

A random external output signal can be turned ON/OFF when passing through a path point. By using this function, for example, a trigger signal can be output to a peripheral device when passing through a random path point even without describing it in the robot program.

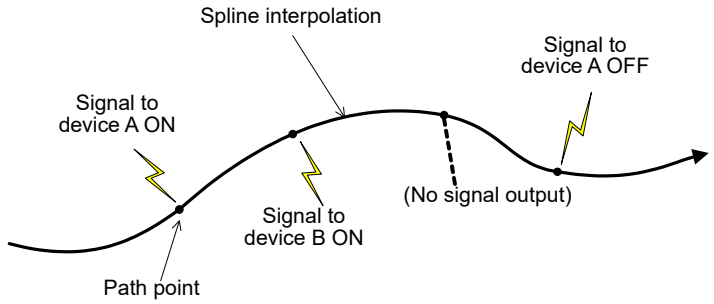


Fig.7-20:Signal output

To use signal output, set the items in Table 7-7 as the path point data.

Table 7-7:Signal output setting items

Setting item	Explanation
Head No.	Designate the head No. of the output signal. Setting range: –1 to 32767 * When –1 is set, this function will be disabled at that path point. The default value is –1. The setting range is shown above, but the head No. that can actually be output will depend on the type of connected device.
Bit width	Designate the bit width of the signal to be output. Setting range: 1, 8, 16, 32-bit The default value is 1-bit.

<table>
<tr><th>Setting item</th><th>Explanation</th></tr>
<tr><td>Bit mask</td><td>Designate the mask pattern for enabling the signal output in bit units in respect to the designated bit width. The setting range changes according to the bit width.
<table>
<tr><th>Bit width</th><th>Bit mask setting range [hex]</th></tr>
<tr><td>1</td><td>0, 1</td></tr>
<tr><td>8</td><td>0 to FF</td></tr>
<tr><td>16</td><td>0 to FFFF</td></tr>
<tr><td>32</td><td>0 to FFFFFFFF</td></tr>
</table>
The bit with 1 bit mask applies the designated output data’s ON/OFF status onto the signal output.
The bit with 0 bit mask maintains the ON/OFF status of the external output signal at that point.
The default value is 0, so the bit mask must be set to output the signal.</td></tr>
<tr><td>Output data</td><td>Designates the data to be signal output.
The setting range differs according to the bit width.
<table>
<tr><th>Bit width</th><th>Bit mask setting range [hex]</th></tr>
<tr><td>1</td><td>0, 1</td></tr>
<tr><td>8</td><td>0 to FF</td></tr>
<tr><td>16</td><td>0 to FFFF</td></tr>
<tr><td>32</td><td>0 to FFFFFFFF</td></tr>
</table>
The default value is 0.</td></tr>
<tr><td>Pulse output</td><td>When pulse output is enabled, the above signal output designation is applied onto the external output signal for approx. 14 ms after the path point is passed. The external output signal status returns to the original status after approx. 14 ms has passed.
This is disabled as the default value.</td></tr>
</table>

An example of the signal output data set at the path point and the changes in the external output signal status is shown in Table 7-8.

Table 7-8:Example of signal output (all signals are OFF in default state)

Path point	Head No.	Bit width	Bit mask	Output data	Pulse output	External output signal status [● : ON/ ○ : OFF]							
						107	106	105	104	103	102	101	100
1	100	1	1	1	Invalid	○	○	○	○	○	○	○	●
2	102	1	1	1	Invalid	○	○	○	○	○	●	○	●
3	-1	1	0	0	Invalid	○	○	○	○	○	●	○	●
4	100	8	5C	FA	Invalid	○	●	○	●	●	○	○	●
5	100	8	FF	AF	Valid	●	●	●	●	●	●	●	●
	Approx. 14 ms later ⇒					○	●	○	●	●	○	○	●

If a dedicated output signal is assigned to the signal output designation, an error L0091 (signal already assigned to dedicated output) will occur when the signal output is executed. Do not output a signal to a bit having a dedicated output assignment.
The external output signal designated with signal output is not occupied. Note that if the target external output signal is changed with another robot program using multi-task, those changes will be applied.

Judgment of path point passage

Passage of the path point is judged by the position command level. It is not judged by the actual robot position (feedback position).

(4) Numerical setting

A random value can be set as the path point data.

Setting range: –1 to 32767 (integer)
* –1 indicates not set.
* The default value is –1.

When the status variable M_SplVar is referred to, the value set for the path point passed through most recently can be confirmed. If the same value is set for multiple path points and referring to the status variable M_SplVar with multi-task, a preset process can be executed when the specified path point is passed through during spline interpolation.
When a non-set path point is passed, the status variable M_SplVar value will not change. The value held at that time will be returned.

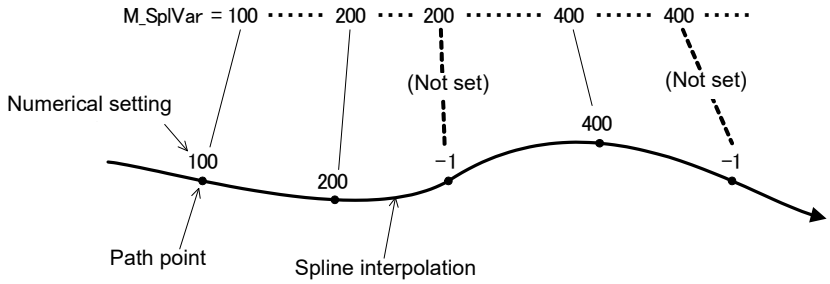


Fig.7-21:Numerical setting

(5) Frame transformation

Frame transformation is a function applied on a random path, and moves the path to another position while maintaining the shape.
The following processes can be handled with this function:

- Transformation from position on drawing to actual position
- Adjustment of work target's position deviation
- Change of work area with spline interpolation

■Outline of process

Frame transformation is explained using Fig. 7-22 as an example.
Set a coordinate system to be used as a reference in respect to a random path. The coordinate system is designated with three position data in the same manner as the Fram function. In this example, position data PR1, PR2 and PR3 are designated, and reference coordinate system "Xfr-Zfr-Yfr" is set.
Next, set the transformed reference coordinate system in the same manner. In this example, position data PC1, PC2 and PC3 is designated, and the transformed reference coordinate system "Xfc-Zfc-Yfc" is set.
When frame transformation is executed, all path points are calculated so that the relative displacement from the reference coordinate system's origin to the path point is the same before and after the transformation. In the example, the path points P11 ~ P14 to P21 ~ P24 is calculated.
The same value as the corresponding pre-transformation path points are set for the transformed path point configuration flags, multi-rotation flags and additional axis data.
In Ex-T spline interpolation, frame transformation is executed including "XE1-ZE1-YE1" of the Ex-T coordinate system origin.

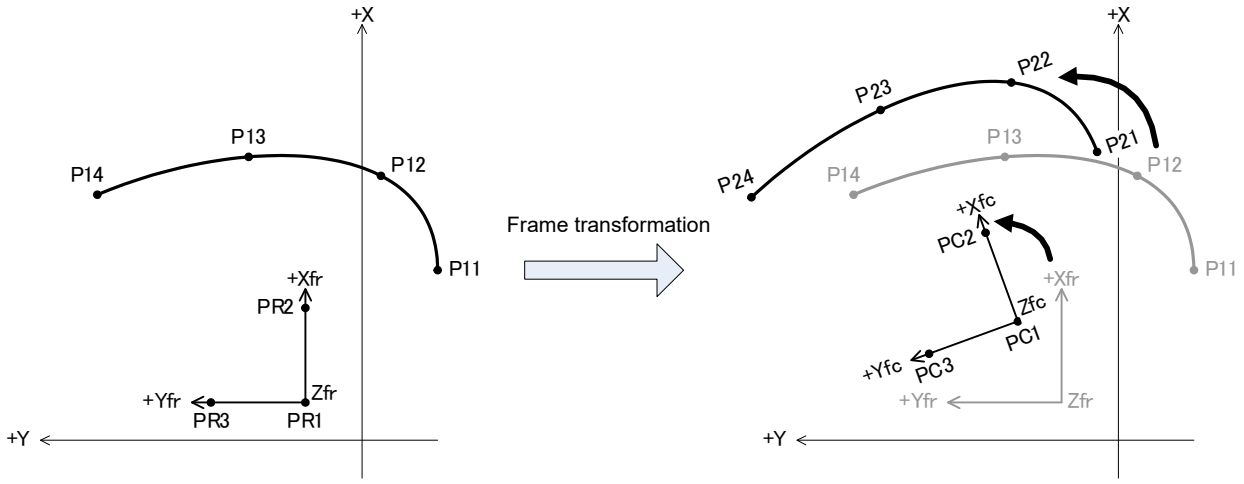


Fig.7-22:Frame transformation (Spline interpolation)

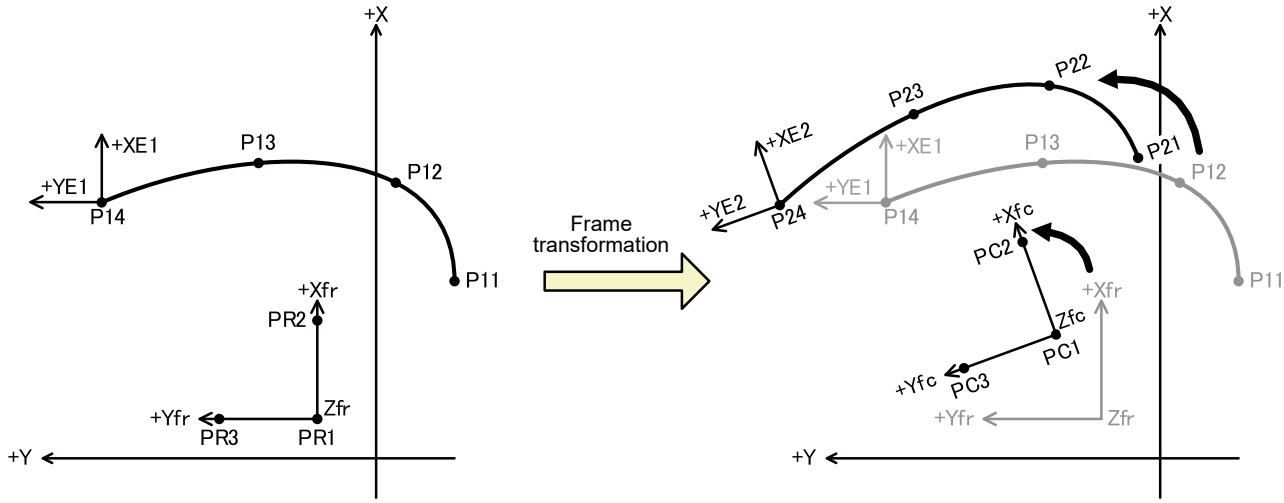


Fig.7-23:Frame transformation (Ex-T spline interpolation)

■Set data

Define the coordinate system by setting the following three position data items in the same manner as the Fram function.

Origin position ..... Equivalent to PR1, PC1 in Fig. 7-22 and Fig. 7-23.
Position on X axis ..... Equivalent to PR2, PC2 in Fig. 7-22 and Fig. 7-23.
XY plan +Y direction position .... Equivalent to PR3, PC3 in Fig. 7-22 and Fig. 7-23.

The X-, Y- and Z-axis coordinate values for a total of six position data items are set for the pre-transformation and transformed.

If the three position data items contain the same point or if the three position data items are on the same line, the coordinate system cannot be defined and error L2041 (can't calculate frame transformation coordinates) will occur.

■Execution method

There are three methods of executing frame transformation.

Table 7-9:Frame transformation execution method

Execution method	Explanation	Usage case
Execute with RT ToolBox3	Frame transformation is executed with the RT ToolBox3 Spline File Edit screen. A spline file is created using the transformation results as the path points.	• When individually compensating the transformed path point data using jog operation, etc. • When using the transformed path point data as the source data for another path.

Save coordinate system in spline file	The coordinate system settings are saved in a spline file, and the path points are sequentially frame transformed when spline interpolation is executed.	• To keep the coordinate values on the drawing as the coordinate values for path points • When coordinate system is fixed to one
Set coordinate system with robot program	The coordinate system is set in the robot program using the SetCalFrm command. The path points are sequentially frame transformed using set coordinate system when spline interpolation is executed.	• To use multiple coordinate systems for one path • To use sensors and compensate the path deviation each time

7.2.4 Work procedures

The work procedures for executing spline interpolation are listed below.

Table 7-10:Work procedures

No.	Step	Work details
1	Creating the spline file (Chapter "7.2.5")	The spline file is created using the RT ToolBox3 Spline File Edit screen, DXF File Import function, or SplWrt command. (1) Register the path points using the position teaching or import function. (2) Set the signal output and numerical settings as necessary.
2	Creating the robot program (Chapter "7.2.6")	Create a robot program that contains the spline interpolation movement.
3	Confirming the movement (Chapter "7.2.7")	Confirm the spline interpolation movement with the simulation function[Note1)] , and correct the path point position and robot program as needed.
4	Saving to robot controller (Chapter "7.2.8")	Write the robot program and spline file to the robot controller.
5	Adjustment work (Chapter "7.2.9")	Make fine adjustments to the actual system using the RT ToolBox3 Spline File Edit screen. (1) Using debugging operations, confirm the spline interpolation movement in the actual system. (2) Read the spline file to the Spline File Edit screen. (3) Correct the path point position and path adjustment setting as needed. (4) Write the spline file to the controller. (5) Repeat steps (1) to (4) to create the required path, and adjust the spline file data.
6	Operation	Run the robot program and execute spline interpolation.

Note1) Not supported with the RT ToolBox3 mini version.

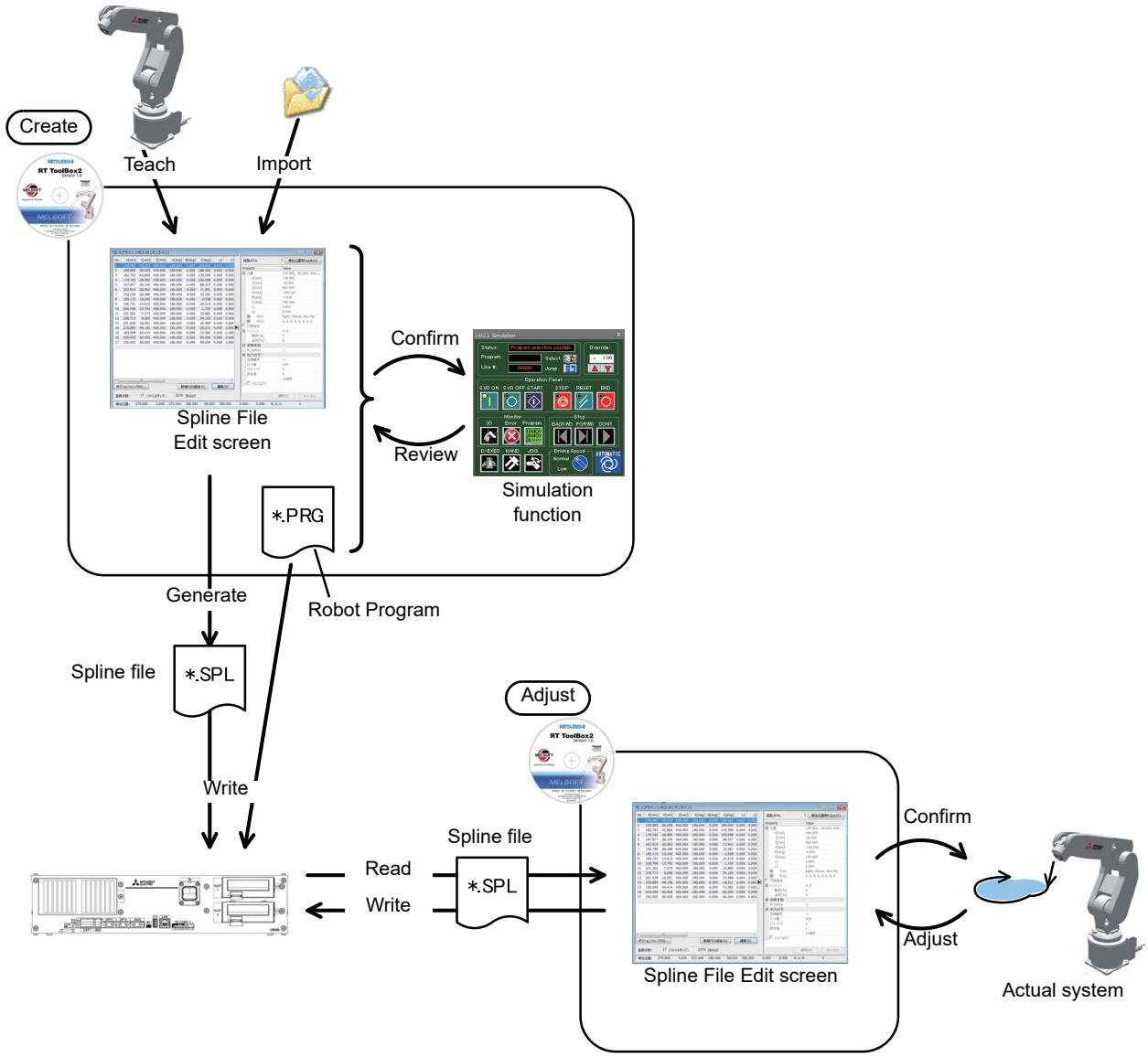


Fig.7-24:Schematic diagram of work details

7.2.5 Creating the spline file

The spline file is created with the RT ToolBox3 Spline File Edit screen, DXF File Import function, or SplWrt command. The methods for creating the spline file are explained below.

(1) New file

■Creating a new file in the personal computer

Select [Offline] → [Spline] for the project to be newly created, and click the right mouse button. The Context menu will open, so click [New] and open the Spline File Edit screen.

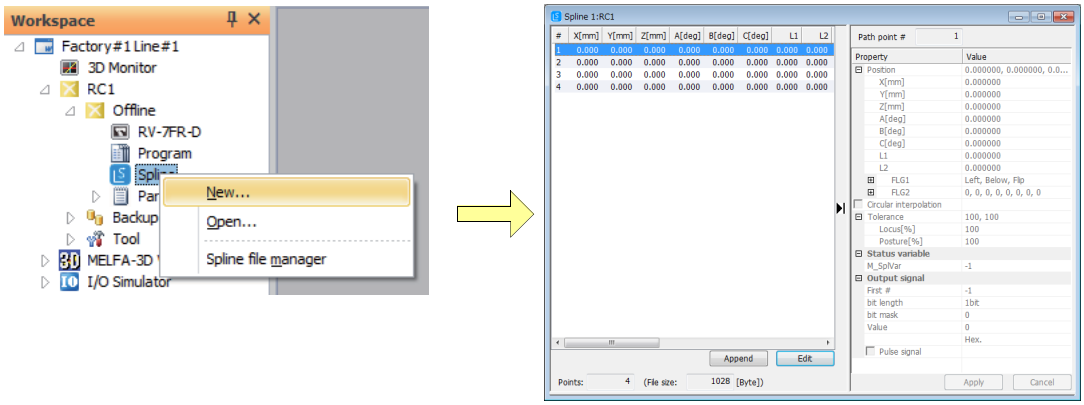


Fig.7-25:Creating a new file in the personal computer

■Newly creating in the controller

Select [Online] → [Spline] for the project to be newly created, and click the right mouse button. The Context menu will open, so click [New] and open the Spline File Edit screen.

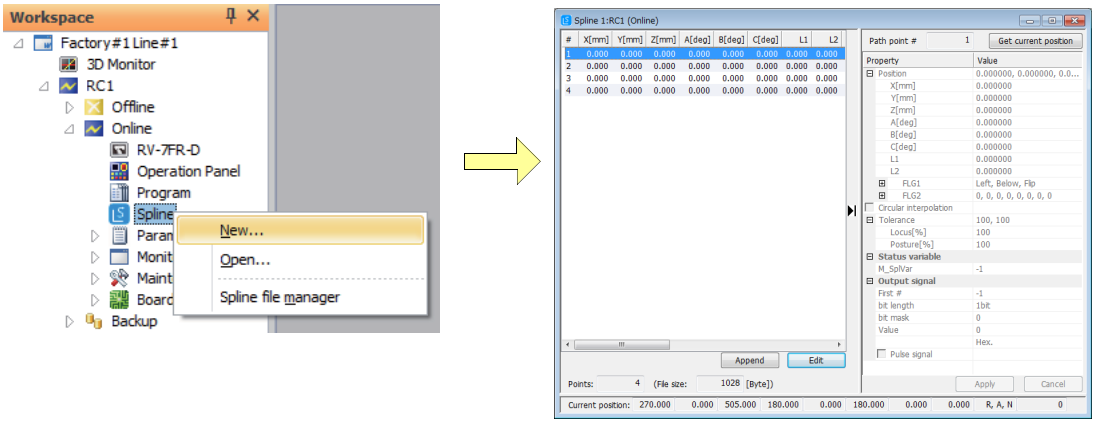


Fig.7-26:Creating a new file in the controller

Robots that do not support spline interpolation

If the robot used in the target project does not support spline interpolation, [Spline] will not appear in the [Online] project tree. [Spline] is always displayed in the [Offline] project tree.

(2) Creating a file by DXF File Import function

AutoCAD versions up to 2014 support DXF files.

1) Selecting a DXF file

Select [Tool] → [DXF File Import] from the project tree and click the right mouse button. A context menu will appear, and a dialog to open a DXF file will appear when [Open] is clicked.

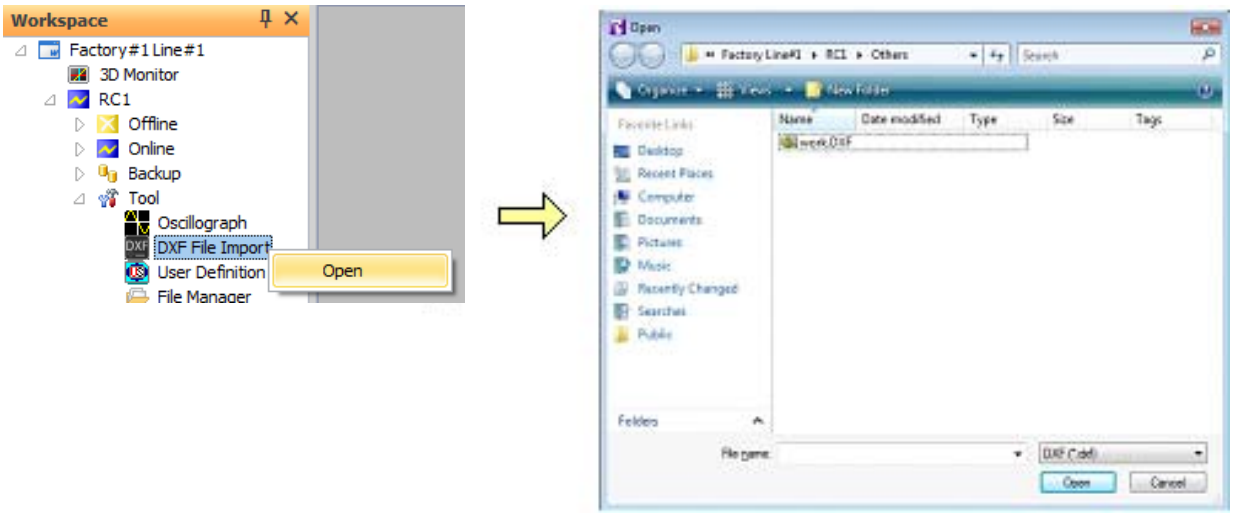


Fig.7-27:File selection screen

The contents of selected DXF file is displayed in the DXF file viewer.

* The following graphic data can import by a DXF files.
  - LINE
  - ARC
  - CIRCLE
  - SPLINE
  - POLYLINE/LWPOLYLINE (Light weight polyline)

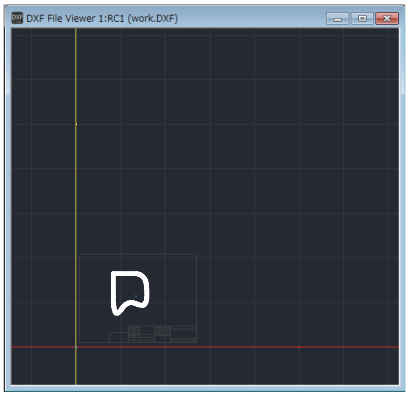


Fig.7-28:DXF file viewer

2) Selecting layers

Select layers to import from [Layer list] in [Layers] page.

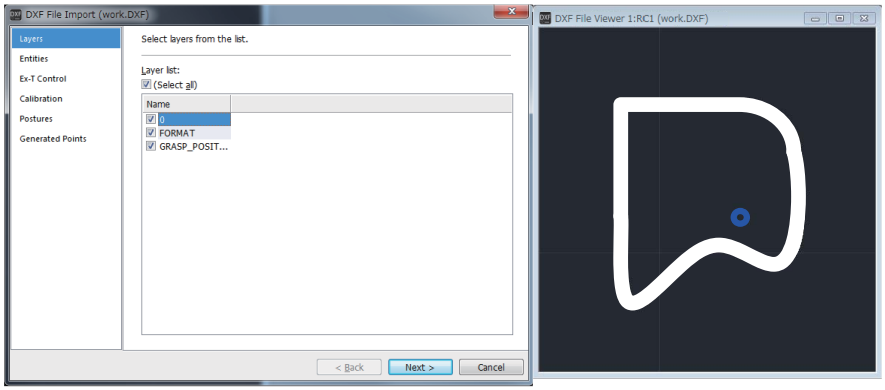


Fig.7-29:Layers selection screen

3) Selecting entities

Select entities to import from [Entities] page.

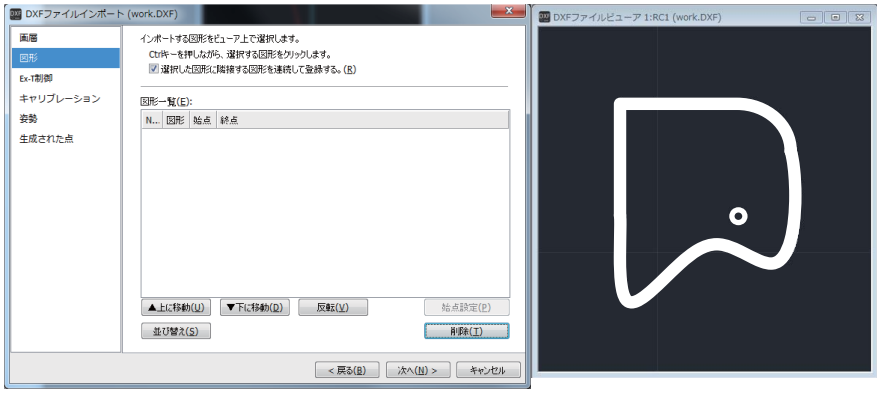


Fig.7-30:Entities selection screen

Hold down the Ctrl key and click entities to be imported by DXF file, they will be added to [Entity list]..

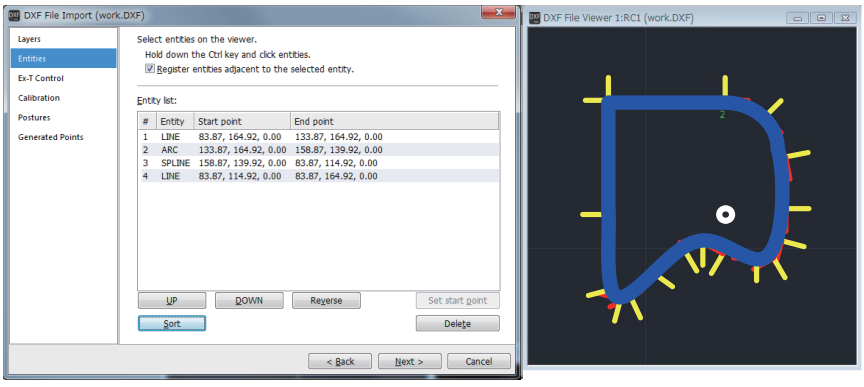


Fig.7-31:Selecting entities

Entities with consecutive start points and end points for the previous and next entities are displayed in blue from the first entity. Non-consecutive entities are displayed in purple.

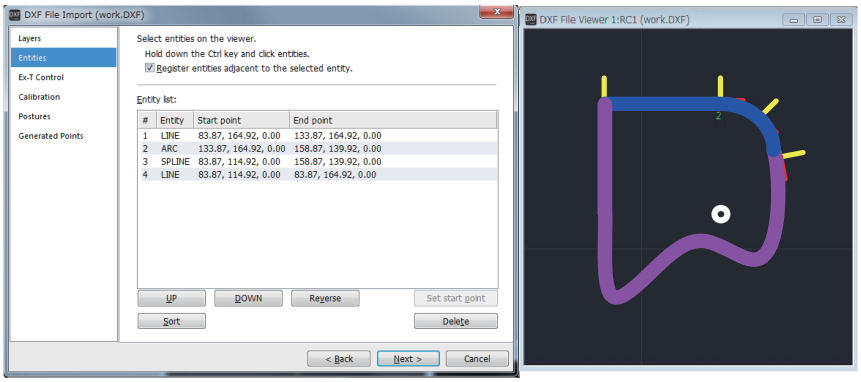


Fig.7-32:Selecting entities

By selecting an entity from the [Entity list] and clicking the [UP] and [DOWN] buttons, the position of the selected entity can be changed.

Entity list:

#	Entity	Start point	End point
1	LINE	83.87, 164.92, 0.00	133.87, 164.92, 0.00
2	ARC	133.87, 164.92, 0.00	158.87, 139.92, 0.00
3	SPLINE	158.87, 139.92, 0.00	83.87, 114.92, 0.00
4	LINE	83.87, 114.92, 0.00	83.87, 164.92, 0.00

Down

Entity list:

#	Entity	Start point	End point
1	ARC	133.87, 164.92, 0.00	158.87, 139.92, 0.00
2	LINE	83.87, 164.92, 0.00	133.87, 164.92, 0.00
3	SPLINE	158.87, 139.92, 0.00	83.87, 114.92, 0.00
4	LINE	83.87, 114.92, 0.00	83.87, 164.92, 0.00

Fig.7-33:Selecting entities (Down)

By selecting an entity from the [Entity list] and clicking the [Reverse] button, the <Start point> and <End point> of the selected entity can be reversed.

Entity list:

#	Entity	Start point	End point
1	LINE	83.87, 164.92, 0.00	133.87, 164.92, 0.00
2	ARC	133.87, 164.92, 0.00	158.87, 139.92, 0.00
3	SPLINE	158.87, 139.92, 0.00	83.87, 114.92, 0.00
4	LINE	83.87, 114.92, 0.00	83.87, 164.92, 0.00

Reverse

Entity list:

#	Entity	Start point	End point
1	LINE	133.87, 164.92, 0.00	83.87, 164.92, 0.00
2	ARC	133.87, 164.92, 0.00	158.87, 139.92, 0.00
3	SPLINE	158.87, 139.92, 0.00	83.87, 114.92, 0.00
4	LINE	83.87, 114.92, 0.00	83.87, 164.92, 0.00

Fig.7-34:Selecting entities (Reverse)

When the [Sort] button is clicked, a confirmation dialog will appear.

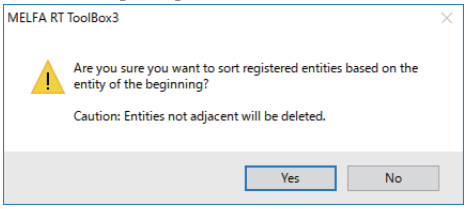


Fig.7-35:Sort confirmation dialog

By clicking [Yes] at the confirmation dialog box, entities registered in the [Entity list] are sorted with the first entity as a reference.
* If non-adjacent entities are registered in the [Entity list], those entities are deleted.

Entity list:

#	Entity	Start point	End point
1	LINE	133.87, 164.92, 0.00	83.87, 164.92, 0.00
2	ARC	133.87, 164.92, 0.00	158.87, 139.92, 0.00
3	SPLINE	158.87, 139.92, 0.00	83.87, 114.92, 0.00
4	LINE	83.87, 114.92, 0.00	83.87, 164.92, 0.00

Sort

Entity list:

#	Entity	Start point	End point
1	LINE	133.87, 164.92, 0.00	83.87, 164.92, 0.00
2	LINE	83.87, 164.92, 0.00	83.87, 114.92, 0.00
3	SPLINE	83.87, 114.92, 0.00	158.87, 139.92, 0.00
4	ARC	158.87, 139.92, 0.00	133.87, 164.92, 0.00

Fig.7-36:Selecting entities (Sort)

4) Ex-T control setting

Grasp positions for imported entities, and Ex-T coordinates registered in spline files are set at the [Ex-T Control] page.

Select the [Use Ex-T control] check box here if using Ex-T spline interpolation. Clear the check box if using normal spline interpolation.

