MELFA

Industrial Robots

Instruction Manual
(Detailed explanations of functions
and operations)

CR1/CR2/CR3/CR4/CR7/CR8/CR9
Controller

Art. no.: 132315
14 07 2005
Version K
BFP-A5992

MITSUBISHI ELECTRIC INDUSTRIAL AUTOMATION

Caution

Users of the robot given as a "Object Model" in "Table 1: List of origin position joint angles" must observe the details below.

⚠ **Warning** Do not release the brakes from an external source and forcibly move the robot arm at a high speed.

If the operation is carried out, a warning error indicating positional deviation (error No.: L1820) may occur. If it is confirmed that the position has deviated after carrying out "1. Operation to confirm positional deviation of origin position", the origin data has been lost.
In this case, reset the origin with the ABS method. Refer to section "ABS method" in the separate "Instruction Manual/Robot Arm Setup to Maintenance" for the operation methods.
If operation is carried out without resetting the origin, interference with peripheral devices or unforeseen operation could occur due to the loss of origin data.

1. Operation to confirm positional deviation of origin position
   (1)Set each axis of the robot to the ABS mark using the teaching box's joint jog operation.
   (2)Confirm that the joint angle displayed on the teaching box screen matches the value corresponding to the object model given in Table 1. If the values do not match, reset the origin with the ABS method.

Table 1: List of origin position joint angles (Position aligned with ABS mark arrow)

Object Model	Joint angle					
	J1	J2	J3	J4	J5	J6
RH-1000GHDC-SA	0degree	0degree	150mm	0degree		
RH-1000GJDC-SA	0degree	0degree	150mm	0degree	0degree	
RH-1000GHLC-SA	0degree	0degree	0degree	0degree		
RH-1000GHLLC-SA	0degree	0degree	0degree	0degree		
RH-1000GJLC-SA	0degree	0degree	0degree	0degree	0degree	
RH-1500GJC-SA/SB	138.7 degree	140 degree	0degree	180 degree	0degree	
RH-1500GC-SA**/SA5* -SB**/SB5*	138.7 degree	140 degree	0degree	180 degree	0degree	0degree
RC-1000GHWDC-SA	0degree	0degree	0degree	180 degree		
RC-1000GHWLC-SA	0degree	0degree	0degree	180 degree		
RC-1300G*	0mm	0mm	0mm	0degree	0mm	

⚠ Safety Precautions

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

⚠CAUTION

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

⚠CAUTION

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

⚠WARNING

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

⚠CAUTION

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

⚠WARNING

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

⚠CAUTION

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

⚠CAUTION

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

⚠CAUTION

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

⚠CAUTION Never carry out modifications based on personal judgments, or use non-designated maintenance parts.
Failure to observe this could lead to faults or failures.

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

⚠CAUTION

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

⚠CAUTION

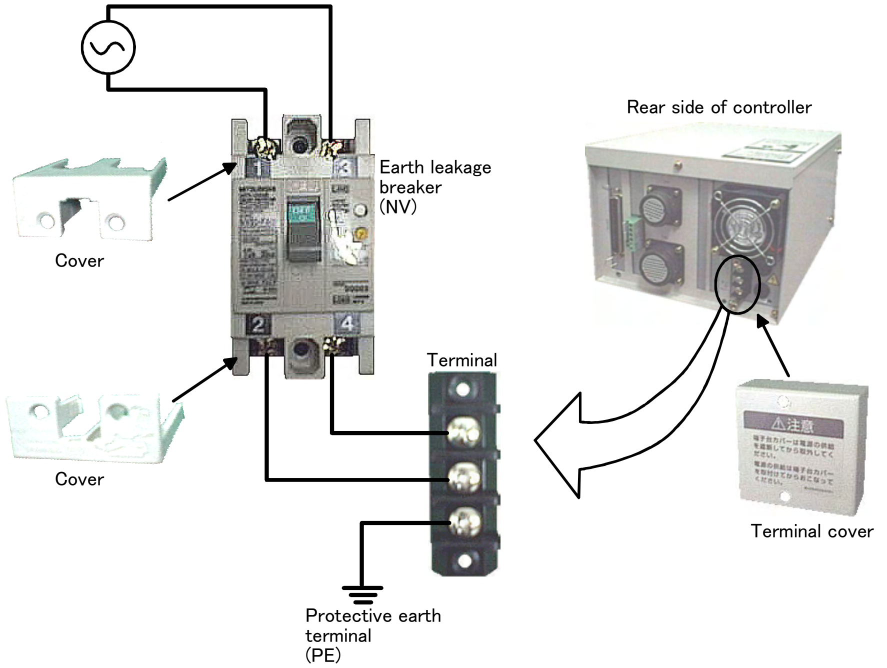
Do not turn off the main power to the robot controller while rewriting the internal information of the robot controller such as the program or parameters.
If the main power to the robot controller is turned off while in automatic operation or rewriting the program or parameters, the internal information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

Precautions for the basic configuration are shown below.(When CR1-571/CR1B-571 is used for the controller.)

⚠CAUTION

Provide an earth leakage breaker that packed together on the primary power supply of the controller as protection against electric leakage. Confirm the setting connector of the input power supply voltage of the controller, if the type which more than one power supply voltage can be used. Then connect the power supply.
Failure to do so could lead to electric shock accidents.

Power supply *RV-1A/2AJ series and RP-1AH/3AH/5AH series: Single phase 90-132VAC, 180-253VAC.
*Except the above: Single phase 180-253VAC.



⚠WARNING

For using RH-5AH/10AH/15AH series or RH-6SH/12SH/18SH series.
While pressing the brake releasing switch on the robot arm, beware of the arm which may drop with its own weight.
Dropping of the hand could lead to a collision with the peripheral equipment or catch the hands or fingers.

Revision history

Date	Specifications No.	Details of revisions
1999-06	BFP-A5992Z-*	First print
1999-09-20	BFP-A5992Z-A	Error in writing correction. The function of RH-1000 was considered.
1999-11-09	BFP-A5992	Error in writing correction.
2000-04-06	BFP-A5992-A	Attention in the power supply connection was added.(CR1 Controller)
2000-06-09	BFP-A5992-B	Parameter CNT was added. Emergency stop input of CR1 controller was added. JRC command was added. The power supply voltage of CR1 controller was corrected.
2000-07-12	BFP-A5992-C	Change title. Error in writing correction.
2001-06-05	BFP-A5992-Da	Major revision. Function list, publication of Q & A, description of system variables, as well as language and similar notation of system functions, and supplementation of various parameter functions.
2001-11-30	BFP-A5992-D	Formal style.
2001-12-12	BFP-A5992-E	Error in writing correction.
2002-11-15	BFP-A5992-F	The explanation and supplementary explanation of the new function corresponding to software version H7 edition were added. The notation of the input-and-output circuit terminal was corrected. Explanation of optimum acceleration/deceleration setting was added. Error in writing correction.
2003-10-14	BFP-A5992-G	The explanation and supplementary explanation of the new function corresponding to software version J1 edition were added. Change title. Error in writing correction.
2003-12-01	BFP-A5992-H	The explanation and supplementary explanation of the new function corresponding to software version J4 edition were added. Error in writing correction.
2005-02-28	BFP-A5992-J	The explanation and supplementary explanation of the new function corresponding to software version K1 edition were added. Error in writing correction.
2005-07-14	BFP-A5992-K	The explanation and supplementary explanation of the new function corresponding to software version K4 edition were added. Error in writing correction.

*Introduction

Thank you for purchasing the Mitsubishi industrial robot.
This instruction manual explains the functions and operation methods of the controller (CR1/CR2/CR3/CR4/CR7/CR8/CR9) and teaching pendant (R28TB), and the functions and specifications of the MELFA-BASIC IV programming language.
Always read through this manual before starting use to ensure correct usage of the robot.
Note that this document is prepared for the following software versions.

Controller	: Version K4 or later
T/B	: Version B2 or later

Contents

Page

Page

Contents

Page

Page

Contents

Page

Page

Contents

Page

Contents

Page

1 Before starting use

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

1.1 Using the instruction manuals

1.1.1 The details of each instruction manuals

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

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

Safety Manual	Explains the common precautions and safety measures to be taken for robot handling, system design and manufacture to ensure safety of the operators involved with the robot.
Standard Specifications	Explains the product's standard specifications, factory-set special specifications, option configuration and maintenance parts, etc. Precautions for safety and technology, when incorporating the robot, are also explained.
Robot Arm Setup & Maintenance	Explains the procedures required to operate the robot arm (unpacking, transportation, installation, confirmation of operation), and the maintenance and inspection procedures.
Controller Setup, Basic Operation and Maintenance	Explains the procedures required to operate the controller (unpacking, transportation, installation, confirmation of operation), basic operation from creating the program to automatic operation, and the maintenance and inspection procedures.
Detailed Explanation of Functions and Operations	Explains details on the functions and operations such as each function and operation, commands used in the program, connection with the external input/output device, and parameters, etc.
Explanations of MOVEMASTER COMMANDS	Explains details on the MOVEMASTER commands used in the program. (For RV-1A/2AJ, RV-2A/3AJ and RV-3S/3SJ/3SB/3SJB series)
Troubleshooting	Explains the causes and remedies to be taken when an error occurs. Explanations are given for each error No.

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

Symbol	Meaning
⚠DANGER	Precaution indicating cases where there is a risk of operator fatality or serious injury if handling is mistaken. Always observe these precautions to safely use the robot.
⚠WARNING	Precaution indicating cases where the operator could be subject to fatalities or serious injuries if handling is mistaken. Always observe these precautions to safely use the robot.
⚠CAUTION	Precaution indicating cases where operator could be subject to injury or physical damage could occur if handling is mistaken. Always observe these precautions to safely use the robot.
[JOINT]	If a word is enclosed in brackets or a box in the text, this refers to a key on the teaching pendant.
[+/FORWD]+[+X] (A) (B)	This indicates to press the (B) key while holding down the (A) key. In this example, the [+/Forward] key is pressed while holding down the [+X/+Y] key.
[STEP/MOVE]+([COND]-[RPL]) (A) (B) (C)	This indicates to hold down the (A) key, press and release the (B) key, and then press the (C) key. In this example, the [Step/Move] key is held down, the [Condition] key is pressed and released, and the [Replace] key is pressed.
T/B	This indicates the teaching pendant.

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.

⚠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.

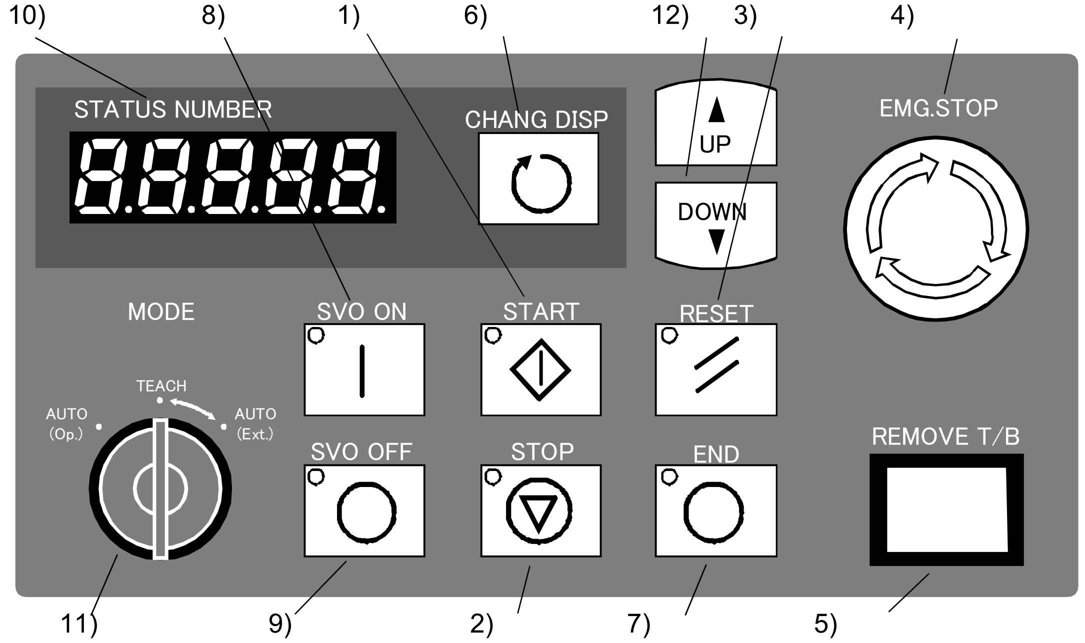
⚠WARNING When the robot arm has to be moved by hand from an external area, do not place hands or fingers in the openings. Failure to observe this could lead to hands or fingers catching depending on the posture.

⚠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.

2 Explanation of functions

2.1 Operation panel (O/P) functions



(1) Explanation of buttons on the operation panel

Table 2-1:Names of each part on operation panel (Controller)

	Button name	Function
1)	Start button	This executes the program and operates the robot. The program is run continuously. The LED (green) lights during operation. When only executing the program to which "ALWAYS" was set as start conditions, the LED not lights.
2)	Stop button	This stops the robot immediately. The servo does not turn OFF. The LED (red) lights while stopped. (Turns on when the program is interrupted.) However, the program to which "ALWAYS" was set as start conditions does not stop.
3)	Reset button	This resets the error. The LED (red) lights while an error is occurring. This also resets the program's interrupted state and resets the program. (Only when program numbers are displayed.)
4)	Emergency stop button	This stops the robot in an emergency state. The servo turns OFF. Turn to the right to cancel.
5)	T/B connection switch	This is used to connect/disconnect the T/B without turning OFF the controller's control power. The T/B should be removed within five seconds after pressing the switch. An error occurs if more than five seconds elapses after pressing the switch. Similarly, when the T/B should be remounted, it should be connected and the switch returned to the original position within five seconds.
6)	Display changeover switch	This changes the details displayed on the display panel in the order of "Program No." - "Line No." - "Override". When an error is occurring, "Program No."- "Line No." - "Override" appear only when the key is pressed. The error No. will appear when the key is released.
7)	End button	This stops the program being executed at the last line or END statement. (Cycle operation) The LED (red) winks during cycle operation. (Cancels continuous operation.) When it is pressed again while flushing in software version J1 or later, the operation returns to continuous operation.
8)	SVO.ON button	This turns ON the servo power. The LED (green) lights during servo ON.
9)	SVO.OFF button	This turns OFF the servo power. The LED (red) lights during servo OFF.
10)	STATUS.NUMBER	The error No., program No., override value (%), etc., are displayed. The program name is shown with simplified symbols if alphabetic characters are used.
11)	MODE changeover switch Note1)	This changes the robot's operation rights. Note2) AUTO(Op.) : Only operations from the controller are valid. Operations for which the operation rights must be at the external device or T/B are not possible. TEACH : When the T/B is valid, only operations from the T/B are valid. Operations for which the operation rights must be at the external device or controller are not possible. AUTO(Ext.): Only operations from the external device are valid. Operations for which the operation rights must be at the T/B or controller are not possible.
12)	UP/DOWN button	This scrolls up or down the details displayed on the display panel (Valid for program numbers, override, and error numbers)

⚠CAUTION

Note1) The servo will turn OFF when the controller's [MODE] switch is changed. Note that axes not provided with brakes could drop with their own weight.

Carry out the following operations to prevent the servo from turning OFF whenthe [MODE] switch is changed.

The servo on status can be maintained by changing the mode with keeping pressing
lightly the deadman switch of T/B. The operating method is shown below.

*When the mode is changed from TEACH to AUTO.
1) While holding down the deadman switch on the T/B, set the [ENABLE/DISABLE] switch to "DISABLE".
2) While holding down the deadman switch on the T/B, set the controller [MODE] switch to "AUTO".
3) Release the T/B deadman switch.

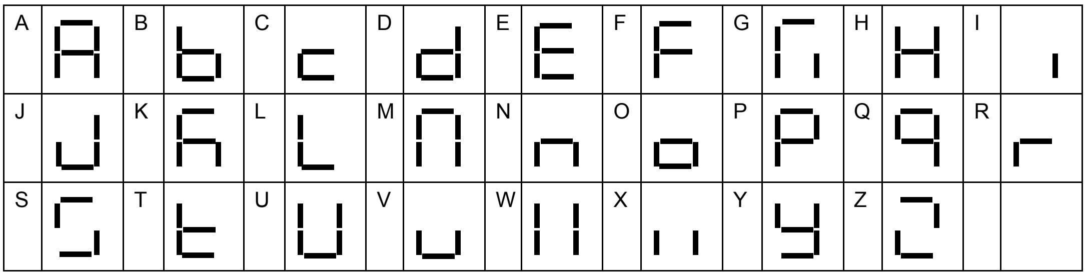
*When the mode is changed from AUTO to TEACH.
1) While the [ENABLE/DISABLE] switch on the T/B is "DISABLE", hold down the deadman switch.
2) While holding down the deadman switch on the T/B, set the controller [MODE] switch to "TEACH".
3) While holding down the deadman switch on the T/B, set the [ENABLE/DISABLE] switch to "ENABLE", then do the operation of T/B that you wish.

Note2) If you want to retain the LED display when switching the mode changeover switch, change the following parameter.

Parameter name	Meaning of the value	Explanation
OPDISP	0:Display the override.(default) 1:Keep display mode.	Specify the action of the LED display when changing the mode changeover switch.

(2) About the status number display

The following is a description of the simplified symbols shown on the 7-segment LED display when displaying a program name specified with alphabetic characters.



The character "P" is fixed at the beginning of the program name display, which means that the number of characters that can be displayed are four or less. Make sure to use no more than four characters when entering the program name.

It is not possible to select a program name consisting of more than four characters from the operation panel. However, it is allowed to create a program name consisting of more than four characters in the case of a program to be executed as a sub-program by the CALLP instruction of the robot language.

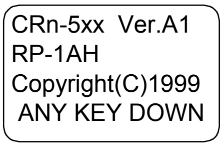
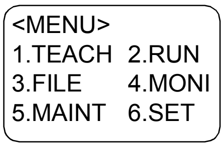
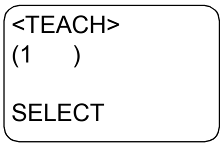
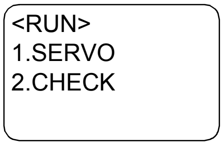
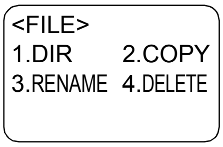
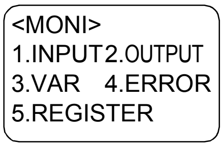
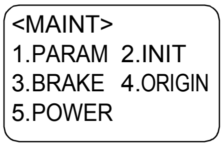
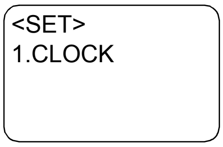
2.2 Teaching pendant (T/B) functions

This chapter explains the functions of R28TB (optional).

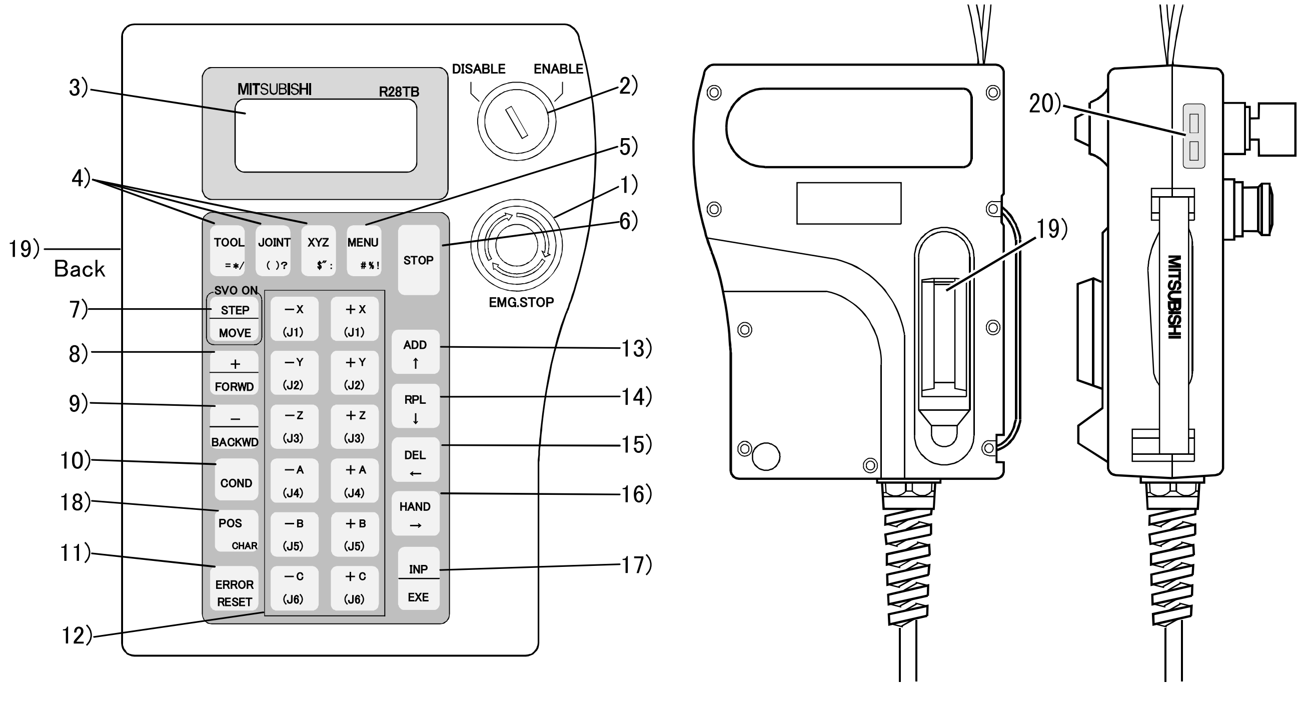
(1) Display screens and functions

Table 2-2 shows the functions corresponding to the screens displayed on the T/B, and the pages on which expla-nations of the operation methods are given.
The screen tree is shown in the Page 13, "(1) Screen tree".