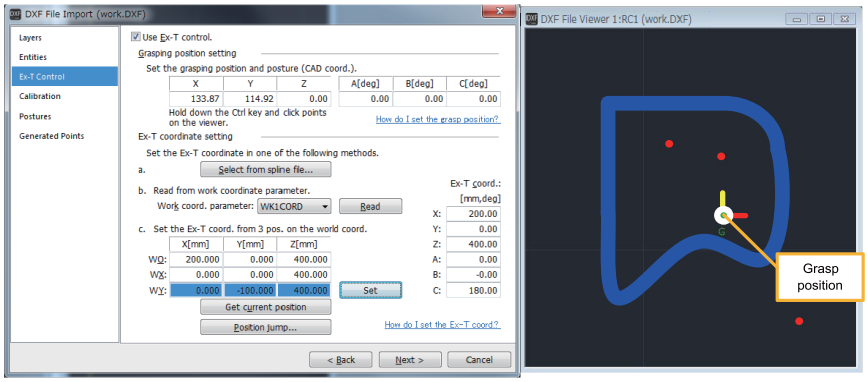


Fig.7-37:Ex-T control setting screen

By clicking a point (red) on the viewer while holding down the Ctrl key, the selected point is registered as the grasp position.

* The layer "GRASP_POSITION" point/circle (center point) is displayed as the default grasp position (X, Y, Z).

The following three types of Ex-T coordinate setting methods are available.

• Selecting from existing spline files

Clicking the [Select from spline file] button displays the spline file selection screen.

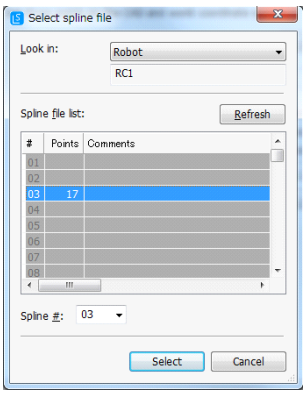


Fig.7-38:Select spline file screen

Ex-T coordinate data stored in the selected spline file is read.

• Reading from work coordinate parameters

By setting the parameter to be read from the drop-down list box and clicking the [Read] button, the work coordinate data registered in the selected parameter is read.

• Setting from three world coordinate system points

Ex-T coordinate data is set by setting the Ex-T coordinate system origin (WO), position on Ex-T coordinate system +X-axis (WX), and +Y-axis position on Ex-T coordinate system X-Y plane (WY).

5) Calibration

Perform calibration at the [Calibration] page to change the coordinate system from the CAD coordinate system to the robot world coordinate system.
If using Ex-T spline interpolation, there is no need to perform calibration.

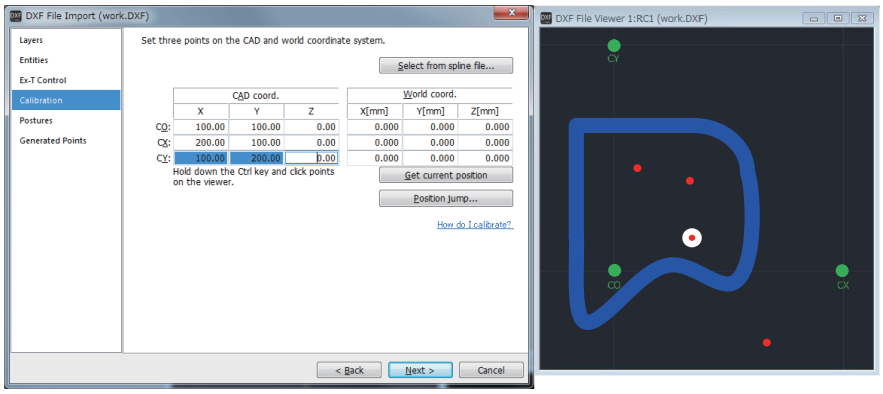


Fig.7-39:Calibration screen

By clicking a point (red) on the viewer while holding down the Ctrl key, the selected point is registered as the CAD coordinate system CO/CX/CY.
* The layer "CALIBRATION" point/circle (center point) is displayed as the CAD coordinate system default.

The following two types of calibration setting methods are available.

- Selecting from existing spline files
  By clicking the [Select from spline file] button, the Select spline file screen similar to that for Ex-T control settings appears. Calibration data stored in the spline selected in the screen is read.
- Setting from three points of the CAD coordinate system, world coordinate system
  Calibration data is set by setting the origin, position on +X-axis, and +Y-axis position on X-Y plane on the CAD coordinate system, and the origin, position on +X-axis, and +Y-axis position on X-Y plane on the world coordinate system.

6) Selecting postures

Set the path point posture at the [Postures] page.

By selecting [Constant with respect to TCP] for the posture type, the posture is registered with the tangent direction for each path point as the tool direction.

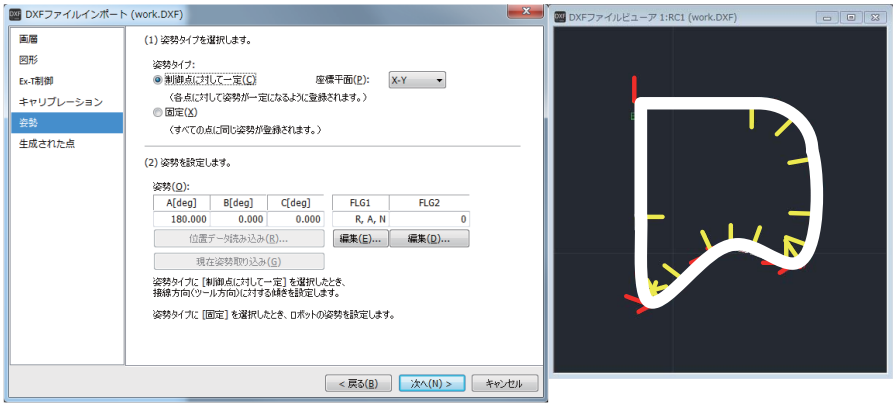


Fig.7-40:Posture registration screen (Constant with respect to TCP)

By selecting [Fixation] for the posture type, the same posture as that for the <Start point> is registered for all path points.

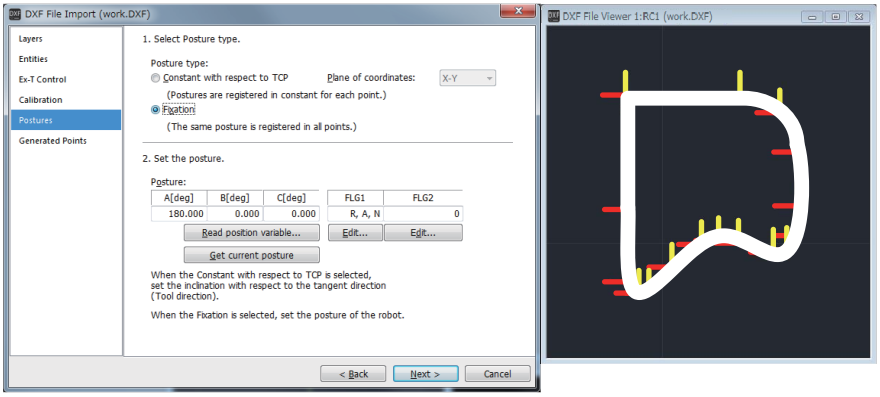


Fig.7-41:Posture registration screen (Fixation)

7) Output to spline file

By selecting [Spline file] as the output format at the [Generated Points] page and clicking the [Finish] button, point sequence data generated to the spline file is output. By clicking [Yes] at the import completion dialog box, the Save spline file screen appears.

By specifying the number of the spline file to be saved and clicking the [Save] button, spline files can be saved.

By selecting the [Tolerance of posture is 0 [%]] check box, the data is output to the spline file with a 0% tolerance for the posture at each path point. To operate the robot in the posture imported by DXF file, select the check box. (The check box is selected in the initial state.)

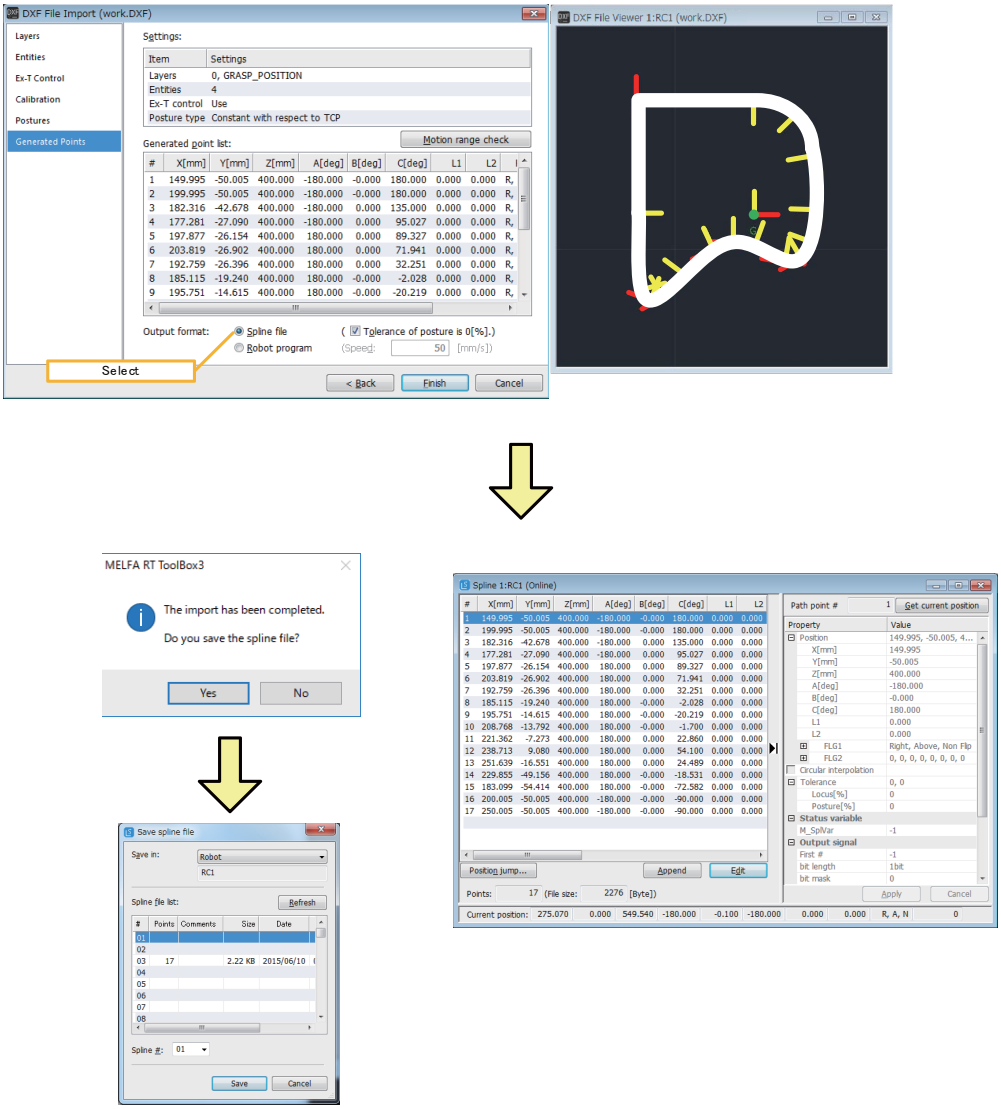


Fig.7-42:Generated Points screen

(3) Creating new spline files with robot language

■Creating new spline files with SplWrt command

Spline files can be created using the SplWrt (spline write) command. Refer to SplWrt (Spline Write) for command details.

Data shown in Table 7-15 is described sequentially separated by commas as path point data in read files. An example can be seen in Fig. 7-75.

Always indicate "Data identification tags" sequentially from <x> in the first line. If there are no "Data identification tags", or if there is an insufficient number of data items (20) in each row, an error will occur.

■Editing existing spline files with SplFWrt command

Frame conversion coordinate data can be registered to spline files saved using the SplFWrt (spline frame write) command. Refer to SplFWrt (Spline Frame Write) for command details.

(4) Opening an existing spline file

■Opening a spline file saved in the personal computer

Select [Offline] → [Spline] for the target project and expand it. The saved spline files will appear in the project tree, so double-click the spline file to be edited.

(The file can also be opened by selecting the spline file and clicking [Open] in the mouse context menu.)

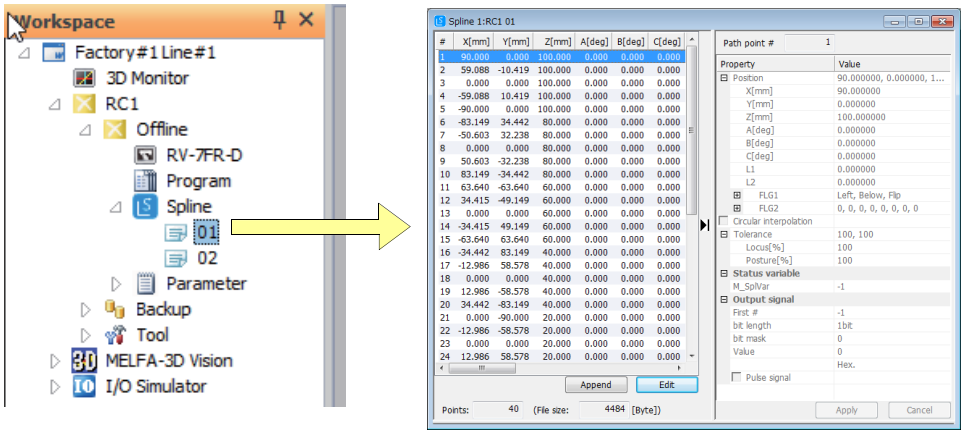


Fig.7-43:Opening a spline file saved in the personal computer

■Opening a spline file saved in the controller

Select [Online] → [Spline] for the target project and expand it. The spline files saved in the controller will appear in the project tree, so double-click the spline file to be edited.

(The file can also be opened by selecting the spline file and clicking [Open] in the mouse context menu.)

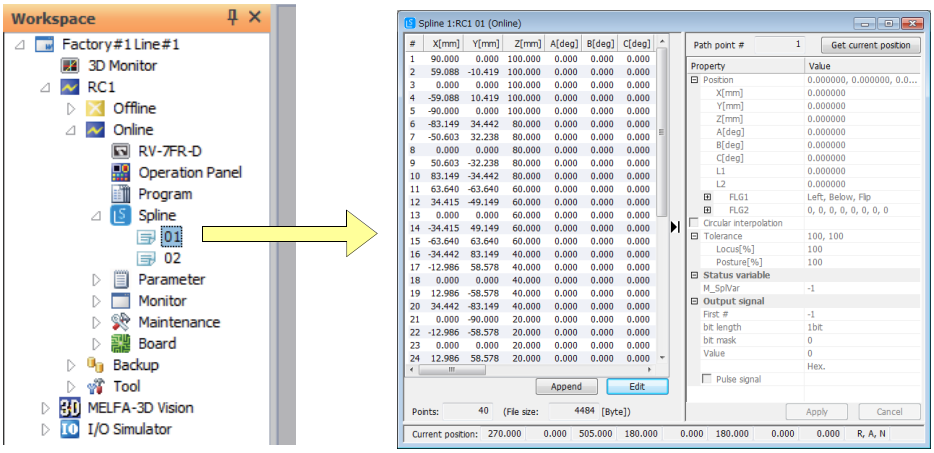


Fig.7-44:Opening a spline file saved in the controller

■Selecting and opening a spline file from the list
Select [Spline] from the project tree and click the right mouse button.
The Context menu will open, so click [Open]. A list of saved spline files will appear.
(The list can also be displayed by clicking menu [File] → [Open].)

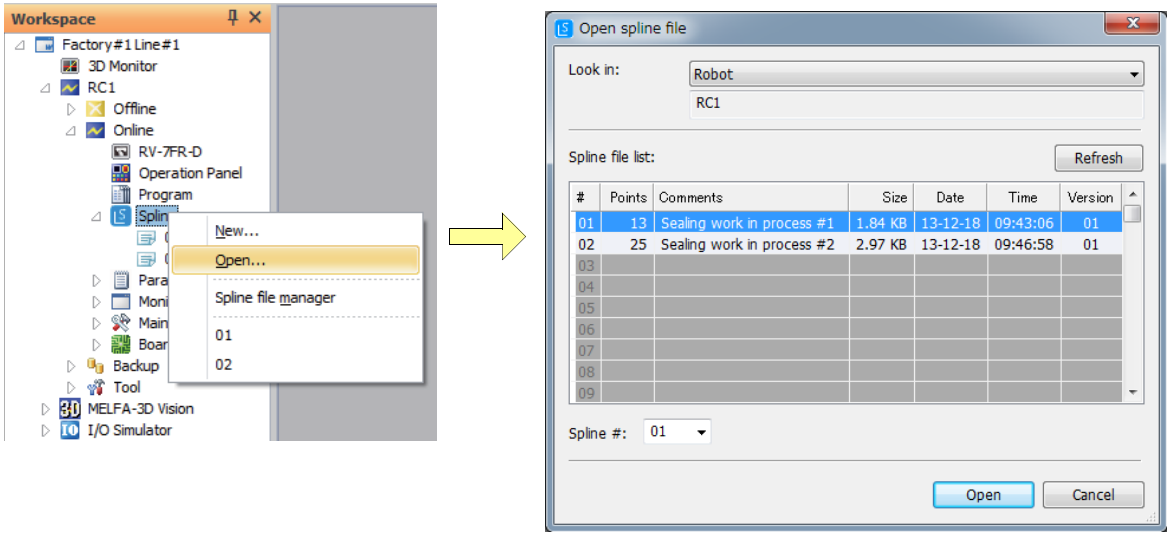


Fig.7-45:Selecting and opening a spline file from the list

The number of path point data items and comments registered in the spline file will appear in the list. Refer to this information and select the target spline file. Click the [Open] button.

(5) Explanation of Spline File Edit screen

The Spline File Edit screen is explained below.

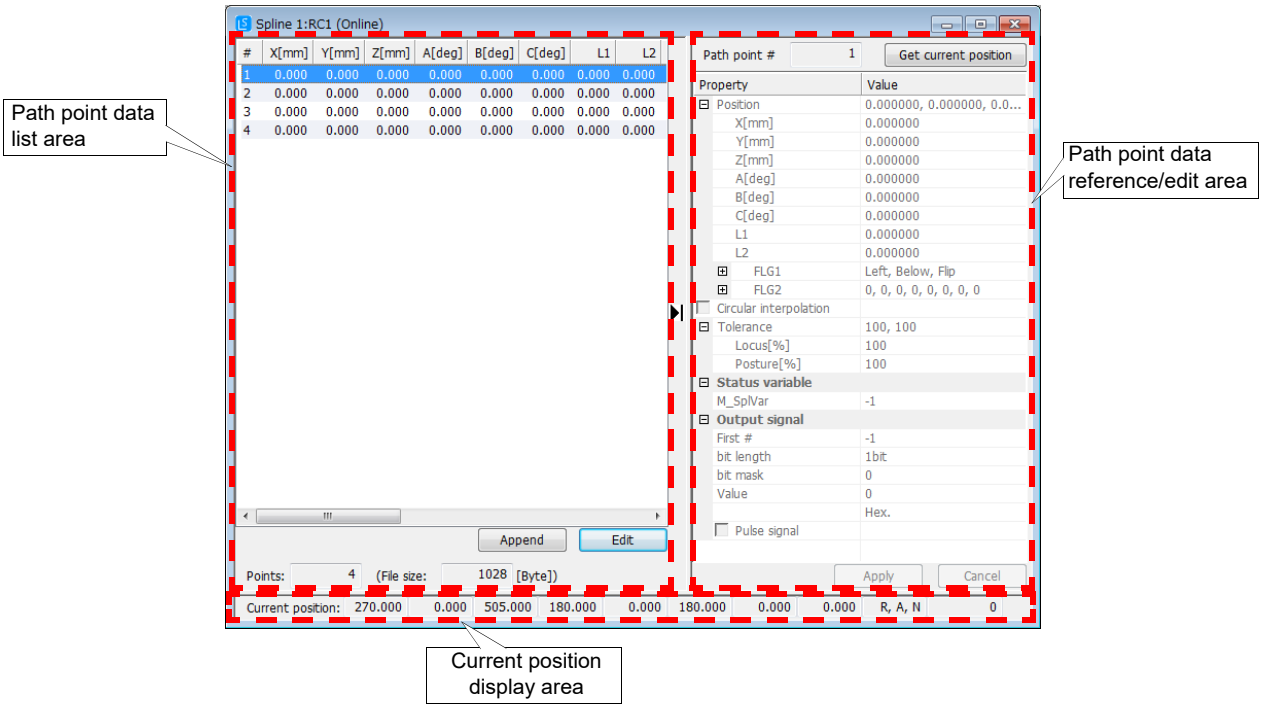


Fig.7-46:Spline File Edit screen

The Spline File Edit screen is configured of three areas.

■Path point data list area
A list of path point data registered in the spline file is displayed. The following operations are performed in this area.

- Add/delete point path data
- Select path point data to edit
- Confirm number of registered points, file size

If the Store mark near the center of the Spline File Edit screen is clicked, the path point data reference/edit area will be stored and the path point data list area display will expand. This makes it easier to confirm the state of changes in data between path points.

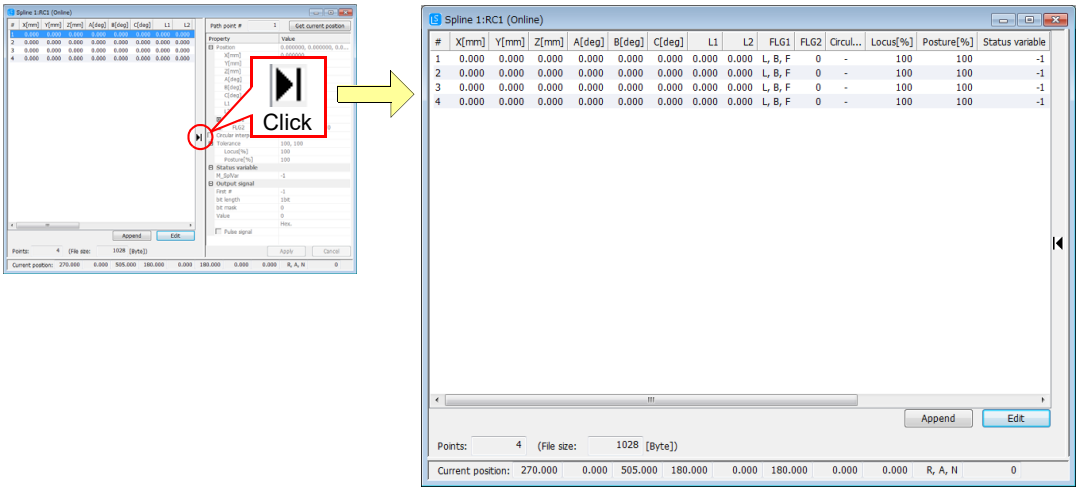
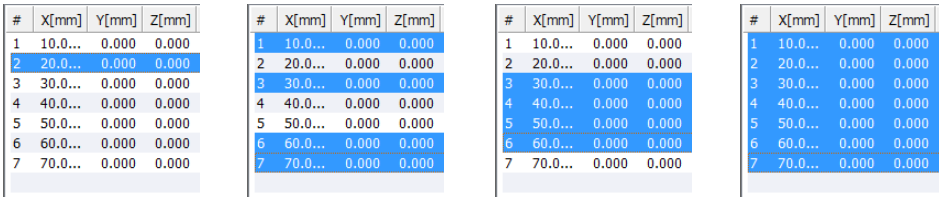


Fig.7-47:State with path point data list area display expanded

Multiple path point data items can be selected simultaneously by using the personal computer's keyboard and mouse.



#	X[mm]	Y[mm]	Z[mm]
1	10.0...	0.000	0.000
2	20.0...	0.000	0.000
3	30.0...	0.000	0.000
4	40.0...	0.000	0.000
5	50.0...	0.000	0.000
6	60.0...	0.000	0.000
7	70.0...	0.000	0.000

1-point (click)

Random point ([Ctrl] key + click)

Consecutive points ([Shift] key + click)

All points (Menu [Edit] → [Edit])

Fig.7-48:Selecting the path point data

■Path point data reference/edit area
The path point data selected in the path point data list area is referred to and edited in this area.
When the path point data is selected in the path point data list area, the details are displayed with gray characters in the path point data reference/edit area. When multiple points are selected, the path point data selected last is displayed. If path point data editing is enabled after that, the characters change to black and the path point data can be edited.
(Refer to Page 714, "■ Editing and registering the path point data" for details on enabling editing.)

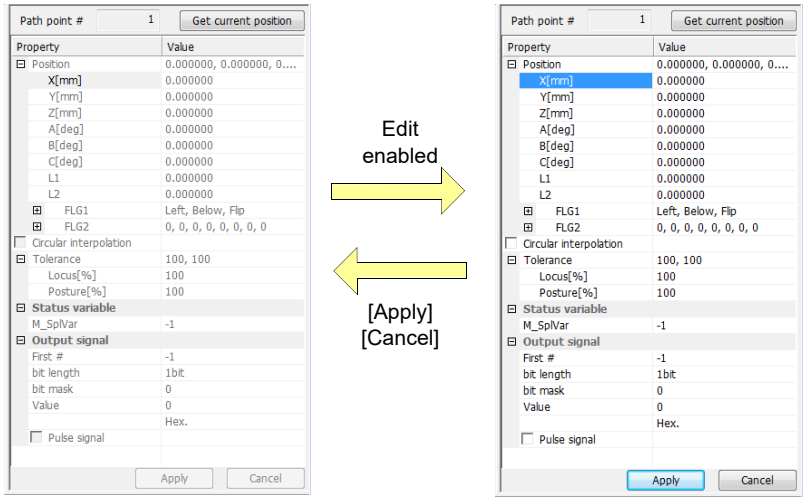


Fig.7-49:Switching between path point data reference/edit area reference state and edit state

If the Store ([–]) or Expand ([+]) mark at the Property filed is clicked, the path point data will be stored or expanded.

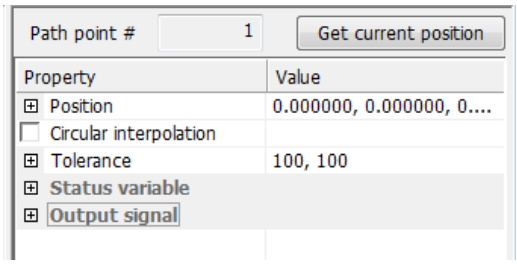


Fig.7-50:State with path point data stored

■Current position display area
The current position data (Cartesian coordinate system) data of the robot connected in the target position is displayed.
If the target project is not connected with a robot (offline mode), the [Get current position] button will not appear in this area or the path point data reference/edit area.

(6) Spline file editing ribbon

The "File (F)", "Edit (E)", and "Tool (T)" tabs are added to the ribbon when editing spline files.



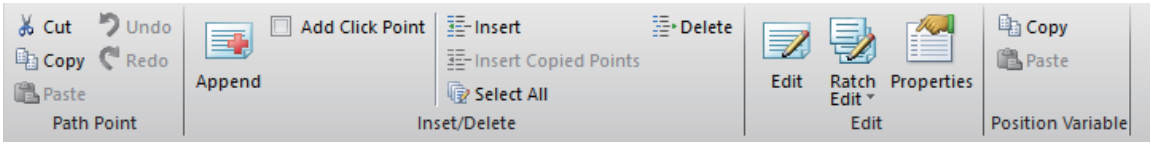
Fig.7-51:Spline file editing ribbon

The details of each menu are explained below.

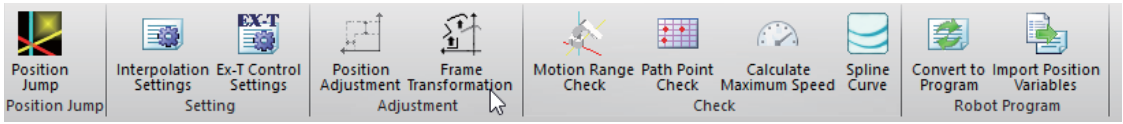
Table 7-11:Details of added menus

Tab options		Explanation	Reference page
File			
	Open	A dedicated screen for opening the spline file opens.	705
	Close	The active Spline File Edit screen closes.	-
	Save	The spline file being edited is overwritten and saved.	720
	Save as	A dedicated screen for saving spline file being edited with a different name opens.	719
	Import	The path point data is imported from a CSV/MXV[Note1)] format file.	729
	Export	The path point data is exported to a CSV format file.	732
	Spline file manager	The Spline file manager screen opens.	722

Tab options	Explanation	Reference page
Edit		



Cut	The selected path point data is saved as cut data. The original path point data is cut out when [Paste] is executed.	732
Copy	The selected path point data is saved as copy data.	732
Paste	The path point data saved with [Cut] or [Copy] is pasted into the designated path point data.	732
Undo	The editing details applied on the path point data last are undone and the original details are restored.	734
Redo	The details canceled with [Undo] are applied again.	734
Append	New path point data is added to the end of the path point data.	712
Add Click Point	Selecting the check box adds the position at which the 3D Monitor screen was clicked to the bottom row.	-
Insert	New path point data is inserted in front of the selected path point data.	712
Insert copied points	The path point data saved with [Copy] is inserted in front of the designated path point data.	733
Select all	All path point data is selected.	-
Delete	The selected path point data is deleted.	715
Edit	Editing of the selected path point data is enabled.	714
Batch Edit - FLG1 structure flag	Edits the FLG1 value of selected path point at one time.	715
Batch Edit - FLG2 multiple rotation data	Edits the FLG2 value of selected path point at one time.	716
Batch Edit - Tolerance	Edits the tolerance of selected path point at one time.	716
Batch Edit - Circular interpolation	Switches the arc designation of the selected path point between enabled and disabled.	716
Properties	The spline file Property screen opens for referring to and entering comments and changing file version.	718
Copy - Position variable	The selected position data is saved to be copied.	733
Paste - Position variable	The saved position data is pasted into the designated path point data.	733

Tab options	Explanation	Reference page
Tools 		
Position jump	A dedicated screen for position jump function opens.	745
Interpolation settings	The spline file Interpolation Setting screen opens.	717
Ex-T control settings	The spline file Ex-T control setting screen opens.	718
Position adjustment	The dedicated screen for the position adjustment function opens.	741
Frame transformation	A dedicated screen for the frame transformation function opens.	743
Motion range check	Checking whether the spline file being edited is within the operating range of the robot or not. * This function can be executed in the simulations.	735
Path point check	A dedicated screen for path point check function opens.	734
Calculate maximum speed	Calculates the maximum speed which can be specified without an error in the spline interpolation commands using the spline file being edited.	735
Spline curve	The spline curve registered in the spline file being edited is displayed.	734
Convert to program	Converts the spline file being edited to the robot program.	736
Import position variables	Imports the position data registered in the robot program to the spline file being edited.	737

Note1) MXT file is a file that contains the dot sequence data and outputted by RT ToolBox3 Pro.

(7) Details of path point data

The details of the path point data registered in the spline file are explained below.

Table 7-12:Details of path point data

Data item		Details	Default value when newly added
Position		Designate the path point position data with the Cartesian coordinate system (Y, Z, A, B, C, L1, L2, FLG1, FLG2). The distance unit is mm and the angle unit is deg.	X to L2: 0.0 FLG1: Left, Below, Flip FLG2: 0
Arc designation		Designate the path points for the arc path. Setting range: Check (designation enabled), No check (designation invalid)	No check
Tolerance	Position	Designate the tolerance for the position (path). Setting range: 0 to 100 [%]	100
	Posture	Designate the tolerance for the posture. Setting range: 0 to 100 [%]	100
Status variable		Designate the value referred to by status variable M_SplVar. Setting range: –1 to 32767 (–1 means not set.)	-1

<table>
<tr><th colspan="2">Data item</th><th>Details</th><th>Default value when newly added</th></tr>
<tr><td rowspan="4">Output signal</td><td>Head No.</td><td>Designate the head address of the output signal.
Setting range: –1 to 32767 (–1 means signal output invalid.)</td><td>-1</td></tr>
<tr><td>Bit width</td><td>Designate the bit width of the output signal.
Setting range: 1, 8, 16, 32 [bit]</td><td>1</td></tr>
<tr><td>Bit mask</td><td>Designate the mask pattern of the bit with valid signal output.
<table><tr><th>Bit width</th><th>Setting range [hex]</th></tr><tr><td>1</td><td>0, 1</td></tr><tr><td>8</td><td>0 to FF</td></tr><tr><td>16</td><td>0 to FFFF</td></tr><tr><td>32</td><td>0 to FFFFFFFF</td></tr></table></td><td>0</td></tr>
<tr><td>Setting value</td><td>Designate the data to be signal output.
<table><tr><th>Bit width</th><th>Setting range [hex]</th></tr><tr><td>1</td><td>0, 1</td></tr><tr><td>8</td><td>0 to FF</td></tr><tr><td>16</td><td>0 to FFFF</td></tr><tr><td>32</td><td>0 to FFFFFFFF</td></tr></table></td><td>0</td></tr>
<tr><td>Pulse output</td><td>Designate the pulse output.
Setting range: Check (designation enabled), No check (designation invalid)</td><td>No check</td></tr>
</table>

(8) Editing the spline file

■Adding path point data

If the [Append] button in the path point data list area is clicked, new path point data is added to the end of the list.

The default values are the values shown in Table 7-12 "Default value when newly added".

(A line can also be added by clicking the menu [Edit] → [Append] from the ribbon.)