Table 2-2:Display screens and functions

Screen display	Function	Explanation page
Title screen  CRn-5xx Ver.A1 RP-1AH Copyright(C)1999 ANY KEY DOWN	Type and software version display	Page 13, "3.1 Operation of the teaching pendant menu screens"
Menu screen  <MENU> 1.TEACH 2.RUN 3.FILE 4.MONI 5.MAINT 6.SET	Selection of following screens	
Teach screen  <TEACH> (1) SELECT	Selection and editing of program No.	Page 24, "3.5.1 Creating a program"
Operation menu screen  <RUN> 1.SERVO 2.CHECK	Servo ON/OFF Step operation	Page 42, "3.8 Turning the servo ON/OFF"
Control menu screen  <FILE> 1.DIR 2.COPY 3.RENAME 4.DELETE	Program list display	Page 45, "(1) Program list display"
	Program protection	Page 46, "(2) Program protection function"
	Program copy	Page 47, "(3) Copying programs"
	Program name change	Page 48, "(4) Changing the program name (Renaming)"
	Program deletion	Page 49, "(5) Deleting a program"
Monitor menu screen  <MONI> 1.INPUT2.OUTPUT 3.VAR 4.ERROR 5.REGISTER	Input signal status display	Page 50, "(1) Input signal monitor"
	Output signal status change and display	Page 51, "(2) Output signal monitor"
	Variable details display	Page 52, "(3) Variable monitor"
	Error history display	Page 53, "(4) Error history"
	Register display	Use when CC-Link option is used. Separate manual "CC-Link Interface".
Maintenance screen  <MAINT> 1.PARAM 2.INIT 3.BRAKE 4.ORIGIN 5.POWER	Parameter setting value display and change	Page 54, "(1) Setting the parameters"
	Program initialization	Page 55, "(2) Initializing the program"
	Battery timer initialization	Page 56, "(3) Initializing the battery consumption time"
	Brake release	Page 57, "(4) Releasing the brakes"
	Origin setting	Page 58, "(5) Setting the origin"
	Operation time display	Page 58, "(6) Displaying the clock data for maintenance"
Setting menu screen  <SET> 1.CLOCK	Date and time display and change	Page 59, "(1) Setting the time"

(2) Function of each key



Front Back Side

Fig.2-1:T/B keys

	Key	Explanation
1)	[EMG. STOP] switch	This is a push-button switch with lock function for emergency stop. When this switch is pressed, the servo will turn OFF and the robot will stop immediately regardless of the T/B enable/disable state. To cancel this state, turn the switch clockwise.
2)	[ENABLE/DISABLE] switch	This changeover switch is used to enable or disable the T/B key operations. To carry out operations using the T/B, always set this switch to "ENABLE" (valid). Operations with the T/B will be enabled, and operations from the controller and external sources will be disabled. The T/B will have the operation rights. To operate with the controller or external source, set this switch to "DISABLE" (invalid). It is possible to change modes of operation related to the monitor and the override even in the disabled status. Set this switch to "DISABLE" position while editing in order to save the current program.
3)	Display LCD	The program contents and robot state are displayed with the T/B key operations.
4)	[TOOL] key	This selects the TOOL jog mode
4)	[JOINT] key	This selects the JOINT jog mode. Press twice to select the additional axis jog mode.
4)	[XYZ] key	The XYZ jog mode is selected if the key is pressed while in the TOOL and/or JOINT jog condition, the 3-axis XYZ jog mode is selected if pressed twice, and the cylinder jog mode is selected if pressed three times.
5)	[MENU] key	This returns the display screen to the menu screen. If the key is pressed while editing, the current program is saved.
6)	[STOP] key	This stops the program and decelerates the robot to a stop. This is the same function as the [STOP] switch on the front of the controller, and can be used even when the T/B [ENABLE/DISABLE] switch is set to DISABLE.
7)	[STEP/MOVE] key	Jog operations are possible when this key is pressed simultaneously with the 12) jog operation key. Step jump is carried out when pressed simultaneously with the [INP/EXE] key. Press it when the servo is off to turn the servo on (while the deadman switch is pressed).
8)	[+/FORWD] key	Step feed is carried out when this key is pressed simultaneously with the [INP/EXE] key. On the edit screen, the next program line is displayed. Press it at the same time as the [STEP/MOVE] key during program operation to increase the override (speed). It is possible to perform this operation even when the T/B is disabled. A "+" is input when characters are entered.

	Key	Explanation
9)	[-/BACKWD] key	On the edit screen, the previous line is displayed. When pressed simultaneously with the [INP/EXE] key, the axis will return along the robot's operation path. When pressed simultaneously with the [STEP/MOVE] key, the override (speed) will decrease.It is possible to perform this operation even when the T/B is disabled. A "+" is input when characters are entered.
10)	[COND] key	Use this key to display the program instruction screen.
11)	[ERROR RESET] key	This key resets an error state that has occurred. When pressed simultaneously with the [INP/EXE] key, the program will be reset.
12)	[Jog operation] key (12 keys from [-X (J1) to +C (J6)]	In this manual, these keys are generically called the "jog operation" keys. When JOINT jog is selected, each axis will rotate, and when XYZ jog is selected, the robot will move along each coordinate system. These keys are also used to input numeric values such as when selecting a menu or inputting a step No.
13)	[ADD] key	Moves the cursor upward. It also, you can add or correct position data by pressing it simultaneously with the [STEP] key on the position data edit screen. (T/B version B1 or later.)
14)	[RPL] key	Moves the cursor to the downward. It also, you can display the next screen after the current position display by pressing it simultaneously with the [STEP] key on the position data edit screen. (T/B version B1 or later.)
15)	[DEL]	This deletes the position data. It also moves the cursor to the left.
16)	[HAND] key	The following hand operations When pushed simultaneously with [+C (J6)] or [-C (J6)] key, operate the hand 1. When pushed simultaneously with [+B (J5)] or [-B (J5)] key, operate the hand 2. When pushed simultaneously with [+A (J4)] or [-A (J4)] key, operate the hand 3. When pushed simultaneously with [+Z (J3)] or [-Z (J3)] key, operate the hand 4. This key also moves the cursor to the right.
17)	[INP/EXE] key	This inputs the program, and carries out step feed/return
18)	[POS CHAR] key	Use this key to display the position edit screen and to enter characters and symbols.
19)	Deadman switch	When the [ENABLE/DISABLE] switch 2 is enabled, and this switch is released or pressed with force, the servo will turn OFF. Press this switch lightly when carrying out functions with the servo ON, such as jog operations. If emergency stop or servo OFF have been applied, and the servo is OFF, the servo will not turn ON even when this switch is pressed. In this case, carry out the servo ON operation again.
20)	Contrast setting switch (Top: Shade, bottom: light)	This sets the display LCD brightness.

2.2.1 Operation rights

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

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

Table 2-3:Relation of setting switches and operation rights O:Has operation rights, X:Does not have operation rights

Setting switch	T/B [ENABLE/DISBLE]	DISABLE			ENABLE		
	Controller [MODE]	AUTO(Op.)	AUTO(Ext.)	TEACH	AUTO(Op.)	AUTO(Ext.)	TEACH
Operation rights	T / B	X	X	X	X[Note 2)]	X[Note 2)]	O
	Controller operation panel	O	X	X	X[Note 2)]	X[Note 2)]	X
	Personal computer	X	O[Note 1)]	X	X[Note 2)]	X[Note 2)]	X
	External signal	X	O[Note 1)]	X	X[Note 2)]	X[Note 2)]	X

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

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

Table 2-4:Operations requiring operation rights Operation item: O=Requires operation rights, X= Does not require operation rights

Class	Operation rights	Operation
Operation	O	Servo ON
	X	Servo OFF
	O	Program stop/cycle stop
	X	Slot initialization (program reset)
	O	Error reset
	X	Override change. Note this is always possible from the T/B.
	O	Override read
	X	Program No. change
	O	Program No./line No. read
	X	Program stop/cycle stop
Input/output signal	X	Input/output signal read
	X	Output signal write
	O	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
	O	Hand output signal write
Program editing Note1)	X	Line registration/read/call; Position addition/correction/read; Variable write/read
	O	Step feed/return, execution
	X	Step up/down
	O	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
	O	Origin setting, parameter change

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

2.3 Functions Related to Movement and Control

This controller has the following characteristic functions.

Function	Explanation	Explanation page
Optimum speed control	This function prevents over-speed errors as much as possible by limiting the speed while the robot is tracking a path, if there are postures of the robot that require the speed to be limited while moving between two points. However, the speed of the hand tip of the robot will not be constant if this function is enabled.	Page 213, "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 193, "OADL (Optimal Acceleration)", Page 179, "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 134, "CMP TOOL (Composition Tool)"
Impact Detection	The robot stops immediately if the robot's tool or arm interferes with a peripheral device, minimizing damage. This function can be activated during automatic operation as well as during jog operation. (Limited to the RV-S/ RH-S series.) Note) Please note that this function cannot be used together with the multi-mechanism control function.	Page 141, "COLCHK (Col Check)" Refer to "COL" parameter in Page 306, "5 Functions set with parameters".
Maintenance Forecast	The maintenance forecast function forecasts the robot's battery, belt and grease maintenance information based on the robot's operating status. This function makes it possible to check maintenance information using the optional Personal Computer Support software. (Limited to the RV-S/ RH-S series.) Note) Please note that this function cannot be used together with the multi-mechanism control function.	Use optional Personal Computer Support software. This function can be used in Version E1 or later.
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. This function can be used with the vertical multi-joint robot and the RH-S series.	Use optional Personal Computer Support software. Vertical multi-joint robot: This function can be used in Version E1 or later. RH-S series: This function can be used in Version F2 or later.
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 65, "(4) Continuous movement", Page 138, "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 86, "4.2.1 What is multitasking?", Page 226, "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 306, "5 Functions set with parameters".
Continuity function	With this function, it is possible to store the status at power off and resume from the same status when the power is turned on again.	Refer to "CTN" parameter in Page 306, "5 Functions set with parameters".
Additional axis control	With this function, it is possible to control up to two axes as additional axes of the robot. Since the positions of these additional axes are stored in the robot's teaching data as well, it is possible to perform completely synchronous control. In addition, arc interpolation while moving additional axes (travelling axes) is also possible. The additional axis interface card optional is required of CR1/CR2 series controller.	Separate manual "ADDITIONAL AXIS INTERFACE".
Multi-mechanism control	With this function, it is possible to control up to two (excluding the standard robots) robots (user mechanism) driven by servo motors, besides the standard robots.	Separate manual "ADDITIONAL AXIS INTERFACE".

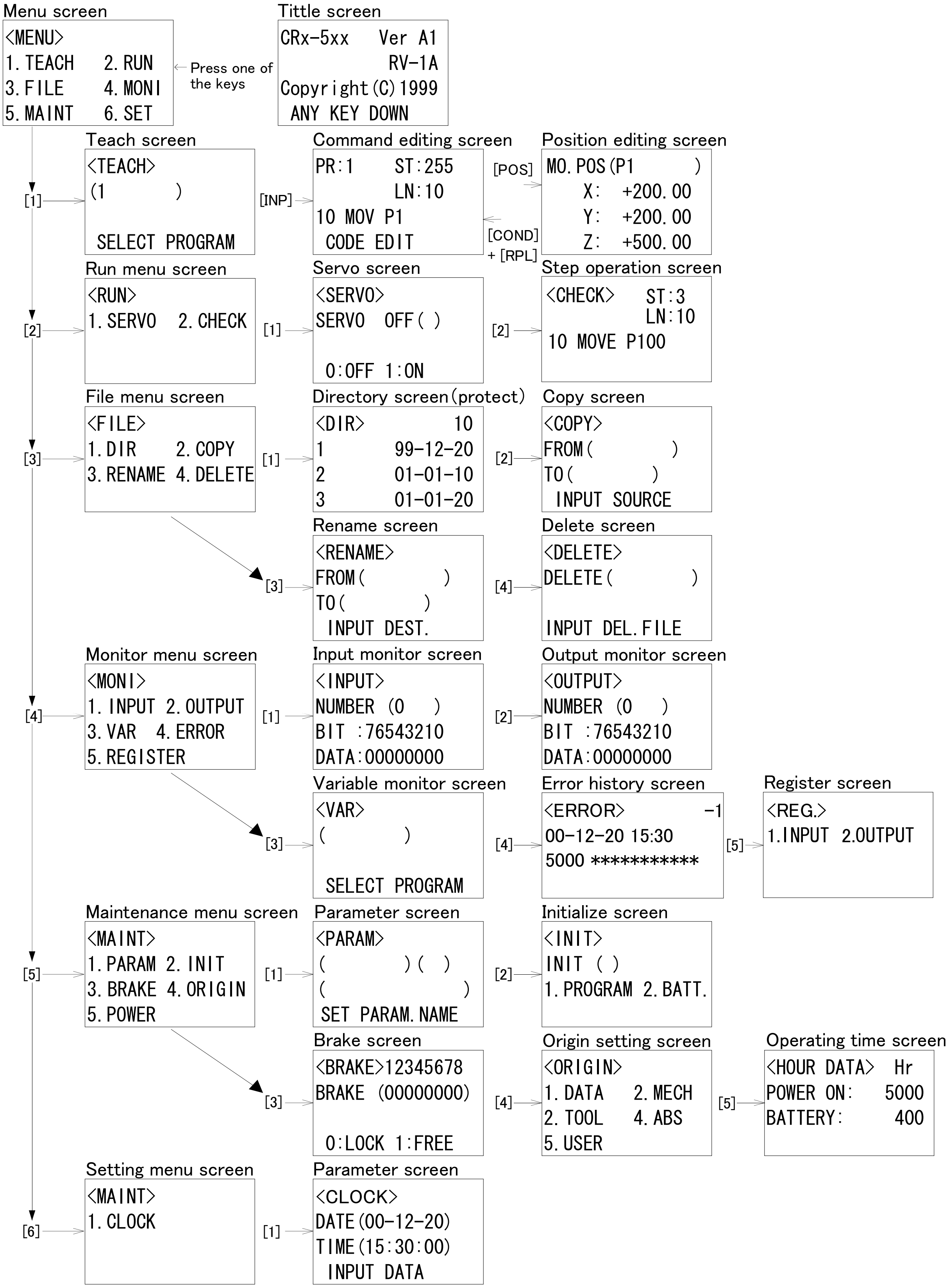
Function	Explanation	Explanation page
External device communication function	The following methods are available for communicating with the external devices For controlling the controller and for interlock within a program 1) Via input/output signals (32 points each for standard input/output in the CR2, CR3, CR4, CR7, CR8 and CR9) 2) Via CC-Link (optional) As a data link with an external device 3) Communication via RS-232C (1 standard port, and up to four optional ports) 4) Communication via Ethernet 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 248, "M_IN/M_INB/ M_INW", Page 254, "M_OUT/M_OUTB/ M_OUTW". Page 337, "5.15 About the communication setting" Separate manual "Ethernet Interface".
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 146, " DEF ACT (Define act)", Page 121, " 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 125, " 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 71, "4.1.2 Pallet operation", Page 157, "DEF PLT (Define pallet)",Page 200, "PLT (Pallet)"
User-defined area function	With this function, it is possible to specify an arbitrary space consisting of up to eight 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 328, "5.8 About user-defined area",Page 262, "M_UAR". Page 304, "ZONE", Page 305, "ZONE 2"
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 306, "5 Functions set with parameters" Refer to Page 329, "5.9 Free plane limit"

3 Explanation of operation methods

This chapter describes how to operate R28TB (optional)

3.1 Operation of the teaching pendant menu screens

(1) Screen tree



Menu screen
```
<MENU>
1.TEACH   2.RUN
3.FILE    4.MONI
5.MAINT   6.SET
```

Tittle screen
```
CRx-5xx   Ver A1
          RV-1A
Copyright(C)1999
 ANY KEY DOWN
```

Teach screen
```
<TEACH>
(1         )

 SELECT PROGRAM
```

Command editing screen
```
PR:1     ST:255
         LN:10
10 MOV P1
 CODE EDIT
```

Position editing screen
```
MO.POS(P1        )
    X:  +200.00
    Y:  +200.00
    Z:  +500.00
```

Run menu screen
```
<RUN>
1.SERVO  2.CHECK
```

Servo screen
```
<SERVO>
SERVO  OFF( )

 0:OFF 1:ON
```

Step operation screen
```
<CHECK>   ST:3
          LN:10
10 MOVE P100
```

File menu screen
```
<FILE>
1.DIR     2.COPY
3.RENAME 4.DELETE
```

Directory screen(protect)
```
<DIR>          10
1        99-12-20
2        01-01-10
3        01-01-20
```

Copy screen
```
<COPY>
FROM(          )
TO(          )
 INPUT SOURCE
```

Rename screen
```
<RENAME>
FROM(          )
TO(          )
 INPUT DEST.
```

Delete screen
```
<DELETE>
DELETE(          )

INPUT DEL.FILE
```

Monitor menu screen
```
<MONI>
1.INPUT 2.OUTPUT
3.VAR   4.ERROR
5.REGISTER
```

Input monitor screen
```
<INPUT>
NUMBER  (0    )
BIT :76543210
DATA:00000000
```

Output monitor screen
```
<OUTPUT>
NUMBER  (0    )
BIT :76543210
DATA:00000000
```

Variable monitor screen
```
<VAR>
(          )

 SELECT PROGRAM
```

Error history screen
```
<ERROR>          -1
00-12-20 15:30
5000 **********
```

Register screen
```
<REG.>
1.INPUT  2.OUTPUT
```

Maintenance menu screen
```
<MAINT>
1.PARAM 2.INIT
3.BRAKE 4.ORIGIN
5.POWER
```

Parameter screen
```
<PARAM>
(          )(   )
(                )
 SET PARAM.NAME
```

Initialize screen
```
<INIT>
INIT ( )
1.PROGRAM 2.BATT.
```

Brake screen
```
<BRAKE>12345678
BRAKE  (00000000)

 0:LOCK 1:FREE
```

Origin setting screen
```
<ORIGIN>
1.DATA    2.MECH
2.TOOL    4.ABS
5.USER
```

Operating time screen
```
<HOUR DATA>  Hr
POWER ON:    5000
BATTERY:      400
```

Setting menu screen
```
<MAINT>
1.CLOCK
```

Parameter screen
```
<CLOCK>
DATE(00-12-20)
TIME(15:30:00)
 INPUT DATA
```

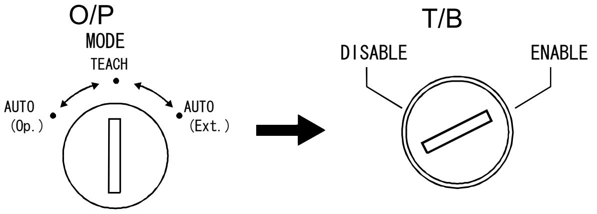
(2) Selecting a menu

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 [INP] key.

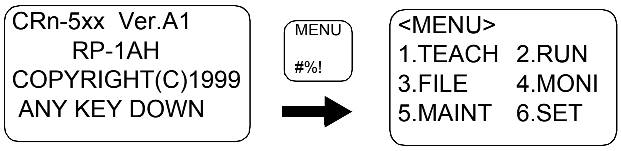
How to select the TEACH screen ("1. TEACH") from the menu screen with each method is shown below.



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

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

Display the MENU screen from the TEACH screen.

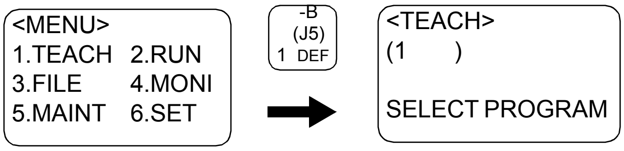


```
CRn-5xx  Ver.A1
       RP-1AH
COPYRIGHT(C)1999
 ANY KEY DOWN
```

```
<MENU>
1.TEACH   2.RUN
3.FILE    4.MONI
5.MAINT   6.SET
```

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

*Press the number key method

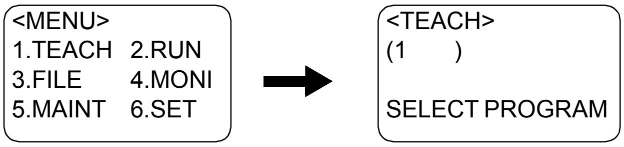


```
<MENU>
1.TEACH   2.RUN
3.FILE    4.MONI
5.MAINT   6.SET
```

```
<TEACH>
(1       )

SELECT PROGRAM
```

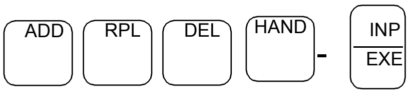
1) Press the [1] key. The <TEACH> screen will appear.

*Use the arrow key method



```
<MENU>
1.TEACH   2.RUN
3.FILE    4.MONI
5.MAINT   6.SET
```

```
<TEACH>
(1       )

SELECT PROGRAM
```



Move the cursor - set

1) Press the arrow keys and move the cursor to "1. TEACH", and then press the [INP] key. The <TEACH> screen will appear. The same operations can be carried out on the other menu screens.

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

Using the T/B

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

Inputting numbers and spaces

To input a number, press the key having a number on the lower left.

To input a space, press the key having "SPACE" on the lower left.

Correcting incorrect numbers

Press the [DEL] key while holding down the [CHAR] key to delete the character, and then input it again.

If the cursor is returned by pressing the [<-] key, and a character is input, it will be inserted.

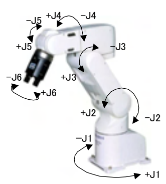
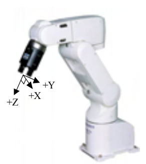
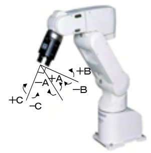
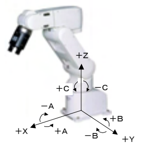
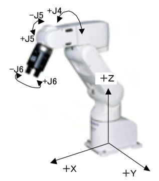
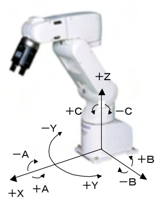
3.2 Jog Feed (Overview)

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 "RV-1A" 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  −J5 +J4 −J4 +J5 −J3 −J6 +J3 +J6 +J2 −J2 −J1 +J1	1) Set the key switch to the [ENABLE] position. 2) Hold the deadman lightly. 3) Press the [STEP/MOVE] key. (The servo is turned on.) 4) Press the [JOINT] key to change to the JOINT jog mode. 5) Press the key corresponding to each of the axes from J1 to J6. 6) Press the [JOINT] key twice to shift to the additional axis mode.	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 [TOOL] 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 [XYZ] key to change to the XYZ jog mode.	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 [XYZ] key twice to switch to the 3-axis XYZ jog mode.	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 [XYZ] key twice to switch to the CYLNDER jog mode.	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.

If the robot's control point comes near a singular point during the operation of TOOL jog, XYZ jog or CYLINDER jog mode among the types of jog feed listed in Table 3-1, a warning mark is displayed on the T/B screen together with the sound of buzzer to warn the operator. It is possible to set this function valid or invalid by parameter MESNGLSW. (Refer to Page 306, "5 Functions set with parameters".) Please refer to Page 344, "5.17 About the singular point adjacent alarm" for details of this function.

3.2.2 Speed of jog feed

Press the [STEP/MOVE] key to display the current position and speed (%) on the screen. To change these values, hold the [STEP/MOVE] key down and press either the [+] key or the [-] key. The following types of jog feed speed are available.

[-] key -- [+] key

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

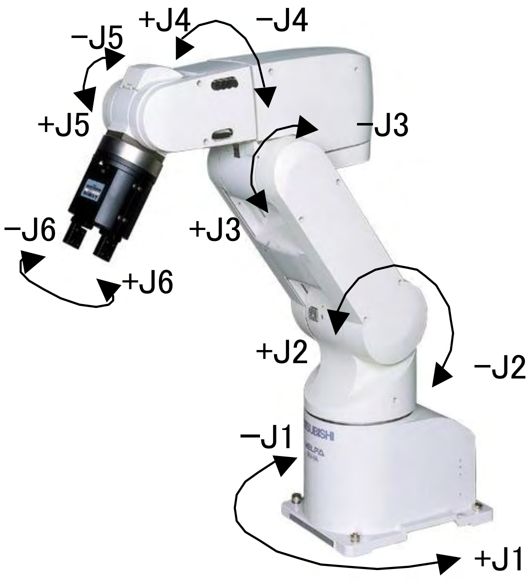
LOW and HIGH are fixed-dimension feed. In fixed-dimension feed, the robot moves a fixed amount every time the key is pressed. The amount of movement depends on the individual robot.

Table 3-2:Fixed-dimension of RV-1A

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

3.2.3 JOINT jog

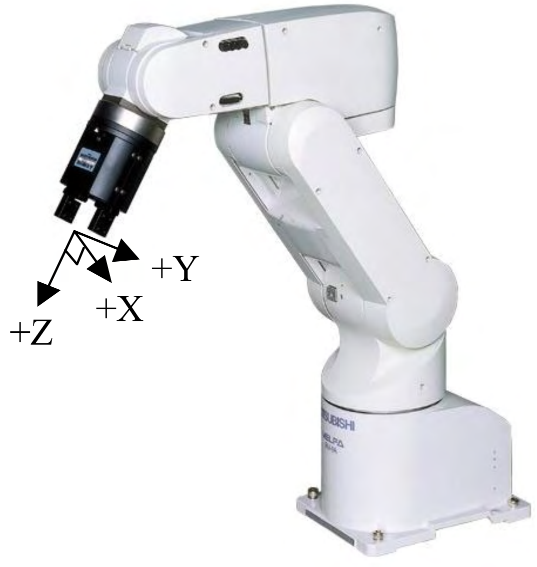
Adjusts the coordinates of each axis independently in angle units.

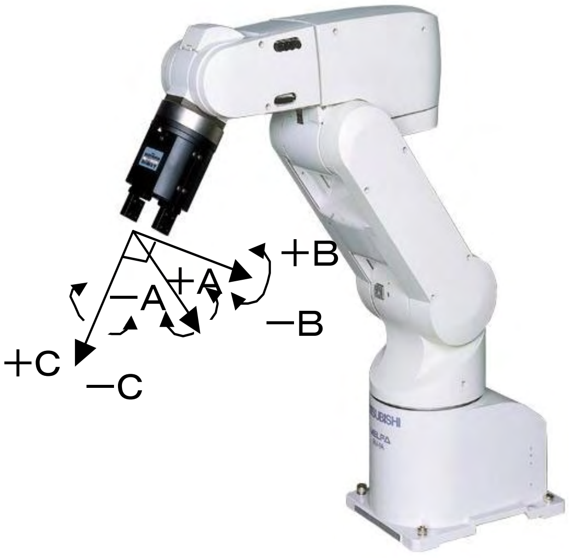


3.2.4 TOOL jog

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

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

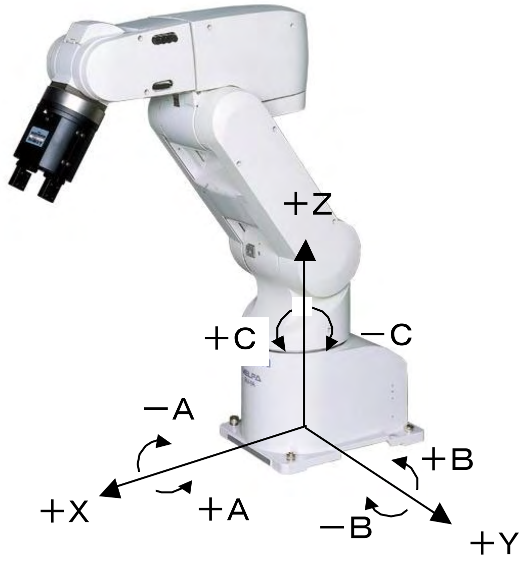




3.2.5 XYZ jog

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

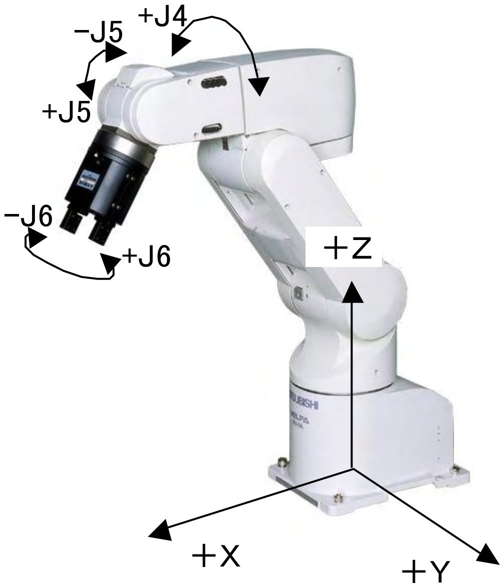
The X, Y, and Z axis coordinates are adjusted in mm units. The A, B, and C axis coordinates are adjusted in angle units.



3.2.6 3-axis XYZ jog

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

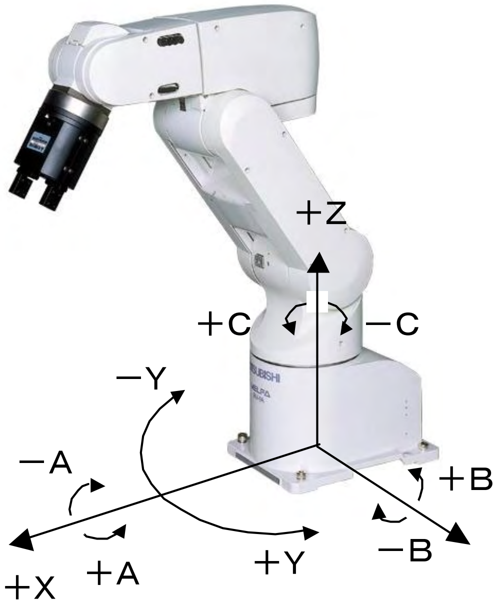
The X, Y, and Z axis coordinates are adjusted in mm units. The J4, J5, and J6 axis coordinates are adjusted in angle units.



3.2.7 CYLNDER jog

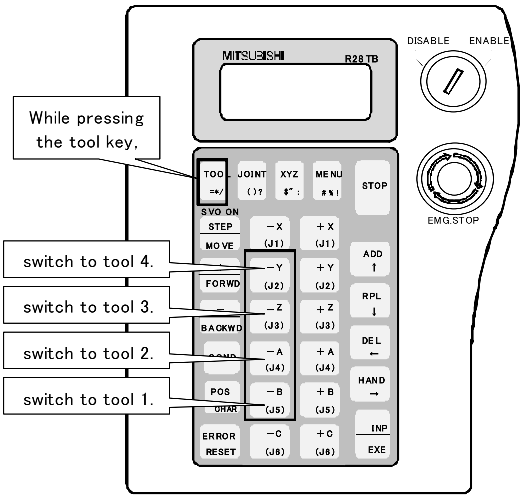
Adjusting the X-axis coordinate moves the hand in the radial direction away from the robot's origin. Adjusting the Y-axis coordinate rotates the arm around the J1 axis. Adjusting the Z-axis coordinate moves the hand in the Z direction of the robot coordinate system. Adjusting coordinates of the A, B, 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.

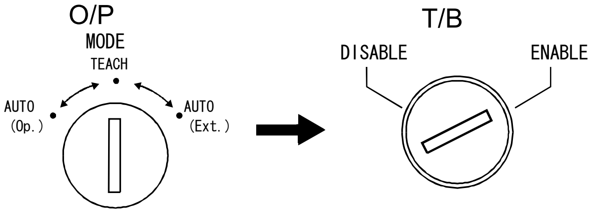


3.2.8 Switching Tool Data

With the combination of the controller's software version J1 or later and the teaching pendant's version A2 or later, tool data can be switched easily via the teaching pendant.



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



1) Set the controller's [MODE] switch to "TEACH."

2) Set the teaching pendant to "ENABLE."

```
<TOOL SETTING>
TOOL NUMBER:0

PUSH 1 TO 4
```

```
<TOOL SETTING>
TOOL NUMBER:0

PUSH 1 TO 4
```

3) While holding down the tool key, switch tool data by pressing from the [1] key to the [4] key.

TOOL + -B (J5) 1 DEF

Selecting Tool Data 1

```
JOINT     T2  100%
J1          +34.50
J2          +20.00
J3          +80.00
```

4) A tool number appears on the jog screen.
A tool number is displayed after T in the upper right of the T/B screen.

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

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

Verifying the Tool Number

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

Related Information

MEXTL, MEXTL1, MEXTL2, MEXTL3 and MEXTL4 parameters
TOOL instruction, M_TOOL variable
The MEXTL parameter holds tool data at that point. When using the MEXTL1 to 4 parameters, be careful as the MEXTL parameter is overwritten once a tool number is selected.
Execute the TOOL instruction to return the tool number to 0.

3.2.9 Impact Detection during Jog Operation

The RV-S/RH-S series is installed with the impact detection function. Impact detection can be enabled even during jog operation. (The initial value is set as disabled.) If the controller detects interference with a peripheral device during jog operation, an error numbered in 1010's will be issued (the first digit is the axis number).

Other models do not function even if the parameter is enabled.

Parameter	Name	No. of elements	Description	Initial value
Impact Detection * This function can be used in the controller's software version J2 or later.	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.	RV-S series 0,0,1 RH-S series 1,0,1 Other models 0,0,0
Detection level during jog operation	COLLVLJG	Integer 8	Set the detection sensitivity during jog operation for each joint axis. Unit (%) To increase detection sensitivity, reduce the numeric value. If an impact error occurs even when no impact occurs during jog operation, increase a numeric value. Setting range: 1 to 500 (%)	The standard is 200,200,200,200,200,200,200,200, but it varies with models.
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	It is released only with the RV-S/RH-S series. The value may vary with models. The maximum load is set as the load.
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	It is released only with the RV-S/RH-S series. 0.0,0.0,0.0,0.0,0.0,0.0,0.0

(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

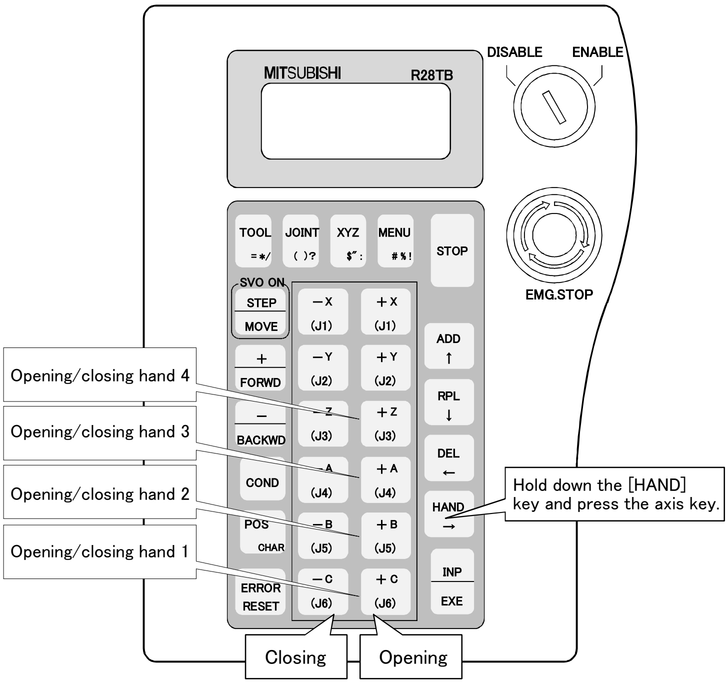
Enabling the impact detection function does not completely prevent the robot, hand, workpiece and others from being damaged, which may be caused by interference with peripheral devices. In principle, operate the robot by paying attention not to interfere with peripheral devices.

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 44, "Operation to Temporarily Reset an Error that Cannot Be Canceled".

3.3 Opening/Closing the Hands

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



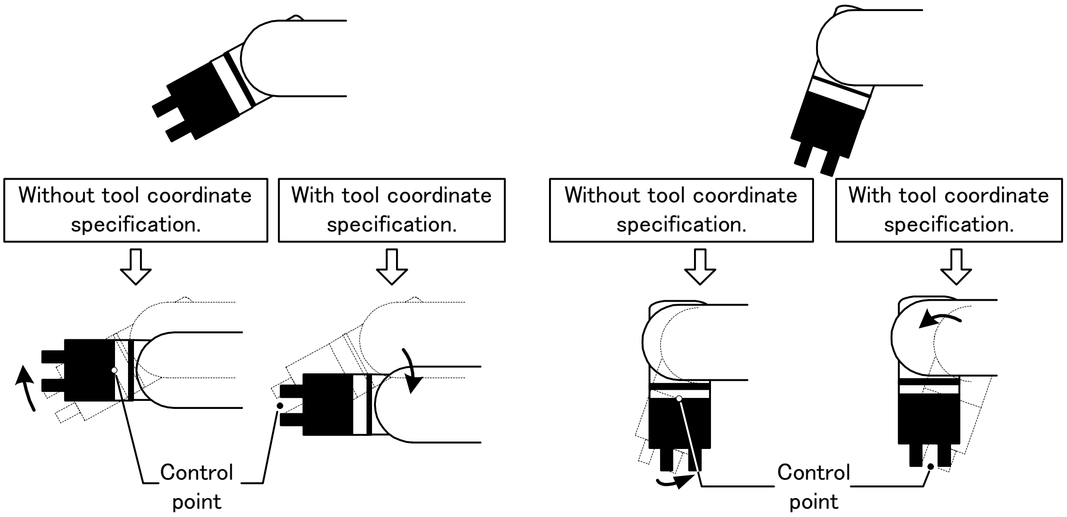
1) Set the key switch to the [ENABLE] position.

2) Hold the [HAND] key down and press each axis key. For example, hold down the [HAND] key and press the [+C] key to open hand 1.

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 333, "5.12 About the hand type" and Page 334, "5.13 About default hand status".

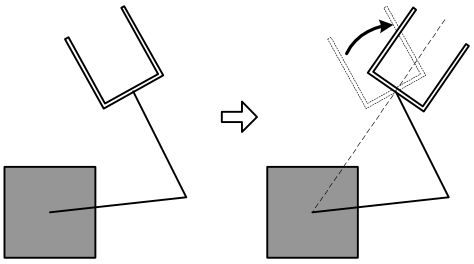
3.4 Aligning the Hand

The posture of the hand attached to the robot can be aligned in units of 90 degrees.
This feature moves the robot to the position where the A, B and C components of the current position are set at the closest values in units of 90 degrees.

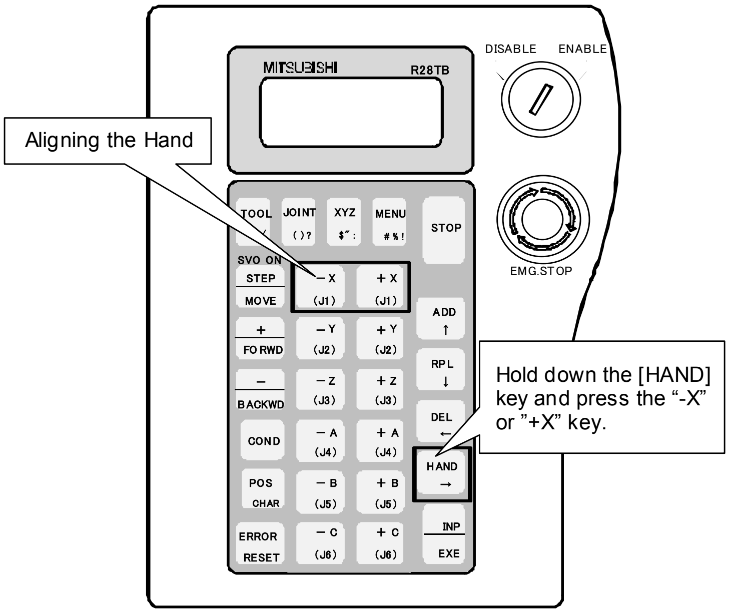

Without tool coordinate specification.
With tool coordinate specification.
Without tool coordinate specification.
With tool coordinate specification.
Control point
Control point

If the tool coordinates are specified by the TOOL instruction or parameters, the hand is aligned at the specified tool coordinates. If the tool coordinates are not specified, the hand is aligned at the center of the mechanical interface. The above illustration shows an example of a small vertical robot. [With Tool Coordinate Specification] indicates when the tool coordinates are specified at the tip of the hand. For more information about the tool coordinates, refer to Page 324, "5.6 Standard Tool Coordinates".

However, in the case of the RH-15UHC robot, the hand aligns radially from the center position of the robot, as shown in the right figure.



The hand alignment procedure is as follows:



Aligning the Hand

Hold down the [HAND] key and press the “-X” or ”+X” key.

1) Set the key switch to the [ENABLE] position.

2) Hold the deadman switch lightly.

3) Press the [STEP/MOVE] key and turn on the servo.

4) Hold down the [HAND] key and press the "-X" or "+X" key.

The robot stops when any of the above keys is released while aligning the hand.

3.5 Programming

MELFA-BASIC IV used with this controller allows advanced work to be described with ample operation functions. The programming methods using the T/B are explained in this section. The functions shown in Table 3-3 are used to input one line. (Refer to Page 118, "4.11 Detailed explanation of command words" in this manual for details on the MELFA-BASIC IV commands and description methods.)

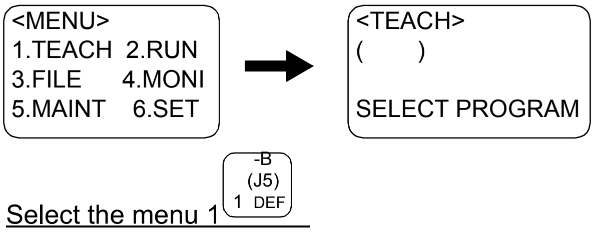
Table 3-3:Process for inputting one line

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

3.5.1 Creating a program

(1) Opening the program edit screen

Open the screen for editing the program to be created.

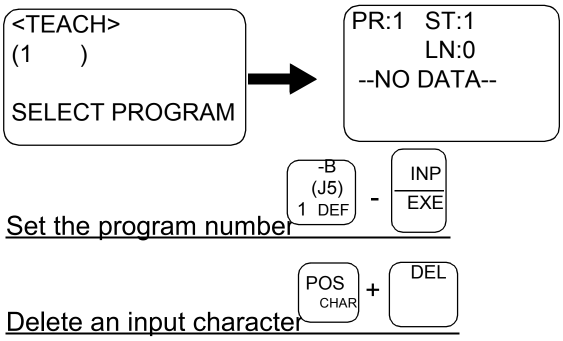


```
<MENU>
1.TEACH  2.RUN
3.FILE     4.MONI
5.MAINT    6.SET
```

```
<TEACH>
(       )

SELECT PROGRAM
```

Select the menu 1

1) Press the [1] key.
The PROGRAM SELECTION screen will appear.



```
<TEACH>
(1      )

SELECT PROGRAM
```

```
PR:1   ST:1
         LN:0
--NO DATA--
```

Set the program number

Delete an input character

2) Press the [1] key.
The program name 1 edit screen will appear, and the head line will appear.