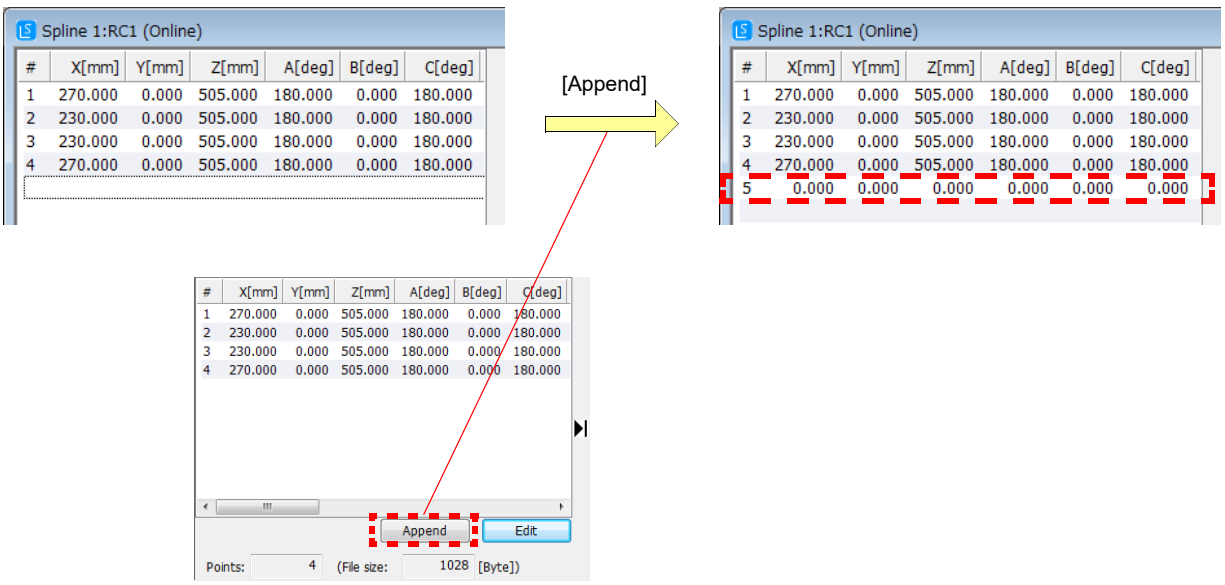


Fig.7-52:Adding path point data

Select random path point data and click the right mouse button. The context menu will open, so click [Insert].
New path point data will be inserted before the selected path point data.
(A line can also be added by clicking menu [Edit] → [Insert] from the ribbon.)

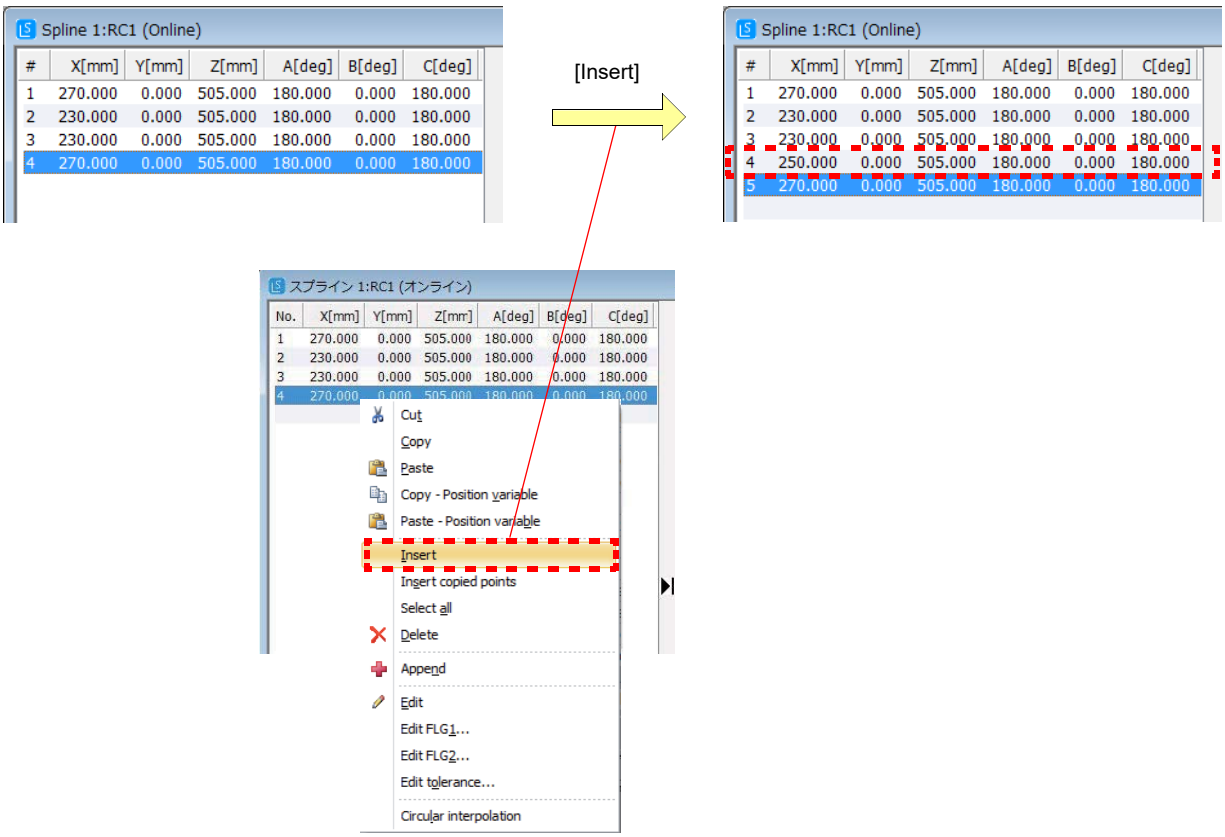


Fig.7-53:Inserting path point data

At this time, the default values of the path point data inserted before the selected path point data line will be the values shown in Table 7-13.

Table 7-13:Default values of inserted path point data

Selected line	Default value of path point data		
Head line	The same values as Table 7-12 "Default value when newly added" are applied.		
Other than head line	Position	X to L2	The average value of the path point data on the line before the selected path point data.
		FLG1, FLG2	Same value as selected path point data.
	Other than position	The same values as Table 7-12 "Default value when newly added" are applied.	

■Editing and registering the path point data
To edit the path point data, select the path point data to be edited in the path point data list area, and double-click it. When double-clicked, editing of the target path point data is enabled, and path point data reference/edit area is set to the editing state.
(Editing can also be enabled by clicking the [Edit] button at the bottom of the path point data list area, or clicking the menu [Edit] → [Edit] from the ribbon.)
Select the item to be edited in the path point data reference/edit area, and set the values.

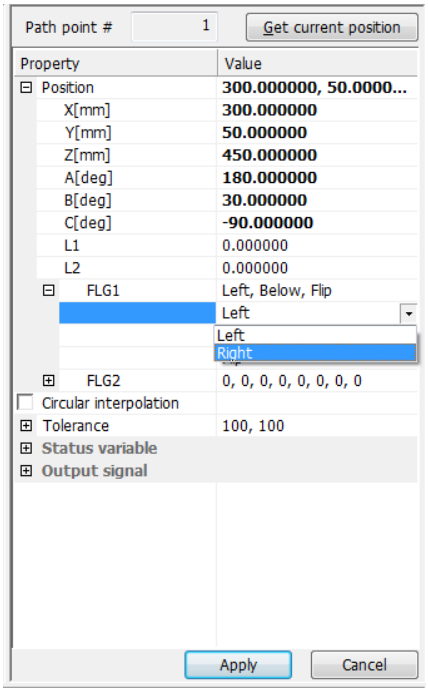


Fig.7-54:Editing the path point data

When connected to a robot, the [Get current position] button appears at the top of the path point data reference/edit area. The robot's current position is imported as the path point position data when this button is clicked.
Click the [Apply] button to register the set details as the path point data. The set details will not be registered as the path point data until this [Apply] button is press. Click the [Cancel] button when not registering the details. Editing is disabled when the [Apply] and [Cancel] buttons are clicked, and the path point data reference/edit area changes to the reference state.

Selection of path point data in editing state

When the path point data reference/edit area is in the editing state, the path point data cannot be selected in the path point data list area. To select another path point data, click the [Apply] or [Cancel] button, and switch the path point data reference/edit area to the reference state.

■Deleting the path point data
Select a path point data in the path point data list area, and click the right mouse button. The context menu will open, so click on [Delete]. The selected path point data will be deleted.
(The data can also be deleted by clicking menu [Edit] → [Delete] from the ribbon.)

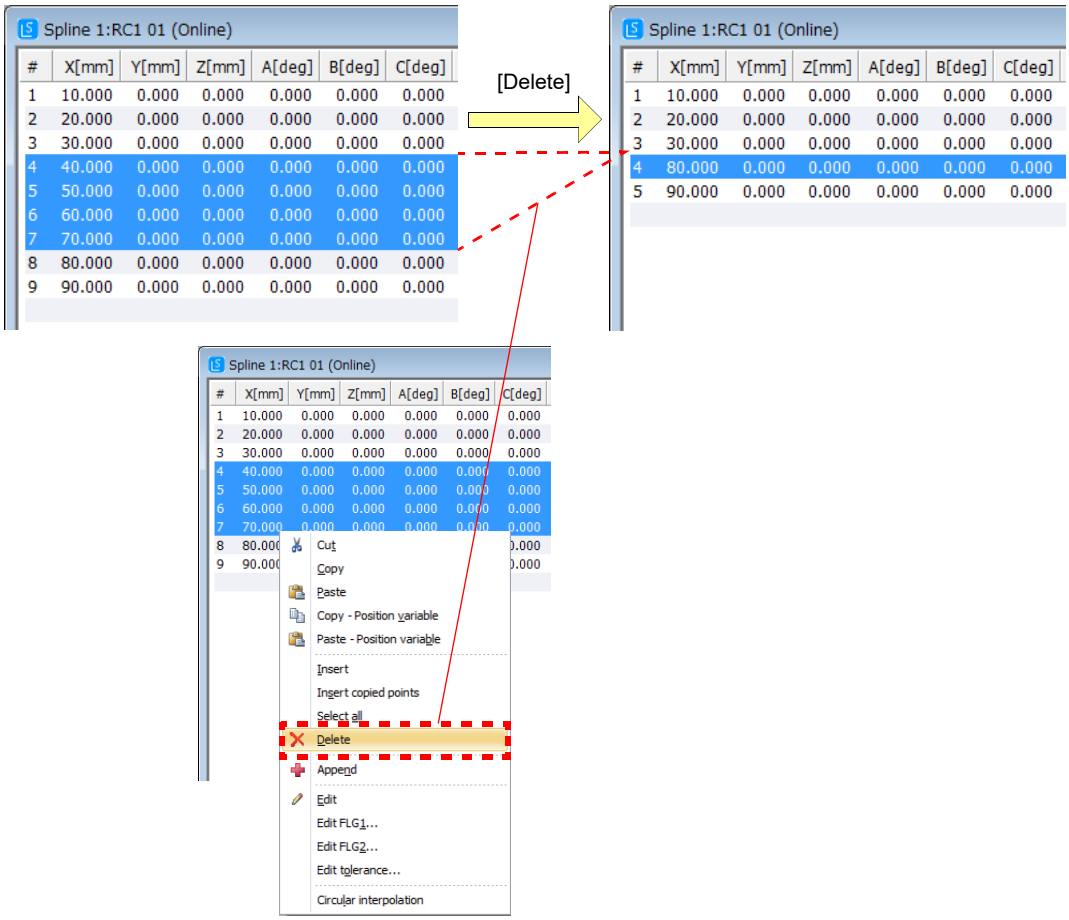


Fig.7-55:Deleting the path point data

■Edit FLG1
Edits the FLG1 value of selected path point at one time.
To batch edit FLG1 values, select the path point data to be edited in the path point data list area, and right-click. Click [Edit FLG1] in the context menu that appears to display an FLG1 batch editing screen.
(This screen can also be displayed by clicking [Edit] → [Batch Edit] → [FLG1 Structure Flag] from the ribbon.)

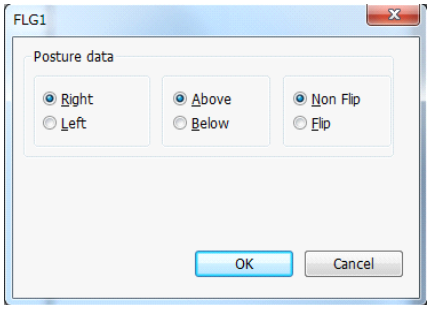


Fig.7-56:Edit FLG1 screen

By changing the arm posture construction and clicking the [OK] button, changes are updated to the selected path point. When the [Cancel] button is clicked, the FLG1 batch editing screen is closed without updating changes.

■Edit FLG2
Edits the FLG2 value of selected path point at one time.
To batch edit FLG2 values, select the path point data to be edited in the path point data list area, and right-click. Click [Edit FLG2] in the context menu that appears to display an FLG2 batch editing screen.
(This screen can also be displayed by clicking [Edit] → [Batch Edit] → [FLG2 multiple rotation data] from the ribbon.)

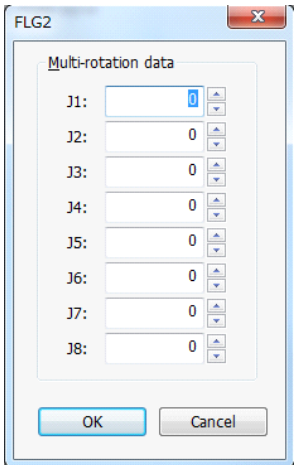


Fig.7-57:Edit FLG2 screen

By changing multi-rotation data for each axis and clicking the [OK] button, changes are updated to the selected path point. When the [Cancel] button is clicked, the FLG2 batch editing screen is closed without updating changes.

■Edit tolerance
Edits the tolerance of selected path point at one time.
To batch edit the amount of tolerance, select the path point data to be edited in the path point data list area, and right-click. Click [Edit tolerance] in the context menu that appears to display the Tolerance batch editing screen. (This screen can also be displayed by clicking [Edit] → [Batch Edit] → [Tolerance] from the ribbon.)

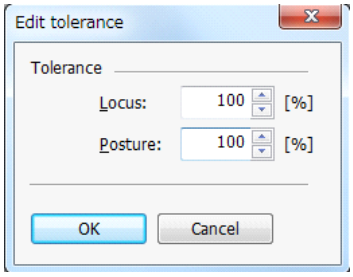


Fig.7-58:Edit tolerance screen

By changing the amount of tolerance and clicking the [OK] button, changes are updated to the selected path point. When the [Cancel] button is clicked, the Tolerance batch editing screen is closed without updating changes.

■Circular interpolation
Edits the arc designation of the selected path point at one time.
To batch edit the arc designation, select the path point data to be edited in the path point data list area, and right-click. By clicking [Circular interpolation] in the context menu that appears, arc designation for the selected path point turns ON and OFF. (The arc designation can also be edited by clicking [Edit] → [Batch Edit] → [Circular interpolation] from the ribbon.)

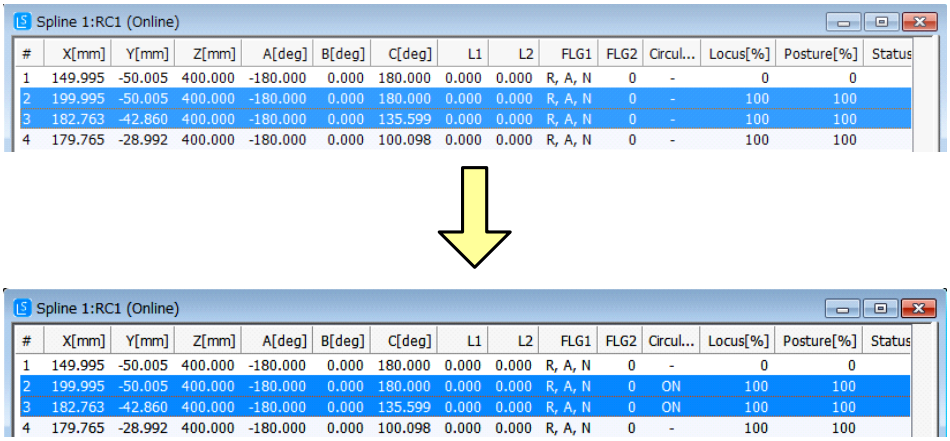


Fig.7-59:Circular interpolation

■Interpolation settings

Set the "Operation mode", "Cancel angle" and "Straight ratio" for the spline interpolation.

When the menu [Tool] → [Interpolation settings] is clicked from the ribbon, the interpolation settings set in the spline file on the active Spline editing screen appear in the setting screen.

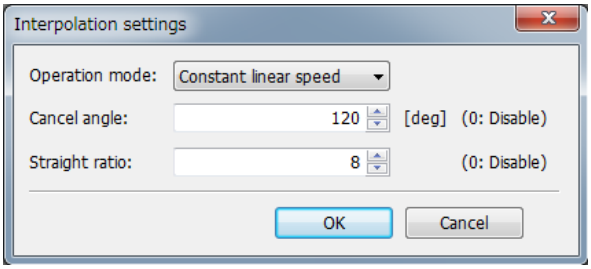


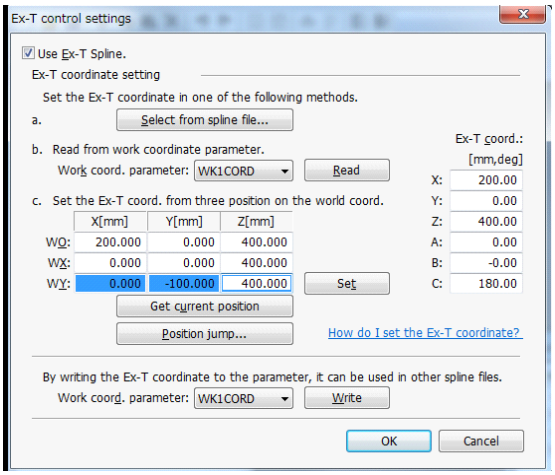
Fig.7-60:Interpolation settings screen

If the [OK] button is clicked after changing the settings, the changes will be saved and the setting screen will close. If the [Cancel] button is clicked, the setting screen will close without saving the changes. The setting range for each item is shown in Table 7-14.

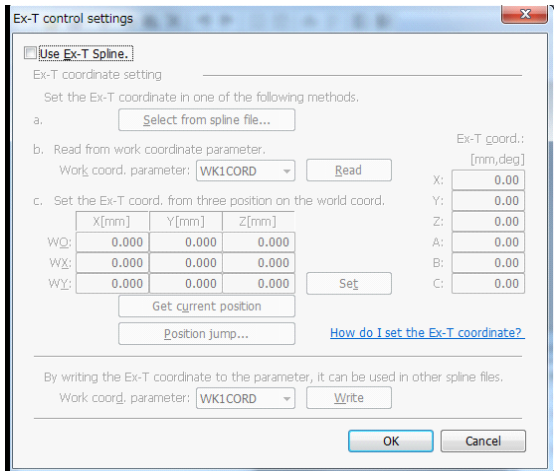
Table 7-14:Setting range for interpolation setting items

Item	Setting range	Default value
Operation mode	Constant linear speed/variable linear speed	Constant linear speed
Cancel angle	0 to 180 [deg]	120
Straight ratio	0 to 100 [fold]	8

■Ex-T control settings
Specify "Ex-T control settings" for spline interpolation.
By clicking [Tool] → [Ex-T control settings] from the ribbon, the content of Ex-T control settings set to the spline file at the active spline file editing screen is displayed in the settings screen.



Ex-T control setting is valid.



Ex-T control setting is invalid.

Fig.7-61:Ex-T control settings screen

■Entering a comment
A comment can be entered for the spline file. The details entered for the comment are displayed in the Comments filed of the spline file list. Make it easier to select the spline file by entering the work details or conditions, etc., as a comment.
When the menu [File] → [Properties] is clicked from the ribbon, the comment entered in the spline file on the active Spline editing screen appears on the Property screen.

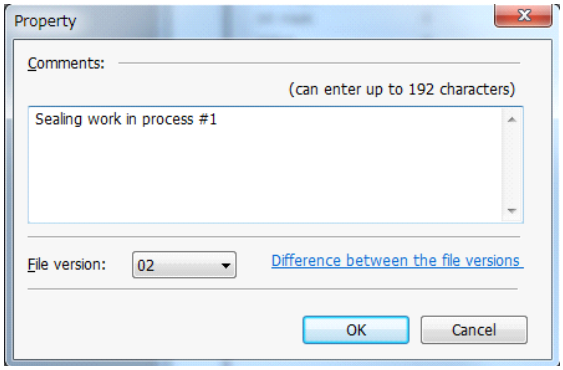


Fig.7-62:Property screen

Up to 192 characters can be entered for the comment. Note that a Line Return cannot be entered. The default state is blank.
When the comment is entered and the [OK] button is clicked, the details are saved and the Property screen will close. If the [Cancel] button is clicked, the Property screen will close without saving the entered comment.

■File version
The spline file version can be changed.
When the menu [File] → [Properties] is clicked from the ribbon, the comment entered in the spline file on the active Spline editing screen appears on the Property screen.

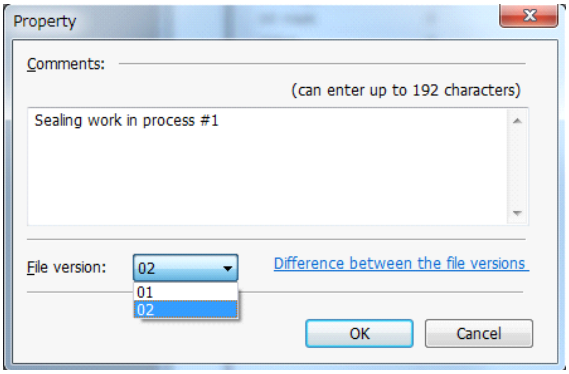


Fig.7-63:Property screen

By changing the file version and clicking the [OK] button, the content is recorded, and the Property screen closes. When the [Cancel] button is clicked, the Property screen is closed without recording the file version change.
* Refer to Page 687, "■ Arc designation" for details on differences between file versions.

(9) Saving the spline file
The details edited on the Spline File Edit screen are applied onto the spline file and saved.

■Click menu [File] → [Save as]
The Save spline file screen will open.

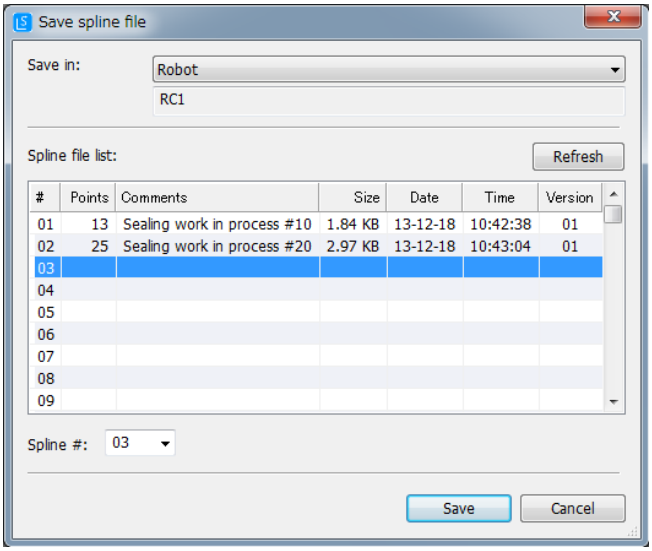


Fig.7-64:Save spline file screen

Select the destination from "Personal computer" or "Robot". ("Robot" cannot be selected in the offline mode.)
Next, select the number of the spline file to be saved from the Spline file list. Click the [Save] button. The edited details will be saved in the spline file, and the Save spline file screen will close.
If the [Cancel] button is clicked, the Save spline file screen will close without saving the details.

Spline file save destination folder

The spline files in the personal computer are managed in workspace project units. The save destination folder is Workspace work folder\project name\Spline.

Spline file name

The spline file name is determined by the following format.
SPLFILE**.SPL (Uppercase fixed)
The spline No. 01 to 99 is set in "**". This corresponds to the MvSpl/EMvSpl commands argument's <Spline No.>.
On the Spline File Edit screen, the file is displayed with a spline No. from 01 to 99, and there is no need to worry about the file name.

CAUTION Spline interpolation cannot be executed if the spline file name is changed with the RT ToolBox3 File Control screen or Microsoft Windows Explorer, etc.
Do not change the name of the spline file.

■Click menu [File] → [Save]
When editing an existing spline file, the edited details are saved in the target spline file.
When creating a new spline file, the same action as when [Save As] is clicked will take place.

■Click Spline File Edit Close button
When the [×] (Close) button at the upper right of the Spline File Edit screen is clicked, a confirmation dialog will appear.

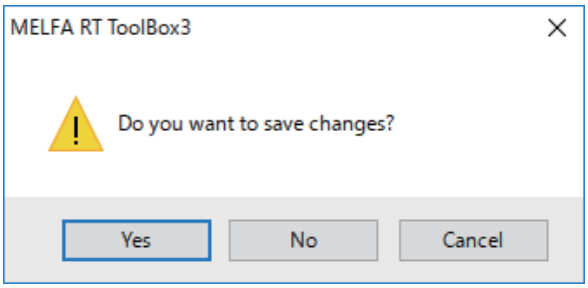


Fig.7-65:Save confirmation dialog

When the [Yes] button on the confirmation dialog is clicked, the edited details are applied on the spline file and saved.
When editing an existing spline file, the edited details are saved in the spline file and the Spline File Edit screen will close.
When creating a new spline file, the Spline file save screen will open.
When the destination spline No. is selected from the spline file list and the [Save] button is clicked, the edited details are saved in the spline file, and the Spline File Edit screen will close. When the [Cancel] button is clicked, the Save screen will close without saving the details and the Spline File Edit screen will open.

If the [No] button on the confirmation dialog is clicked, the edited details will be discarded and the Spline File Edit screen will close.
If the [Cancel] button on the confirmation dialog is clicked, the details will not be saved and the Spline File Edit screen will open.

CAUTION The details edited and saved on the Spline File Edit screen are not applied on the spline file unless they are saved. Save the edited changes into the spline file often while editing.

CAUTION Changes cannot be saved in a spline file being used for spline interpolation. If saving is attempted, error L2610 (can't change spline file) will occur. Do not perform save to a spline file being used.

(10) Deleting a spline file

Select the spline file to be deleted from the project tree, and right-click the mouse.
A context menu will appear, and a confirmation dialog will appear when [Delete] is clicked.

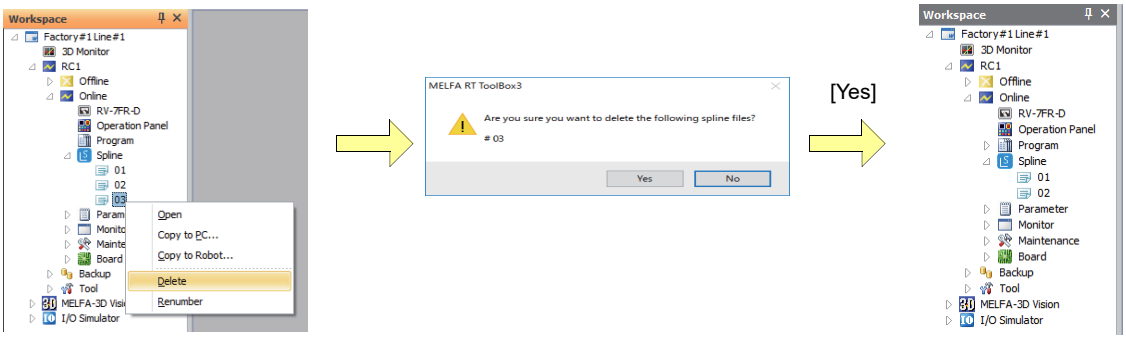


Fig.7-66:Deleting a spline file

When the [Yes] button is clicked, the designated spline file will be deleted.
When the [No] button is clicked, the deletion process will be canceled.

(11) Changing the spline No.

Select the target spline file from the project tree, and click the right mouse button.
A context menu will appear. When [Renumber] is clicked, a dialog for designating a new spline No. will appear.

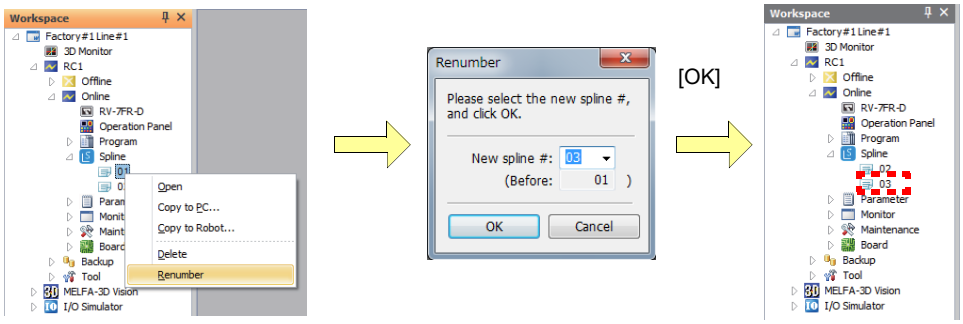


Fig.7-67:Changing the spline No.

When a new spline No. is designated and the [OK] button is clicked, the spline No. of the designated spline file will change.
The spline No. change process is canceled when the [Cancel] button is clicked.

(12) Copying the spline file

Spline files can be copied on the project tree.

By selecting the spline file to be copied on the project tree, and then dragging and dropping it to [Spline]/[Online]/[Offline] in the copy destination project tree, a dialog box appears to confirm the selection.

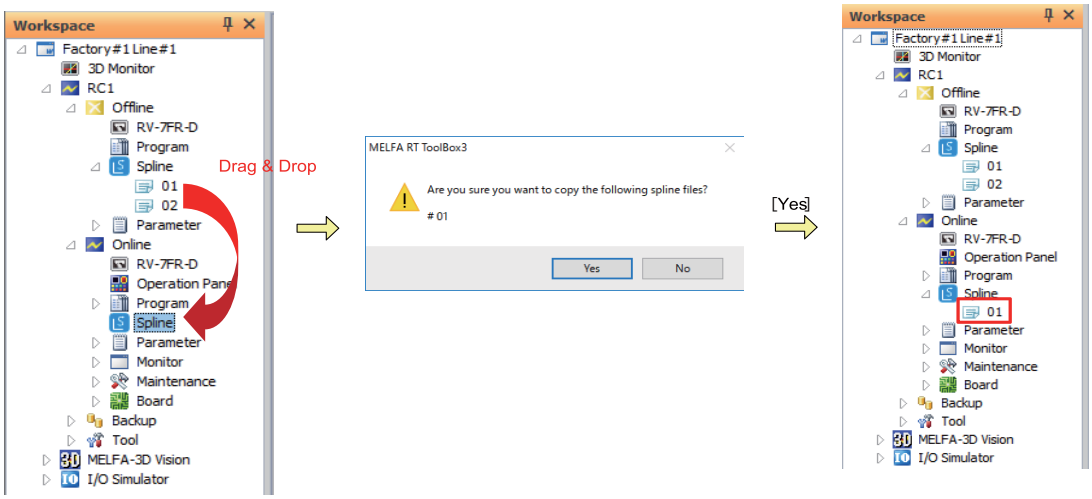


Fig.7-68:Copying the spline file

By clicking the [Yes] button, the specified spline file is copied.

By clicking the [No] button, copying is canceled.

If a spline file with spline No. that already exists in the copy destination is copied, an overwrite confirmation dialog box appears.

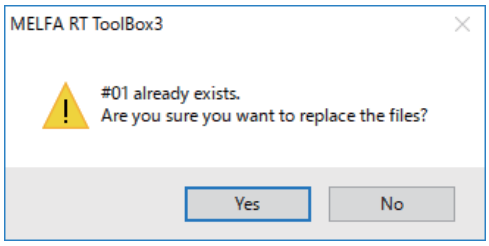


Fig.7-69:Overwrite confirmation dialog box

(13) Spline file manager

A Spline file manager screen has been added.

Select [Online] → [Spline] for projects for which spline file management is to be performed, and right-click.

Click [Spline file manager] in the context menu that appears to display the Spline file manager screen.

(Menu [File] → [Spline file manager] from the ribbon)

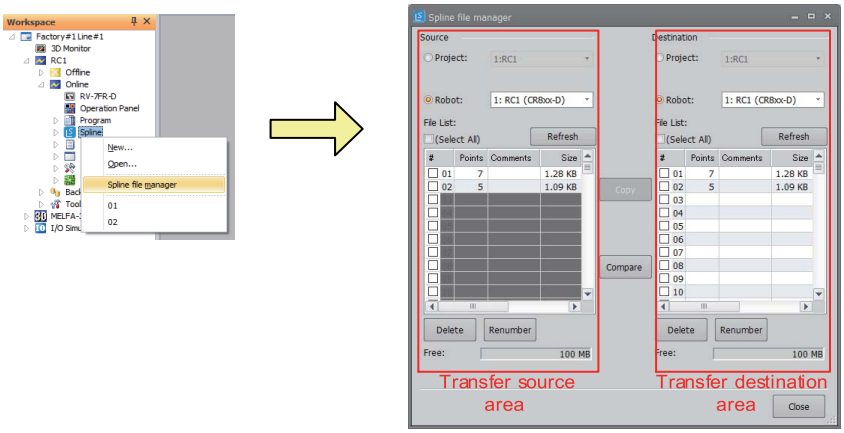


Fig.7-70:Spline file manager screen

The Spline file manager screen is made up of two areas used for copying and deleting spline files, and for changing spline numbers.