Editing a program in constant execution mode (ALWAYS attribute)

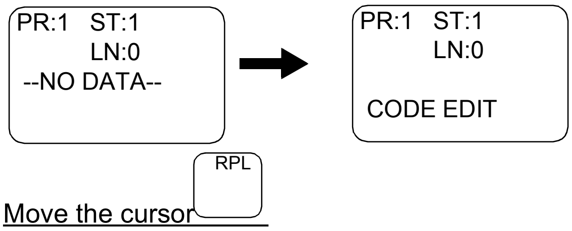
In order to edit a program set to be in constant execution mode (the ALWAYS attribute in the SLTn parameter is set), the constant execution attribute must be canceled first. Since programs in constant execution mode are executed continually, they cannot be edited. Change ALWAYS to START in the SLTn parameter, turn the controller's power on again, and stop the constant execution.

Selection from the program list

When the [INP] key is pressed in a blank field on the program selection screen, the program list appears, making it possible for you to edit a program by selecting it with the cursor and then pressing the [INP] key.

(2) Creating a program

1) Press the [RPL] key three times.
The cursor will move to the command editing line.

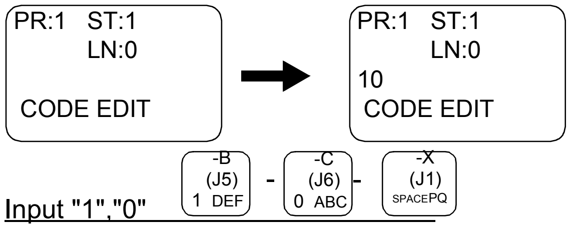


```
PR:1   ST:1
       LN:0
--NO DATA--
```

```
PR:1   ST:1
       LN:0

CODE EDIT
```

Move the cursor

2) Press the [1], [0] and [SPACE] keys. The line No. "10" will be input.

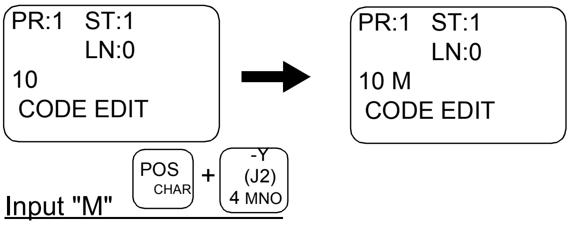


```
PR:1   ST:1
       LN:0

CODE EDIT
```

```
PR:1   ST:1
       LN:0
10
CODE EDIT
```

Input "1","0"

3) Press the [-Y/MNO] key once while holding down the [CHAR] key. "M" will appear.

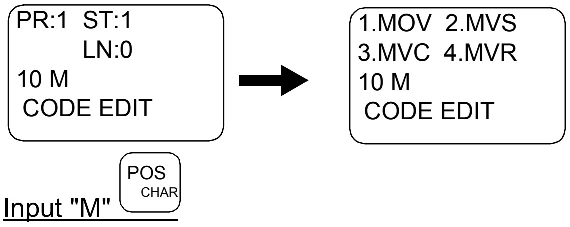


```
PR:1   ST:1
       LN:0
10
CODE EDIT
```

```
PR:1   ST:1
       LN:0
10 M
CODE EDIT
```

Input "M"

4) Release the [CHAR] key once, and then hold it down. The four commands assigned to "M" will appear.

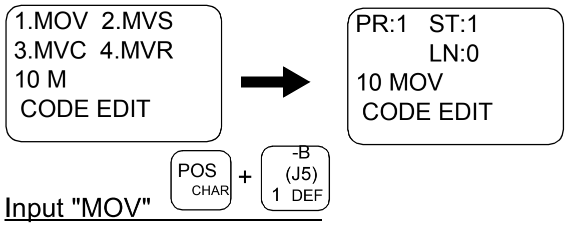


```
PR:1  ST:1
      LN:0
10 M
CODE EDIT
```

```
1.MOV  2.MVS
3.MVC  4.MVR
10 M
CODE EDIT
```

Input "M"

5) Press the [1] key while holding down the [CHAR] key.
The "MOV" command will be input.



```
1.MOV  2.MVS
3.MVC  4.MVR
10 M
CODE EDIT
```

```
PR:1   ST:1
       LN:0
10 MOV
CODE EDIT
```

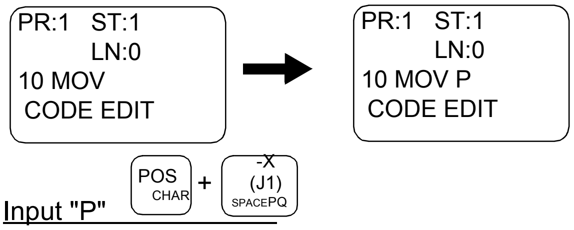
Input "MOV"

Inputting characters

The characters that can be input are indicated, three in a group, on the lower right of each key.
To input a character, hold down the [CHAR] key and press the key having the character to be input. Each time the corresponding character key is pressed while the [CHAR] key is pressed, the three characters will appear alternately.
Release the [CHAR] key when the target character appears, and set the character.

Inputting commands

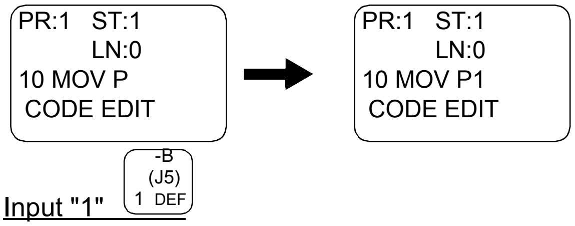
The commands can be input one character at a time (ex., for "M" "O" "V" for the MOV command), but if the head character of the command is input, the command can be selected as a number from the list of commands that appears.
After inputting the head character of the command, press the [CHAR] key. The list of commands will appear. While holding down the [CHAR] key, press the numeral key for the target command No., and select the command. If the target command is not found in the list, press the [CHAR] key again to update the list.



```
PR:1   ST:1
       LN:0
10 MOV
 CODE EDIT
```

```
PR:1   ST:1
       LN:0
10 MOV P
 CODE EDIT
```

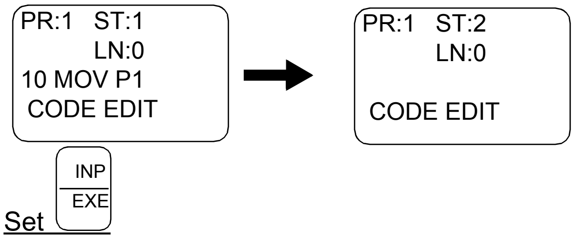
6) Press the [-X/PQR] key once while holding down the [CHAR] key, and then release the [CHAR] key.
"P" will be input.



```
PR:1   ST:1
       LN:0
10 MOV P
 CODE EDIT
```

```
PR:1   ST:1
       LN:0
10 MOV P1
 CODE EDIT
```

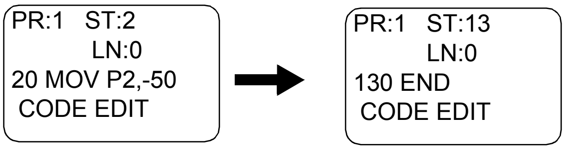
7) Press the [1] key. "1" will be input.



```
PR:1   ST:1
       LN:0
10 MOV P1
 CODE EDIT
```

```
PR:1   ST:2
       LN:0

 CODE EDIT
```

8) Press the [INP] key.
"10 MOV P1" will be set.



```
PR:1   ST:2
       LN:0
20 MOV P2,-50
 CODE EDIT
```

```
PR:1   ST:13
       LN:0
130 END
 CODE EDIT
```

9) Input the remaining program in the same manner.

This completes the inputting of the program.

(3) Completion of program creation and saving programs

Press the [MENU] key, or

Set the [ENABLE/DISABLE] switch of the T/B to the "DISABLE" position.

The program is saved when either of these operations is performed.

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 PC support software (optional).

Displaying the previous and next command line

To display the previous line, press the [BACKWD] key, and to display the next line, press the [FORWD] key.

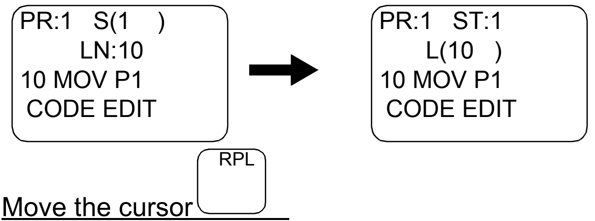
Displaying a specific line

Press the [ADD] key and move the cursor to LN:. Input the No. of the line to be displayed in the parentheses, and then press the [INP] key. The designated line will appear.

(4) Correcting a program

Before correcting a program, refer to Page 24, "3.5.1 Creating a program" in "(1)Opening the program edit screen", and open the program edit screen.

Call the line No.

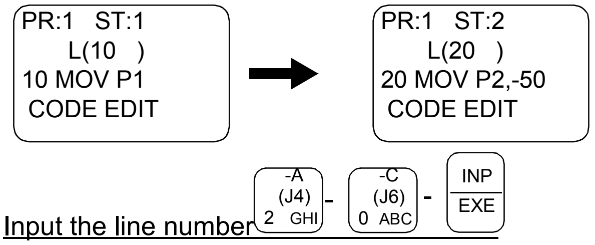


```
PR:1   S(1    )
        LN:10
10 MOV P1
 CODE EDIT
```

```
PR:1   ST:1
      L(10   )
10 MOV P1
 CODE EDIT
```

Move the cursor

1) Press the [RPL] key, and move the cursor to LN:( ).



```
PR:1   ST:1
      L(10   )
10 MOV P1
 CODE EDIT
```

```
PR:1   ST:2
      L(20   )
20 MOV P2,-50
 CODE EDIT
```

Input the line number

2) Press the [2], [0], [INP] key to display line 20.

Cursor movement

When the cursor is at a command line display, the command can be edited. When at a line No. display (LN:), the line No. is designated.

The cursor is moved with the [ADD], [RPL], [DEL] and [HAND] keys.

Calling out a line No.

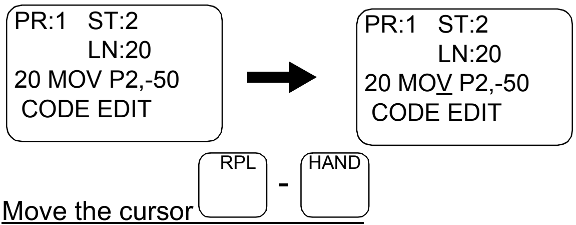
When designating and calling out a line No., move the cursor to the line No. display (LN:), input the line No., and then press the [INP] key.

The displayed line can be scrolled up or down by pressing the [FORWD] or [BACKWD] key.

Caution for Editing Array Variables

The number of elements in the array variable definition (DIM) can be changed in the software version K3 or later. If the number of elements is reduced, exercise caution as the data of the reduced elements will be deleted. Also, the number of dimensions cannot be changed (for example, changing from one dimension to two dimensions is not possible).

Change the command

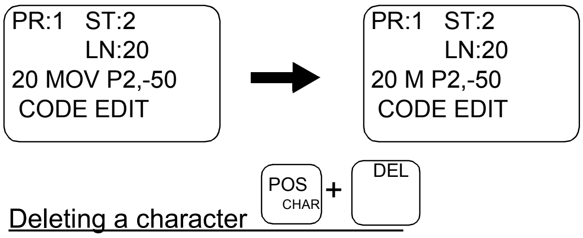


```
PR:1   ST:2
       LN:20
20 MOV P2,-50
 CODE EDIT
```

```
PR:1   ST:2
       LN:20
20 MOV P2,-50
 CODE EDIT
```

Move the cursor

3) Press the [RPL] key and move the cursor to the command line.
Press the [HAND] key six times, and move the cursor to the right of "V".

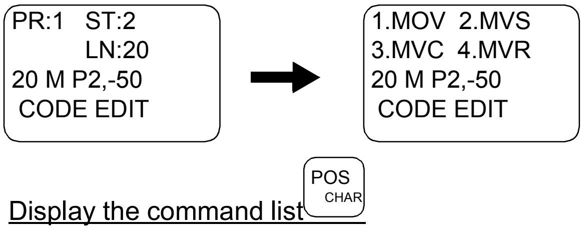


```
PR:1   ST:2
       LN:20
20 MOV P2,-50
 CODE EDIT
```

```
PR:1   ST:2
       LN:20
20 M P2,-50
 CODE EDIT
```

Deleting a character

4) Press the [DEL] key two times while holding down the [CHAR] key, and delete "OV". "M" will remain displayed.

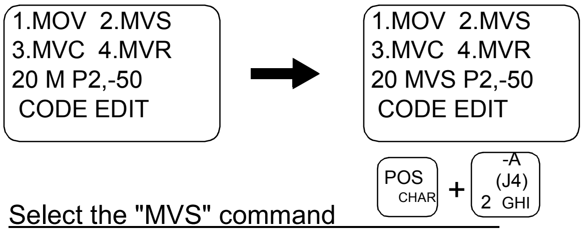


```
PR:1   ST:2
       LN:20
20 M P2,-50
 CODE EDIT
```

```
1.MOV  2.MVS
3.MVC  4.MVR
20 M P2,-50
 CODE EDIT
```

Display the command list

5) Hold down the [CHAR] key.
The four commands assigned to "M" will appear.

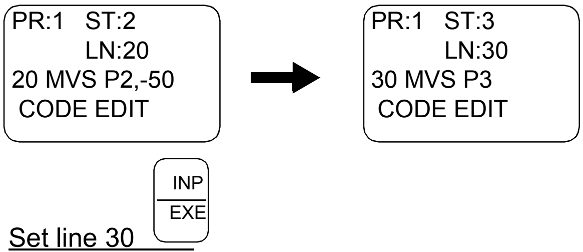


```
1.MOV  2.MVS
3.MVC  4.MVR
20 M P2,-50
 CODE EDIT
```

```
1.MOV  2.MVS
3.MVC  4.MVR
20 MVS P2,-50
 CODE EDIT
```

Select the "MVS" command

6) Press the [2] key while holding down the [CHAR] key, and select "MVS". MVS will appear on the screen.



```
PR:1   ST:2
       LN:20
20 MVS P2,-50
 CODE EDIT
```

```
PR:1   ST:3
       LN:30
30 MVS P3
 CODE EDIT
```

Set line 30

7) Press the [INP] key, and set line No. 70.
The next line will appear on the screen.

8) After finishing modifying an instruction, press the [MENU] key to save it.

Line No. 20 has been changed to linear movement with the above operation.

Correcting a character

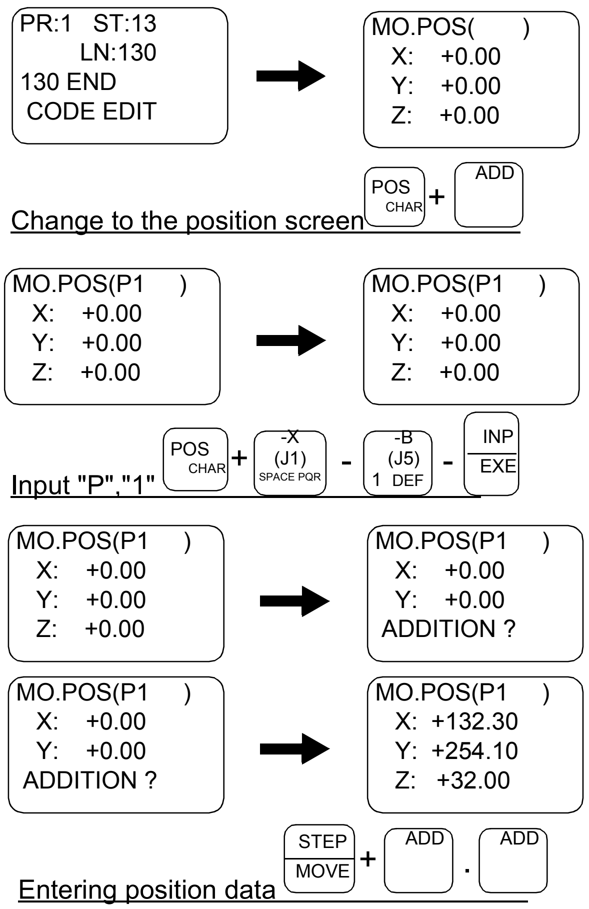
Move the cursor to the right of the incorrect character, and press the [DEL] key to delete in the left direction. Then, input the correct character.

After correcting a program

After modifying a program, make sure to perform the save operation (pressing the [MENU] key) and perform a step operation to check that the content of the program is properly changed.

(5) Registering the current position data

Entering position data(Teaching P1)



```
PR:1   ST:13
       LN:130
130 END
 CODE EDIT
```

```
MO.POS(      )
  X:    +0.00
  Y:    +0.00
  Z:    +0.00
```

Change to the position screen

```
MO.POS(P1    )
  X:    +0.00
  Y:    +0.00
  Z:    +0.00
```

```
MO.POS(P1    )
  X:    +0.00
  Y:    +0.00
  Z:    +0.00
```

Input "P","1"

```
MO.POS(P1    )
  X:    +0.00
  Y:    +0.00
  Z:    +0.00
```

```
MO.POS(P1    )
  X:    +0.00
  Y:    +0.00
 ADDITION ?
```

```
MO.POS(P1    )
  X:    +0.00
  Y:    +0.00
 ADDITION ?
```

```
MO.POS(P1    )
  X:  +132.30
  Y:  +254.10
  Z:   +32.00
```

Entering position data

1) On the command editing screen, press the [ADD] key or [RPL] while holding down the [POS] key.
The position editing screen will appear.

2) Input "P1" in the parentheses at MO.POS, and then press the [INP] key.
The position variable name P1 will be called, and the currently registered coordinate value will appear.

Refer to Page 25, "Inputting characters" for details on inputting characters.

3) ÅmPress the [ADD] key or [RPL] key while holding down the [STEP] key, and release only the [ADD] key or [RPL] key.
The buzzer will sound a "beep", and a confirmation message will appear.
While holding down the [STEP] key, press the [ADD] key or [RPL] key again.
The buzzer will sound a "beep", and the message "ADDING" will appear. Then, the current position will be registered.

4) After finishing modifying an instruction, press the [MENU] key to save it.

The robot's operation position can be taught with the above operations.

Changing between the command editing screen and position editing screen.

The commands are edited on the command editing screen, and the positions are edited on the position editing screen. To change from the command editing screen to the position editing screen, press [POS] + ([ADD] or [RPL] key). If the cursor is not displayed, press the [COND] key and then the [POS] key to display the cursor.

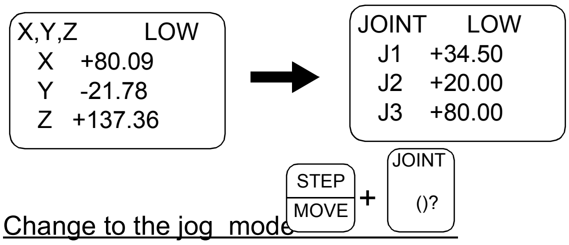
(6) Confirming the position data (Position jump)

Move the robot to the registered position data place.
The robot can be moved with the "MO position movement" or "MS position movement" 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
MO position movement	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.
MS position movement	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.

Move the MO position

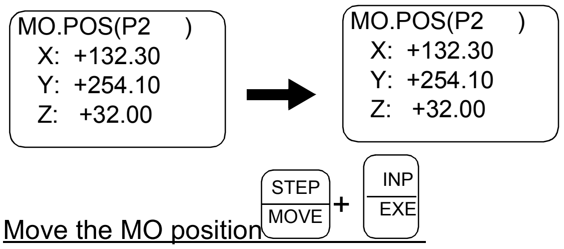


```
X,Y,Z          LOW
  X    +80.09
  Y    -21.78
  Z   +137.36
```

```
JOINT      LOW
  J1    +34.50
  J2    +20.00
  J3    +80.00
```

Change to the jog mode

1) MO position movement is only possible in the CYLNDER jog mode. Thus, change the mode if the jog mode is other than JOINT jog.



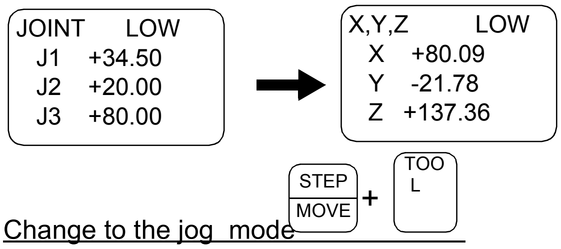
```
MO.POS(P2     )
  X:  +132.30
  Y:  +254.10
  Z:   +32.00
```

```
MO.POS(P2     )
  X:  +132.30
  Y:  +254.10
  Z:   +32.00
```

Move the MO position

2) When the [EXE] key is pressed while holding down the [MOVE] key, the robot will move with joint interpolation to the currently displayed position data place only while the [EXE] key is held down.

Move the MS position

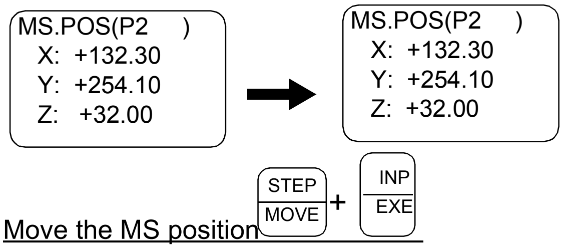


```
JOINT      LOW
  J1    +34.50
  J2    +20.00
  J3    +80.00
```

```
X,Y,Z          LOW
  X    +80.09
  Y    -21.78
  Z   +137.36
```

Change to the jog mode

3) MS position movement is possible in jog modes other than the XYZ jog, 3-axis XYZ jog, CYLNDER jog and TOOL jog. Thus, change the mode if these modes are entered.



```
MS.POS(P2     )
  X:  +132.30
  Y:  +254.10
  Z:   +32.00
```

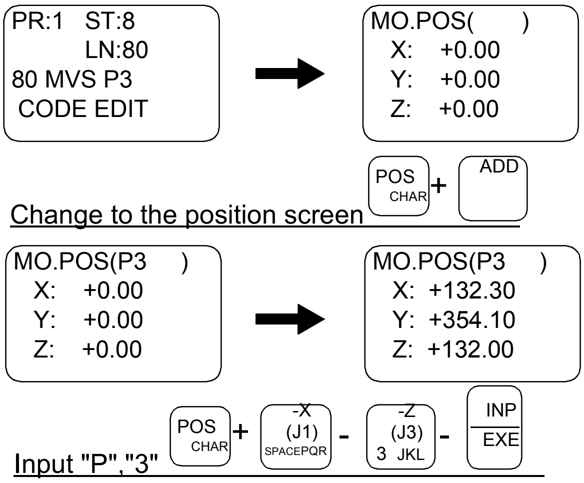
```
MS.POS(P2     )
  X:  +132.30
  Y:  +254.10
  Z:  +32.00
```

Move the MS position

4) When the [EXE] key is pressed while holding down the [MOVE] key, the robot will move with linear interpolation to the currently displayed position data place only while the [EXE] key is held down.
If linear movement from the current position to the designated position is not possible, the robot will not move.

(7) Correcting the current position data

Change the movement position



```
PR:1   ST:8
       LN:80
80 MVS P3
 CODE EDIT
```

```
MO.POS(      )
  X:    +0.00
  Y:    +0.00
  Z:    +0.00
```

Change to the position screen

```
MO.POS(P3     )
  X:    +0.00
  Y:    +0.00
  Z:    +0.00
```

```
MO.POS(P3     )
  X:  +132.30
  Y:  +354.10
  Z:  +132.00
```

Input "P","3"

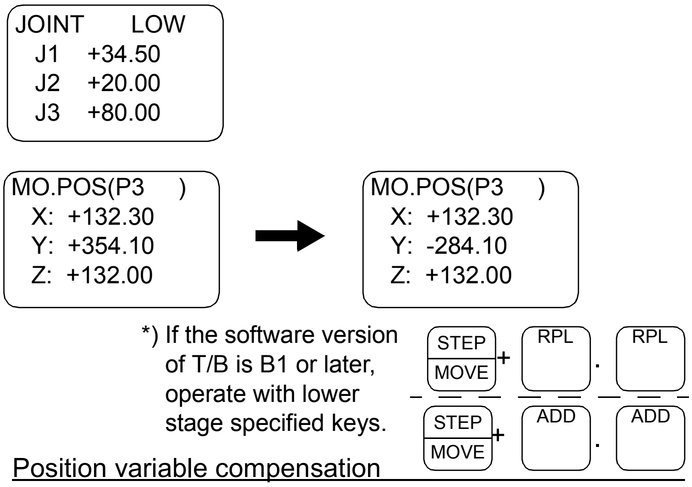
1) On the command editing screen, press the [ADD] key or [RPL] key while holding down the [POS] key.
The position editing screen will appear.

2) Input "P3" in the parentheses at MO.POS, and then press the [INP] key.
The position variable name P3 will be called, and the currently registered coordinate value will appear.

3) After finishing modifying an instruction, press the [MENU] key to save it.

Calling out a position variablec

Input the name of the variable to be called out in the parentheses at MO. POS on the position editing screen. Then, press the [INP] key.
The position variable can be scrolled up or down by pressing the [FORWD] or [BACKWD] key.



```
JOINT       LOW
  J1    +34.50
  J2    +20.00
  J3    +80.00
```

```
MO.POS(P3     )
  X:  +132.30
  Y:  +354.10
  Z:  +132.00
```

```
MO.POS(P3     )
  X:  +132.30
  Y:  -284.10
  Z:  +132.00
```

*) If the software version of T/B is B1 or later, operate with lower stage specified keys.

Position variable compensation

4) Move the robot to the new wait position with jog operation.

5) Press the [RPL] key while holding down the [STEP] key, and release only the [RPL] key.
The buzzer will sound a "beep", and a confirmation message will appear.
While holding down the [STEP] key, press the [RPL] key again.
The buzzer will sound a "beep", and the message "Replacing" will appear. Then, the current position will be corrected.

*) If the software version of T/B is B1 or later, change the [RPL] key to [ADD], and operate.(like the left figure)

This completes correction of the wait position.

After correcting a program

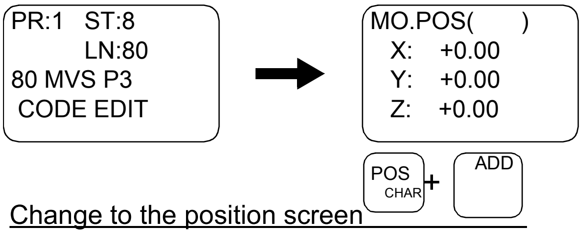
After modifying a program, make sure to perform the save operation (pressing the [MENU] key) and perform a step operation to check that the content of the program is properly changed.

The check of the software version of T/B

After turning on the controller power, it can check by the T/B screen before title screen is displayed.

(8) Correcting the MDI (Manual Data Input)

MDI compensation method

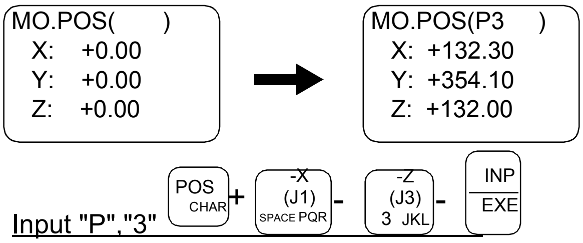


```
PR:1   ST:8
       LN:80
80 MVS P3
 CODE EDIT
```

```
MO.POS(      )
  X:   +0.00
  Y:   +0.00
  Z:   +0.00
```

Change to the position screen

1) On the command editing screen, press the [ADD] key or [RPL] key while holding down the [POS] key. The position editing screen will appear.

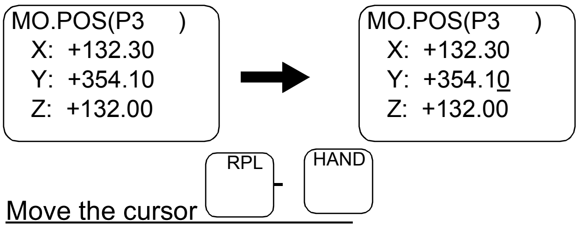


```
MO.POS(      )
  X:   +0.00
  Y:   +0.00
  Z:   +0.00
```

```
MO.POS(P3    )
  X:  +132.30
  Y:  +354.10
  Z:  +132.00
```

Input "P","3"

2) Input "P3" in the parentheses at MO.POS, and then press the [INP] key. The position variable name P3 will be called, and the currently registered coordinate value will appear.

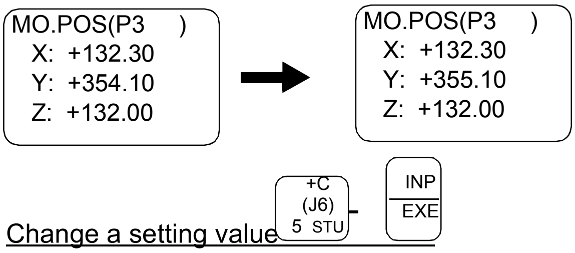


```
MO.POS(P3    )
  X:  +132.30
  Y:  +354.10
  Z:  +132.00
```

```
MO.POS(P3    )
  X:  +132.30
  Y:  +354.10
  Z:  +132.00
```

Move the cursor

3) Using the [RPL]key, move the cursor to the Y: line, and then using the [HAND] key, move the cursor to above 4.



```
MO.POS(P3    )
  X:  +132.30
  Y:  +354.10
  Z:  +132.00
```

```
MO.POS(P3    )
  X:  +132.30
  Y:  +355.10
  Z:  +132.00
```

Change a setting value

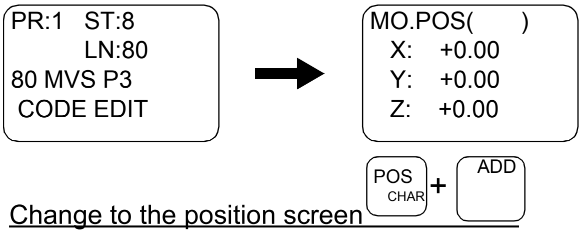
4) Press the [5] key, and then press the [INP] key. 5 will be written over 4, and the Y coordinate value will be changed.

5) After finishing modifying an instruction, press the [MENU] key to save it.

(9) Deleting position data

Only position data that is not currently being used by the program can be deleted. If position data being used in the program is deleted, an error will occur.

Position data deletion method

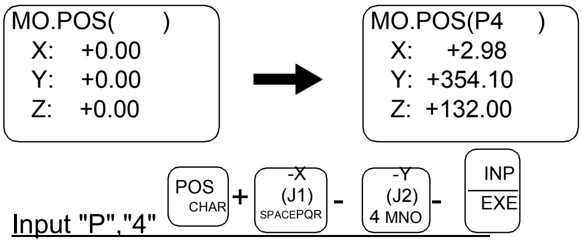


```
PR:1   ST:8
       LN:80
80 MVS P3
 CODE EDIT
```

```
MO.POS(      )
  X:    +0.00
  Y:    +0.00
  Z:    +0.00
```

Change to the position screen

1) On the command editing screen, press the [ADD] key or [RPL] key while holding down the [POS] key.
The position editing screen will appear.

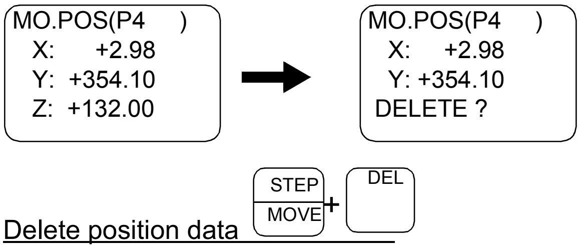


```
MO.POS(      )
  X:    +0.00
  Y:    +0.00
  Z:    +0.00
```

```
MO.POS(P4     )
  X:     +2.98
  Y:  +354.10
  Z:  +132.00
```

Input "P","4"

2) Input "P4" in the parentheses at MO.POS, and then press the [INP] key.
The position variable name P4 will be called, and the currently registered coordinate value will appear.

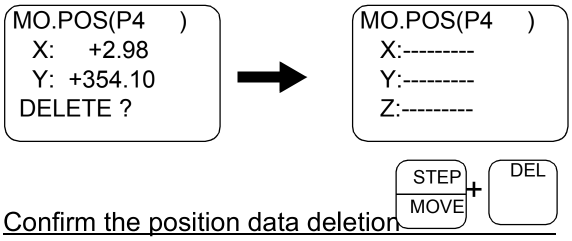


```
MO.POS(P4     )
  X:      +2.98
  Y:  +354.10
  Z:  +132.00
```

```
MO.POS(P4     )
  X:     +2.98
  Y: +354.10
 DELETE ?
```

Delete position data

3) Press the [DEL] key while holding down the [STEP] key, and release only the [DEL] key. The buzzer will sound a "beep", and a confirmation message will appear.



```
MO.POS(P4     )
  X:    +2.98
  Y:  +354.10
 DELETE ?
```

```
MO.POS(P4     )
  X:---------
  Y:---------
  Z:---------
```

Confirm the position data deletion

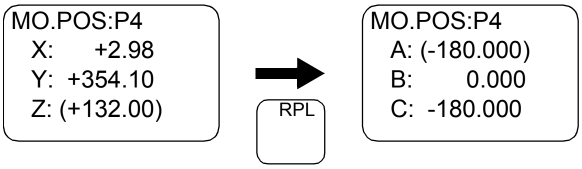
4) While holding down the [STEP] key, press the [DEL] key again.
The buzzer will sound a "beep", and the message indicating the deletion will appear at the bottom line of the screen.
The position data will be deleted.

5) After finishing modifying an instruction, press the [MENU] key to save it.

(10) Display on the position edit screen

The XYZ coordinate values of the X, Y, and Z axes, the posture data of the A, B, and C axes, the posture structure flags, and the multiple rotation information of each axis, are saved as the robot's position data. Perform the following operations to display each screen.

Position edit screen

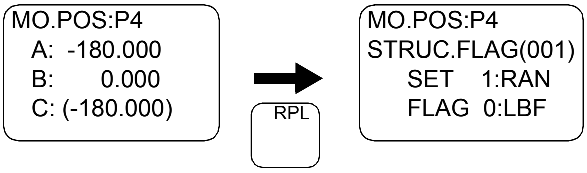


```
MO.POS:P4
  X:      +2.98
  Y:  +354.10
  Z: (+132.00)
```

```
MO.POS:P4
  A: (-180.000)
  B:        0.000
  C:  -180.000
```

Change to the position screen (A,B,C axis)

1) When the position edit screen is displayed, the coordinate values of the X, Y, and Z axes are displayed first. Press the [RPL] key to display the data of the A, B, and C axes.

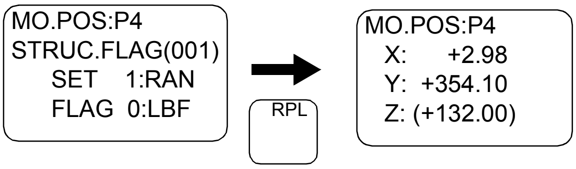


```
MO.POS:P4
  A:  -180.000
  B:       0.000
  C: (-180.000)
```

```
MO.POS:P4
STRUC.FLAG(001)
    SET    1:RAN
    FLAG  0:LBF
```

Change to the position screen (Flg 1)

2) Press the [RPL] key again to display the information of structure flag 1.



```
MO.POS:P4
STRUC.FLAG(001)
    SET    1:RAN
    FLAG  0:LBF
```

```
MO.POS:P4
  X:      +2.98
  Y:  +354.10
  Z: (+132.00)
```

Return to the position screen (XYZ)

3) Press the [RPL] key once again to return to the X, Y, and Z axes display.

(11) Saving the program

When completed creating or revising a program, save the edited details with one of the following operations.

* Press the [MENU] key and display the Menu screen.

* Set the T/B [ENABLE/DISABLE] switch to "DISABLE"

Precaution when saving the edited program/data

Please be aware that any updates to the program or data, including teaching data, will be lost if the power is shut down while in the program edit screen. Please be aware that any updates to the program or data, including teaching data, will be lost if the power is shut down while in the program edit screen. Please be aware that any updates to the program or data, including teaching data, will be lost if the power is shut down while in the program edit screen.

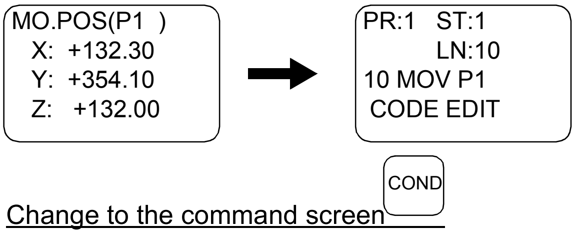
3.6 Debugging

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

(1) Step feed

The program is run one line at a time in the feed direction. The program is run in line order from the head or the designated line.
Confirm that the program runs correctly with this process.

Using the T/B execute the program line by line (step operation), and confirm the operation.
Perform the following operations while pressing lightly on the deadman switch of the T/B after the servo has been turned on.

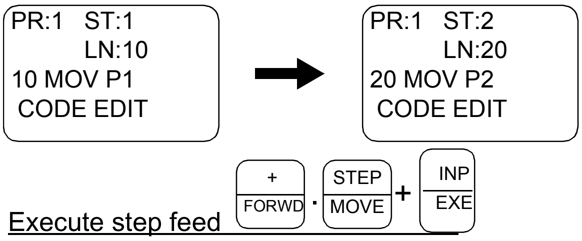


```
MO.POS(P1  )
  X:  +132.30
  Y:  +354.10
  Z:   +132.00
```

```
PR:1   ST:1
       LN:10
10 MOV P1
 CODE EDIT
```

Change to the command screen

1) Press the [COND] key, and display the command editing screen.



```
PR:1   ST:1
       LN:10
10 MOV P1
 CODE EDIT
```

```
PR:1   ST:2
       LN:20
20 MOV P2
 CODE EDIT
```

Execute step feed

2) While holding down the [FORWD] key or [STEP] key, hold down the [EXE] key. 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 [EXE] is released during this step, the robot will stop.

⚠CAUTION Take special care to the robot movements during operation. If any abnormality occurs, such as interference with the peripheral devices, release the [EXE] key or deadman switch, or press the deadman switch with force and stop the robot.

Step operation

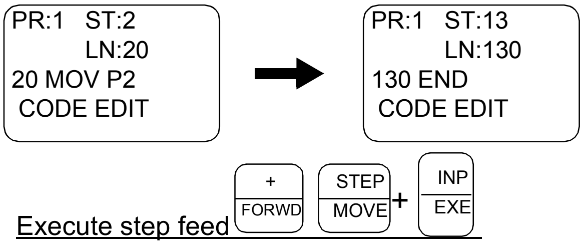
"Step operation" executes the program line by line. The operation speed is slow, and the robot stops after each line, so the program and operation position can be confirmed.
During execution, the lamp on the controller's [START] switch will light.

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 error, turn the servo ON, and start step operation.

*Release or forcibly press the "deadman" switch.
The servo will turn OFF, and the moving robot will immediately stop. Error 2000 will occur.
To resume operation, reset the error, lightly press the [Deadman] switch, press the [SVO ON] key to turn ON the servo, and then start step operation.

*Release the [EXE] key.
The step execution will be stopped. The servo will not turn OFF.
To resume operation, press the [EXE] key.



```
PR:1   ST:2
       LN:20
20 MOV P2
 CODE EDIT
```

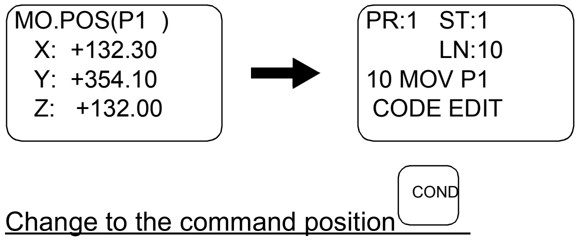
```
PR:1   ST:13
       LN:130
130 END
 CODE EDIT
```

Execute step feed

3) Carry out step operation of the entire program, and confirm the operation in the same manner.
If the robot operation or position is incorrect, refer to the following operations and make corrections.

(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.

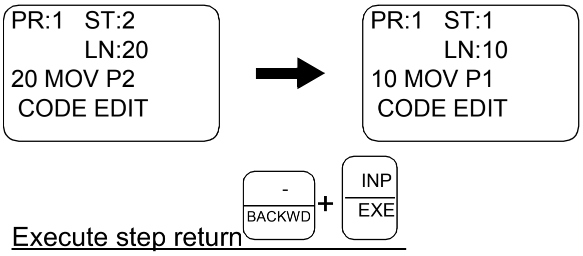


```
MO.POS(P1  )
  X:  +132.30
  Y:  +354.10
  Z:   +132.00
```

```
PR:1   ST:1
       LN:10
10 MOV P1
 CODE EDIT
```

Change to the command position

1) Press the [COND] key, and display the command editing screen. After this, carry out step feed referring to the previous page.



```
PR:1   ST:2
       LN:20
20 MOV P2
 CODE EDIT
```

```
PR:1   ST:1
       LN:10
10 MOV P1
 CODE EDIT
```

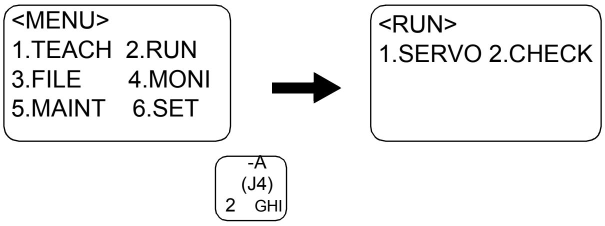
Execute step return

2) While holding down the [BACKWD] key, hold down the [EXE] key. The robot will start step return.

When the execution of one line is completed, the robot will stop, and the previous line will appear on the screen. If [EXE] is released during this step, the robot will stop.

(3) Step feed in another slot

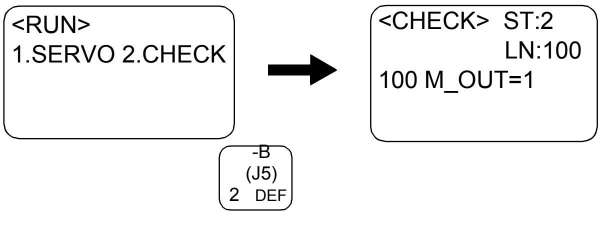
When checking a multitask program, it is possible to perform step feed in the confirmation screen of the operation menu, not in the edit screen.



```
<MENU>
1.TEACH  2.RUN
3.FILE      4.MONI
5.MAINT    6.SET
```

```
<RUN>
1.SERVO 2.CHECK
```

1) Press the "2" key to display the operation menu screen.



```
<RUN>
1.SERVO 2.CHECK
```

```
<CHECK>  ST:2
              LN:100
100 M_OUT=1
```

2) Press the "2" key to display the confirmation screen.