■Copy the spline file
By selecting the check box for spline file displayed at the source and then clicking the [Copy] button, the selected spline file is copied to the destination. If a spline file with the same number exists in the destination, the spline No. can be changed and copied.

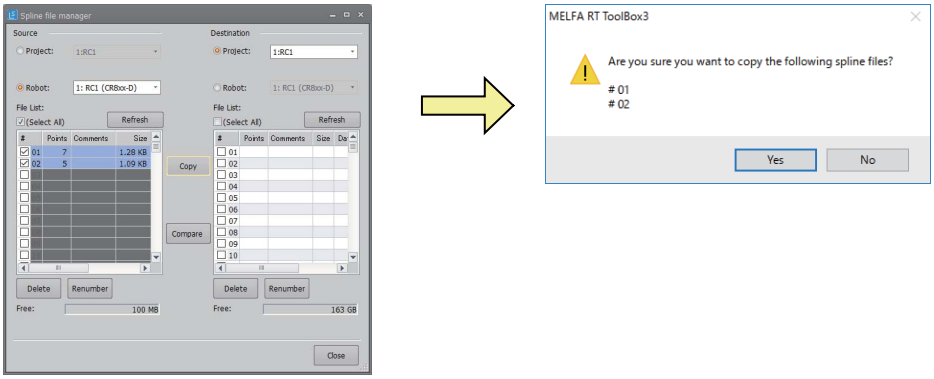


Fig.7-71:File copy on the Spline file manager screen

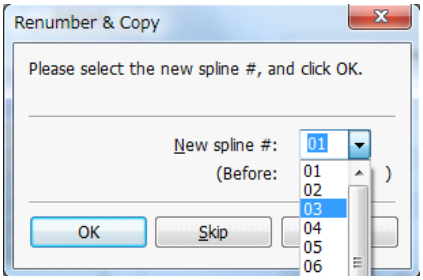


Fig.7-72:Renumber & Copy

■Delete the spline file
By selecting the check box of the file to be deleted from the source or destination spline file list and then clicking the [Delete] button, the selected spline file can be deleted.

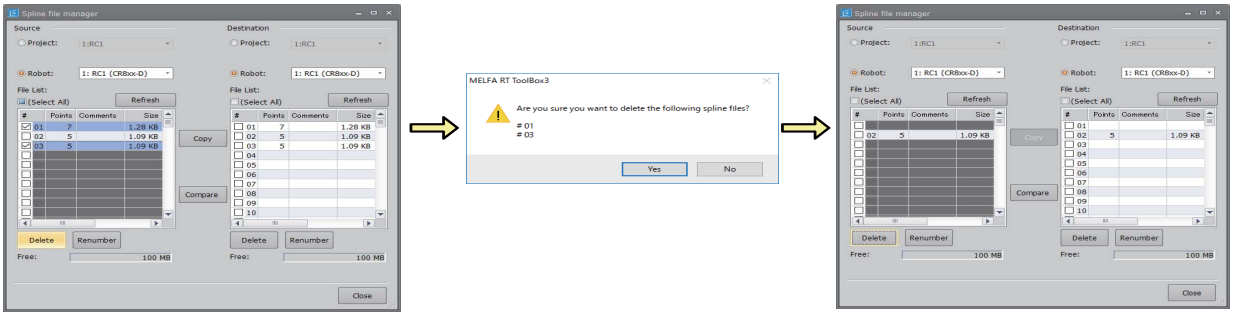


Fig.7-73:Delete the spline file

■Change the spline No.
By selecting the check box of the file to be changed from the source or destination spline file list and then clicking the [Renumber] button, the selected spline file No. can be changed.

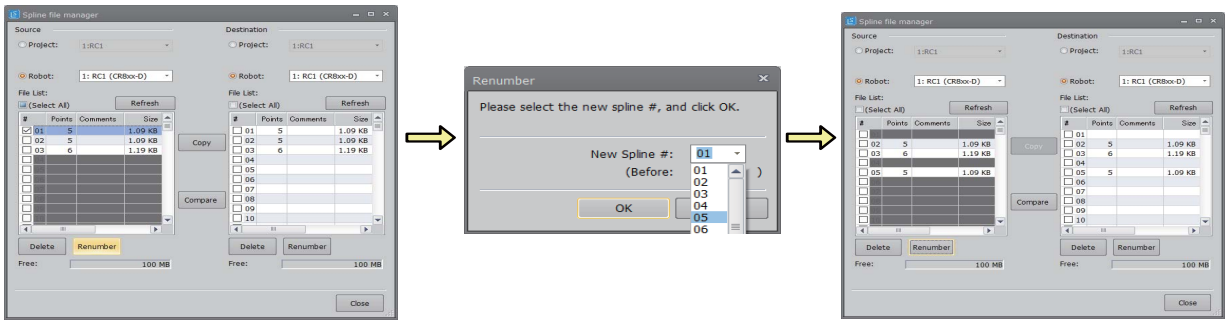


Fig.7-74:Change the spline No.

(14) Import/export function

If the path point data is descried with the specified format of CSV file, that file can be imported into the Spline File Edit screen. Using this function, position data extracted from a CAD drawing, for example, can be written into a CSV file and then registered in a batch as the path point data. In addition to CSV files, point sequence data (MXT files) output from RT ToolBox3 Pro can be imported.
The path point data registered from the Spline File Edit screen can be exported as a CSV file. Using this function, the details of the path point data can be confirmed even without starting the Spline File Edit screen.

■CSV file format
The data shown in Table 7-15 is described in the CSV file as the path point data. Commas (,) are used as delimiters. The data is written so that one path point data is shown on one line.

Table 7-15:Data described in CSV file

Data identification tag	Details
<X>	Designates the X-axis coordinate value of the path point. Unit: mm
<Y>	Designates the Y-axis coordinate value of the path point. Unit: mm
<Z>	Designates the Z-axis coordinate value of the path point. Unit: mm
<A>	Designates the A-axis coordinate value of the path point. Unit: degree
<B>	Designates the B-axis coordinate value of the path point. Unit: degree
<C>	Designates the C-axis coordinate value of the path point. Unit: degree
<L1>	Designates the L1-axis (additional axis 1) coordinate value of the path point. Unit: mm or degree
<L2>	Designates the L2-axis (additional axis 2) coordinate value of the path point. Unit: mm or degree
<FL1>	Designates the path point configuration flag value. Setting range: 0 to 7
<FL2>	Designates the path point multi-rotation flag value. Setting range: 0 to 4294967295
<Info>	Designates the validity of the arc designation Setting range: Designated (1), Not designated (0)
<Tol1>	Designates the position (path) tolerance amount with an integer. Setting range: 0 to 100 [%]

<table>
<tr><th>Data identification tag</th><th>Details</th></tr>
<tr><td><Tol2></td><td>Designates the posture tolerance amount with an integer.
Setting range: 0 to 100 [%]</td></tr>
<tr><td><Tol3></td><td>Not used. Describe as 100.</td></tr>
<tr><td><M_SplVar></td><td>Designates the value referred to by status variable M_SplVar as an integer.
Setting range: –1 to 32767 (–1 means not set.)</td></tr>
<tr><td><OutPort></td><td>Designates the head address of the output signal with an integer.
Setting range: –1 to 32767 (–1 means signal output invalid.)</td></tr>
<tr><td><Bit></td><td>Designates the bit width of the signal to be output.
Setting range: 1, 8, 16, 32 [bit]</td></tr>
<tr><td><Mask></td><td>Designates the mask pattern of the bit to enable the signal to be output with a hexadecimal.
<table>
<tr><th>Bit width</th><th>Setting range [hex]</th></tr>
<tr><td>1</td><td>0, 1</td></tr>
<tr><td>8</td><td>0 to FF</td></tr>
<tr><td>16</td><td>0 to FFFF</td></tr>
<tr><td>32</td><td>0 to FFFFFFFF</td></tr>
</table></td></tr>
<tr><td><OutVal></td><td>Designates the data to be signal output with a hexadecimal.
<table>
<tr><th>Bit width</th><th>Setting range [hex]</th></tr>
<tr><td>1</td><td>0, 1</td></tr>
<tr><td>8</td><td>0 to FF</td></tr>
<tr><td>16</td><td>0 to FFFF</td></tr>
<tr><td>32</td><td>0 to FFFFFFFF</td></tr>
</table></td></tr>
<tr><td><Pulse></td><td>Designates the pulse output.
Setting range: Designated (1), Not designated (0)</td></tr>
</table>

An example of the CSV file is shown below.

```
<X>,<Y>,<Z>,<A>,<B>,<C>,<L1>,<L2>,<FL1>,<FL2>,<Info>,<Tol1>,<Tol2>,<Tol3>,<M_SplVar>,<OutPort>,<Bit>,<Mask>,<OutVal>,<Pulse>
300.00, 125.00, 325.00, 180.00,   0.00, 180.00, 0.00, 0.00, 7, 0, 0, 100, 100, 100,    0, 10100, 8,  a, ff, 0
250.00, 100.00, 325.00, 180.00,   0.00, 180.00, 0.00, 0.00, 7, 0, 0, 100, 100, 100,    1,    -1, 1,  0,  0, 0
250.00,   0.00, 325.00, 180.00,  15.00, 180.00, 0.00, 0.00, 7, 0, 1, 100, 100, 100,   10, 10100, 8,  5, ff, 1
300.00, -50.00, 325.00, 165.00,   0.00, 180.00, 0.00, 0.00, 7, 0, 1, 100, 100, 100,  100,    -1, 1,  0,  0, 0
350.00,   0.00, 325.00, 180.00, -15.00, 180.00, 0.00, 0.00, 7, 0, 1, 100, 100, 100,   -1,    -1, 1,  0,  0, 0
300.00,  50.00, 325.00,-165.00,   0.00, 180.00, 0.00, 0.00, 7, 0, 1, 100, 100, 100,   -1,    -1, 1,  0,  0, 0
250.00,   0.00, 325.00, 180.00,  15.00, 180.00, 0.00, 0.00, 7, 0, 1, 100, 100, 100,    0, 10100, 8, ff,  0, 0
250.00,-100.00, 325.00, 180.00,   0.00, 180.00, 0.00, 0.00, 7, 0, 0,  50, 100, 100,   -1,    -1, 1,  0,  0, 0
300.00,-125.00, 325.00, 180.00,   0.00, 180.00, 0.00, 0.00, 7, 0, 0,  50, 100, 100,   -1,    -1, 1,  0,  0, 0
```

Fig.7-75:Example of CSV file

In the first line, always describe the Table 7-15 "Data identification tag" in order from <X>. The file cannot be imported if the "Data identification tag" is not described, or if the data on each line is insufficient (there should be 20 data items).

■MXT file format

MXT files contain map data output by the MELFA-Works function in RT ToolBox3 Pro.

MXT files are created using the following procedure.

* For more information on using MELFA-Works, refer to "MELFA-Works Instruction Manual (BFP-A3554)".
* The procedure differs in places between spline and Ex-T spline, and therefore caution is advised.

1) MELFA-Works startup
   Start the MELFA-Works function.

2) Grasping workpieces
   * This is only required for Ex-T spline. For spline, perform the procedure in "4)Creating paths".

Attach the picking hand to the robot.
From the ribbon, select [MELFA-Works] → [Hand Setting] to display the Hand Setting screen. Select picking hand as the connected hand and then click the Hand State button to pick up the workpiece.

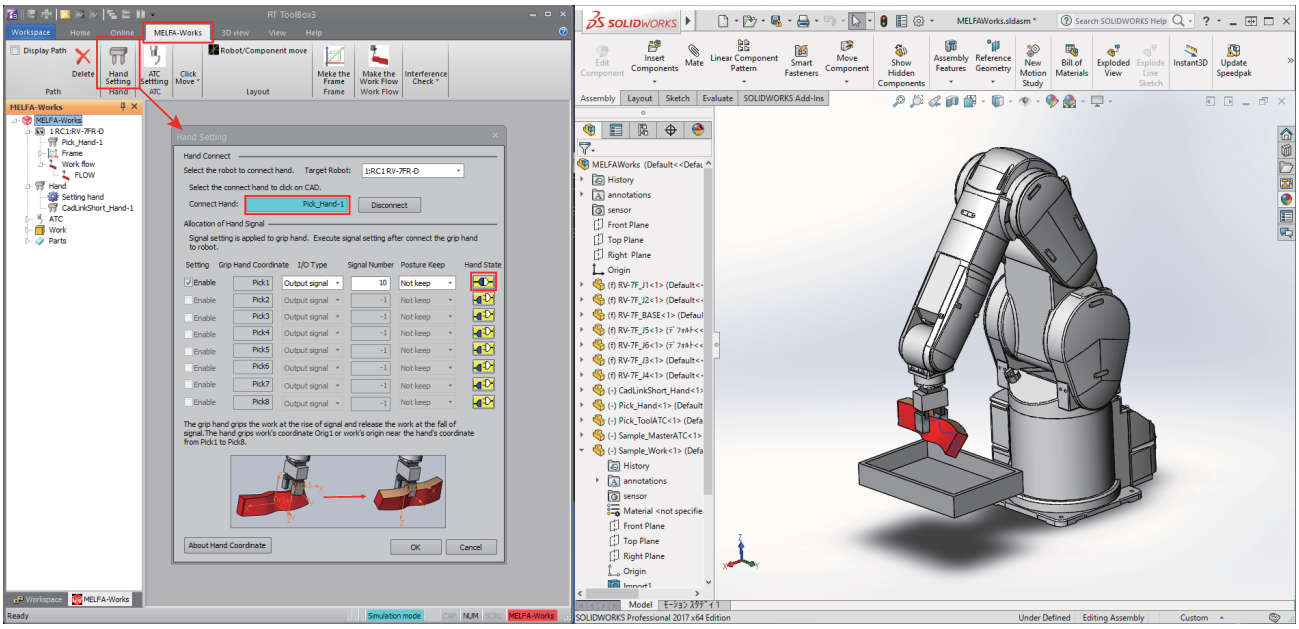


Fig.7-76:Grasping workpieces

3) Teaching the grasp position
* This is only required for Ex-T spline. For spline, perform the procedure in "4)Creating paths".
From the ribbon, select [MELFA-Works] → [Make the Work Flow] to display the Make Teaching Point and Path screen. To edit an existing work flow, display the same screen from the MELFA-Works tree. On the Teach tab, select [Add] → [Get Current Position] to teach the robot the position at which to pick up workpieces.
* The data taught here is the grasp position.
* Ensure not to move the workpiece after teaching the grasp position.

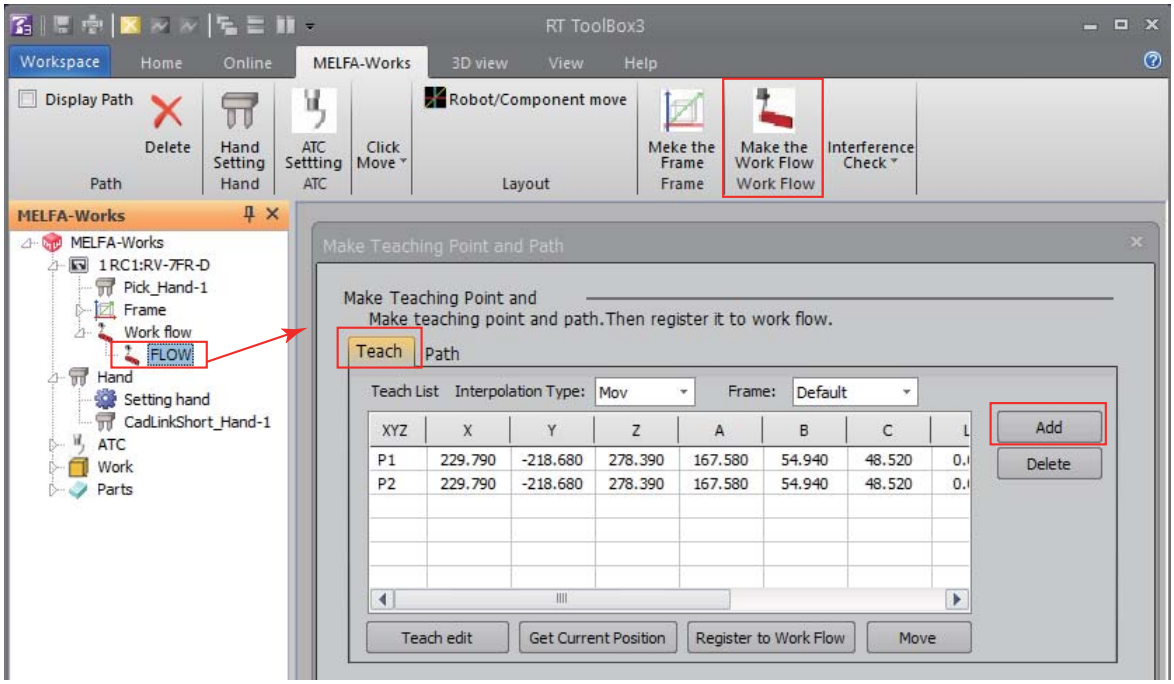


Fig.7-77:Teaching the grasp position

4) Creating paths

Switch to the CAD link hand to create path data.

Click [Work-flow] to display the Work-flow screen.

Select the Path tab and click the [Add] button to add a process path. Select the added process path and then click the [Edit Path Data] button to display the Edit path data screen. Select the workpiece work path and then click the [Add] button to register the process path.

* With Ex-T spline, reverse the Z axis according to the Ex-T coordinate.

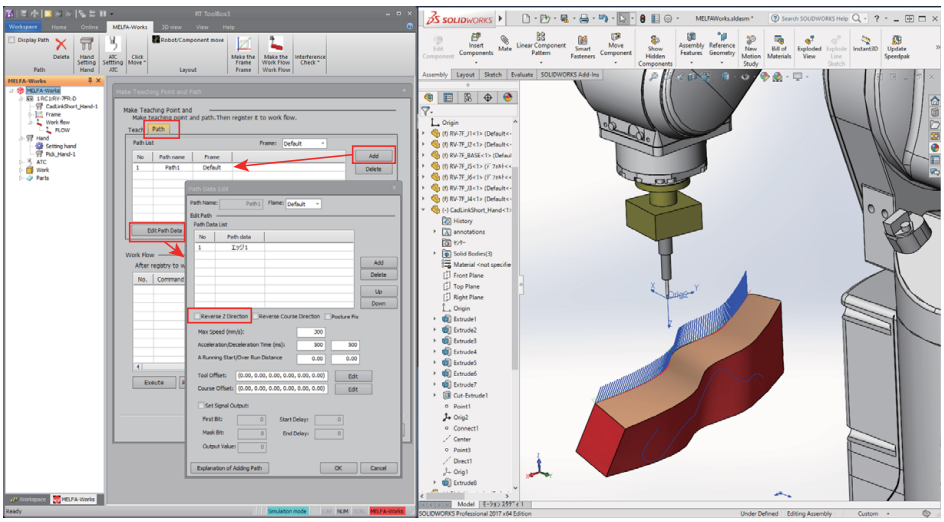


Fig.7-78:Creating paths

5) Creating the work-flow

Create the work-flow.

* The work-flow procedure differs between spline and Ex-T spline, and therefore caution is advised.

Click [Work-flow] to display the Work-flow screen.

Click the flow [Add] button to add a flow.

Select the Work-flow screen "Path" tab.

Select a process path and then click the [Register to Work Flow] button to add the selected path to the flow.

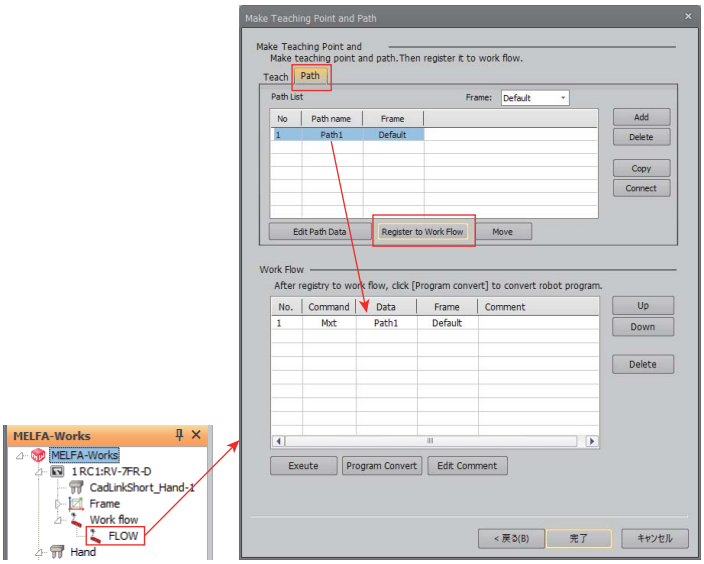


Fig.7-79:Creating flows (processing path registration)

With Ex-T spline, it is necessary to register the grasp position taught at "3)Teaching the grasp position" in the work-flow.

Select the "Teaching" tab at the Work-flow screen.

Select the registered pickup position and then click the [Register to Work Flow] button to add the selected position data to the flow.

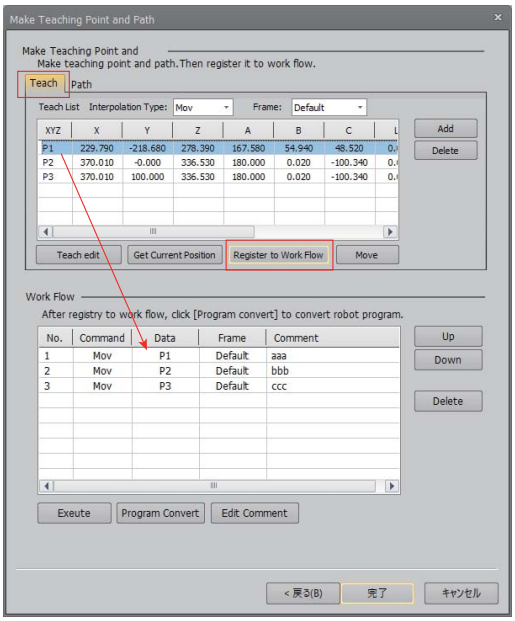


Fig.7-80:Flow creation (grasp position registration)

6) Outputting MXT files

Output an MXT file for the created work-flow.

From the Make Teaching Point and Path screen, click the [Program Convert] button to display the Program Convert screen. Enter a name for the output program and then the [OK] button to output a program into the current workspace and create an MXT file in the following folder.

MXT file output location:

<Workspace>\<Project name>\<MELFA-Works>\Program name***.MXT

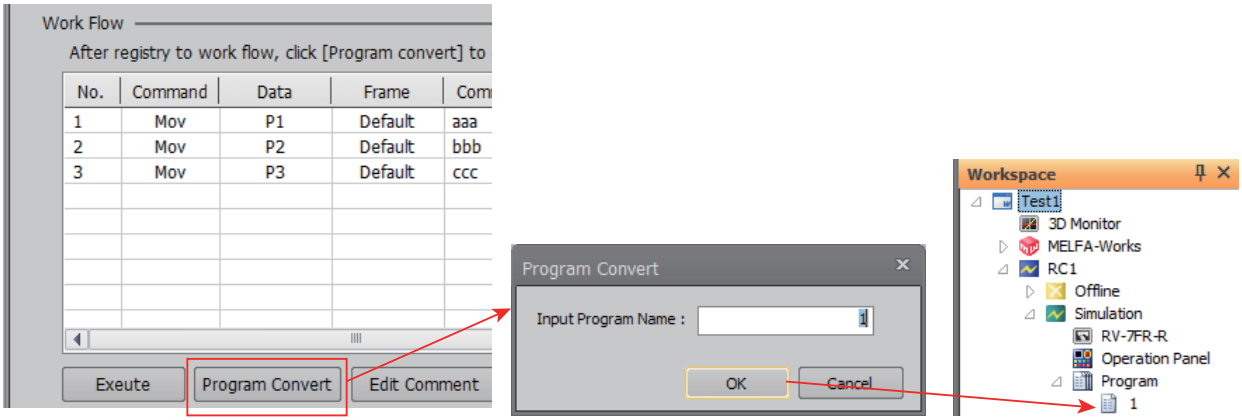


Fig.7-81:Outputting MXT files

■Import
Importing is possible not only for CSV files, but also for MXT files.

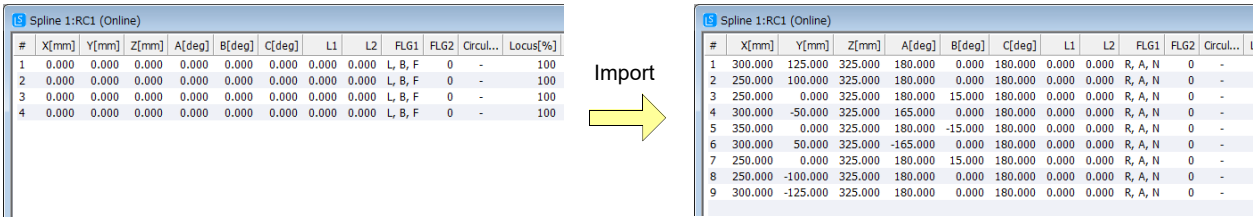


Fig.7-82:Import

CAUTION When the import function is used, all path point data registered in the Spline File Edit screen will be overwritten by the newly imported path point data.

• CSV file
From the ribbon, select [File] → [Import] to display the file selection dialog. When the required CSV file is selected and opened, its contents will be imported into the active Spline File Edit screen.
* With Ex-T spline, Ex-T control settings must be specified after importing data.

• MXT file
From the ribbon, select [File] → [Import] to display the file selection dialog.
When the required MXT file is selected and opened, the Import settings screen appears.

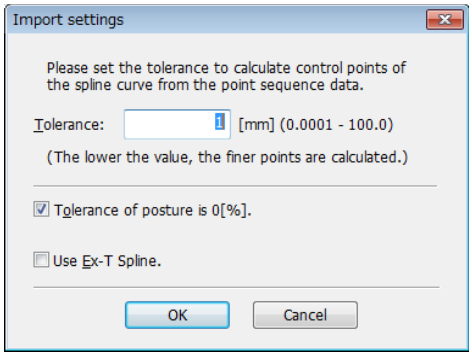


Fig.7-83:Import setting screen

By setting the tolerable error (mm) for the point sequence data and spline curve and then clicking the [OK] button.
By selecting the [Tolerance of posture is 0 [%]] check box on the Import settings screen, the data is imported with a 0% tolerance for the posture at each path point. To operate the robot in the posture imported by MXT file, select the check box. (The check box is selected in the initial state.)
With spline, by clicking the [OK] button, the content of the MXT file is imported to the active spline file editing screen. (The smaller the tolerable error, the more detailed the calculated path points.)

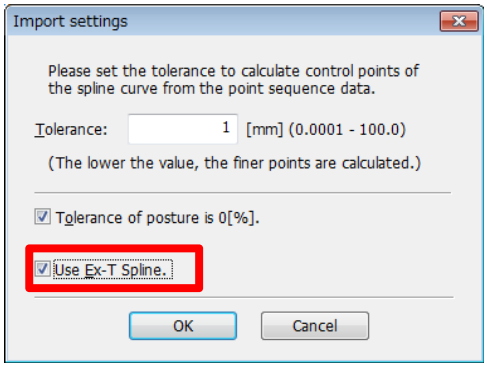


Fig.7-84:Import setting screen (in Ex-T spline)

With Ex-T spline, by selecting the [Use Ex-T control] check box and clicking the [OK] button, the Ex-T control settings screen for setting the grasping position and Ex-T coordinates appears.
By clicking the [Read position variable] button, the Select robot program screen appears, allowing the program created when outputting the MXT file to be selected.

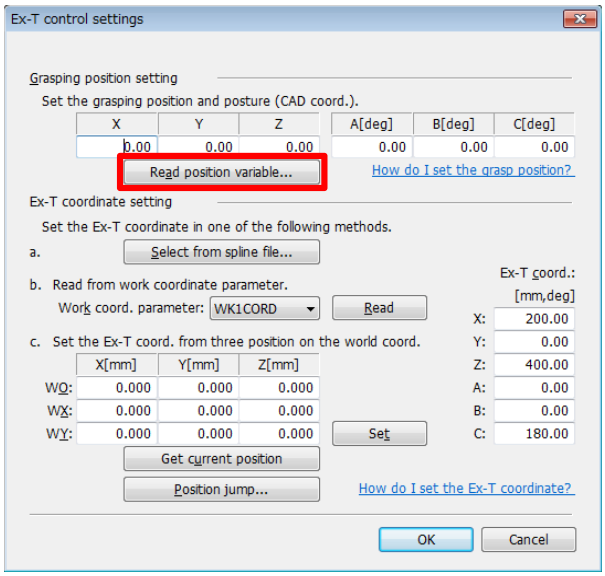


Fig.7-85:Ex-T control settings screen (settings)

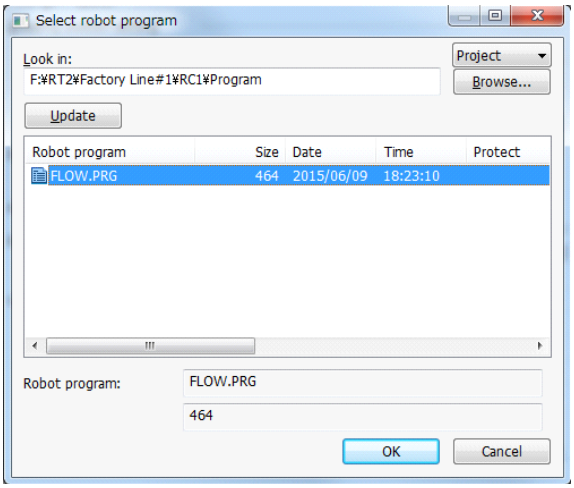


Fig.7-86:Select robot program screen

By selecting the program and clicking the [OK] button, the Select XYZ Position Variables screen appears.

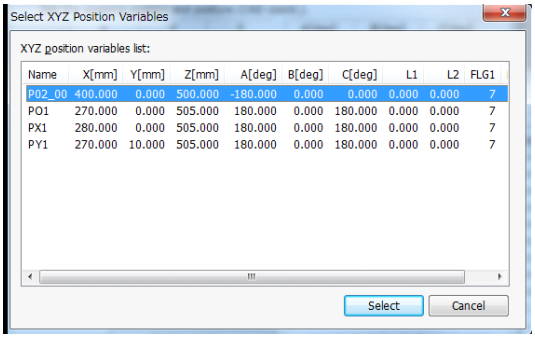


Fig.7-87:Select XYZ Position Variables screen

By selecting the position data (grasp position data taught when creating MXT file) to be read and clicking the [OK] button, the grasp position is set.

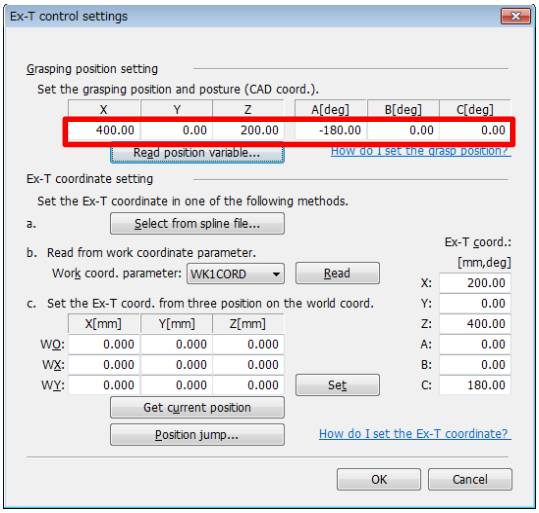


Fig.7-88:Ex-T control settings (Setting the grasping position)

By setting the Ex-T coordinates and clicking the [OK] button, the content of the MXT file is converted to Ex-T spline path point data and then imported.

CAUTION The corresponding point sequence data type is XYZ data only.
The movement speed (acceleration/deceleration time, maximum speed) in the MXT file header information is not imported, and therefore it must be updated to the robot program MvSpl command argument.

CAUTION By reducing the tolerance values, the accuracy of the calculated path points will improve, however, the distance between path points will become shorter, thus curtailing speed.

■Export
A dialog for designating the CSV file save destination opens when the menu [File] → [Export] is clicked. When the save destination folder and CSV file name is designated and saved, the path point data details on the active Spline File Edit screen is exported to the CSV file.

(15) Auxiliary editing functions
■Cut & Paste
The designated path point data is cut and pasted into another path point data.
Select the source path point data in the path point data list area, and click menu [Edit] → [Cut] from the ribbon. Select the desired position in the path point data at which to paste the cut data. From the ribbon, select [Edit] → [Paste] to paste the cut path point data in the same order as displayed when the [Cut] operation. At the same time, the cut source path point data will be deleted from the list.

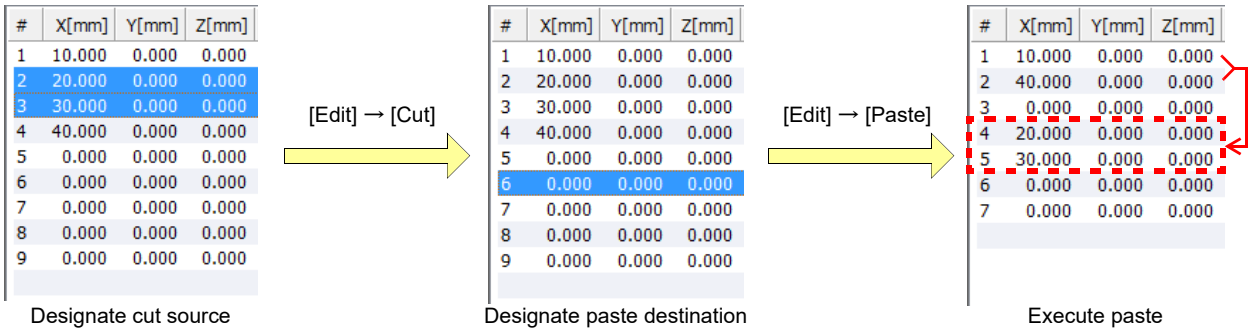


Fig.7-89:Cut & Paste

When there are not enough path point data items at paste destination

If the number of subsequent path point data items at the paste destination is low compared to the number of path point data items designated as the cut source, the path point data with no paste destination will not be pasted.

■Copy & Paste
The designated path point data is copied and pasted into another path point data.
Select the desired path point data in the path point data list and select [Edit] → [Copy] from the ribbon. Select the path point data of the copy destination and click [Edit] → [Paste] from the menu. The path point data recorded by the copy operation will be copied to the path point data of the copy destination.

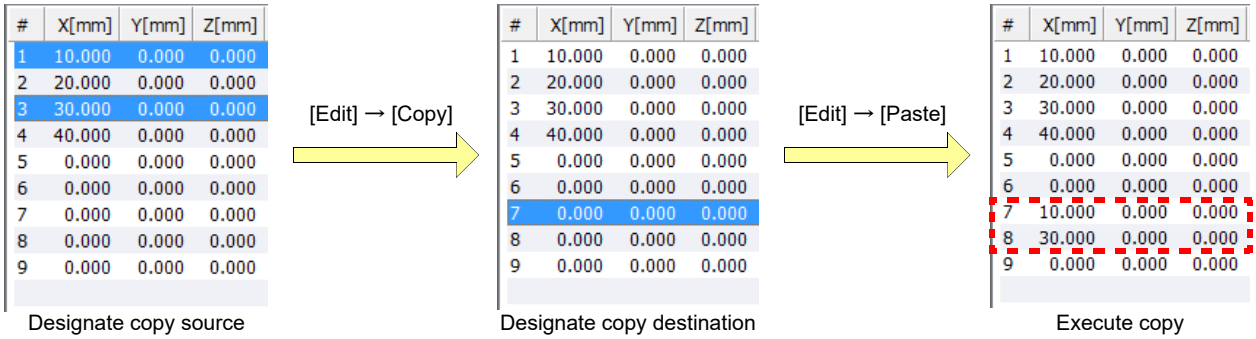


Fig.7-90:Copy & Paste

When there are not enough path point data items at copy destination

If the number of subsequent path point data items is low at the copy destination is low compared to the number of path point data items designated as the cut source, the path point data with no copy destination will not be copied.

■Copy & Paste (Position variable)

The designated position data is copied and pasted into path point data in the spline file.

Select the desired position data from the robot program and select [Edit] → [Copy - Position variable] from the ribbon. Select the desired position in the path point data at which to paste the copied data. From the ribbon, select [Edit] → [Paste - Position variable] to paste the copied position data into the path point data in the same order as displayed when copied.

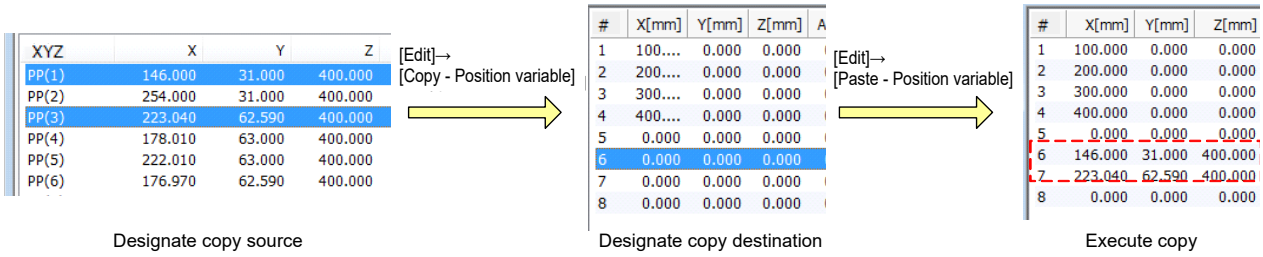


Fig.7-91:Copy & Paste (Position variable)

■Insert copied path point

The details of the designated path point data are inserted into another line as new path point data.

Select the desired path point data in the path point data list and select [Edit] → [Copy] from the ribbon. Select the desired position in the path point data at which to paste the copied data. From the ribbon, select [Edit] → [Insert copied path point] to paste the copied path point data, in the same order as displayed when the [Copy] operation, directly before then path point data selected as the copy location.

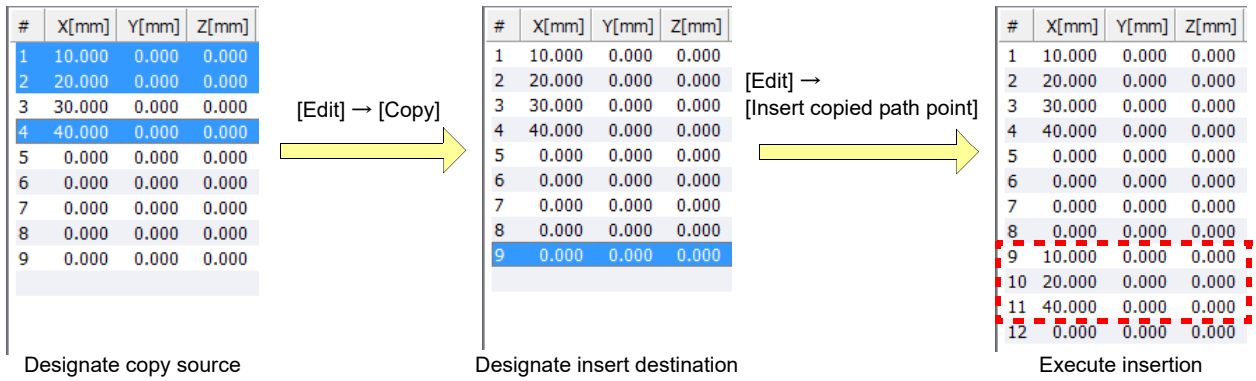


Fig.7-92:Insert copied path point

■Undo
From the ribbon, select [Edit] → [Undo] to undo the last edit and restore the data to the previous state before the edit was made. The following operations can be undone.
- Path point data add, insert, delete
- Path point data change (applied to edited details)
- Path point data cut & paste, copy & paste, insert copied path point
- Import
- Position adjustment function
- Frame transformation (execution of transformation)

■Redo
From the ribbon, select [Edit] → [Redo] to remake the edit that was undone with the [Undo] feature.

(16) Displays the spline curve

Displays the spline curve.
From the ribbon, select [Tool] → [Spline curve] to display the spline curve, configured for the spline file being edited on the 3D Monitor screen, on the 3D Monitor screen.

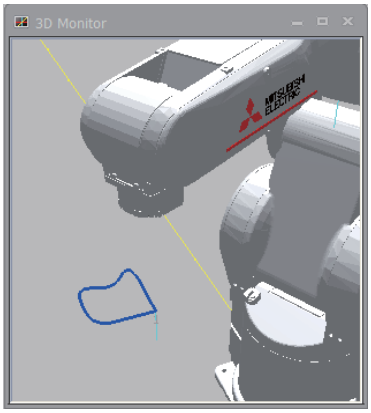


Fig.7-93:Displays the spline curve

Refer to the "RT ToolBox3/RT ToolBox3 mini Instruction Manual" for details on using the 3D Monitor screen.

(17) Checking edited data

■Path point check
Check the distance between path points, amount of change in posture, and the value of the structure flag for each path point beforehand.
From the ribbon, select [Tool] → [Path point check] to display the Path point check screen for the active Spline editing screen.

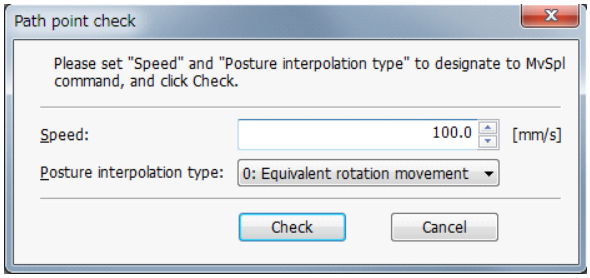


Fig.7-94:Path point check screen

By setting the speed and posture interpolation type specified with the MvSpl and EMvSpl commands and then clicking the [Check] button, a check of each path point is performed. A completion dialog box appears

if successful. An error dialog box appears if an error occurs. Refer to the separate "Troubleshooting" manual for error details.
* The path point check is performed at speed with override at 100%.

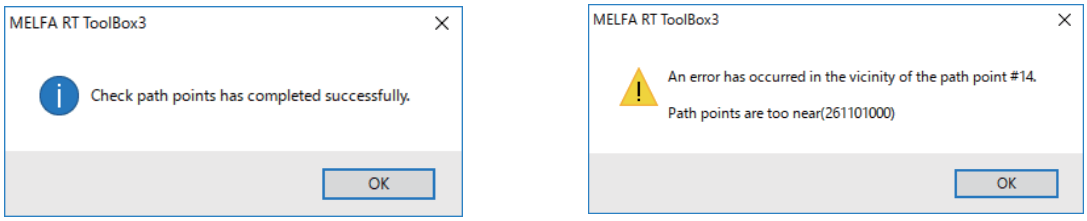


Fig.7-95:Path point check complete dialog (left)/Error dialog (right)

■Motion range check
Check whether the robot is able to move to each path point beforehand.
From the ribbon, select [Tool] → [Motion range check] to execute the operation for each path data of the active Spline editing screen. A completion dialog box appears if successful. An error dialog box appears if an error occurs. Refer to the separate "Troubleshooting" manual for error details.
* The robot trajectory when performing the motion range check is not guaranteed.

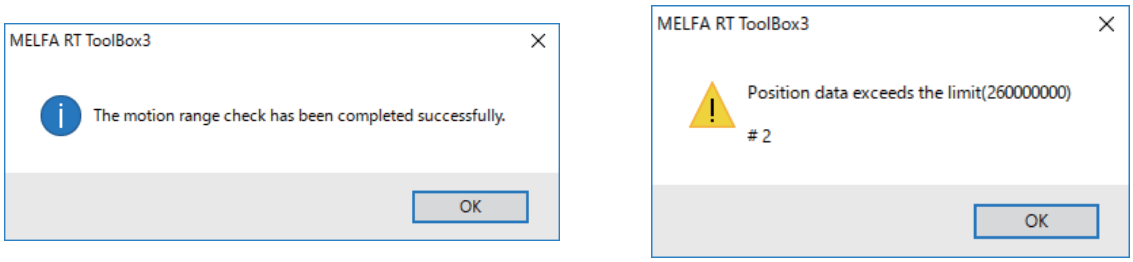


Fig.7-96:Motion range check complete dialog (left)/Error dialog (right)

■Calculate maximum speed
Calculate the maximum speed at which the motion range check is possible from the distance between path points without an error occurring when executing spline interpolation commands (MvSpl command, EMvSpl command) beforehand.
From the ribbon, select [Tool] → [Calculate maximum speed] to display the maximum speed calculation result on the active Spline editing screen.

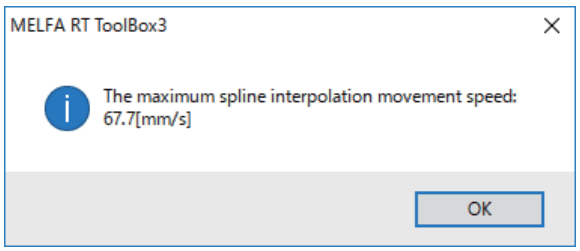


Fig.7-97:Maximum speed calculation result

(18) Robot program

■Convert to program

Create a robot program using the path point data registered in the spline file.

From the ribbon, select [Tool] → [Robot program] → [Convert to program] to display the robot program dialog on the active Spline editing screen.

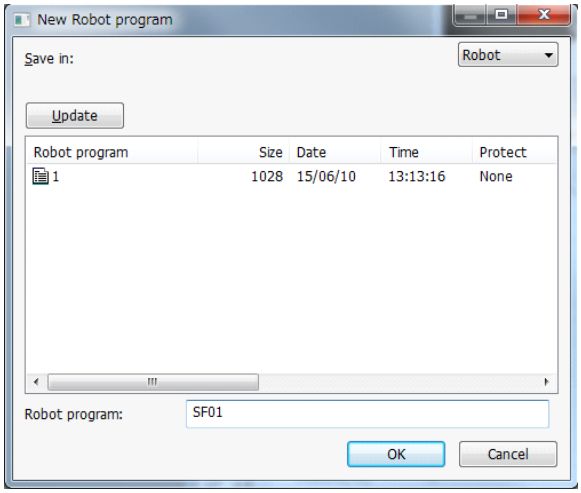


Fig.7-98:New Robot program screen

By entering the name of the robot program to be saved and clicking the [OK] button, the Speed setting dialog box appears.

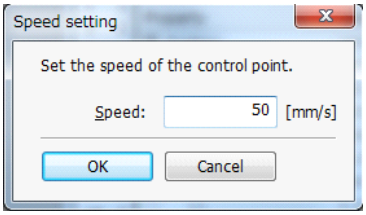
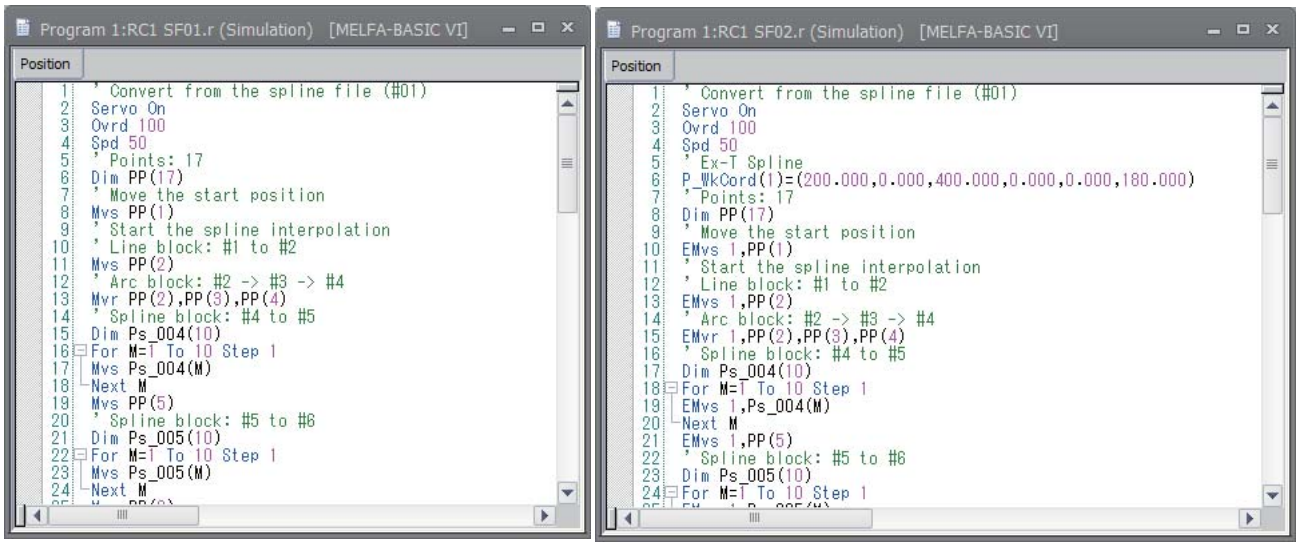


Fig.7-99:Speed setting dialog

By setting the control point speed and clicking the [OK] button, the content of the spline file is converted to a robot program.



Spline interpolation　　Ex-T spline interpolation

Fig.7-100:Program created by each interpolations

■Import position variables

Robot program position data can be imported as spline file path point data.

From the ribbon, select [Tool] → [Robot program] → [Import position variables] to display the Select robot program screen used to load a program onto the active Spline editing screen.

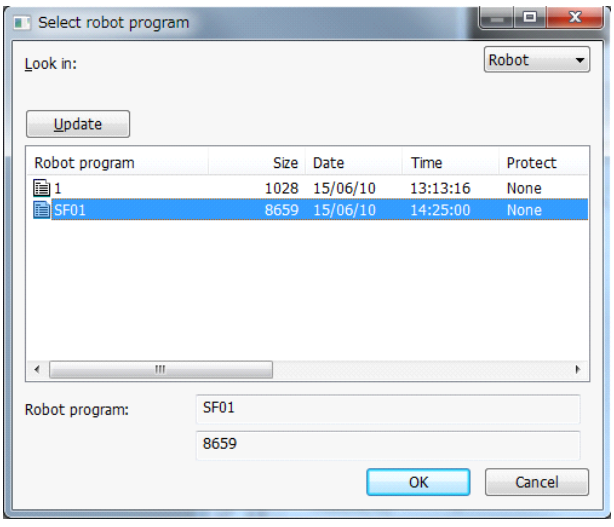


Fig.7-101:Select robot program

By selecting the robot program for which position data is to be imported and clicking the [OK] button, position data is imported to the active Spline editing screen.

* Position data that can be imported is PP (n) (n: path point number) or P* (*: path point number).
* If no path point corresponding to the active spline file exists, create new path points. (Default values are used for other than position data.)

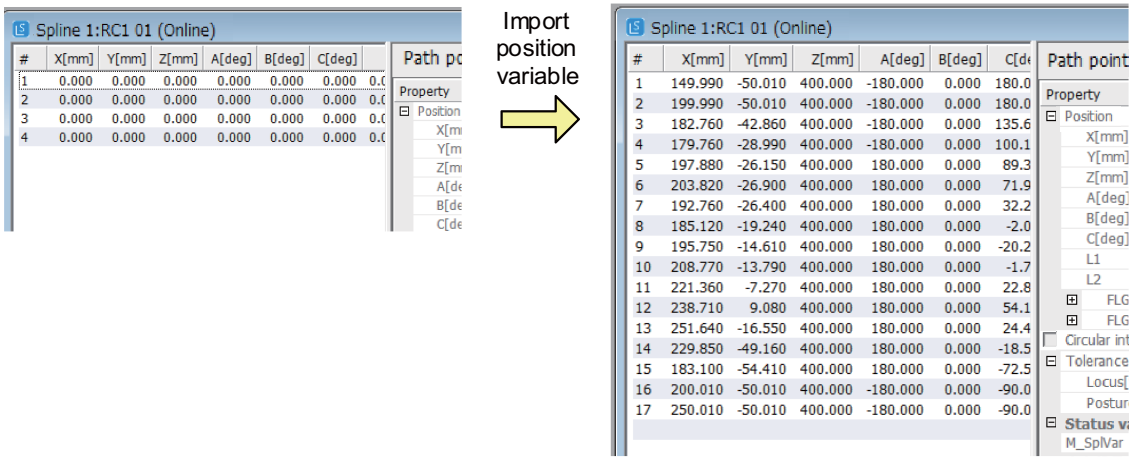


Fig.7-102:Import position variable

7.2.6 Creating the robot program

The robot program is created in the program edit screen of RT ToolBox3 or the teaching pendant.
To execute spline interpolation, use the commands and robot status variables listed in Table 7-16 and Table 7-18. Refer to sections Page 170, "4.12 Detailed explanation of command words" and Page 346, "4.13 Detailed explanation of Robot Status Variable".

Table 7-16:Commands used with spline interpolation

Command word	Explanation	Reference page
MvSpl (Move Spline)	Executes spline interpolation.	283
EMvSpl (E Move Spline)	Executes Ex-T spline interpolation.	238
SetCalFrm (Set Calibration Frame)	Sets the coordinate system used with frame transformation.	322

Table 7-17:Functions used with spline interpolation

Function name	Explanation	Reference page
SplPos	Substitute the path point data registered in the specified spline file for the position variable.	475
SplSpd	Substitute the speed at which an L2611 (path point is too close) error does not occur when executing the MvSpl/EMvSpl commands for the speed variable based on the specified spline file path point data.	476
SplECord	Substitute the Ex-T coordinate system origin data registered in the specified spline file for the position variable.	474

Table 7-18:Robot status variables related to spline interpolation

Variable name	No. of array elements	Details	Attribute Note1)	Data type	Reference page
M_SplPno	Mechanism No. (1 to 3)	The number of the path point passed through most recently is returned.	R	Integer	404
M_SplVar	Mechanism No. (1 to 3)	The setting value of the most recent path point data is returned. The value can be rewritten.	RW	Integer	405

Note1) R ..... Read-only.
RW ..... Read and write possible.

■Sample program

The path 1 path point data is registered in spline file 05. Spline interpolation is executed on path 1 and path 2 by using frame transformation. At this time, the path point data numerical settings are used by the slot 2 program to turn the output signal No. 100 and No. 101 signals ON and OFF during the movement.

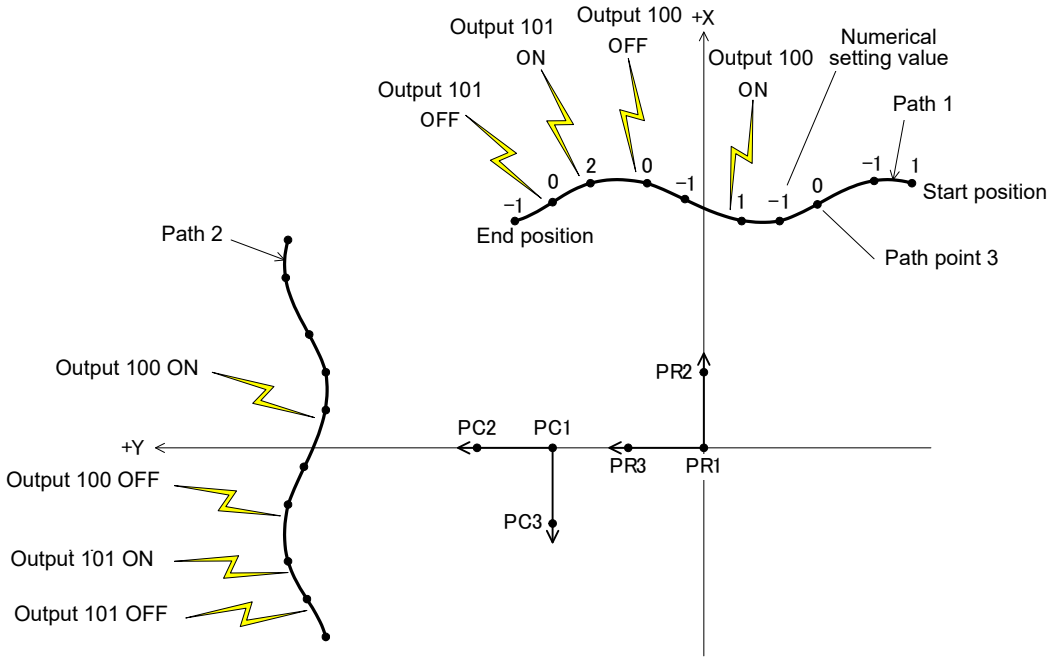


Fig.7-103:Movement of sample program

Slot 1

```
Ovrd 100
Mov P1                                  ' Move to wait position (P1) with joint interpolation
 '
Fine 200                                ' Change positioning pulse to 200
Spd 100                                 ' Set linear interpolation speed to start position to 100mm/s
M_00=1                                  ' Turn slot 2 synchronization flag ON
MvSpl 5, 50, 10                         ' Spline interpolate path 1 with acceleration/decelerate distance
                                          50mm/s
M_00=0                                  ' Turn synchronization flag OFF
Fine 0                                  ' Disable positioning pulse designation
Mvs P1                                  ' Move to wait position with linear interpolation
 '
PR1=(0, 0, 0, 0, 0, 0)(0, 0)            ' Set the reference coordinate system for frame transformation
                                          (PR1 to PR3)
PR2=(20, 0, 0, 0, 0, 0)(0, 0)
PR3=(0, 20, 0, 0, 0, 0)(0, 0)
PC1=(0, 40, 0, 0, 0, 0)(0, 0)           ' Set the position of the transformed reference coordinate system
                                          (PC1 to PC3)
PC2=(0, 60, 0, 0, 0, 0)(0, 0)           ' (90-degree turn around Z axis in respect to pre-transformation,
PC3=(-20, 40, 0, 0, 0, 0)(0, 0)         ' coordinate system moved 40mm in Y axis direction)
SetCalFrm PR1, PR2, PR3, PC1, PC2, PC3  ' Calculate and set coordinate system for frame transformation
 '
Fine 200
M_00=1
MvSpl 5, 50, 10, 2                      ' Execute frame transformation on spline No. 5 path point
                                        ' and spline interpolation path 2
M_00=0
Fine 0
Mvs P1
End
```

Slot 2

```
Def IO PORT1=Byte, 100, &H03              ' Assign output signal 100 and 101 to variable PORT1
M_SplVar=0                                ' Reset M_SplVar value to 0
Wait M_OO=1                               ' Wait for spline interpolation to start
*L1:If M_SplPno<3 Then GoTo *L1           ' Wait for path point 3 to be passed
*L2
Select M_SplVar
Case 1                                    ' Pass through path point for which numerical setting value is set
                                            to 1
PORT1=1                                   ' Turn output signal 100 ON
Break
Case 2                                    ' Pass through path point for which numerical setting value is set
                                            to 2
PORT1=2                                   ' Turn output signal 101 ON
Break
Default                                   ' Numerical setting value is not 1 or 2
PORT1=0                                   ' Turn output signal 100 and 101 OFF
Break
End Select
If M_OO=1 Then Goto *L2                   ' Repeat until spline interpolation is finished
End
```

7.2.7 Confirming the movement

Using the RT ToolBox3 simulation function, confirm the spline interpolation movement of the created robot program. If the movement differs from the required movement, review and correct the path point data settings and robot program.

Refer to the "RT ToolBox3/RT ToolBox3 mini Instruction Manual" for details on using the simulation function. Note that the simulation function cannot be used with RT ToolBox3 mini.

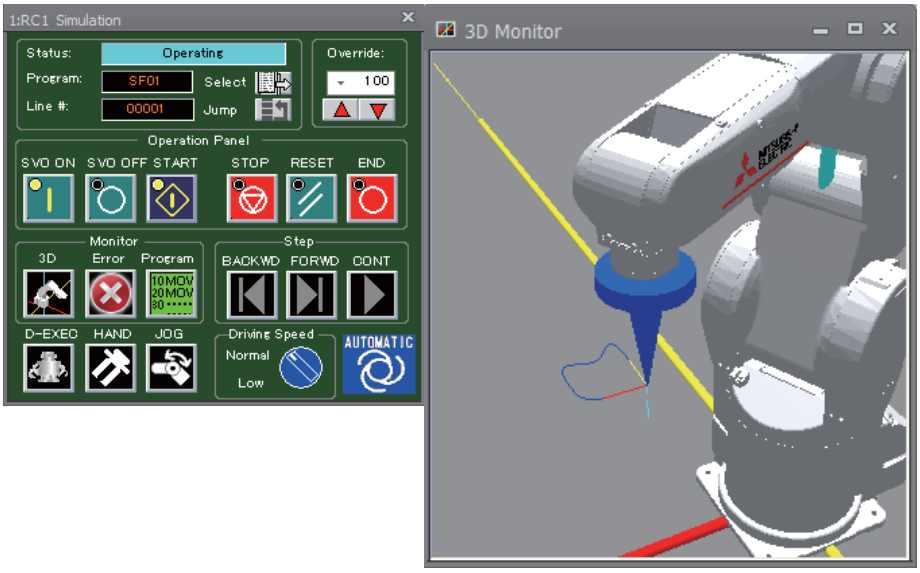


Fig.7-104:Simulation function

Robot program and spline file used for simulation

When spline interpolation is simulated, the robot program and spline file must be registered into the virtual controller in the RT ToolBox3. If the robot program and spline file are not registered into the [Online] section of the project starting the simulation, copy them into the virtual controller with the following method.

- Robot program: Use the RT ToolBox3 program control.
- Spline file: Open the target file in the Spline File Edit screen, and save with the procedure give on Page 719, "(9) Saving the spline file" or copy with the procedure gibe on Page 722, "(12) Copying the spline file".

7.2.8 Saving in the robot controller

Save the robot program and spline file into the robot controller.
Refer to Page 719, "(9) Saving the spline file" for details on saving the spline file.

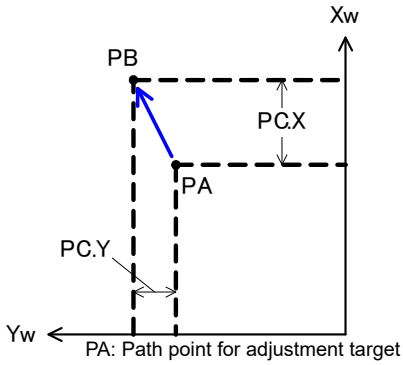
7.2.9 Adjustment work

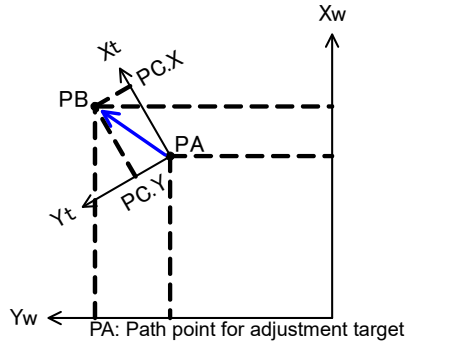
Using the actual system, confirm the spline interpolation movement with debugging (step feed).
If the movement differs from the required movement, review and revise the path point data and robot program. To revise the path point data, import the spline file into the RT ToolBox3 Spline File Edit screen and change the path point data setting values. Then, export the file to the controller.
The "position adjustment function", "frame transformation function", and "position jump function" provided in the RT ToolBox3 for adjusting the position data, and the parameter SPLOPTGC (active gain control gain compensation rate) are explained in this section.

(1) Position adjustment function

The same type of adjustment as the MELFA-BASIC VI position data's relative calculation can be applied on the path point data's robot position. The two compensation methods shown in Table 7-19 can be used.

Table 7-19:Position adjustment method

Adjustment method	Explanation
Vector sum operation (P+P)	The adjustment data value is added (addition of each coordinate element) in respect to the path point data's robot position data. The adjustment goes along the world coordinate system. The configuration flag, multi-rotation flag and additional axis data are not changed from the original value.  PA: Path point for adjustment target PB: Adjustment results PC: Adjustment data Vector sum calculation (P+P)

Adjustment method	Explanation
Vector product operation (P×P)	The adjustment data values are multiplied in respect to the path point data's robot position data. (Position after adjustment = Path point position × adjustment data) The adjustment follows the tool coordinate system. The configuration flag, multi-rotation flag and additional axis data are not changed from the original value.  PA: Path point for adjustment target PB: Adjustment results PC: Adjustment data Xt-Yt: Orientation of tool coordinates at PA Vector product calculation (P×P)

To use the position adjustment function, select [Tool] → [Position adjustment] from the ribbon. The Position adjustment screen opens when [Position adjustment] is clicked.

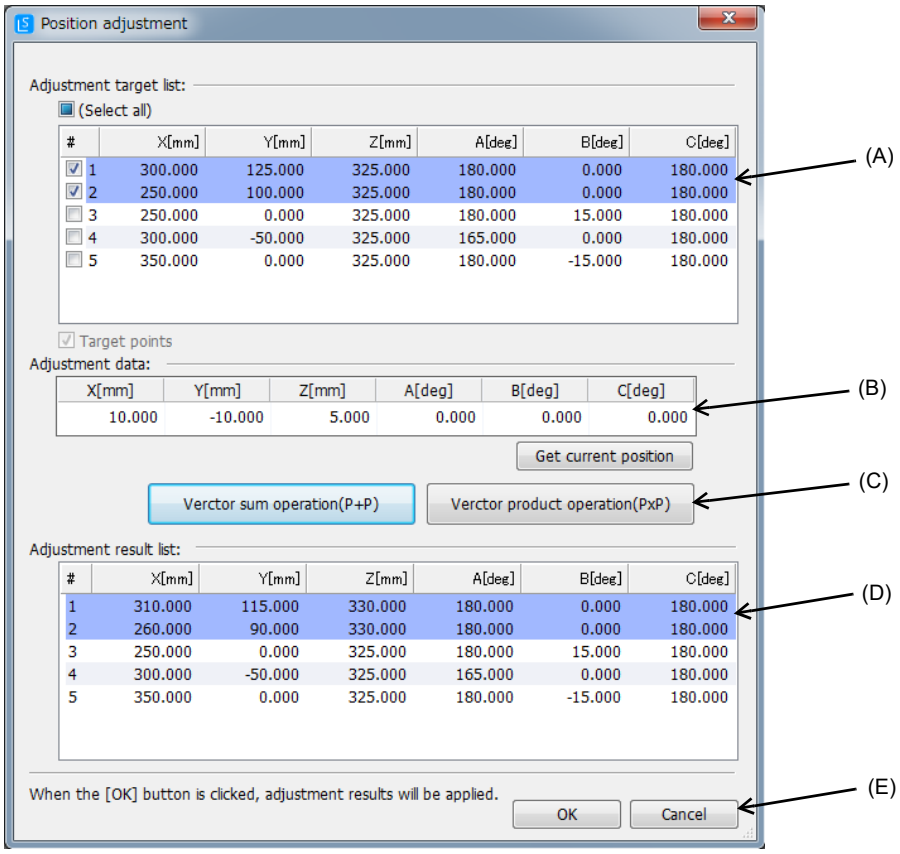


Fig.7-105:Position adjustment screen

The position is adjusted with the following steps.