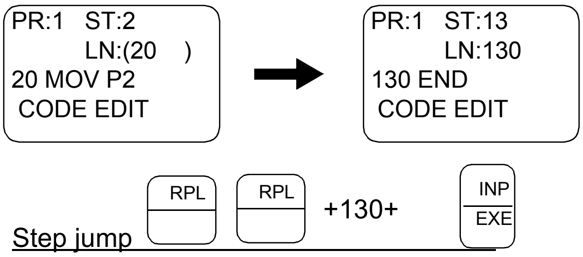
3) Change the slot number to display and allow step feed for another task slot.

4) The operations for step feed are the same as in the edit screen. Hold down "+" (or "-") and press the [INP/EXE] key.

Specify 0 for the slot number in order to perform step feed for all the task slots at the same time.

(4) Step jump

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



```
PR:1   ST:2
       LN:(20    )
20 MOV P2
 CODE EDIT
```

```
PR:1   ST:13
       LN:130
130 END
 CODE EDIT
```

1) Use the arrow keys to move to the ST column or LN column and enter the step number or line number you want to display, and then press the [INP] key.

2) It is possible to perform step feed from the step after the change. However, an undefined error or similar will occur if lines for initialization of variables, etc. are skipped.

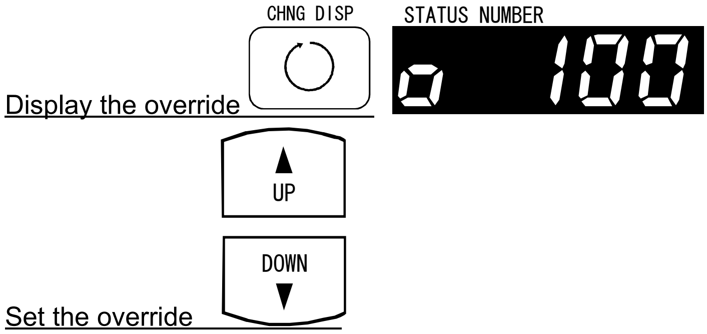
3.7 Automatic operation

(1) Setting the operation speed

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

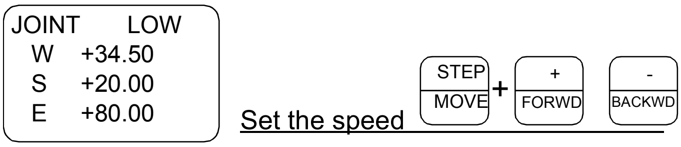
The actual speed during automatic operation will be the operation speed = (controller (T/B) setting value) x (program setting value).

*Operating with the controller



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

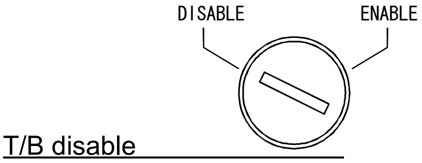
*Operating with the T/B



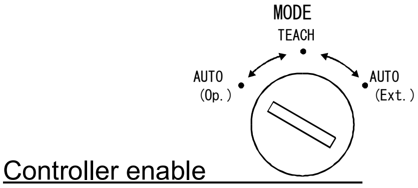
1) Each time the [MOVE] + [+] keys are pressed, the override will increase in the order of (LOW - HIGH - 3 - 5 - 10 - 30 - 50 - 70 -100%). The speed will decrease in reverse each time the [MOVE] + [%] keys are pressed. The currently set speed will appear on the upper right of the screen.

(2) Selecting the program No.

Prepare the control

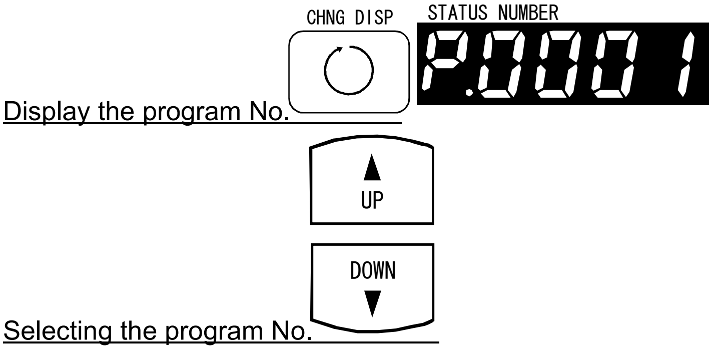


1) Set the T/B [ENABLE/DISABLE] switch to "DISABLE".



2) Set the controller [MODE] switch to "AUTO (Op.)".

Selecting the program No.

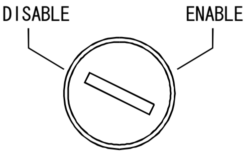


3) Press the [CHNG DISP] switch and display "PROGRAM NO." on the STATUS NUMBER display.
4) When the [UP] switch is pressed, the registered program Nos. will scroll up, and then the [DOWN] switch is pressed, the program Nos. will scroll down. Display the program No. to be used for automatic operation. *They are not displayed if a program name consisting of five or more characters is specified. If these are selected from an external device, "P - - - - " is displayed.

(3) Starting automatic operation

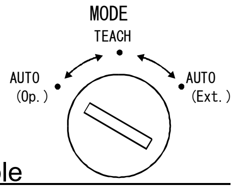
⚠CAUTION Before starting automatic operation, always confirm the following items. Starting automatic operation without confirming these items could lead to property damage or physical injury.
*Make sure that there are no operators near the robot.
*Make sure that the safety fence is locked, and operators cannot enter unintentionally.
*Make sure that there are no unnecessary items, such as tools, inside the robot operation range.
*Make sure that the workpiece is correctly placed at the designated position.
*Confirm that the program operates correctly with step operation.

Prepare the control



T/B disable

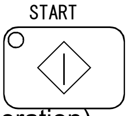
1) Set the T/B [ENABLE/DISABLE] switch to "DISABLE".



Controller enable

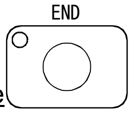
2) Set the controller [MODE] switch to "AUTO (Op.)".

Start automatic operation



Start
(Continuous operation)

➡ End with one cycle



3) Automatic operation will start when the controller [START] switch is pressed. (Continuous operation)
If the [END] switch is pressed during the continuous operation, the program will stop after one cycle. The LED blinks during the cycle stop.
When the [END] key is pressed again during a cycle stop in software version H8 or later, the operation returns to continuous operation.

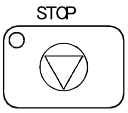
⚠CAUTION Before starting automatic operation, always confirm that the target program No. is selected.

⚠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.

(4) Stopping

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

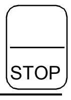
*Operating with the controller



Stop

1) Press the [STOP] switch.

*Operating with the T/B



Stop

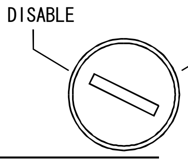
1) Press the [STOP] key.

Operation rights not required

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

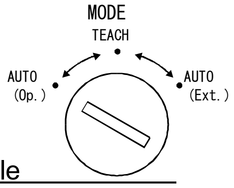
(5) Resuming automatic operation from stopped state

⚠CAUTION Before starting automatic operation, always confirm the following items. Starting automatic operation without confirming these items could lead to property damage or physical injury.
*Make sure that there are no operators near the robot.
*Make sure that the safety fence is locked, and operators cannot enter unintentionally.
*Make sure that there are no unnecessary items, such as tools, inside the robot operation range.
*Make sure that the workpiece is correctly placed at the designated position.
*Confirm that the program operates correctly with step operation.



T/B disable

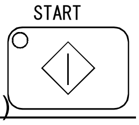
1) Set the T/B [ENABLE/DISABLE] switch to "DISABLE".



Controller enable

2) Set the controller [MODE] switch to "AUTO (Op.)".

Start automatic operation



Start (Continuous operation)

3) Automatic operation will start when the controller [START] switch is pressed. (Continuous operation)
The continuous operation/one cycle operation state will be the same as before the operation was stopped.

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

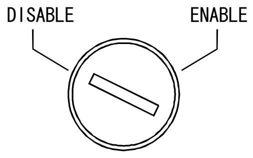
⚠CAUTION Take special care to the robot movements during automatic operation. If any abnormality occurs, press the [EMG. STOP] switch and immediately stop the robot.

⚠CAUTION Do not turn the controller's power off during the automatic operation. The memory in the controller may be malfunctioned and programs may be destroyed. Use the emergency stop to stop the robot immediately.

(6) Resetting the program

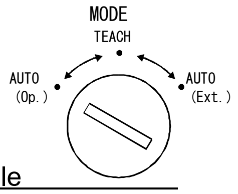
The program's stopped state is canceled, and the execution line is returned to the head.

*Operating with the controller



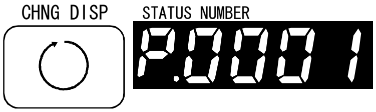
T/B disable

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



Controller enable

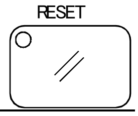
2) Set the controller [MODE] switch to "AUTO (Op.)".



Display the program No.

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

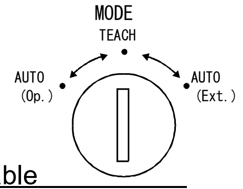
Execute of program reset



Reset

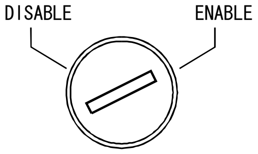
4) Press the controller [RESET] switch.
The STOP lamp will turn OFF, and the program's stopped state will be canceled.

*Operating with the T/B



Controller disable

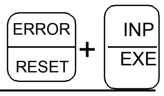
1) Set the [Mode selection switch] on the front of the controller to "TEACH".



T/B enable

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

Execute of program reset



Program reset

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

Valid only while program is stopped

The program cannot be reset while the program is running. Always carry out this step while the program is stopped.
When resetting the program from the controller operation panel, display the "program No." on the STATUS NUMBER display, and then reset.

STOP lamp turns OFF

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

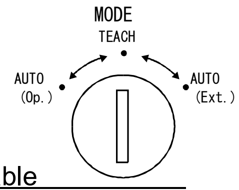
3.8 Turning the servo ON/OFF

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

*Turning servo ON with T/B [SVO ON] key

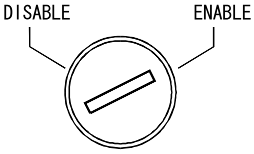
Prepare the T/B

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



Controller disable

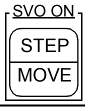
2) Set the T/B [ENABLE/DISABLE] switch to "ENABLE".



T/B enable

Execute servo ON

3) The servo will turn ON when the [SVO ON] key ([STEP/MOVE] key) is pressed.

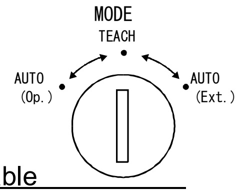


Servo ON operation

*Turning servo ON/OFF with T/B Servo screen

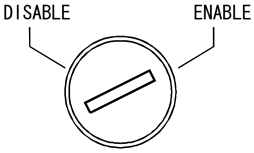
Prepare the T/B

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



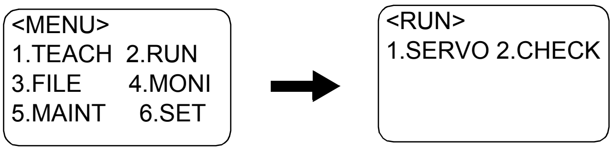
Controller disable

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



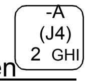
T/B enable

3) Press the [2] key, and select the operation screen.



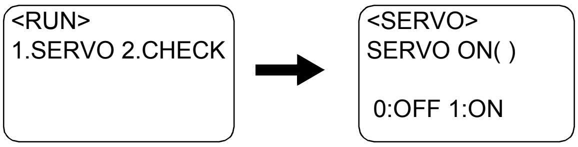
<MENU>
1.TEACH 2.RUN
3.FILE 4.MONI
5.MAINT 6.SET

<RUN>
1.SERVO 2.CHECK



Select the RUN screen

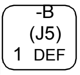
4) Press the [1] key, and select the servo screen.



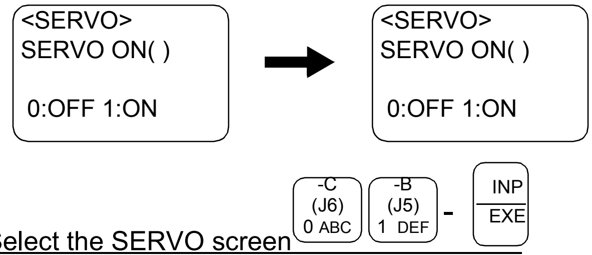
<RUN>
1.SERVO 2.CHECK

<SERVO>
SERVO ON( )

0:OFF 1:ON



Select the SERVO screen



```
<SERVO>
SERVO ON( )

 0:OFF 1:ON
```

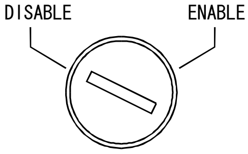
```
<SERVO>
SERVO ON( )

 0:OFF 1:ON
```

Select the SERVO screen

5) To turn ON/OFF the servo, press the [0] key when the servo is OFF or the [1] key when the servo is ON while lightly holding the deadman switch, and then press the [INP] key. (It is also possible to turn the servo on by pressing the [STEP/MOVE] key.)

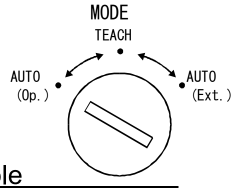
*Operating with the controller

Prepare the controller



T/B disable

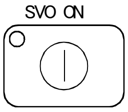
1) Set the T/B [ENABLE/DISABLE] switch to "DISABLE".



Controller enable

2) Set the controller [MODE] switch to "AUTO (Op.)".

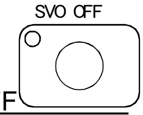
Execute servo ON



Servo ON

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

Execute servo OFF



Servo OFF

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

Brakes will activate

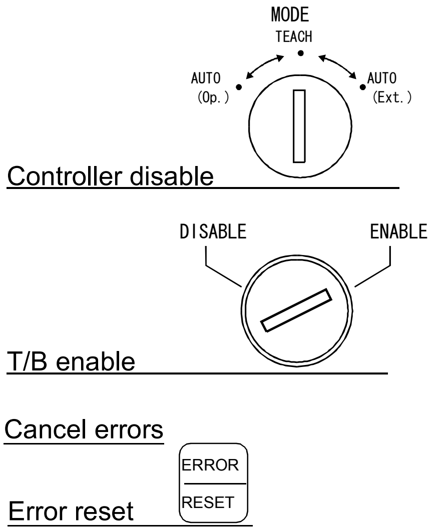
The brakes will automatically activate when the servo is turned OFF. Depending on the type of robot, some axes may not have brakes.

3.9 Error reset operation

*Error reset operation from the operation panel

Set the key switch to the AUTO (OP) position, and then press the reset key on the operation panel.

*Error reset operation from the T/B



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

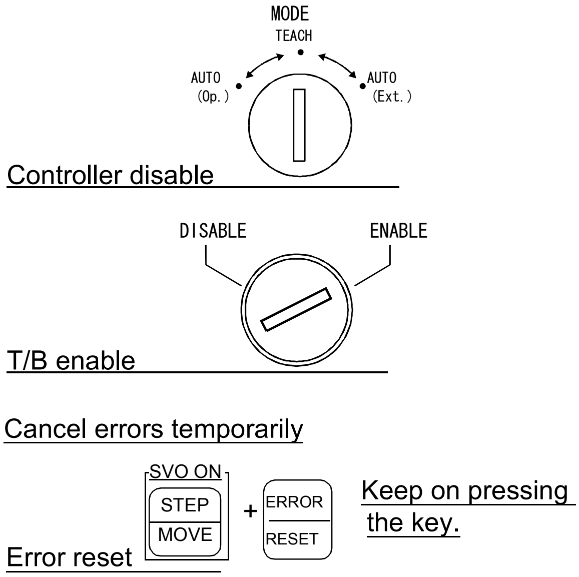
2) Set the T/B [ENABLE/DISABLE] switch to "ENABLE".

3) Press the [ERROR RESET] key.

3.10 Operation to Temporarily Reset an Error that Cannot Be Canceled

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

*Operation to cancel errors temporarily from the T/B



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

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

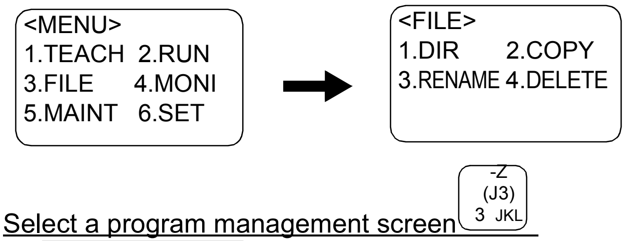
3) Hold the deadman switch lightly, hold down the [SVO] key and keep on pressing the [ERROR RESET] key.

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

3.11 Operating the program control screen

(1) Program list display

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

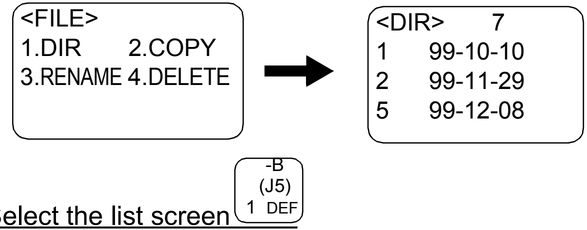


```
<MENU>
1.TEACH  2.RUN
3.FILE     4.MONI
5.MAINT   6.SET
```

```
<FILE>
1.DIR       2.COPY
3.RENAME 4.DELETE
```

Select a program management screen

1) Press the [3] key, and select the program control screen.

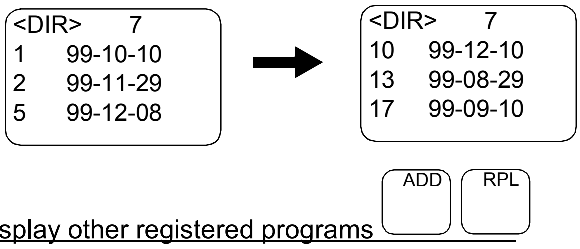


```
<FILE>
1.DIR       2.COPY
3.RENAME 4.DELETE
```

```
<DIR>      7
1      99-10-10
2      99-11-29
5      99-12-08
```

Select the list screen

2) When the [1] key is pressed, the program list (date of creation) will appear. The No. of registered programs will appear at the upper right of the screen.

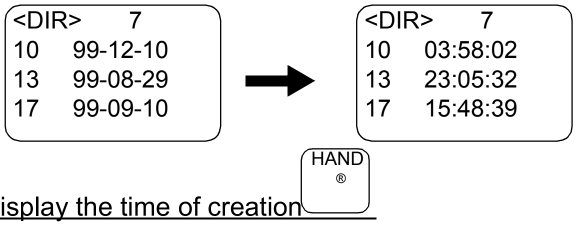


```
<DIR>      7
1      99-10-10
2      99-11-29
5      99-12-08
```

```
<DIR>      7
10     99-12-10
13     99-08-29
17     99-09-10
```

Display other registered programs

3) The other registered programs can be displayed by pressing the [ADD] and [RPL] keys.

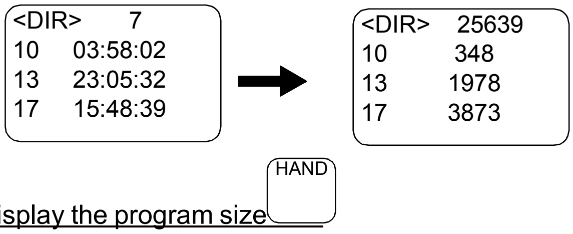


```
<DIR>      7
10     99-12-10
13     99-08-29
17     99-09-10
```

```
<DIR>      7
10     03:58:02
13     23:05:32
17     15:48:39
```

Display the time of creation

4) If the [HAND] key is pressed when the date of creation is displayed, the time that the program was created will appear. When the [DEL] key is pressed, the display will return to the date of creation.

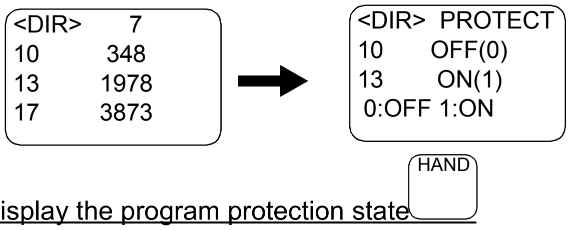


```
<DIR>      7
10     03:58:02
13     23:05:32
17     15:48:39
```

```
<DIR>    25639
10         348
13         1978
17         3873
```

Display the program size

5) If the [HAND] is pressed when the time of creation is displayed, the program size (unit: byte) will appear. The currently usable memory size will appear at the upper right of the screen. When the [DEL] key is pressed, the display will return to the date of creation.