(A) The path point data for the Spline File Edit screen currently active is displayed in the "Adjustment target list". Select and check the path point data for which the position is to be adjusted. Multiple path point data items can be selected.

(B) Set the adjustment data.
If a controller is connected, the [Get current position] button will appear. When this button is clicked, the robot's current position is imported as the adjustment data.
(C) Select the adjustment method and click the button.
(D) The adjusted path point data is displayed in the "Adjustment result list". The path point data, not selected as an adjustment target, remains the unchanged.
(E) When the [OK] button is clicked, the adjustment results will be applied onto the path point data on the Spline File Edit screen, and the Position adjustment screen will close.
When the [Cancel] button is clicked, the adjustment results will be discarded and the Position adjustment screen will close.

(2) Frame transformation function

From the ribbon, select [Tool] → [Frame transformation] to display the screen used to perform frame conversion on the path point data on the active Spline editing screen.
(Refer to Page 693, "(5) Frame transformation" for details on frame transformation.)

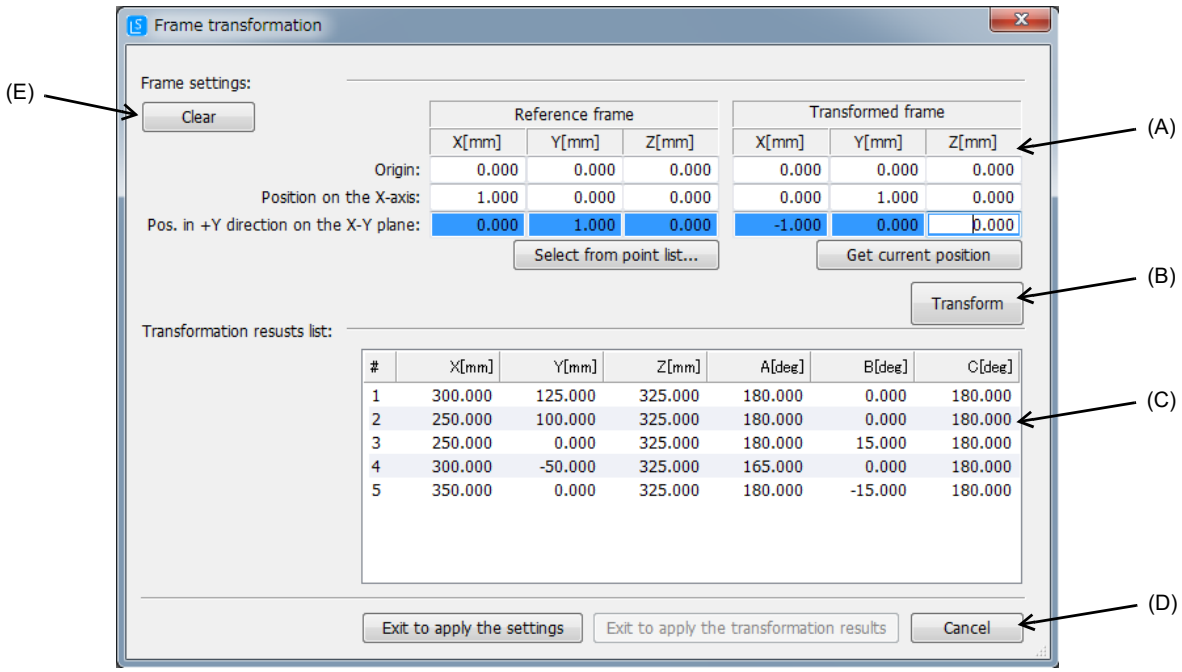


Fig.7-106:Frame transformation screen

■Setting the coordinate system
In the (A) section of Fig. 7-106, set the X, Y and Z axis coordinate values for the three positions (origin, position on X axis, position in +Y direction on X-Y plane) that define the reference coordinate system and transformed reference coordinate system.
For the reference coordinate system, the coordinate values can be selected from registered path point data that appears when the [Select from point list...] button is clicked.

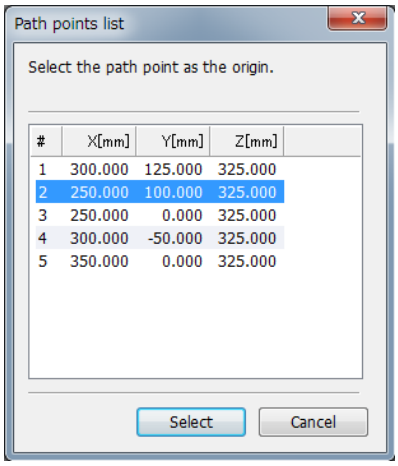


Fig.7-107:Selecting from Path points list

If a controller is connected, the [Get current position] button will appear. When this button is clicked, the robot's current position will be set to the selected transformed reference coordinate system.

■Executing frame transformation with RT ToolBox3
When the [Transform] button in Fig. 7-106 (B) is clicked after setting the coordinate system, frame transformation is applied on all path point data. The results are shown in Fig. 7-106 (C) "Transformation results list".
When frame transformation is executed, the [Exit to apply the transformation results] button in Fig. 7-106 (D) is enabled.

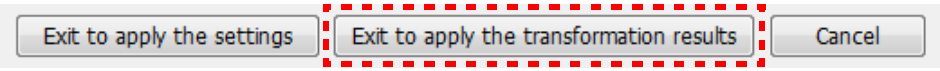


Fig.7-108:[Exit to apply the transformation results] button

When this button is clicked, the transformation results will be applied on the path point data in the Spline File Edit screen, and the Frame transformation will close.
When the [Cancel] button is clicked, the transformation results will be aborted and the Frame transformation screen will close.
If the coordinate system settings are not correct (on same point or 3 points on the same line), the message "Invalid the frame settings." will appear, and the frame transformation cannot be executed.

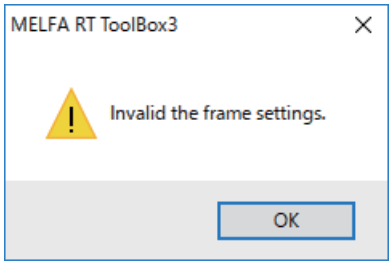


Fig.7-109:Dialog when coordinate system settings are not correct (when executing transformation)

■Saving coordinate system in spline file
When the Fig. 7-106 (D) [Exit to apply the settings] button is clicked after setting the coordinate system, the position data that defines the coordinate system will be saved in the Spline File Edit screen, and the Frame transformation screen will close.
If "1" is set for the MvSpl/EMvSpl command's argument <Frame transformation>, the coordinate system set here will be used for frame transformation during spline interpolation.
When the [Cancel] button is clicked, the transformation results will be aborted and the Frame transformation screen will close. (If the coordinate system has already been saved in the Spline File Edit screen, the details will not be changed.)

If the coordinate system settings are not correct (on same point or 3 points on the same line), a dialog indicating that the coordinate system settings are incorrect will appear.

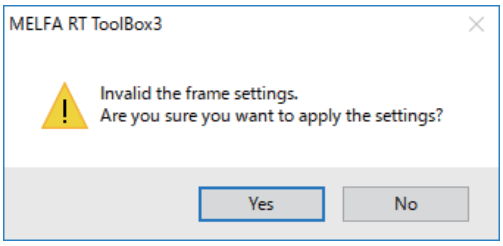


Fig.7-110:Dialog when coordinate system settings are not correct (when saving coordinate system)

When the [Yes] button is clicked, the settings will be saved in the Spline File Edit screen and the Frame transformation screen will close.
When the [No] button is clicked, the settings will not be saved in the Spline File Edit screen and the Frame transformation screen will reappear.

■Clearing the coordinate system settings
Click the Fig. 7-106 (E) [Clear] button to clear the coordinate system settings saved in the Spline File Edit screen. A confirmation dialog will appear when this button is clicked.

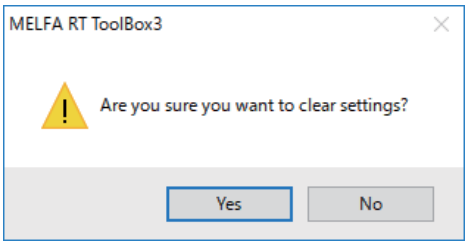


Fig.7-111:Coordinate system clear confirmation dialog

When the [Yes] button is clicked, the coordinate system settings saved in the Spline File Edit screen are cleared.
When the [No] button is clicked, the settings are not changed, and the Frame transformation screen reappears.

(3) Position jump

Jump to the selected path point position.
By opening the program in the debug state and then clicking the [Position] button at the bottom of the path point data list area, the Position jump screen appears. (This screen can also be displayed by clicking [Tool] → [Position Jump] from the ribbon.)

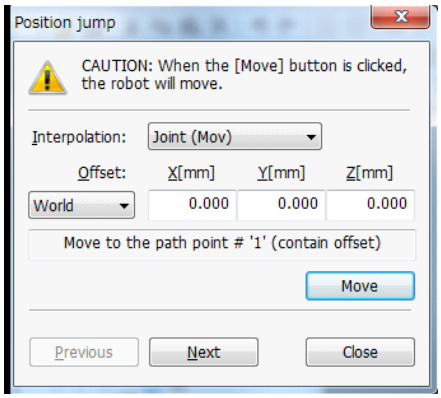


Fig.7-112:Position jump screen

By clicking the [Next] ([Previous]) button, the target path point changes to the next (previous) point.
By clicking the [Move] button, the robot moves to the selected path point position (including offset amount) with the specified interpolation movement.

⚠CAUTION The robot moves at the same speed as that for the actual system, and therefore a safety check should be conducted around the robot
Always use the robot in such a way that the EMERGENCY STOP button can be pushed at any time.

If the applicable project is in offline mode, the [Position jump] button does not appear. (Similarly, [Position Jump] will not be available under [Tool] from the ribbon.)

(4) Parameter SPLOPTGC

The robot's control characteristics applied when moving with spline interpolation can be adjusted with parameter SPLOPTGC.
Usually it is not necessary to change the value from the default setting value.
For the movement path at the curve section, to further improve the inner turn in respect to the command, increase the setting value. When the setting value is increased the path accuracy may be improved.
Conversely, to suppress robot sway or vibration, reduce the setting value. The state may be improved.

Table 7-20:Parameter SPLOPTGC

Parameter	Parameter name	No. of arrays No. of characters	Detailed explanation	Factory setting
Gain compensation rate for active gain control	SPLOPTGC	Integer 1	The robot's control characteristics (servo responsiveness) during spline interpolation can be adjusted. The path accuracy may be improved if the setting value is increased, and the vibration may be suppressed if the setting value is decreased. The setting range is 1 to 200[%]. * If the value is too large, the motor may oscillate or the robot may vibrate. Conversely, if the value is too small, the robot may not move and a servo error may occur. Change the value gradually while confirming the behavior, and set the value within a range that does not cause a problem. * This does not function unless the force sensor control is valid. * The power does not need to be reset after setting this parameter.	100

7.2.10 High speed spline interpolation command

(1) Outline

A higher spline interpolation operation speed can be specified using the SPLHSPMD parameter.

Table 7-21:Parameter SPLHSPMD

Parameter	Parameter name	No. of arrays No. of characters	Detailed explanation	Factory setting
High speed spline interpolation	SPLHSPMD	Integer 1	The spline interpolation high speed mode can be enabled and disabled. (Disabled = 0 / Enabled = 1) * The power does not need to be reset after setting this parameter.	0

(2) Required software versions

The high speed spline interpolation command function can be used with the following robot controller and RT ToolBox3 software versions.

Table 7-22:Software versions that support the high speed spline interpolation command function

Device	Controller	RT ToolBox3
Compatible software version	Ver. A3a or later	Ver. 1.31H or later

(3) Specifications

The relationship between the commanded speed and the required distance between path points (on the table on Page 685, "(4) Check related to path points", changes depending on the setting value of parameter SPLHSPMD as shown below.

When high speed mode is enabled, a higher operation speed can be specified with the SPLHSPMD parameter for the same distance between path points.

Table 7-23:Command speed and required distance between path points

Command speed [mm/s]	Required distance between path points	
	SPLHSPMD=0	SPLHSPMD=1
10	0.6 mm or more	0.08 mm or more
20	1.2 mm or more	0.16 mm or more
50	3.0 mm or more	0.4 mm or more
100	6.0 mm or more	0.8 mm or more
200	12.0 mm or more	1.6 mm or more
500	30.0 mm or more	4.0 mm or more

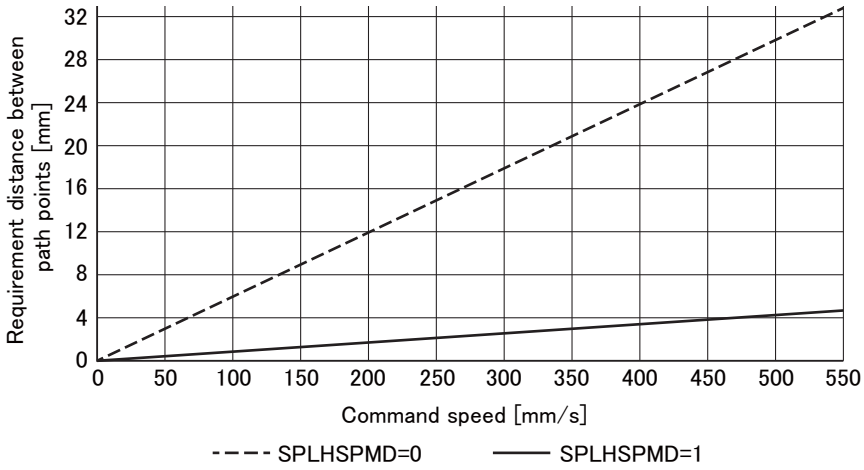
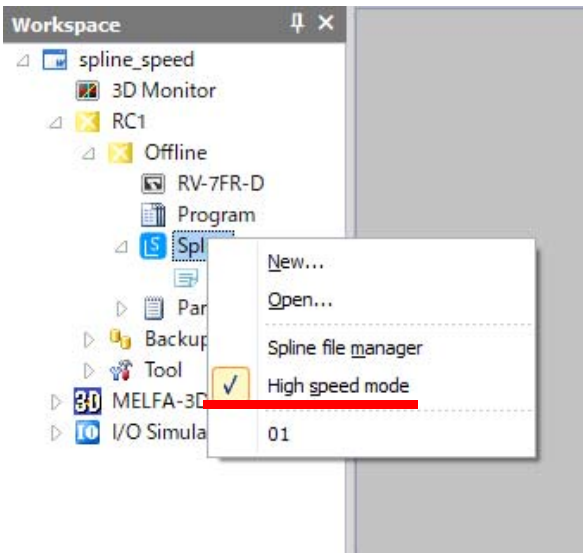


Fig.7-113:Command speed and required distance between path points

(4) RT ToolBox3 settings

High speed spline interpolation command can also be set by right clicking [Spline] in the project tree.
The high speed spline interpolation command can be enable/disabled by clicking the check mark next to "High speed mode" as shown below.



When the high speed spline interpolation command is enabled, high speed spline interpolation is applied to "path point check" and "calculate maximum speed" (shown in Table 7-11). The path point check window will be displayed as shown below.

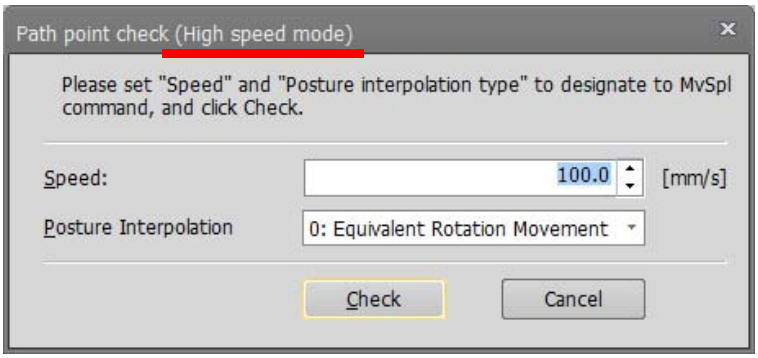


Fig.7-114:RT ToolBox3 path point check (High speed mode)

(5) Precautions and limitations

Take note of the following precautions and restrictions regarding the high speed spline interpolation command function.

1) Enabling the high speed mode makes it possible to specify a higher speed than previously possible. However, overspeed can occur depending on the robot's position or posture, and it may not be possible to move at the desired speed.
2) Furthermore, the position command may change slightly if high speed mode is enabled during spline interpolation. In such cases, disable the high speed mode.
3) The maximum speed calculated by the SplSpd function changes. If the SplSpd function is already being used, be aware of changes in the calculated value. Furthermore, the speed which is calculated by the maximum speed calculation function of RT ToolBox3 shown on Page 734, "(17) Checking edited data" also changes.
4) When pulse output is enabled, the specified signal is usually output at roughly 14 ms after the path points are passed. However, if the specified speed is high, the pulse width may become shorter than 14 ms (3.5 ms minimum).
5) If the specified speed is high and the operation mode is set to "variable linear speed", overspeed control will become less effective. If overspeed occurs, take steps such as reducing posture change, increasing the distance between the path points, and reducing the specified speed.
6) When checking the spline interpolation operation using a simulation in RT ToolBox3, ensure that the high speed mode is enabled/disabled for both the simulation and robot controller. Even if movements are possible in the simulator, errors may occur during operation if the high speed mode is not enable/disable for both the simulation and robot controller.

7.3 Ex-T control

Spline interpolation, one of the robot movement commands, is explained in this section.

7.3.1 Outline

(1) Features

The Ex-T control is the function to operate the robot using the origin of the externally fixed coordinates system as the robot control point. The examples of applications include the following.

• Polishing

The robot holds the target workpiece, and pushes it against the fixed grinder or abrasive belt for deburring or surface finishing.

• Coating

The robot holds the target workpiece, and applies solvent or adhesive supplied from the fixed dispenser to the workpiece.



Fig.7-115:Example of polishing

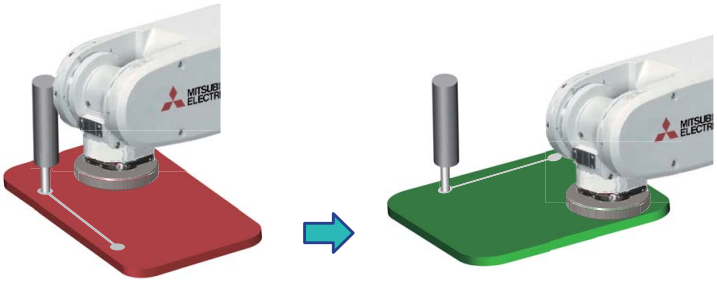


Fig.7-116:Example of coating

When the fixed tool (grinder, dispenser, etc.) is used and processing (deburring, polishing, sealing, etc.) of workpieces held and moved by the robot is performed as shown in the figure above, it is quite difficult to create a program for movement along the specified processing route while maintaining the relative position between the tool and the workpiece.

The Ex-T control is the function to facilitate operations in such applications and programming. The grinder or dispenser position can be registered in the robot, which enables linear/circular arc operations using the position as a reference.

(2) Specifications

<table>
<tr><th>Item</th><th>Specifications</th></tr>
<tr><td>Compatible robot</td><td>Vertical multiple-joint 6-axis robots, horizontal multiple-joint 4-axis robots
* The function is not available for the vertical multiple-joint 5-axis robots and the user mechanisms.</td></tr>
<tr><td>Compatible robot language</td><td>• Commands
<table>
<tr><th>Commands</th><th>Explanation</th><th>Page</th></tr>
<tr><td>EMvs (E Move S)</td><td>Ex-T control linear interpolation</td><td>236</td></tr>
<tr><td>EMvc (E Move C)</td><td>Ex-T control circular interpolation</td><td>228</td></tr>
<tr><td>EMvr (E Move R)</td><td>Ex-T control circular arc interpolation</td><td>230</td></tr>
<tr><td>EMvr2 (E Move R 2)</td><td>Ex-T control circular arc interpolation 2</td><td>232</td></tr>
<tr><td>EMvr3 (E Move R 3)</td><td>Ex-T control circular arc interpolation 3</td><td>234</td></tr>
</table>
• Status variables
<table>
<tr><th>Variable name</th><th>Array designation</th><th>Page</th></tr>
<tr><td>P_WkCord</td><td>Work coordinates number (1 to 8)</td><td>440</td></tr>
</table></td></tr>
<tr><td>Number of Ex-T coordinates that can be specified</td><td>Up to 8 (The coordinates system is specified with the parameters/status variables.)</td></tr>
</table>

(3) Equipment and software version requirements

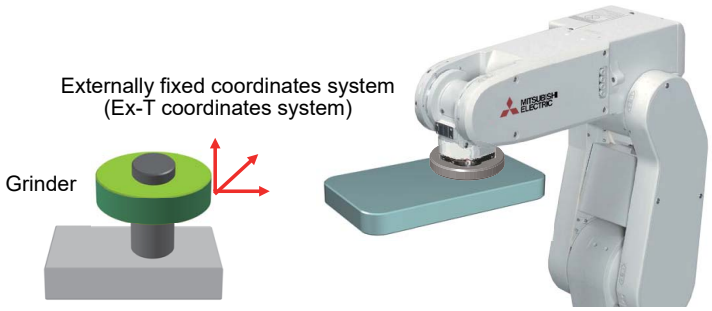
The Ex-T control can be used when the robot controller software version is the one shown in the table below or later.

Table 7-24:Software version compatible with Ex-T control

Device	Controller	T/B
Compatible software version	Ver. A1 or later	R32TB: Ver. 1.3 or later R56TB: Ver. 2.3 or later

7.3.2 Ex-T coordinates setting

When using the Ex-T control, setting is required for the externally fixed reference coordinates (Ex-T coordinates).



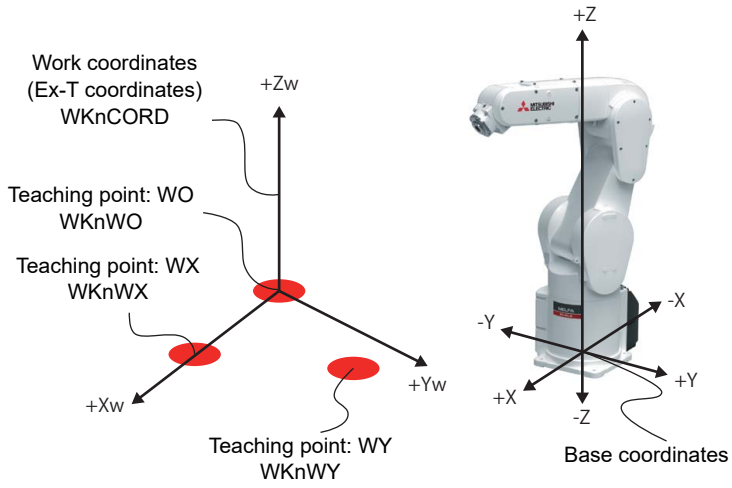
• For polishing using the fixed grinder, the position where polishing is performed is the origin of the Ex-T coordinates.
• For coating using the fixed dispenser, the nozzle tip position is the origin of the Ex-T coordinates.

(1) Setting

The Ex-T coordinates is specified using the same parameters/status variables as those of the work coordinates. As is the case with the work coordinates, perform the following:

• Parameter setting using TB or RT ToolBox3
• System status variable setting in the robot program (MELFA-BASIC VI)

The parameters related to the Ex-T coordinates (work coordinates) are listed in the table below.

Parameter name	Details explanation
WKnCORD "n" is 1 to 8	The work coordinates for work jog operation (X,Y,Z,A,B,C) Unit: mm or degree It is used as standard coordinates and work coordinate data in the work jog. It is also used as the control point in the Ex-T control (Ex-T coordinates). When using it as work coordinate data, the valid axial element differ depending on the robot type. Refer to Page 519, "5.7 About Standard Base Coordinates". The work coordinates defined by operation of T/B are set. However, inputting the coordinate value into this parameter can also define work coordinates. In this case, each coordinate value of the three teaching points for defining the work coordinates is cleared by 0. (Parameter: WKnWO, WKnWX, WKnWY ("n" is 1-8))  Note) To manage easily, you should teach work coordinates (Ex-T coordinates) in the condition that not convert the base coordinates. (Base coordinates and the world coordinate are in agreement.) Especially, it is necessary when defining two or more work coordinates.
WKnWO "n" is 1 to 8	Set the position of the work coordinates (Ex-T coordinates) origin as a teaching position of work coordinates (Ex-T coordinates). (Correspond to "WO" of the teaching operation by T/B. Refer to above figure) (X, Y, Z) Unit: mm Notes) The work coordinates (Ex-T coordinates) are not defined by merely inputting this coordinate value. Calculate the work coordinates (Ex-T coordinates) by performing [DEFINE] on the TB work coordinate setting screen or by performing [Write] on the RT ToolBox3 work coordinate parameter screen.
WKnWX "n" is 1 to 8	Set the position of "+X" axis of work coordinates (Ex-T coordinates) as a teaching position of work coordinates (Ex-T coordinates). (Correspond to "WX" of the teaching operation by T/B. Refer to above figure) (X, Y, Z) Unit: mm Notes) The work coordinates (Ex-T coordinates) are not defined by merely inputting this coordinate value. Calculate the work coordinates (Ex-T coordinates) by performing [DEFINE] on the TB work coordinate setting screen or by performing [Write] on the RT ToolBox3 work coordinate parameter screen.
WKnWY "n" is 1 to 8	Set the position at the side of "+Y" axis on the X-Y plane of work coordinates (Ex-T coordinates). (Correspond to "WY" of the teaching operation by T/B. Refer to above figure) (X, Y, Z) Unit: mm Notes) The work coordinates (Ex-T coordinates) are not defined by merely inputting this coordinate value. Calculate the work coordinates (Ex-T coordinates) by performing [DEFINE] on the TB work coordinate setting screen or by performing [Write] on the RT ToolBox3 work coordinate parameter screen.

When specifying the work coordinates (Ex-T coordinates) by three-point teaching using WKnWO, WKnWX, and WKnWY, it is useful to set the tool conversion data so that the reference position for teaching is used as control point.

For how to perform three-point teaching using R32TB, refer to "Work jog operation" in "Confirming the operation" of the separate manual "ROBOT ARM SETUP & MAINTENANCE".

7.3.3 Ex-T jog

The Ex-T jog is the function to perform jog operation along the work coordinates system (Ex-T coordinates system) by specifying the work coordinates (Ex-T coordinates) as the control point. The operation is similar to the conventional WORK jog operation because the operation is performed along the work coordinates system, but operations related to the posture elements are different between the Ex-T jog and the conventional WORK jog.
The Ex-T jog operation is performed by the WORK jog operation of TB. The conventional WORK jog operation and the Ex-T jog operation modes can be switched by setting the parameters WK1JOGMD to WK8JOGMD of each work coordinates system (Ex-T coordinates system).

Parameter name	Details explanation
WKnJOGMD "n" is 1 to 8	The operation mode in WORK jog operations is specified for each work coordinate. 0: WORK jog (The operation of A, B, and C elements is rotation around the axis parallel to the X, Y, and Z axes of the work coordinates. The control point position does not change.) 1: Ex-T jog (The operation of A, B, and C elements is rotation around the X, Y, and Z axes of the work coordinates with the control point moving.

Differences of operations according to the parameter WK1JOGMD to WK8JOGMD setting are shown below for each robot model.

WORK jog operation of the RV 6-axis type

WORK jog operation mode	WORK jog	Ex-T jog
Parameters WKnJOGMD (n is 1 to8) setting	0 (initial value)	1
XYZ key operation	Moves along each axis of the work coordinates system	Same as the conventional WORK jog
ABC key operation	With the control point position maintained, the direction changes along the work coordinates system.	While the control point position is changed, the direction changes on each axis of the work coordinates system.

RH 4-axis type and RH 4-axis hanging type

WORK jog operation mode	WORK jog	Ex-T jog
Parameters WKnJOGMD (n is 1 to8) setting	0 (initial value)	1
XYZ key operation	Moves along each axis of the work coordinates system	Same as the conventional WORK jog
C key operation	With the control point position maintained, the direction changes along the work coordinates system.	While the control point position is changed, the direction changes on the Z axis of the work coordinates system (Zw).
AB key operation	The robot does not move.	The robot does not move.

The movement of posture elements of the robot differs between the conventional WORK jog and the Ex-T jog operations. For example, the difference of the movement of C element is described below.

(1) Movement of the posture element in the WORK jog

The jog operation of the posture element in the WORK jog is rotation around the axes parallel to the X, Y, and Z axes of the work coordinates at the control point. The position remains fixed.

Fig. 7-117 shows the example of C element movement in the WORK jog.

W0-Wx-Wy shows the work coordinates system (when viewed from +Wz). ● represents the robot control point (TCP), and the square with rounded corners represents the workpiece held by the robot. The dotted line shows the workpiece position (posture) after the movement.

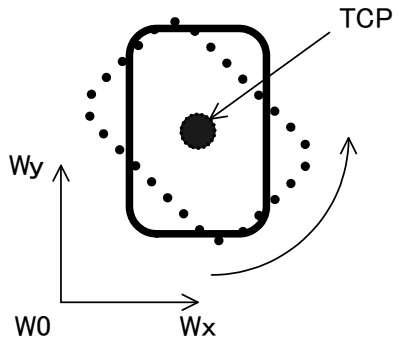


Fig.7-117:Example of C element movement in the WORK jog

(2) Movement of the posture element in the Ex-T jog

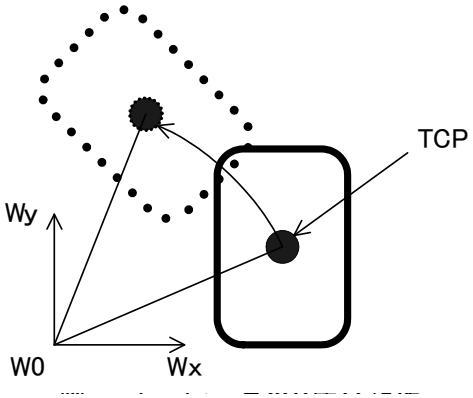
The jog operation of the posture element in the Ex-T jog is rotation around the X, Y, and Z axes of the Ex-T coordinates system (work coordinates system). The robot position also changes.

<Movement example 1> and <Movement example 2> in Fig. 7-118 show the example of C element movement in the Ex-T jog.

W0-Wx-Wy shows the Ex-T coordinates system (work coordinates system) (when viewed from +Wz). ● represents the robot control point (TCP), and the square with rounded corners represents the workpiece held by the robot. The dotted line shows the workpiece position (posture) after the movement.

In <Movement example 1>, the Ex-T coordinates system (work coordinates system) is located away from the workpiece. In <Movement example 2>, the origin of the Ex-T coordinates system (work coordinates system) is located on the workpiece. Rotating operation is performed around W0 in both cases.

<Movement example 1>



<Movement example 2>

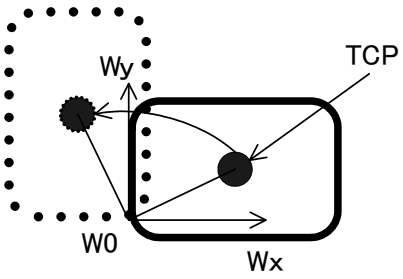
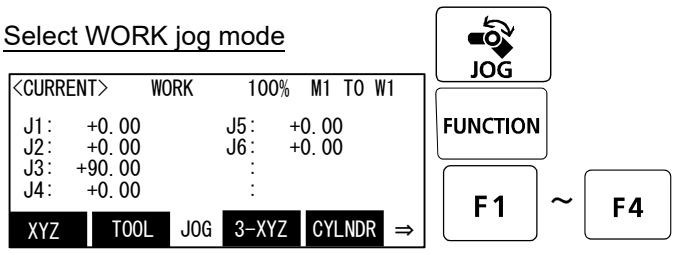


Fig.7-118:Example of C element movement in the Ex-T jog

(3) Ex-T jog operation

The Ex-T jog operation is the same as the WORK jog operation.
Advance setting of the Ex-T coordinates (work coordinates) and the operation mode (parameters WK1-JOGMD to WK8JOGMD) is required.

Select WORK jog mode

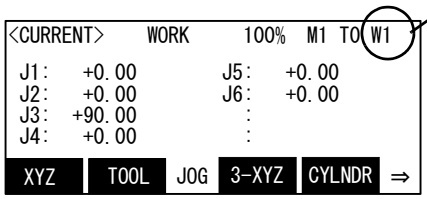


WORK jog mode

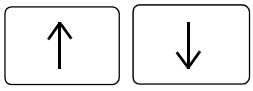
[JOG] Press the key and display the jog screen. ("JOG" is displayed on the screen bottom)
Check that the "WORK" in jog mode is displayed on the screen.
If other jog modes are displayed, please press the function key corresponding to the "WORK." (If the jog mode which he wishes under the screen is not displayed, it is displayed that the [FUNCTION] key is pressed)
If it finishes jog operation, press the [JOG] key again, or function key which correspond to "close."