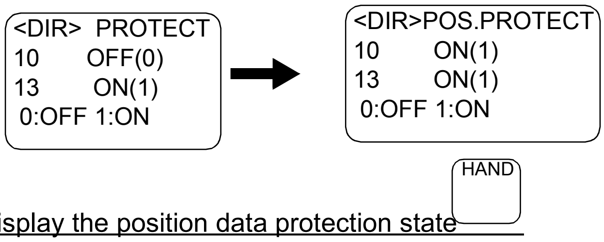


```
<DIR>      7
10         348
13         1978
17         3873
```

```
<DIR>  PROTECT
10       OFF(0)
13        ON(1)
 0:OFF 1:ON
```

Display the program protection state

6) If the [HAND] key is pressed when the program size is displayed, the program protection state will appear. Refer to the section Page 46, "* Program protection function" for details.



```
<DIR>  PROTECT
10       OFF(0)
13        ON(1)
 0:OFF 1:ON
```

```
<DIR>POS.PROTECT
10        ON(1)
13        ON(1)
 0:OFF 1:ON
```

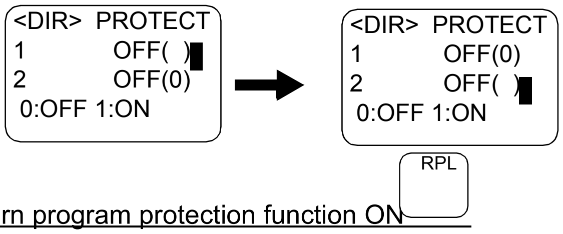
Display the position data protection state

7) If the [HAND] key is pressed when the program protection state is displayed, the variable protection state will appear. Refer to the section Page 46, "* Position variable protection function" for details.

(2) Program protection function

*Program protection function

This function protects the program from being deleted or changed inadvertently.

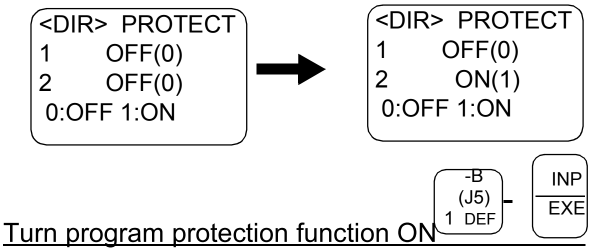


```
<DIR>  PROTECT
1          OFF( )
2          OFF(0)
 0:OFF 1:ON
```

```
<DIR>  PROTECT
1          OFF(0)
2          OFF( )
 0:OFF 1:ON
```

Turn program protection function ON

1) Display the "program list display" explained on the previous page, and then display the program protection state. Move the cursor to the program targeted for protection with the [RPL] and [ADD] keys.



```
<DIR>  PROTECT
1       OFF(0)
2       OFF(0)
 0:OFF 1:ON
```

```
<DIR>  PROTECT
1       OFF(0)
2         ON(1)
 0:OFF 1:ON
```

Turn program protection function ON

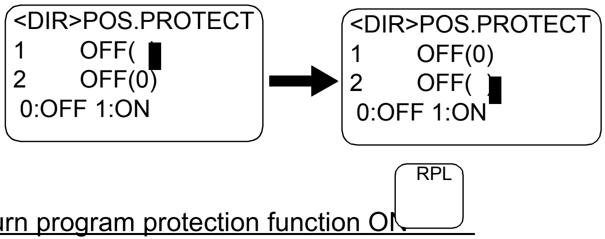
2) The program protection function for that program will turn ON or OFF when the [INP] key is pressed after the following key:
Program protection function ON ... [1] key
Program protection function OFF ... [0] key

Program protection

This function protects the program from inadvertent program deletion, renaming and command changing.
*The protection function is not copied with the copy operation.
*The protected information is ignored during the initialization process.

*Position variable protection function

This function protects the program from inadvertent variable deletion and changing.

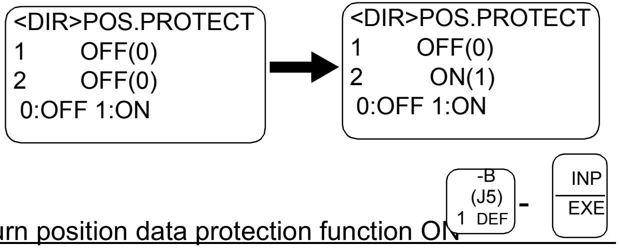


```
<DIR>POS.PROTECT
1       OFF( )
2       OFF(0)
 0:OFF 1:ON
```

```
<DIR>POS.PROTECT
1       OFF(0)
2       OFF( )
 0:OFF 1:ON
```

Turn program protection function ON

1) Display the "program list display" explained on the previous page, and then display the variable protection state. Move the cursor to the program targeted for protection with the [RPL] and [ADD] keys.



```
<DIR>POS.PROTECT
1       OFF(0)
2       OFF(0)
 0:OFF 1:ON
```

```
<DIR>POS.PROTECT
1       OFF(0)
2         ON(1)
 0:OFF 1:ON
```

Turn position data protection function ON

2) The variable protection function for that program will turn ON or OFF when the [INP] key is pressed after the following key:
Variable protection function ON ... [1] key
Variable protection function OFF ... [0] key

Variable protection

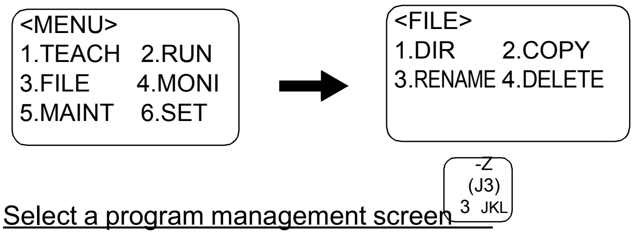
This function protects the variable from inadvertent position data registration or changing, and from substitution to each variable during incorrect program execution.
*The protection function is not copied with the copy operation.
*The protected information is ignored during the initialization process.

(3) Copying programs

This function copies a registered program to another program.
If an existing program name is designated as the copy destination program, an error will occur.

An example for copying program 1 to program 5 is shown below.

1) Press the [3] key, and select the program control screen.

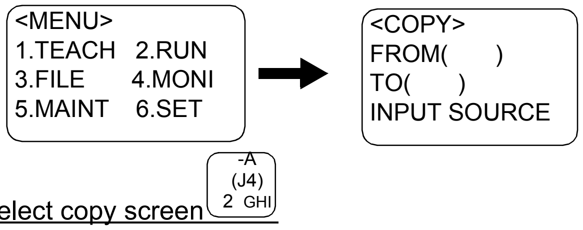


```
<MENU>
1.TEACH   2.RUN
3.FILE    4.MONI
5.MAINT   6.SET
```

```
<FILE>
1.DIR      2.COPY
3.RENAME 4.DELETE
```

Select a program management screen

2) Press the [2] key, and select the program copy screen.

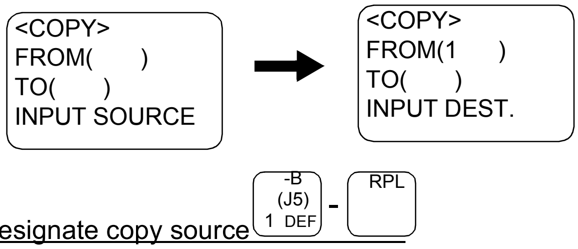


```
<MENU>
1.TEACH   2.RUN
3.FILE    4.MONI
5.MAINT   6.SET
```

```
<COPY>
FROM(      )
TO(      )
INPUT SOURCE
```

Select copy screen

3) Press the [1] key and then the [(] key to move the cursor to the copy destination program name input line.

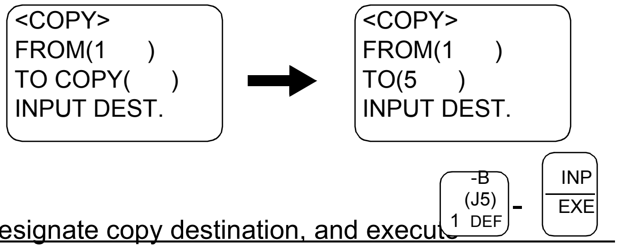


```
<COPY>
FROM(      )
TO(      )
INPUT SOURCE
```

```
<COPY>
FROM(1      )
TO(      )
INPUT DEST.
```

Designate copy source

4) When the [INP] key is pressed after pressing [5], the program will be copied. The program control screen will appear after execution.



```
<COPY>
FROM(1      )
TO COPY(      )
INPUT DEST.
```

```
<COPY>
FROM(1      )
TO(5      )
INPUT DEST.
```

Designate copy destination, and execute

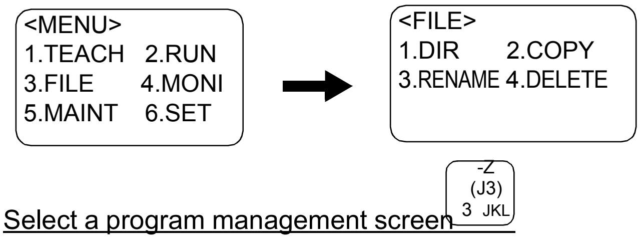
Protected information is not copied

The program protection information and variable protection information is not copied with the copy operation.
Reset this information as necessary.

(4) Changing the program name (Renaming)

This function renames a registered program's name.
If an existing program name is designated as the rename destination program, an error will occur.

An example for renaming program 1 to program 5 is shown below.

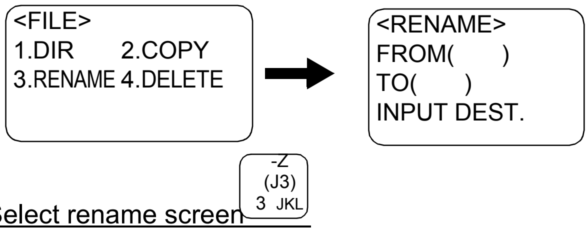


```
<MENU>
1.TEACH   2.RUN
3.FILE    4.MONI
5.MAINT   6.SET
```

```
<FILE>
1.DIR      2.COPY
3.RENAME 4.DELETE
```

Select a program management screen

1) Press the [3] key, and select the program control screen.

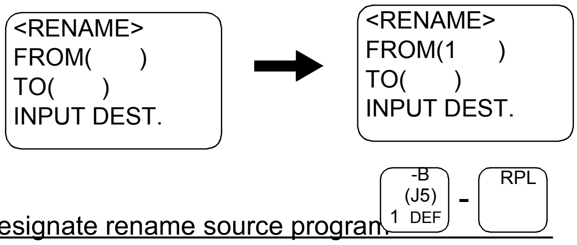


```
<FILE>
1.DIR      2.COPY
3.RENAME 4.DELETE
```

```
<RENAME>
FROM(      )
TO(      )
INPUT DEST.
```

Select rename screen

2) Press the [3] key, and select the program rename screen.

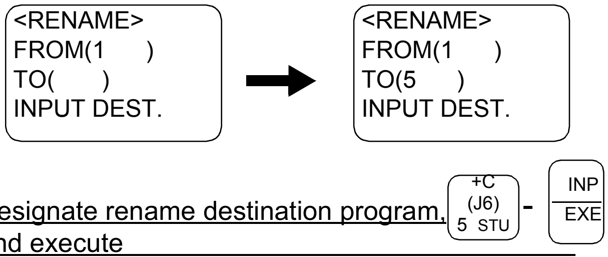


```
<RENAME>
FROM(      )
TO(      )
INPUT DEST.
```

```
<RENAME>
FROM(1     )
TO(      )
INPUT DEST.
```

Designate rename source program

3) Press the [1] key and then the [RPL] key to move the cursor to the rename destination program name input line.



```
<RENAME>
FROM(1     )
TO(      )
INPUT DEST.
```

```
<RENAME>
FROM(1     )
TO(5     )
INPUT DEST.
```

Designate rename destination program, and execute

4) When the [INP] key is pressed after pressing [5], the program will be renamed. The program control screen will appear after execution.

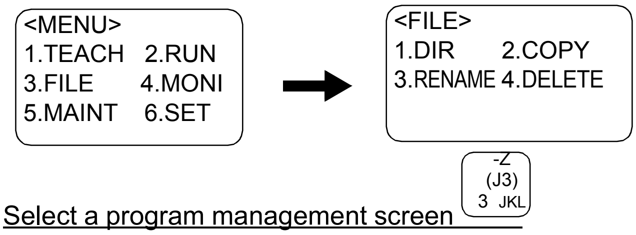
Renaming is not possible when protected

If either the program protection or variable protection is ON, the program cannot be renamed.
In this case, turn the protection OFF before renaming.

(5) Deleting a program

This function deletes a registered program.

The case for deleting program 1 is shown below.

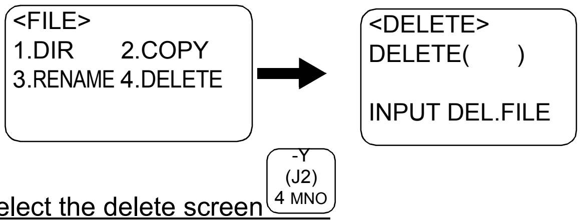


```
<MENU>
1.TEACH  2.RUN
3.FILE   4.MONI
5.MAINT  6.SET
```

```
<FILE>
1.DIR      2.COPY
3.RENAME 4.DELETE
```

Select a program management screen

1) Press the [3] key, and select the program control screen.

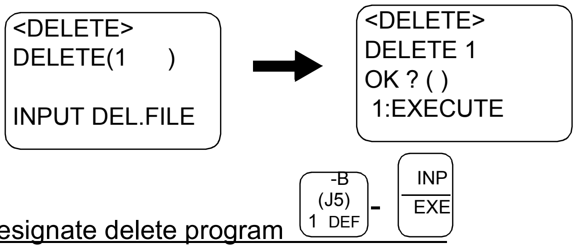


```
<FILE>
1.DIR      2.COPY
3.RENAME 4.DELETE
```

```
<DELETE>
DELETE(      )

INPUT DEL.FILE
```

Select the delete screen

2) Press the [4] key, and select the program rename screen.

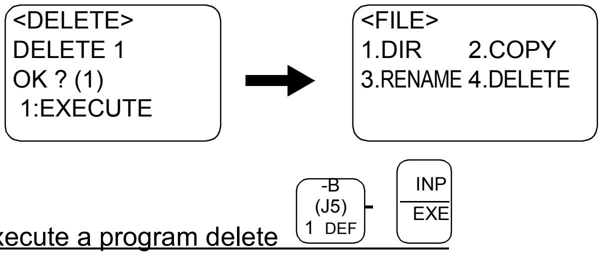


```
<DELETE>
DELETE(1     )

INPUT DEL.FILE
```

```
<DELETE>
DELETE 1
OK ? ( )
 1:EXECUTE
```

Designate delete program

3) When the [INP] key is pressed after pressing [1], a deletion confirmation message will appear.



```
<DELETE>
DELETE 1
OK ? (1)
 1:EXECUTE
```

```
<FILE>
1.DIR      2.COPY
3.RENAME 4.DELETE
```

Execute a program delete

4) When the [INP] key is pressed after pressing [1], the program will be deleted. The program control screen will appear after execution.

Deletion not possible when protected

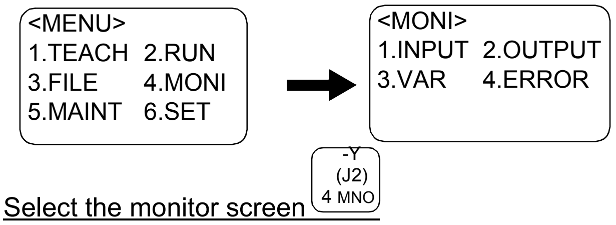
If either the program protection or variable protection is ON, the program cannot be deleted.
In this case, turn the protection OFF before deleted.

3.12 Operating the monitor screen

(1) Input signal monitor

This function allows the state of the input signals from an external source to be confirmed at real-time.

An example for confirming the states of the input signal bits 8 to 15 is shown below.

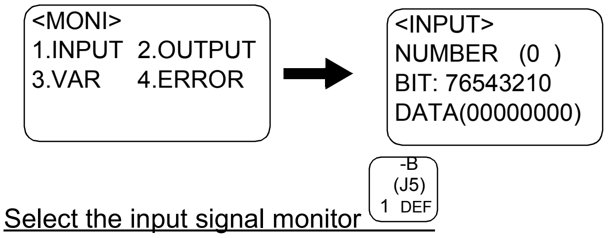


```
<MENU>
1.TEACH  2.RUN
3.FILE   4.MONI
5.MAINT  6.SET
```

```
<MONI>
1.INPUT  2.OUTPUT
3.VAR    4.ERROR
```

Select the monitor screen

1) Press the [4] key, and select the monitor screen.

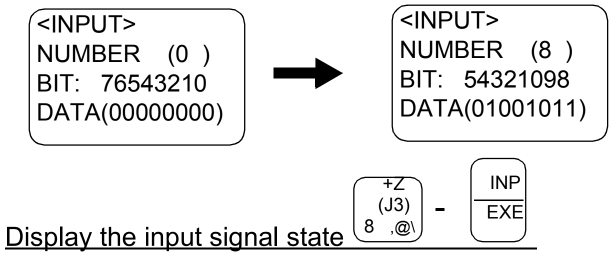


```
<MONI>
1.INPUT  2.OUTPUT
3.VAR    4.ERROR
```

```
<INPUT>
NUMBER   (0  )
BIT: 76543210
DATA(00000000)
```

Select the input signal monitor

2) Press the [1] key, and select the input signal screen.



```
<INPUT>
NUMBER   (0  )
BIT:  76543210
DATA(00000000)
```

```
<INPUT>
NUMBER   (8  )
BIT:  54321098
DATA(01001011)
```

Display the input signal state

3) When the [INP] key is pressed after pressing the [8] key, the states of the input signal bits 8 to 15 will appear.

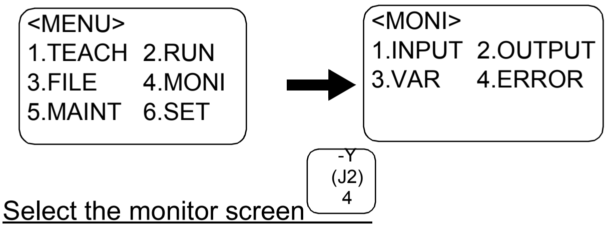
Operation rights not required

This operation can be carried out even if the T/B does not have the operation rights.

(2) Output signal monitor

This function allows the state of the signals output to an external source to be confirmed and set.

An example for turning the input signal bit 8 ON is shown below.

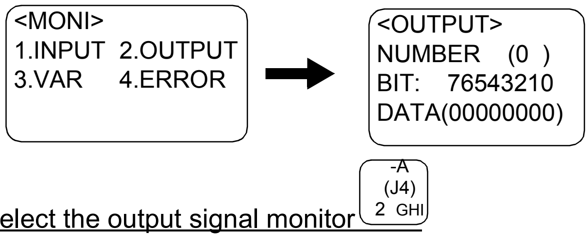


```
<MENU>
1.TEACH  2.RUN
3.FILE     4.MONI
5.MAINT  6.SET
```

```
<MONI>
1.INPUT  2.OUTPUT
3.VAR     4.ERROR
```

Select the monitor screen

1) Press the [4] key, and select the monitor screen.

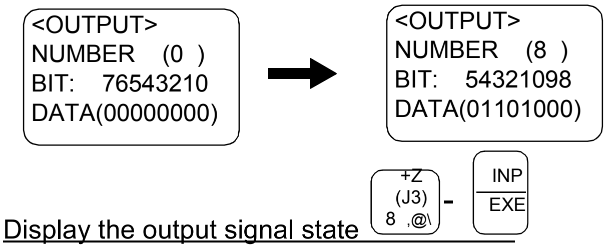


```
<MONI>
1.INPUT  2.OUTPUT
3.VAR     4.ERROR
```

```
<OUTPUT>
NUMBER   (0  )
BIT:   76543210
DATA(00000000)
```

Select the output signal monitor

2) Press the [2] key, and select the output signal screen.

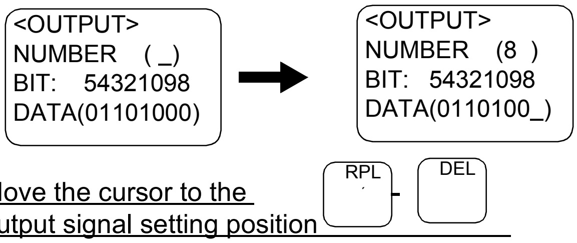


```
<OUTPUT>
NUMBER   (0  )
BIT:   76543210
DATA(00000000)
```

```
<OUTPUT>
NUMBER   (8  )
BIT:   54321098
DATA(01101000)
```

Display the output signal state

3) When the [INP] key is pressed after pressing the [8] key, the states of the output signal bits 8 to 15 will appear.

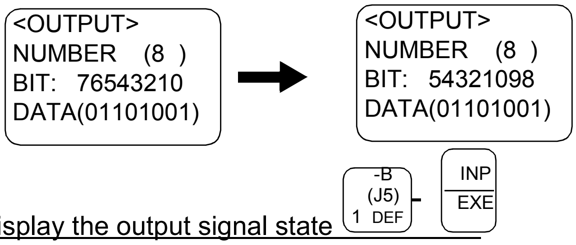


```
<OUTPUT>
NUMBER   ( _)
BIT:   54321098
DATA(01101000)
```

```
<OUTPUT>
NUMBER   (8  )
BIT:   54321098
DATA(0110100_)
```

Move the cursor to the output signal setting position

4) Press the [RPL] key and then the [DEL] key to move the cursor below bit 8.



```
<OUTPUT>
NUMBER   (8  )
BIT:   76543210
DATA(01101001)
```

```
<OUTPUT>
NUMBER   (8  )
BIT:   54321098
DATA(01101001)
```

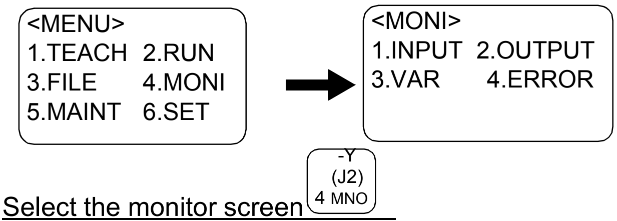
Display the output signal state

5) When the [INP] key is pressed after pressing the [1] key, the output signal bit 8 will be set to ON.