* Displayed coordinate values are based on XYZ coordinates system.

Confirmation and selection of the work coordinates system (Ex-T coordinates system)



Target work coordinates system (Ex-T coordinates system)



Select the work coordinates system (Ex-T coordinates system)

Confirm the target work coordinates system (Ex-T coordinates system). The current target number is displayed on the screen upper right. (W1 - W8)
The number of work coordinates (Ex-T coordinates) can be changed by the arrow key [Upper arrow], [Lower arrow]
Push the key [Upper arrow], the number will increase. (W1, W2,..... W8) Conversely, push the key [Lower arrow], the number will decrease

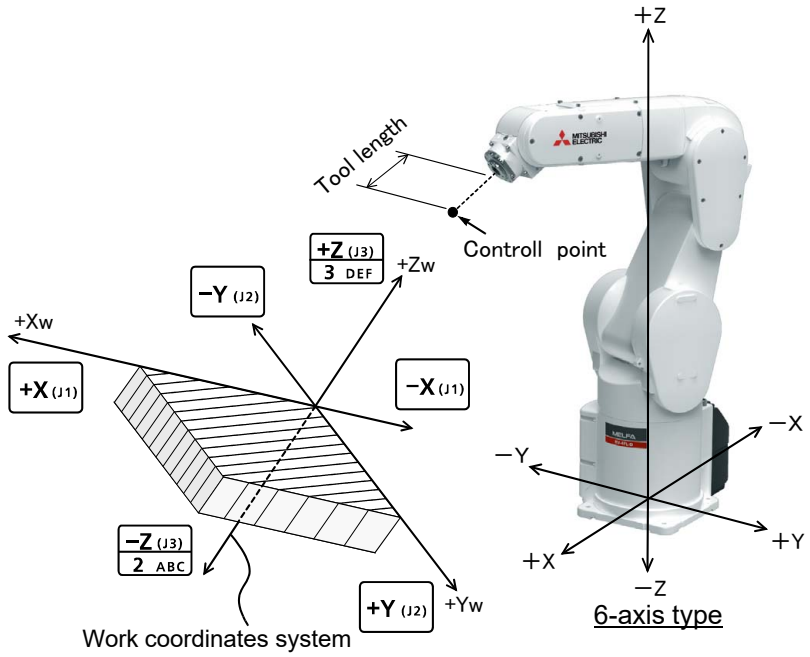
⚠ CAUTION

Always confirm that the number of the target work coordinates system (Ex-T coordinates system) is displayed correctly (Display of W1-W8 at the upper right of the screen)
If mistaken, the robot will move in the direction which is not meant and will cause the damage and the personal injuries.

(4) WORK jog operation of the RV 6-axis type

When the X, Y, or Z keys are used, the operation is the same in the WORK jog and the Ex-T jog modes.

The jog movement based on work coordinates system



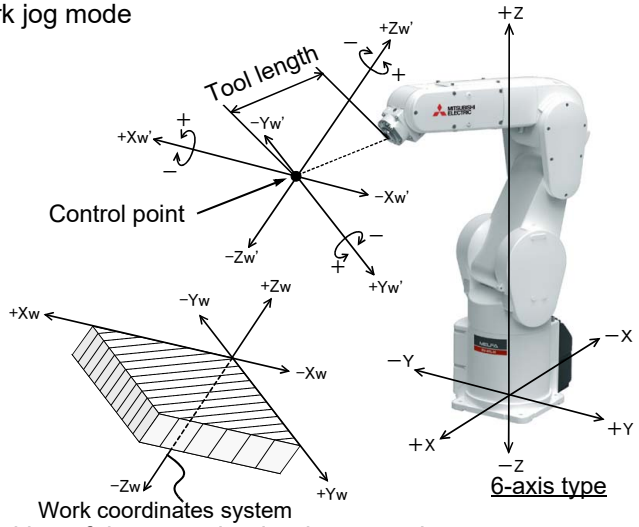
* The direction of the flange will not change.
Move the control point with a straight line in accordance with the work coordinates system.

- When the[+X (J1)] keys are pressed, the robot will move along the X axis (Xw) plus direction on the work coordinates system.
  When the[-X (J1)] keys are pressed, Move along the minus direction.
- When the[+Y (J2)] keys are pressed, the robot will move along the Y axis (Yw) plus direction on the work coordinates system.
  When the[-Y (J2)] keys are pressed, Move along the minus direction.
- When the[+Z (J3)] keys are pressed, the robot will move along the Z axis (Zw) plus direction on the work coordinates system.
  When the[-Z (J3)] keys are pressed, Move along the minus direction.

When the A, B, and C keys are used, the operation is different in the WORK jog and the Ex-T jog modes.

Changing the flange surface posture

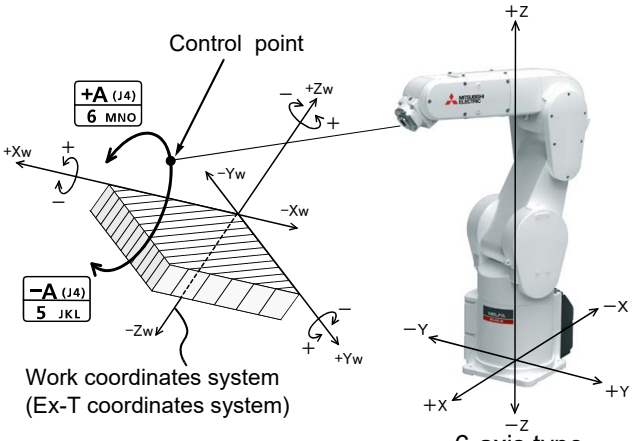
<1> Work jog mode



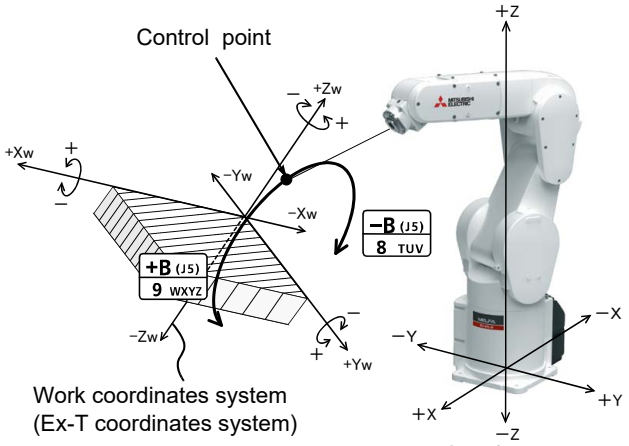
* The position of the control point does not change.
  Change the direction of the flange in accordance with the work coordinates system.

- When the[+A (J4)] keys are pressed, the X axis will rotate in the plus direction of the work coordinate system.
  When the[-A (J4)] keys are pressed, rotate in the minus direction.
- When the[+B (J5)] keys are pressed, the Y axis will rotate in the plus direction of the work coordinate system.
  When the[-B (J5)] keys are pressed, rotate in the minus direction.
- When the[+C (J6)] keys are pressed, the Z axis will rotate in the plus direction of the work coordinate system.
  When the[-C (J6)] keys are pressed, rotate in the minus direction.

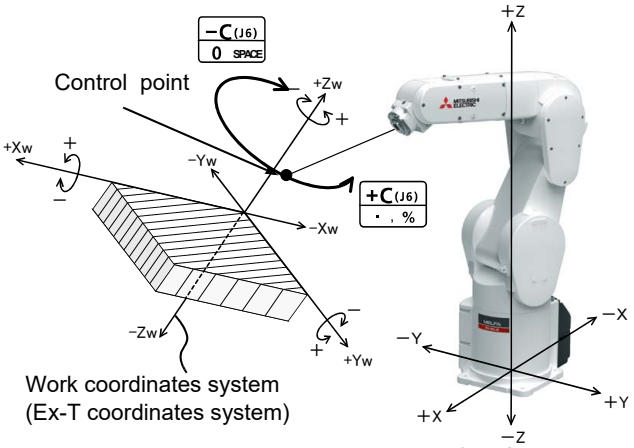
<2> Ex-T jog mode



6-axis type



6-axis type



6-axis type

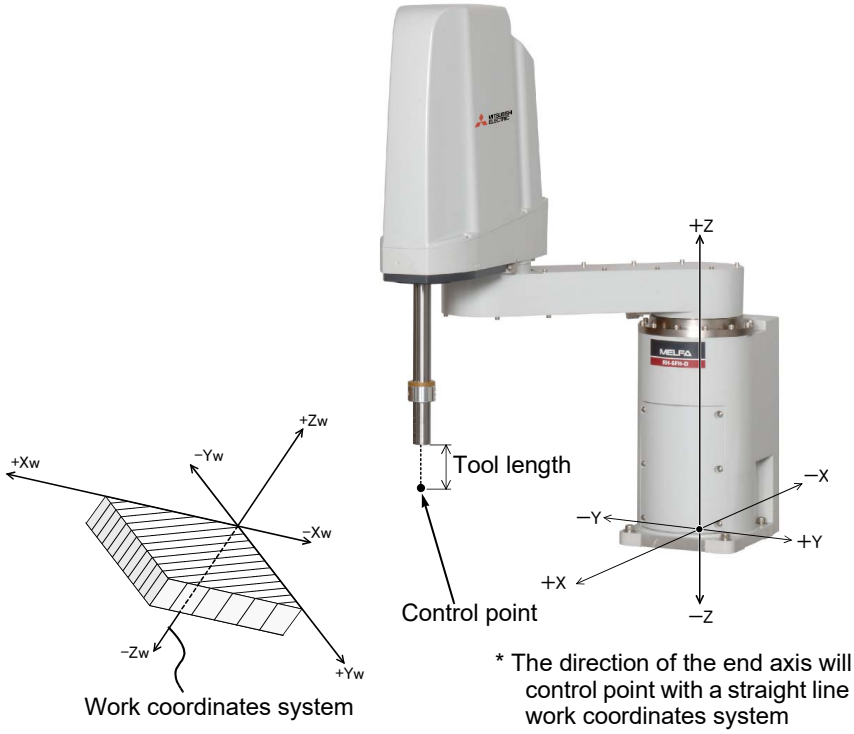
* The control point rotates around each axes of work coordinates system (Ex-T coordinates system). When the [+A (J4)] or the [-A (J4)] key is pressed, the control point rotates around the Xw axis. When the [+B (J5)] or the [-B (J5)] key is pressed, the control point rotates around the Yw axis. When the [+C (J6)] or the [-C (J6)] key is pressed, the control point rotates around the Zw axis.

- When the [+A (J4)] keys are pressed, the control point will rotate in the plus direction around the X axis (Xw) of work coordinates system (Ex-T coordinates system).
  When the[-A (J4)] keys are pressed, the control point will rotate in the minus direction.
- When the [+B (J5)] keys are pressed, the control point will rotate in the plus direction around the Y axis (Yw) of work coordinates system. (Ex-T coordinates system).
  When the[-B (J5)] keys are pressed, the control point will rotate in the minus direction.
- When the [+C (J6)] keys are pressed, the control point will rotate in the plus direction around the Z axis (Zw) of work coordinates system. (Ex-T coordinates system).
  When the[-C (J6)] keys are pressed, the control point will rotate in the minus direction.

(5) WORK jog operation of the RH 4-axis type

When the XYZ keys are used, the operation is the same in the WORK jog and the Ex-T jog modes.

The jog movement based on work coordinates system



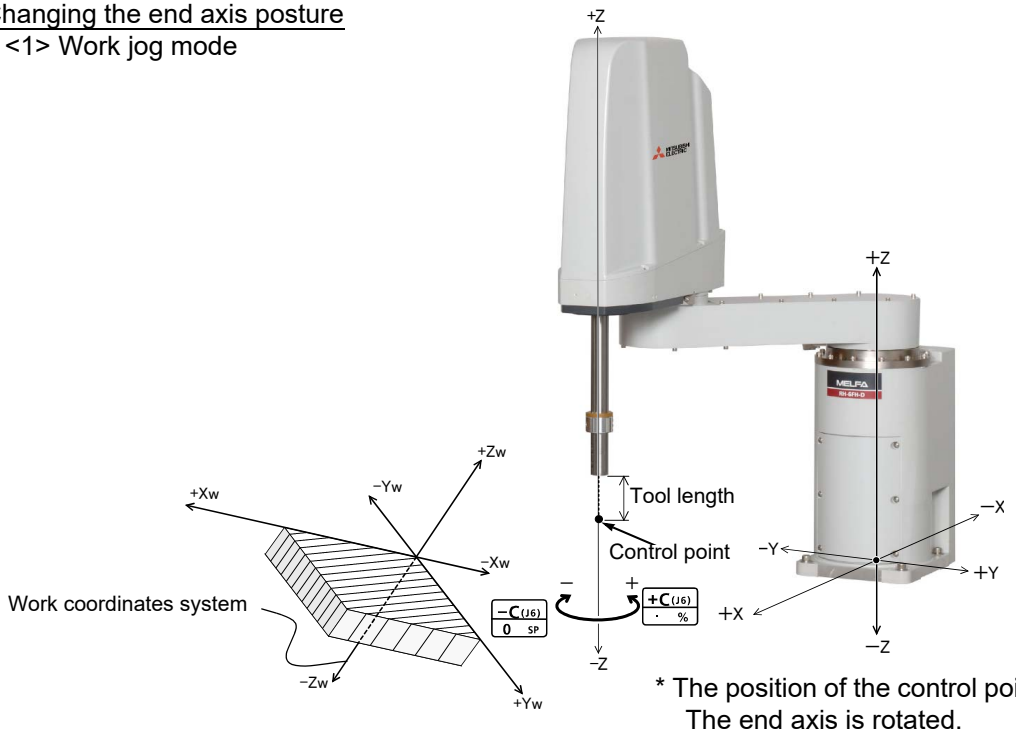
* The direction of the end axis will not change. Move the control point with a straight line in accordance with the work coordinates system

- When the[+X (J1)] keys are pressed, the robot will move along the X axis (Xw) plus direction on the work coordinates system.
  When the[-X (J1)] keys are pressed, Move along the minus direction.
- When the[+Y (J2)] keys are pressed, the robot will move along the Y axis (Yw) plus direction on the work coordinates system.
  When the[-Y (J2)] keys are pressed, Move along the minus direction.
- When the[+Z (J3)] keys are pressed, the robot will move along the Z axis (Zw) plus direction on the work coordinates system.
  When the[-Z (J3)] keys are pressed, Move along the minus direction.

When the C key is used, the operation is different in the WORK jog and the Ex-T jog modes.
The robot does not move when the A or B key is used.

Changing the end axis posture

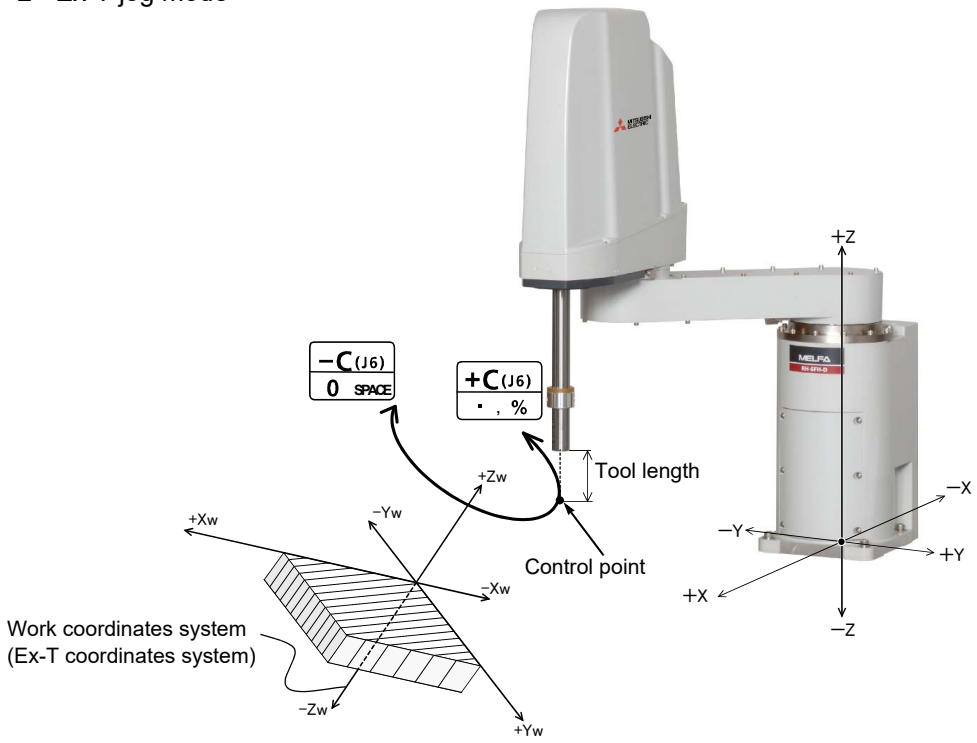
<1> Work jog mode



* The position of the control point does not change.
The end axis is rotated.

• When the[+C (J6)] keys are pressed, the Z axis will rotate in the plus direction of the XYZ coordinate system.
When the[-C (J6)] keys are pressed, rotate in the minus direction.

<2> Ex-T jog mode

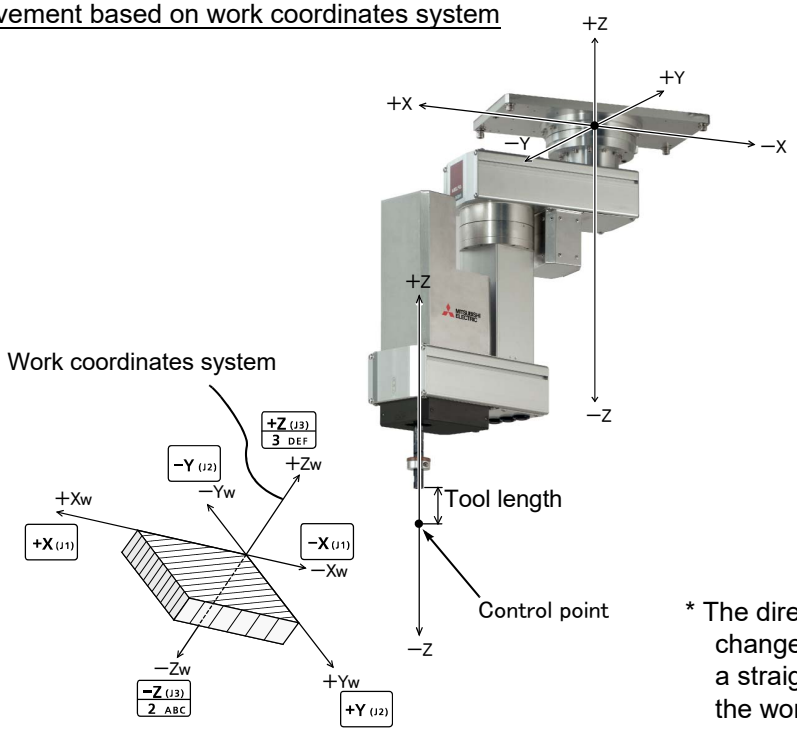


• When the [+C (J6)] keys are pressed, the control point will rotate in the plus direction around the Z axis (Zw) of work coordinates system (Ex-T coordinates system).
When the[-C (J6)] keys are pressed, the control point will rotate in the minus direction.

(6) WORK jog operation of the RH 4-axis hanging type

When the XYZ keys are used, the operation is the same in the WORK jog and the Ex-T jog modes.

The jog movement based on work coordinates system



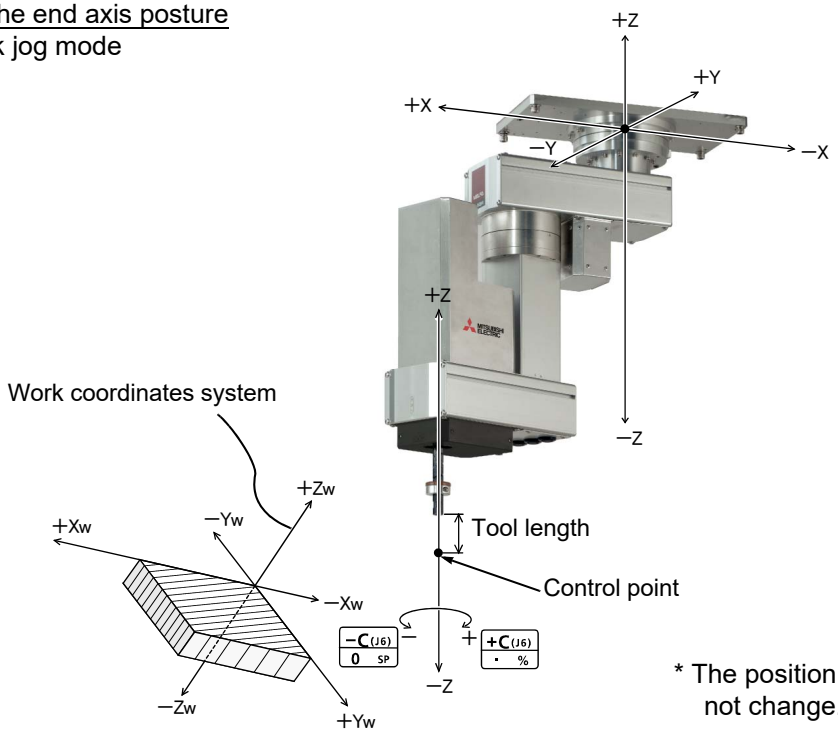
* The direction of the end axis will not change. Move the control point with a straight line in accordance with the work coordinates system

- When the[+X (J1)] keys are pressed, the robot will move along the X axis (Xw) plus direction on the work coordinates system.
  When the[-X (J1)] keys are pressed, Move along the minus direction.
- When the[+Y (J2)] keys are pressed, the robot will move along the Y axis (Yw) plus direction on the work coordinates system.
  When the[-Y (J2)] keys are pressed, Move along the minus direction.
- When the[+Z (J3)] keys are pressed, the robot will move along the Z axis (Zw) plus direction on the work coordinates system.
  When the[-Z (J3)] keys are pressed, Move along the minus direction.

When the C key is used, the operation is different in the WORK jog and the Ex-T jog modes.
The robot does not move when the A or B key is used.

Changing the end axis posture

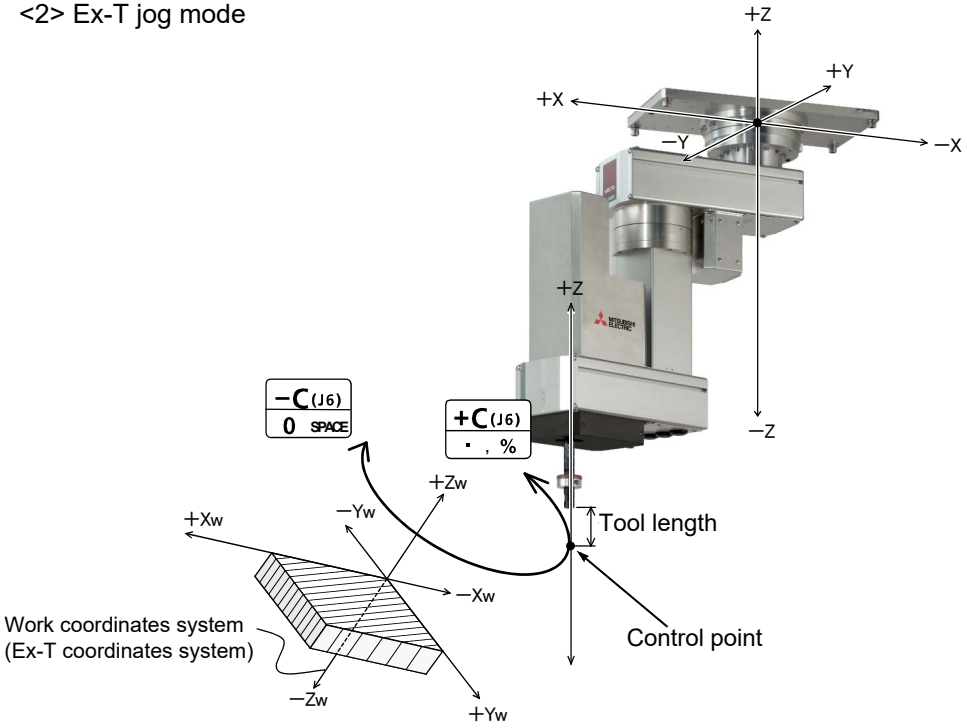
<1> Work jog mode



* The position of the control point does not change. The end axis is rotated.

• When the[+C (J6)] keys are pressed, the Z axis will rotate in the plus direction of the XYZ coordinate system.
When the[-C (J6)] keys are pressed, rotate in the minus direction.

<2> Ex-T jog mode



• When the[+C (J6)] keys are pressed, the Z axis will rotate in the plus direction of the XYZ coordinate system.
When the[-C (J6)] keys are pressed, rotate in the minus direction.

7.3.4 Creation of robot program

(1) List of commands/variables related to the Ex-T control

The commands and variables related to the Ex-T control are listed below.
For the command details, refer to the reference page shown in the table.

Table 7-25:List of commands related to the Ex-T control

No.	Type of commands	Added commands	Functions	Reference page
1	Linear interpolation	EMvs (E Move S)	Linear interpolation along the work coordinates (Ex-T coordinates)	236
2	Circular interpolation	EMvc (E Move C)	Circular interpolation along the work coordinates (Ex-T coordinates)	228
3	Circular arc interpolation	EMvr (E Move R)	Circular arc interpolation along the work coordinates (Ex-T coordinates)	230
4		EMvr2 (E Move R 2)		232
5		EMvr3 (E Move R 3)		234

Table 7-26:List of variables related to the Ex-T control

No.	Variable name	Outline of the function	Supplementary explanation	Reference page
1	P_WkCord	Reading and setting of the work coordinates	Also used for the Ex-T coordinates	440

(2) Programming example



The example shows a program to perform the operation as shown in the figure.
The robot holds the workpiece, and moves the workpiece along the fixed processing tool. (Figures <1> to <5>)

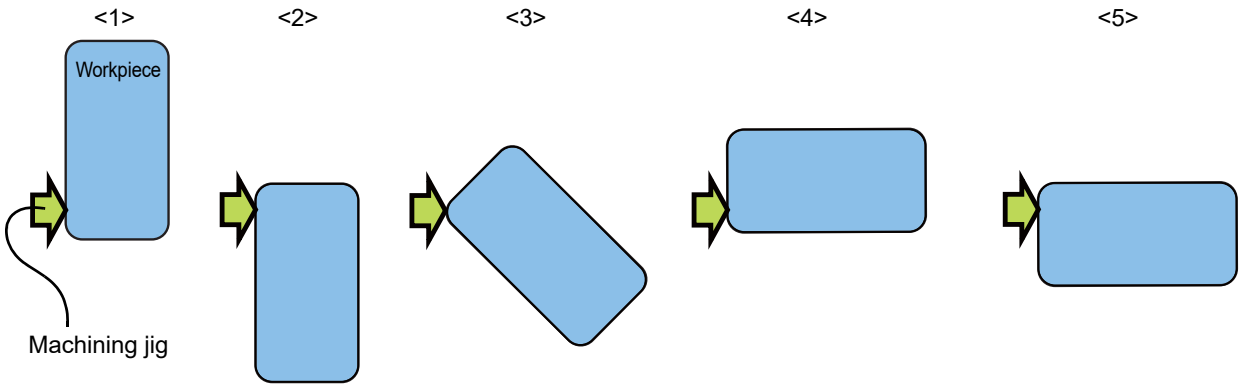


Fig.7-119:Operation example 1

By overlaying the figures, the actual operation is shown as follows.

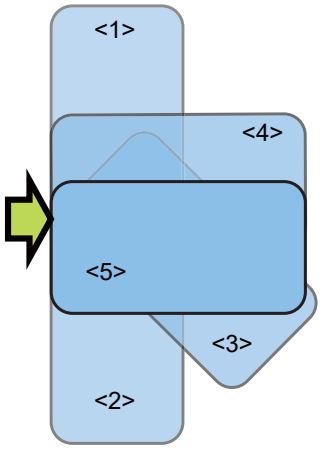
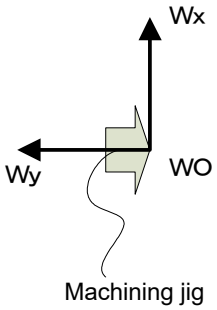


Fig.7-120:Operation example 2

■Step 1: Setting of the work coordinates (Ex-T coordinates)
Specify the work coordinates (Ex-T coordinates) so that the contact point between the processing tool and the workpiece shown in the figure is used as the origin of the work coordinates (Ex-T coordinates). (On the work coordinate 1 in this example)



In order to enable jog operation along this work coordinates system, set "1 (Ex-T jog mode)" in the parameter WK1JOGMD.

■Step 2: Teaching of the positions
Let the robot actually hold the workpiece for teaching of the positions.
The positions <1> to <5> are taught in this example.
For teaching of the positions, performing jog feed of the robot in WORK jog (Ex-T jog) enables jog feed along the processing tool.

■Step 3: Creation of the program
Create the program.
(Some of the actual necessary operations such as workpiece holding movement and input/output of signals are omitted in this example.)

[Example program]

```
Mov P001                    ' Moves to the position <1>.
Dly 0.5
Spd 50                      ' Sets the processing speed (workpiece moving speed) to 50 mm/sec.
EMvs 1, P002                ' Moves to the target position <2> along the work coordinate 1 by Ex-T
                              linear interpolation.
EMvr 1, P002,P003,P004      ' Moves to the target position <4> from position <2> via position <3>
                              along the work coordinate 1 by Ex-T circular arc interpolation.
EMvs 1, P005                ' Moves to the target position <5> along the work coordinate 1 by Ex-T
                              linear interpolation.
  :
```

In this example, the teaching positions are five. More teaching positions are required depending on the actual workpiece shape or processing operations. Change the program accordingly.

7.4 Get-position-quick function

7.4.1 Outline

The get-position-quick function (GPS function) allows monitoring and responding quickly to the input signal from an external sensor, which enables the accurate robot position data acquisition according to the signal input timing.
This function is used for the alignment correction (correcting a position deviation of a workpiece by the accurate robot position data acquisition according to the timing when an external sensor passes) and the mapping (the workpiece presence recognition).

The GPS function has the following two functions. For details on how to use each function, refer to Page 768, "(1) Position data acquisition at the sensor input timing" or Page 769, "(2) Workpiece presence recognition in a cassette".

Table 7-27:GPS function list

No.	Function	Usage
1	Position data acquisition at the sensor input timing	Alignment correction
2	Workpiece presence recognition in a cassette	Mapping

7.4.2 Specifications of the GPS Function

The specifications of the GPS function are as shown in the table below.

Table 7-28:Specifications of the GPS function

Item		Specifications
Position data acquisition at the sensor input timing	Condition for the position data acquisition	Define a condition for the position data acquisition in the Def Gps command.
	Controlling to start/stop monitoring	Control to start/stop monitoring the input signal with the GpsChk command.
	Position data latch	Store the position data in the P_GpsX() at the point in time when a condition for the position data acquisition is met ("X" indicates the same number as the target monitoring number from 1 to 8). (Up to 400 data)
Workpiece presence recognition in a cassette	Condition for the workpiece presence recognition	Define a condition for the workpiece presence recognition in the Def Map command.
	Controlling to start/stop monitoring	Control to start/stop monitoring the input signal with the GpsChk command.
	Position data latch	Store the position data in the P_GpsX() at the point in time when a condition for the position data acquisition is met ("X" indicates the same number as the target monitoring number from 1 to 8). (Up to 400 data)

7.4.3 Specifications of Digital Input Signal

The specifications of the digital input signal are as shown in the table below.

Table 7-29:Specifications of digital input signal

Item		Specifications	Remarks
Digital input	Number of channels	SKIP input: 3CH	
	Connector Note1)	CNUSR12 connector	Pin name (Pin number): Input signal number SKIP21(3), SKIP22(12): input signal 801 SKIP31(2), SKIP32(11): input signal 802 SKIP41(1), SKIP42(10): input signal 803
	Monitoring period	SKIP input: 444 μs	
Signal input condition	ON	DC 18 V to DC 25.2 V, 9 mA or more	444 μs-wide signals are required.
	OFF	DC 4 V or less, 2 mA or less	

Note1) Location of connector is shown in Fig. 7-121. And refer to the separate manual: "Controller setup, basic operation, and maintenance." for details on the pin assignment.

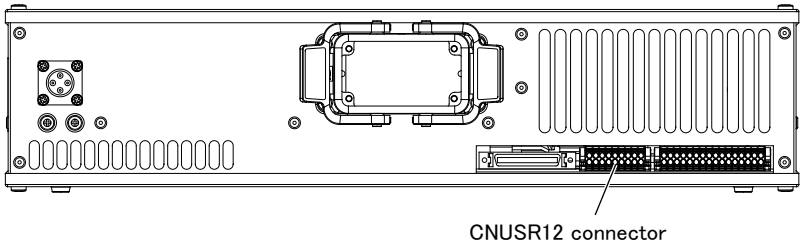
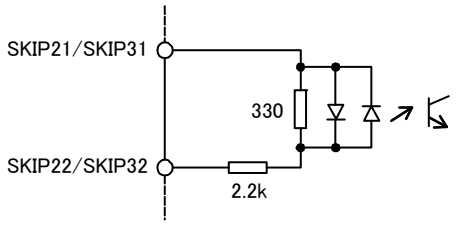


Fig.7-121: Location of connector

7.4.4 Electric Specification of Digital Input

The electric specification of SKIP input is shown in Table 7-30.

Table 7-30:Electric specification of the SKIP input

Item	Specifications	Internal circuit
Type	DC input	
No. of input point	2	
Insulation method	Photo-coupler insulation	
Rated input coltage	DV 24 V ±10%	
Rated input current	Approx. 9 mA	
Input resistance	Approx. 2.5kΩ	
Common method	1 point per common	

7.4.5 How to use the GPS function

(1) Position data acquisition at the sensor input timing

This section shows the basic method to create programs using the MELFA BASIC VI programming language for position data acquisition according to the signal input timing.

Table 7-31: Command list of MELFA BASIC VI

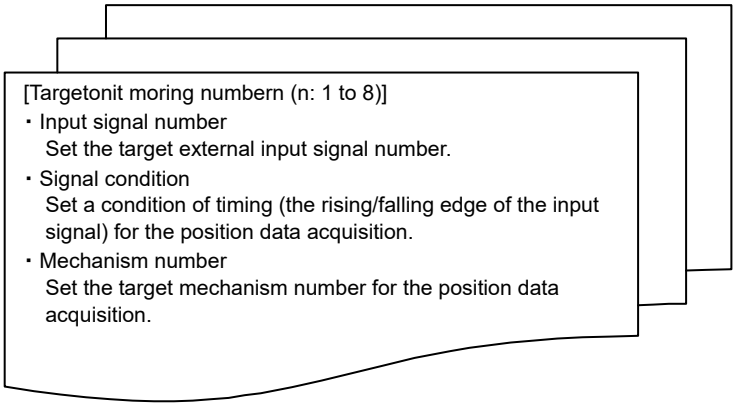
Command	Explanation
Def Gps	This command defines a condition for the position data acquisition using the GPS function.
GpsChk	This command starts/stops monitoring the set condition.

Table 7-32: Status variables list

Variable name	No. of array elements	Details
M_Gps(n) (n:target monitoring number from 1 to 8)	1 to 8 (target monitoring number)	The number of the position data recorded in the status variable P_GpsX() is stored. ("X" indicates the same number as the target monitoring number from 1 to 8.)
P_Gps1(x) to P_Gps8(x)	400 (data)	Up to 400 of the robot position data at the point in time when the condition defined in the Def Gps command holds is stored.

1) Definition of the monitoring condition

First, define a condition (an input signal number, a signal condition, and a mechanism number) for the position data acquisition according to an external signal input timing in the Def Gps command. Up to eight different conditions can be set. Each setting of condition is controlled by the monitoring number.



2) Starting the monitoring

When the GpsChk On command ("GpsChk On,n" statement) is executed, start monitoring for the target monitoring number which is defined in the Def Gps command, and store the robot position data at the point of time when the condition holds. Up to 400 position data can be obtained for one monitoring number.

("n" in "GpsChk On,n" statement indicates the target monitoring number from 1 to 8.)

3) Stopping the monitoring

When the GpsChk Off command ("GpsChk Off,n" statement) is executed, stop monitoring for the target monitoring number, and store the robot position data in the status variable P_GpsX(). ("X" indicates the same number as the target monitoring number from 1 to 8.) Up to 400 position data can be stored for one monitoring number. (A numerical value in the array elements is numbered in turn every time the position data is obtained.)

The number of position data stored in the P_GpsX() is retained in the status variable M_Gps(n). The position data stored in the M_Gps(n) and P_GpsX() is cleared to zero when the GpsChk On command ("GpsChk On,n" statement) is executed next time.

("n" in "GpsChk Off,n" statement and the status variable M_Gps(n) indicates the target monitoring number from 1 to 8.)

<Sample program>

```
1 Def Gps 1,801,On,1      ' The position data of the mechanism No. 1 is recorded for the monitoring No. 1 when the signal
                            No. 801 is turned on.
2 GpsChk On,1             ' Monitoring a condition for the monitoring No. 1 is started.
3 Mvs P1                  ' Moves to P1
4 GpsChk Off,1            ' Monitoring a condition for the monitoring No. 1 is stopped (the position data obtained is stored).
5 M1=M_Gps(1)             ' The number of the position data recorded in the P_Gps1 is stored.
6 If M1=0 Then Error 9000 ' The error 9000 is generated if no position data is recorded.
7 P1=P_Gps1(1)            ' The position data at the first time when the signal No. 801 is turned on is substituted for "P1".
8 Hlt                     ' Halt program
```

(2) Workpiece presence recognition in a cassette

This section shows the basic method to create programs using the MELFA BASIC VI programming language for the mapping (detecting the presence of a workpiece.) in the cassette.

Table 7-33: Command list of MELFA BASIC VI

Command	Explanation
Def Map	This command defines a condition for the workpiece presence recognition in the cassette, using the GPS function.
GpsChk	This command starts/stops monitoring the set condition.

Table 7-34: Status variables list

Variable name	No. of array elements	Details
M_Gps(n) (n:target monitoring number from 1 to 8)	1 to 8 (target monitoring number)	The number of the position data recorded in the status variable P_GpsX() is stored. ("X" indicates the same number as the target monitoring number from 1 to 8.)
P_Gps1(x) to P_Gps8(x)	400 (data)	Up to 400 robot position data at the time when the condition for the target input number defined in the Def Map command is met is stored.
M_Map1(x) to M_Map8(x)	130 (segments)	"1" is stored for the segment number in which the workpiece is present, calculated from the condition defined in the Def Map command.

1) Definition of the monitoring condition

First, define a condition (an input signal number, a signal condition, and a mechanism number) for the position data acquisition according to an external signal input timing and a cassette condition (the coordinates data and the number of segments for a cassette) in the Def Map command. Up to eight different conditions can be set. Each setting of condition is controlled by the monitoring number.

2) Starting the monitoring

When the GpsChk On command ("GpsChk On,n" statement) is executed, start monitoring for the target monitoring number which is defined in the Def map command, and store the robot position data at the point of time when the condition holds. Up to 400 position data can be obtained for one monitoring number.
("n" in "GpsChk On,n" statement indicates the target monitoring number from 1 to 8.)

3) Stopping the monitoring

The segment number in which the workpiece is present is calculated from the position data obtained at the time when the GpsChk Off command ("GpsChk Off,n" statement) is executed and from the data of cassette shape and robot position. A result of the calculation is stored in the status variable M_MapX(). ("X" indicates the same number as the target monitoring number from 1 to 8.)
The number of stored position data is retained in the status variable M_Gps(n).
The data stored in the M_Gps(n) and M_MapX() is cleared to zero when the GpsChk On command ("GpsChk On,n" statement) is executed next time.
("n" in "GpsChk Off,n" statement and the status variable M_Gps(n) indicates the target monitoring number from 1 to 8.)

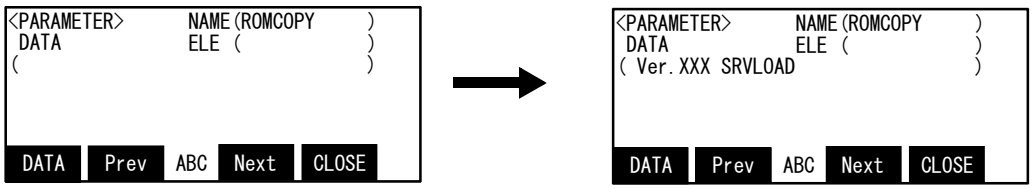
<Sample program>

```
1  Def Map 3,851,On,1,PC1,PC2,20,10
                                    ' The position data of the mechanism No. 1 is recorded for the monitoring No. 3 when the signal
                                      No. 851 is turned on, and the mapping is executed according to the defined condition.
                                      PC1: Lowest point (first segment) in a cassette, PC2: Highest point (last segment) in a cassette,
                                      20: the number of segments in a cassette (20 segments), 10: a sensitive area of a sensor (10
                                      mm)
2  Mov PM1                          ' The robot moves its arm to the mapping start position.
3  GpsChk On,3                      ' Monitoring a condition for the monitoring No. 3 is started.
                                      The position data of the mechanism No. 1 is recorded when the signal No. 851 is turned on.
4  Mvs PM2                          ' The robot moves its arm to the mapping stop position.
5  GpsChk Off,3                     ' Monitoring a condition for the monitoring No. 3 is stopped.
                                    ' Using the recorded position data, the segment number in which the workpiece is present is
                                      stored in the M_Map3().
6  M1=M_Gps(3)                      ' The number of the position data recorded in the P_Gps3 is stored.
7  If M1=0 Then Error 9000          ' The error 9000 is generated if no position data is recorded.
8  For M2=1 To 20                   '
9  M_Out(6100+M2)=M_Map3(M2)        ' The results of the mapping are output with a signal (the signal number "6101" is assigned for the
                                      first result).
10 Next M2                          ' The process performed for the first segment is repeated for the remaining segments.
```

7.5 Upgrade of the servo software

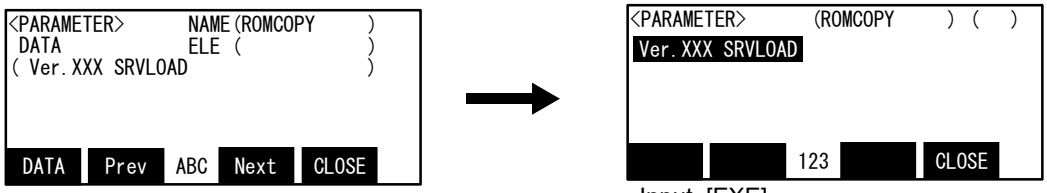
When error H0099 occurs, the upgrade of the servo software is required.
You can upgrade the servo software using the teaching pendant or RT ToolBox3.
The procedure for upgrading the servo software using the teaching pendant is shown below.

1) Display the parameter screen to input "ROMCOPY" into the name field and press the [EXE] key.



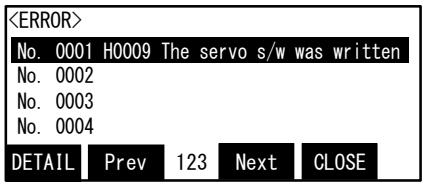
Enter the parameter name [R], [O], [M], [C], [O], [P], [Y]
Input [EXE]

2) Press the function key ([F1]) corresponding to "DATA" to display the parameter input screen. Press the [EXE] key on the parameter input screen.



Setting the data [F1]
Input [EXE]

3) When you check that error H0009 is displayed, reset the power supply of the controller.



Check the display

4) When a buzzer goes off from the controller, turn off the controller's power supply.

The upgrade of the servo software is completed.

For the use of RT ToolBox3, you can upgrade the servo software on the parameter list screen.

7.6 Log function

This section explains the log function which saves robot information at error occurrence.

7.6.1 Outline

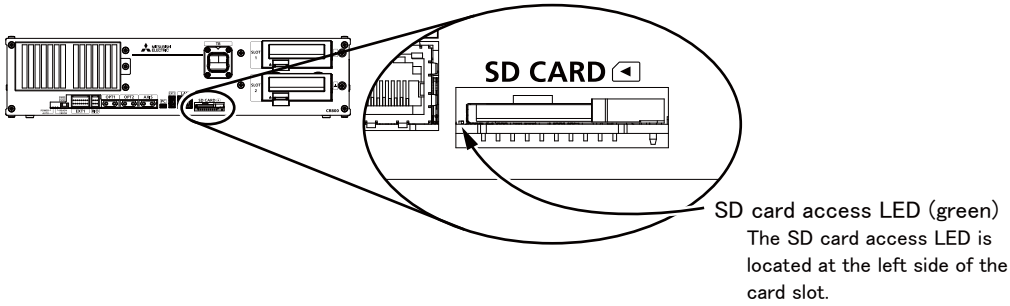


Robot information at error occurrence can be saved in the FTP server or SD memory card (CR800-D only).
The following four types of robot information can be saved.

- Error history
- Event history
- Program execution history
- Internal information of the robot operation

To use the log function, parameters must be set beforehand.

⚠CAUTION Do not remove the SD memory card or turn off the robot controller while the SD card access LED is on. Doing so may damage the SD memory card or files.
When removing the SD memory card, support it with your hands since it may pop out from the card slot.



⚠CAUTION The lock state set by the LOCK switch of the SD memory card is ignored in the CR800 controller. Even if the lock state is set, information is written to the SD memory card.

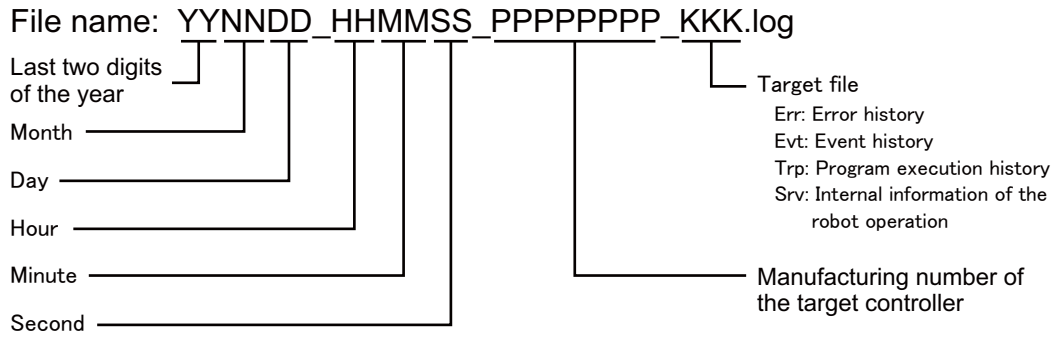
7.6.2 Specifications

(1) Basic specifications

Table 7-35:Basic specifications

Item	Specifications
Logs to be saved	The following four types of logs are saved. • Error history • Event history • Program execution history • Internal information of the robot operation
Save destination	FTP server or SD memory card (CR800-D only)
Target SD memory card	2F-2GBSD (capacity: 2GB)

(2) Log file specifications



Example)

170701_113502_B12345678M_Err.log
170701_113502_B12345678M_Evt.log
170701_113502_B12345678M_Trp.log
170701_113502_B12345678M_Srv.log

For SD memory cards, the log files are saved in the folder with the date.

Table 7-36:Log details

File name	Details	Check method
xxxx_Err.log	Error history	The error history can be checked in the simulation of RT ToolBox3 Ver. 1.10L or later.
xxxx_Evt.log	Event history	The error history can be checked in the simulation of RT ToolBox3 Ver. 1.00A or later.
xxxx_Trp.log	Program execution history	The program execution history can be checked with a text editor such as Notepad.
xxxx_Srv.log	Internal information of the robot operation	-

7.6.3 Error numbers to be saved

The errors shown in Table 7-37 have been registered in advance. To add errors, set the parameter LOGTRGE.

Table 7-37:Target error number

Error number	Error details
H0740	Power supply main circuit error
H0810	Power supply undervoltage
H091n	Servo amplifier overspeed
H092n	Power module overcurrent
H093n	Motor overcurrent
H096n	Excessive error 1
H101n	Collision detection
H1680	Cannot servo ON (timeout)

7.6.4 Parameter setting

Use T/B or RT ToolBox3 to set parameters. Refer to the instruction manuals enclosed with each product for the details of the operation method.

- R32TB/R33TB: This instruction manual “Page 84, "3.15 Operation of parameter screen"”
- R56TB/R57TB: R56TB/R57TB Instruction Manual (BFP-A8684)
- RT ToolBox3: RT ToolBox3/RT ToolBox3 mini User's Manual (BFP-A3495)

Table 7-38:Log function parameter

Parameter	Parameter name	No. of arrays No. of characters	Details explanation	Factory setting
Error log automatic transfer	LOGMODE 1	Integer 1	Set the log function at error occurrence. 1: Valid (FTP) 2: Valid (SD memory card) *CR800-D only 0: Invalid	0
Target error number	LOGTRGE	Integer 16	Set an error number to be saved. Eight types of error numbers can be registered in the order of error number and type. Type 0: The error number is used as it is. 1: The axis number is entered in the last digit of the error number. Setting example) 2000, 0, 1010, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, “2000, 0,” indicates error 2000 only. “1010, 1,” indicates error 1011 to 1018.	0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
FTP transfer Note1)	FTPID	Character string 1	Set a user ID to be used in the FTP communication. Setting range: Alphanumeric characters (uppercase/lowercase) within 8 characters	ftpuser
	FTPPASS	Character string 1	Set a password to be used in the FTP communication. Setting range: Alphanumeric characters (uppercase/lowercase), Symbol (! # $ % & = - @ . ? _) within 16 characters	ftppassword
	FTPSVRIP	Character string 1	Set the IP address of an FTP server to be used in the FTP communication.	192.168.0.99
	FTPPATH	Character string 1	Set the path of an FTP transfer destination. Setting range: Alphanumeric characters (uppercase/lowercase), Symbol (:/)	Blank

Note1) To use the FTP, the following parameters must be set beforehand.

7.6.5 Checking the log file details

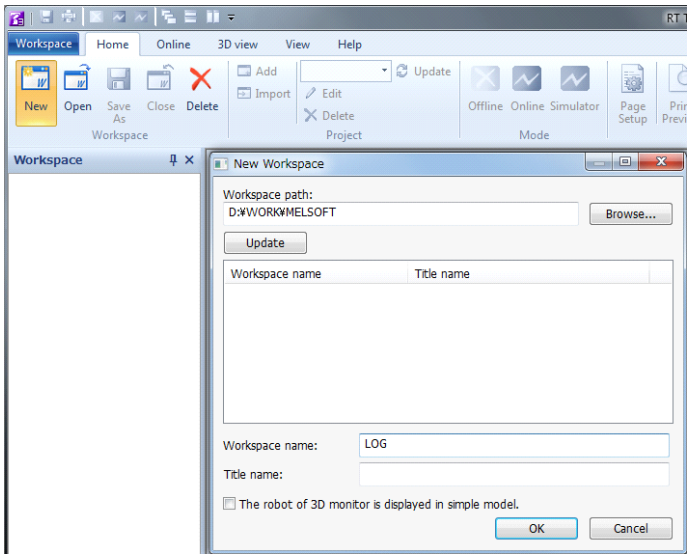
This section explains the check method of each log file.

(1) Error history

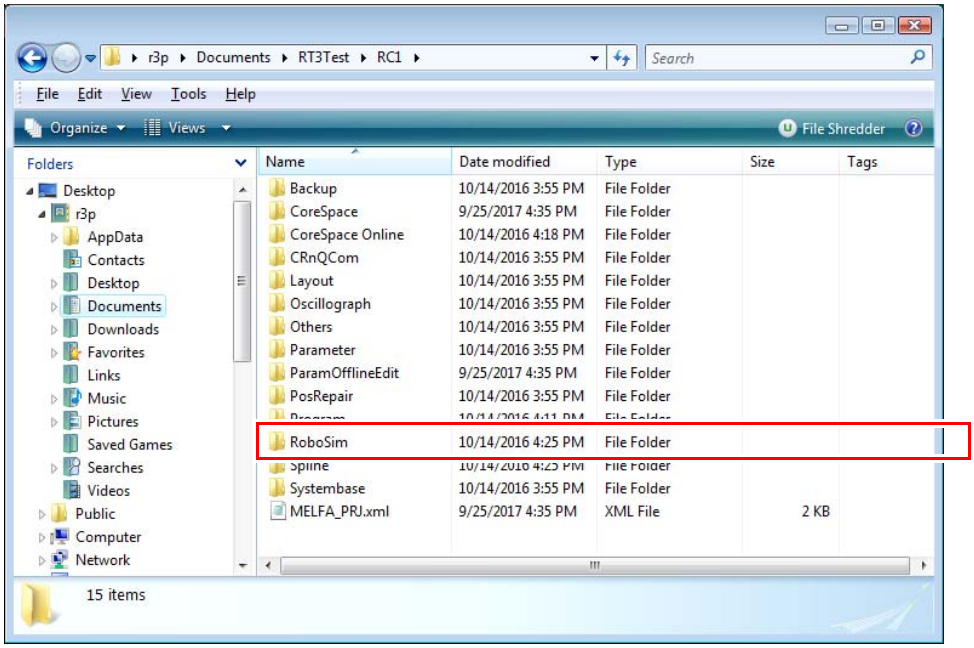
This can be checked with RT ToolBox3 Ver. 1.10L or later.

* Operation procedure

1) Rename xxxx_Err.log to AError.log.
2) Create a workspace with RT ToolBox3.

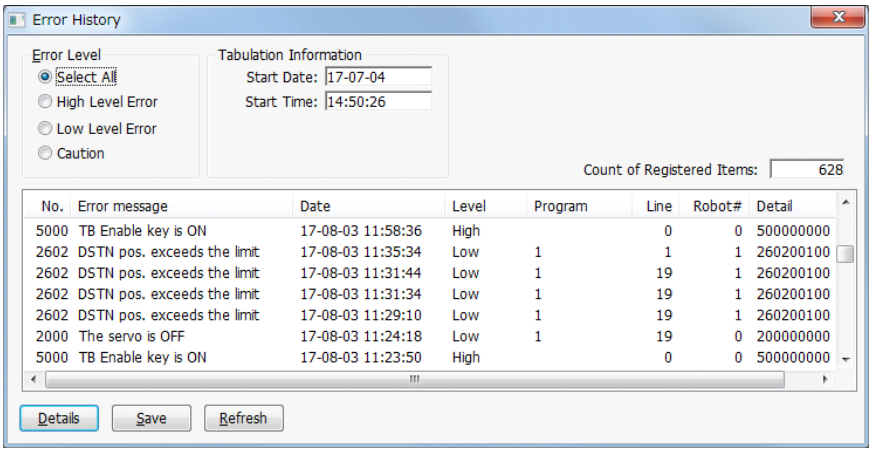


3) Copy AError.log to the simulation folder “RoboSim”.



4) Start the simulation.

5) Check the details in the Error History window.

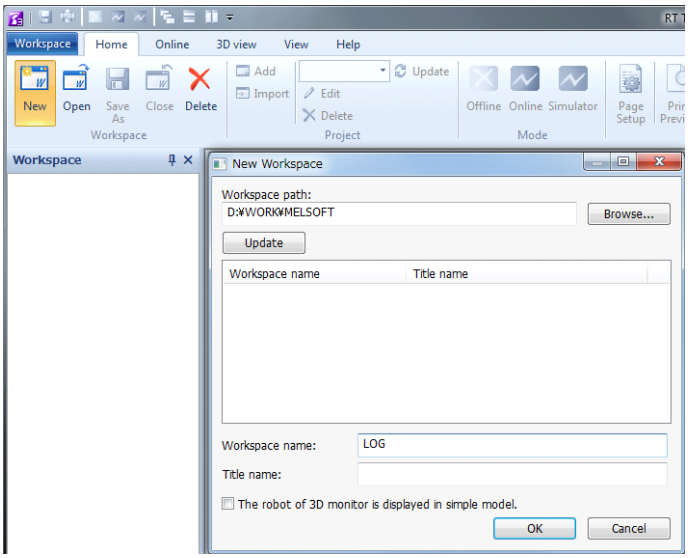


(2) Event history

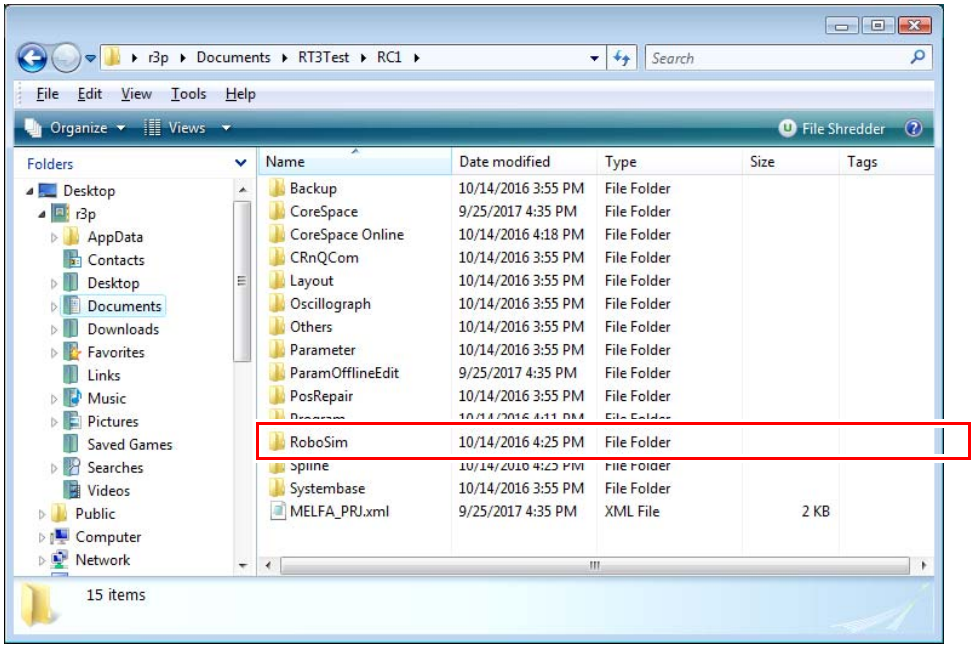
This can be checked with RT ToolBox3 Ver. 1.00A or later.

* Operation procedure

1) Rename xxxx_Evt.log to Event.log.
2) Create a workspace with RT ToolBox3.

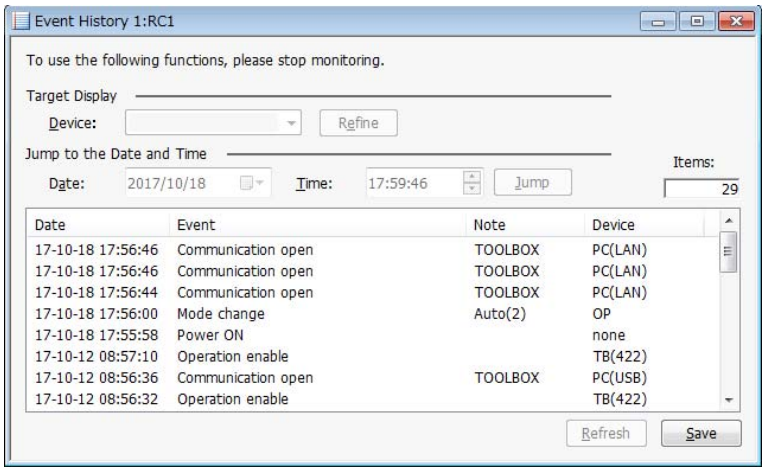


3) Copy Event.log to the simulation folder “RoboSim”.



4) Start the simulation.

5) Check the details in the Event History window.



(3) Program execution history

This can be checked with a text editor such as Notepad.

* Operation procedure

1) Open xxxx_Trp.log with a text editor such as Notepad.
The program execution order can be checked as shown below.

```
NARC Trap system PROG log V1.0
17-10-18 11:34:42;sec/it=0.000889 ←Date and time when data collection is started
17-10-18 11:37:18;0; ← Date and time when the data in the last row is recorded
0;659;PROG;5  ← <Spare>;<Serial number>;<Program name>;<Step number>
0;660;PROG;2
0;661;PROG;3
0;662;PROG;4
0;663;PROG;5
0;664;PROG;2
????
```

7.7 Special devices

The compatible special devices are as follows.

7.7.1 CR800-R series

(1) Special relay

The compatible special relays are as follows.

Table 7-39:R16RTCPU robot CPU-compatible special relays

No.	Name	Remarks
SM0	Latest self-diagnostics error (Including annunciator ON)	
SM1	Latest self-diagnostics error (Not including annunciator ON)	
SM50	Clear error	
SM51	Low battery latch	
SM52	Low battery	
SM203	STOP contact	
SM204	PAUSE contact	

(2) Special register

The compatible special registers are as follows.

Table 7-40:R16RTCPU robot CPU-compatible special registers

No.	Name	Remarks
SD0	Latest self-diagnostics error code	The 0x1800, 0x3000, and 0x3001 codes are stored when robot-specific errors occur.
SD1	Time of occurrence of latest self-diagnostics error code	
SD2	↑	
SD3	↑	
SD4	↑	
SD5	↑	
SD6	↑	
SD7	↑	

No.	Name	Remarks
SD10	Self-diagnostics error code 1	The 0x1800, 0x3000, and 0x3001 codes are stored when robot-specific errors occur.
SD11	Self-diagnostics error code 2	
SD12	Self-diagnostics error code 3	
SD13	Self-diagnostics error code 4	
SD14	Self-diagnostics error code 5	
SD15	Self-diagnostics error code 6	
SD16	Self-diagnostics error code 7	
SD17	Self-diagnostics error code 8	
SD18	Self-diagnostics error code 9	
SD19	Self-diagnostics error code 10	
SD20	Self-diagnostics error code 11	
SD21	Self-diagnostics error code 12	
SD22	Self-diagnostics error code 13	
SD23	Self-diagnostics error code 14	
SD24	Self-diagnostics error code 15	
SD25	Self-diagnostics error code 16	
SD201	LED status	
SD203	CPU operating status	
SD228	Multi-CPU system information	
SD229	↑	
SD230	↑	
SD231	↑	
SD232	↑	
SD233	↑	
SD260	Number of assigned bit devices	
SD261	↑	
SD262	↑	
SD263	↑	
SD264	↑	
SD265	↑	
SD280	Number of assigned word devices	
SD281	↑	

7.7.2 CR800-D series

(1) Special relay

The compatible special relays are as follows.

Table 7-41:CR800-D-compatible special relays

No.	Name	Remarks
SM0	Latest self-diagnostics error (Including annunciator ON)	
SM1	Latest self-diagnostics error (Not including annunciator ON)	
SM51	Low battery latch	
SM52	Low battery	

No.	Name	Remarks
SM203	STOP contact	
SM204	PAUSE contact	

(2) Special register

The compatible special registers are as follows.

Table 7-42:CR800-D-compatible special registers

No.	Name	Remarks
SD0	Self-diagnostics error code	The 0x1800, 0x3000, and 0x3001 codes are stored when robot-specific errors occur.
SD1	Time of occurrence of latest self-diagnostics error code	
SD2	↑	
SD3	↑	
SD4	↑	
SD5	↑	
SD6	↑	
SD7	↑	
SD10	Self-diagnostics error code 1	The 0x1800, 0x3000, and 0x3001 codes are stored when robot-specific errors occur.
SD11	Self-diagnostics error code 2	
SD12	Self-diagnostics error code 3	
SD13	Self-diagnostics error code 4	
SD14	Self-diagnostics error code 5	
SD15	Self-diagnostics error code 6	
SD16	Self-diagnostics error code 7	
SD17	Self-diagnostics error code 8	
SD18	Self-diagnostics error code 9	
SD19	Self-diagnostics error code 10	
SD20	Self-diagnostics error code 11	
SD21	Self-diagnostics error code 12	
SD22	Self-diagnostics error code 13	
SD23	Self-diagnostics error code 14	
SD24	Self-diagnostics error code 15	
SD25	Self-diagnostics error code 16	
SD201	LED status	
SD203	CPU operating status	
SD260	Number of assigned bit devices	
SD261	↑	
SD262	↑	
SD263	↑	
SD264	↑	
SD265	↑	
SD280	Number of assigned word devices	
SD281	↑	

7.7.3 CR800-Q series

(1) Special relay

The compatible special relays are as follows.

Table 7-43:Q172DSRCPU robot CPU-compatible special relays

No.	Name	Remarks
SM0	Latest self-diagnostics error (Including annunciator ON)	
SM1	Latest self-diagnostics error (Not including annunciator ON)	
SM50	Clear error	
SM51	Low battery latch	
SM52	Low battery	
SM53	Momentary interruption detection	
SM203	STOP contact	
SM204	PAUSE contact	

(2) Special register

The compatible special registers are as follows.

Table 7-44:Q172DSRCPU robot CPU-compatible special registers

No.	Name	Remarks
SD0	Latest self-diagnostics error code	The 12000 (H, reset disabled), 12001(H), 12002(L), and 12003(C) codes are stored when robot-specific errors occur.
SD1	Time of occurrence of latest self-diagnostics error code	
SD2	↑	
SD3	↑	
SD4	Error information class code (CPU No.)	
SD5	Common error information code (CPU No.)	
SD6	Common error information code (I/O No.)	
SD51	Low battery latch	
SD52	Low battery	
SD53	Number of momentary interruption detections	
SD200	I/O operation rights	
SD201	LED status	
SD203	CPU operating status	
SD290	Device assignment	
SD291	↑	
SD292	↑	
SD293	↑	
SD294	↑	
SD295	↑	
SD296	↑	
SD297	↑	
SD298	↑	

No.	Name	Remarks
SD299	↑	
SD300	↑	
SD301	↑	
SD302	↑	
SD303	↑	
SD304	↑	
SD305	↑	
SD393	Multi-CPU system information	
SD395	↑	
SD396	↑	
SD397	↑	
SD398	↑	
SD399	↑	

MITSUBISHI ELECTRIC CORPORATION
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
Authorised representative:
Mitsubishi Electric Europe B.V. FA - European Business Group
Mitsubishi-Electric-Platz 1, D-40882 Ratingen, Germany
Tel: +49(0)2102-4860

Feb. 2021 MEE Printed in Japan on recycled paper. Specifications are subject to change without notice.