(3) Variable monitor

This function allows the details of the numeric variables used in the program to be displayed and changed.

An example for changing the program 1 numeric variable M8 value from 2 to 5 is shown below.

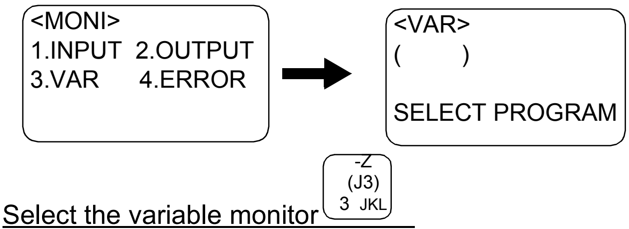


```
<MENU>
1.TEACH  2.RUN
3.FILE   4.MONI
5.MAINT  6.SET
```

```
<MONI>
1.INPUT  2.OUTPUT
3.VAR    4.ERROR
```

Select the monitor screen

1) Press the [4] key, and select the monitor screen.

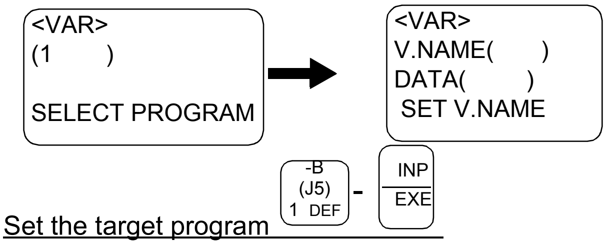


```
<MONI>
1.INPUT  2.OUTPUT
3.VAR    4.ERROR
```

```
<VAR>
(        )

SELECT PROGRAM
```

Select the variable monitor

2) When the [3] key is pressed, the program selection screen for displaying the variables will appear. Input the name of the program for monitoring the variables.

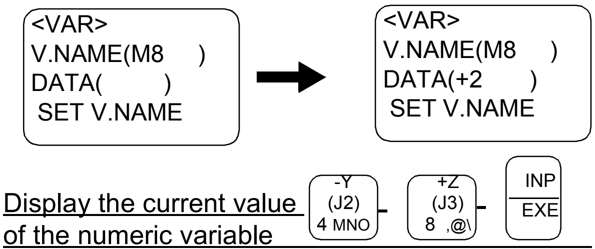


```
<VAR>
(1       )

SELECT PROGRAM
```

```
<VAR>
V.NAME(       )
DATA(         )
 SET V.NAME
```

Set the target program

3) When the [INP] key is pressed after pressing [1], the variable display and setting screen will appear.

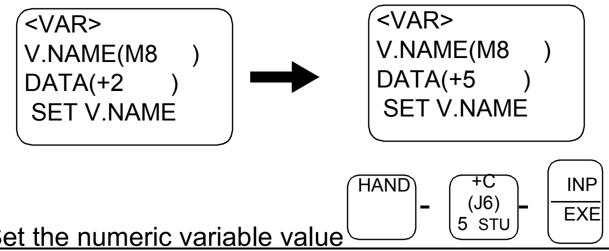


```
<VAR>
V.NAME(M8     )
DATA(         )
 SET V.NAME
```

```
<VAR>
V.NAME(M8     )
DATA(+2       )
 SET V.NAME
```

Display the current value of the numeric variable

4) When the [-Y/MNO] key, [8] key and then [INP] key are pressed, the current value +2 of the program 1 numeric variable M8 will appear.



```
<VAR>
V.NAME(M8     )
DATA(+2       )
 SET V.NAME
```

```
<VAR>
V.NAME(M8     )
DATA(+5       )
 SET V.NAME
```

Set the numeric variable value

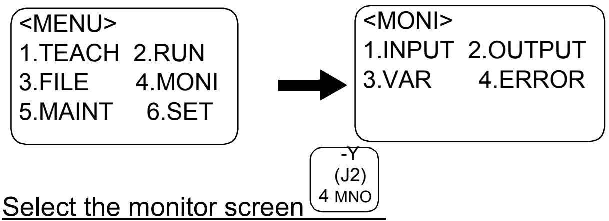
5) When the [HAND] key, [5] key and then [INP] key are pressed, the current value +2 of the program 1 numeric variable M8 will change to +5.

Operation rights not required for only display

If this function is used to only display the values, the T/B does not require the operation rights.
The robot status variables cannot be directly monitored. In this case, the variable must be substituted in the program variables once, and then monitored with the program variable.

(4) Error history

This function displays the history of the errors that have occurred in the robot. Use this as reference when trouble occurs.

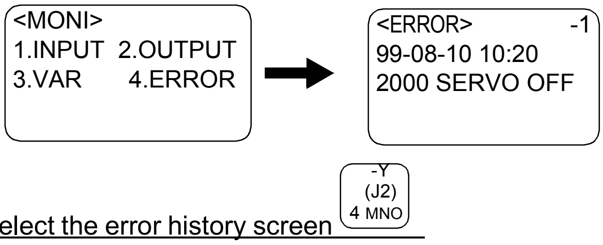


```
<MENU>
1.TEACH  2.RUN
3.FILE     4.MONI
5.MAINT    6.SET
```

```
<MONI>
1.INPUT  2.OUTPUT
3.VAR      4.ERROR
```

Select the monitor screen

1) Press the [4] key, and select the monitor screen.

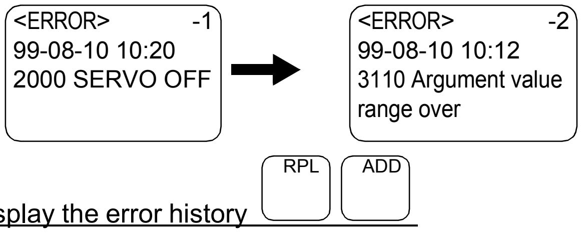


```
<MONI>
1.INPUT  2.OUTPUT
3.VAR      4.ERROR
```

```
<ERROR>              -1
99-08-10 10:20
2000 SERVO OFF
```

Select the error history screen

2) When the [4] key is pressed, the error history will appear.



```
<ERROR>              -1
99-08-10 10:20
2000 SERVO OFF
```

```
<ERROR>              -2
99-08-10 10:12
3110 Argument value
range over
```

Display the error history

3) The error history can be viewed by pressing the following keys:
[ADD] key ... Previous errors
[RPL] key ... Following errors

Operation rights not required

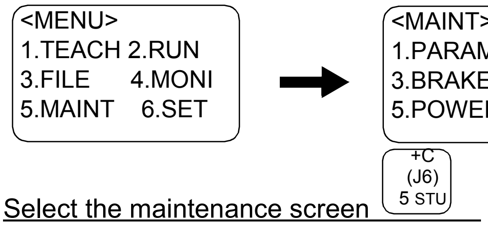
This operation can be carried out even if the T/B does not have the operation rights.

3.13 Operation of maintenance screen

(1) Setting the parameters

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.

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

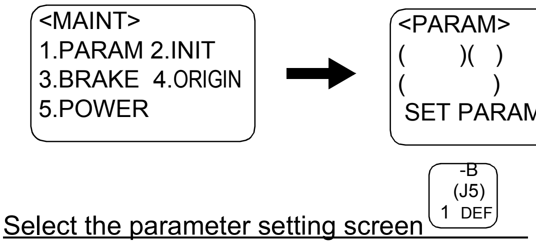


```
<MENU>
1.TEACH 2.RUN
3.FILE    4.MONI
5.MAINT   6.SET
```

```
<MAINT>
1.PARAM 2.INIT
3.BRAKE  4.ORIGIN
5.POWER
```

Select the maintenance screen

1) Press the [5] key, and select the maintenance screen.

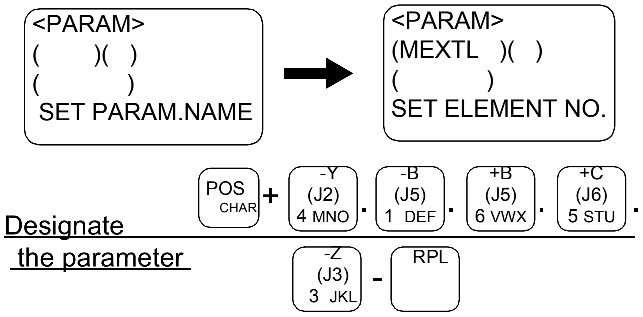


```
<MAINT>
1.PARAM 2.INIT
3.BRAKE  4.ORIGIN
5.POWER
```

```
<PARAM>
(       )(  )
(           )
 SET PARAM.NAME
```

Select the parameter setting screen

2) Press the [1] key, and select the parameter setting screen.

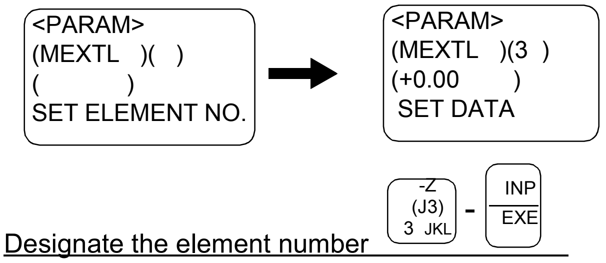


```
<PARAM>
(       )(  )
(           )
 SET PARAM.NAME
```

```
<PARAM>
(MEXTL   )(  )
(           )
SET ELEMENT NO.
```

Designate the parameter

3) Input "MEXTL" and press the [RPL] key to move the cursor to the element No. input line.

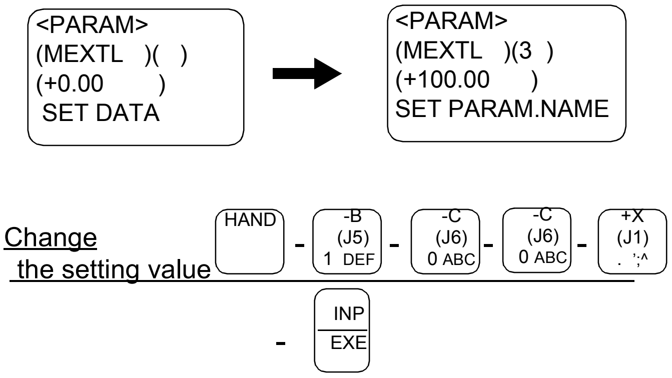


```
<PARAM>
(MEXTL   )(  )
(           )
SET ELEMENT NO.
```

```
<PARAM>
(MEXTL   )(3  )
(+0.00       )
 SET DATA
```

Designate the element number

4) Press the [3] key to designate the 3rd element (Z axis), and press the [INP] key. The parameter MEXTL Z axis current value will appear as +0.00.



```
<PARAM>
(MEXTL   )(  )
(+0.00       )
 SET DATA
```

```
<PARAM>
(MEXTL   )(3  )
(+100.00     )
SET PARAM.NAME
```

Change the setting value

5) Input "+100.00" here, and then press the [INP] key. The parameter MEXTL Z axis setting value will be changed from 0 to 100.

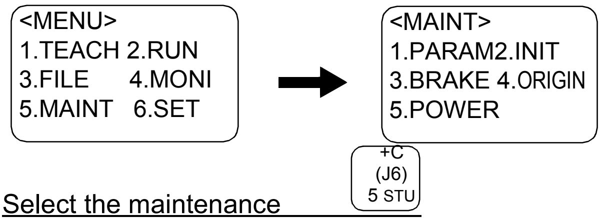
6) Turn the controller's power off and on again; otherwise the changed parameters will not become valid.

Power must be turned ON again

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

(2) Initializing the program

This function erases all programs.

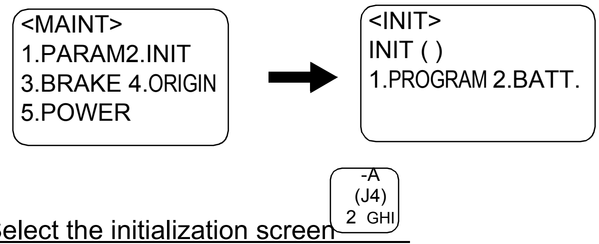


```
<MENU>
1.TEACH 2.RUN
3.FILE      4.MONI
5.MAINT   6.SET
```

```
<MAINT>
1.PARAM2.INIT
3.BRAKE 4.ORIGIN
5.POWER
```

Select the maintenance

1) Press the [5] key, and select the maintenance screen.

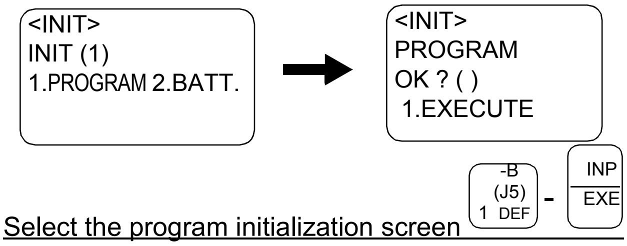


```
<MAINT>
1.PARAM2.INIT
3.BRAKE 4.ORIGIN
5.POWER
```

```
<INIT>
INIT ( )
1.PROGRAM 2.BATT.
```

Select the initialization screen

2) Press the [2] key, and select the initialization screen.

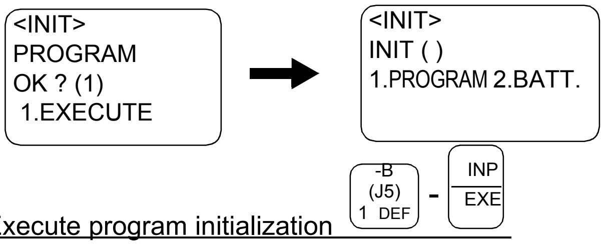


```
<INIT>
INIT (1)
1.PROGRAM 2.BATT.
```

```
<INIT>
PROGRAM
OK ? ( )
 1.EXECUTE
```

Select the program initialization screen

3) Press [1] and then [INP] to select the program initialization screen.



```
<INIT>
PROGRAM
OK ? (1)
 1.EXECUTE
```

```
<INIT>
INIT ( )
1.PROGRAM 2.BATT.
```

Execute program initialization

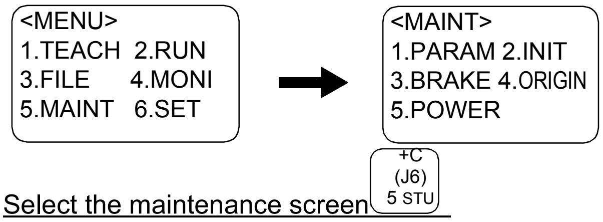
4) When the [INP] key is pressed after pressing [1], the program initialization will start. The initialization screen will appear after the execution.

Executed even when protected

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

(3) Initializing the battery consumption time

The usage time of the battery built into the controller and robot arm is calculated to indicate battery replacement on the caution message screen when the battery is spent. Thus, always initialize this setting after replacing the battery.

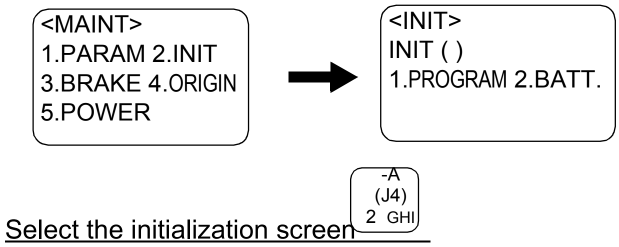


```
<MENU>
1.TEACH  2.RUN
3.FILE     4.MONI
5.MAINT   6.SET
```

```
<MAINT>
1.PARAM 2.INIT
3.BRAKE 4.ORIGIN
5.POWER
```

Select the maintenance screen

1) Press the [5] key, and select the maintenance screen.

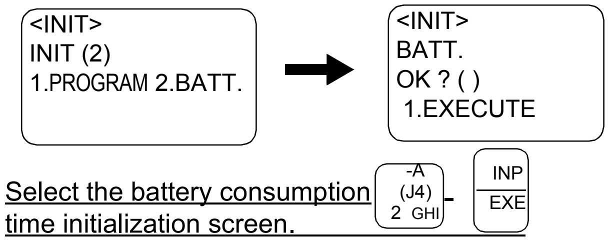


```
<MAINT>
1.PARAM 2.INIT
3.BRAKE 4.ORIGIN
5.POWER
```

```
<INIT>
INIT ( )
1.PROGRAM 2.BATT.
```

Select the initialization screen

2) Press the [1] key, and select the initialization screen.

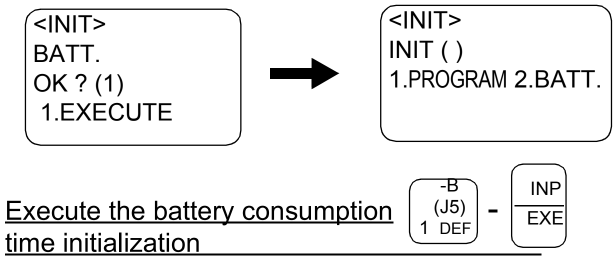


```
<INIT>
INIT (2)
1.PROGRAM 2.BATT.
```

```
<INIT>
BATT.
OK ? ( )
 1.EXECUTE
```

Select the battery consumption time initialization screen.

3) Press [2] and then [INP] to select the battery consumption time initialization screen.



```
<INIT>
BATT.
OK ? (1)
 1.EXECUTE
```

```
<INIT>
INIT ( )
1.PROGRAM 2.BATT.
```

Execute the battery consumption time initialization

4) When the [INP] key is pressed after pressing [1], the battery consumption time initialization will start. The initialization screen will appear after the execution.

Always initialize after battery replacement

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

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

(4) Releasing the brakes

This function releases the servomotor brakes when the servo is OFF. Refer to Page 42, "3.8 Turning the servo ON/OFF" for details on turning the servo OFF.
This function is used to directly move the robot arm by hand, etc.

⚠CAUTION Due to the robot configuration, when the brakes are released, the robot arm will drop with its own weight depending on the released axis.
Always assign an operator other than the T/B operator to prevent the arm from dropping. This operation must be carried out with the T/B operator giving signals.
Refer to Table 3-5 and accurately designate the axis for which the brakes are to be released.

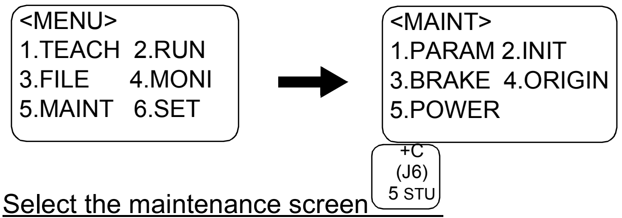
Note that the minimum axis unit for which the brakes can be released at once will differ according to the model of the robot in use. Table 3-5 shows the minimum axis unit for which the brake release operation can be performed at once for each model.
The minimum axis units can be combined to release the brakes form multiple axes at the same time.

Table 3-5:Brake release axis unit per mode

Model	AXIS								Remarks
	1	2	3	4	5	6	7	8	
RP-1AH/3AH/5AH RH-5AH/10AH/15AH RH-6SH/12SH/18SH RH-15UHC RV-100THL	*	*	*	*					The three axes (Z axes) of the RH-10AH/15AH and RH-12SH/18SH are in the intermittent brake mode. The brake release operation repeatedly applies and releases the brake in order to prevent the arm from dropping suddenly.
RV-1A/2A/4A/3AL RV-3S/3SB/6S/6SL/12S/12SL/18S	*	*	*	*	*	*			
RV-2AJ/3AJ/5AJ/4AJL/3SJ/3SJB	*	*	*		*	*			
RH-1000GHLC RC-1000GHWLC/1000GHWDC	* = *		*	*					Axes 1 and 2 are paired. It is not possible to release the brake if both axes are not set to 1 simultaneously.
RH-1000GHLC-RL	* = *		*	*				*	
RH-1000GJLC RH-1500GJC	* = *		*	* = *					Axes 1 and 2, and 4 and 5 are paired. It is not possible to release the brake if both axes are not set to 1 simultaneously.
RH-1000GJLC-RL RH-1500GJC-RL	* = *		*	* = *				*	
RH-1500GC	* = *		*	* = *		*			
RH-1500GC-RL RH1500GVC-RL	* = *		*	* = *		* = = *			Axes 1 and 2, and 4 and 5, and 6 and 8 are paired. It is not possible to release the brake if both axes are not set to 1 simultaneously.
RC-1000GHWLC-RL RC-1000GHWDC-RL	* = *		*	* = = = = *					Axes 1 and 2, and 4 and 8 are paired. It is not possible to release the brake if both axes are not set to 1 simultaneously.
RV-100TH/150TH/150THL RH-1000GHDC RS-30AG/30FG	* = *		* = *						Axes 1 and 2, and 3 and 4 are paired. It is not possible to release the brake if both axes are not set to 1 simultaneously.
RH-1000GHDC-RL	* = *		* = *					*	
RH-1000GJDC	* = *		* = *		*				
RH-1000GJDC-RL	* = *		* = *		*			*	
RV-20A	* = *		* = *		* = *				Axes 1 and 2, and 3 and 4, and 5 and 6 are paired. It is not possible to release the brake if both axes are not set to 1 simultaneously.
RC-1300G	* * * ========== * * ========*								Axes 1 and 5, and 3 and 6 are paired. It is not possible to release the brake if both axes are not set to 1 simultaneously.

Note) The symbol [*=*] means that it is necessary to set two axes at the same time to release the brake.
The symbol [*] means that it is possible to release the brake for an independent axis.

The operation method is shown below. The following operations are carried out while lightly pressing the deadman switch on the T/B.

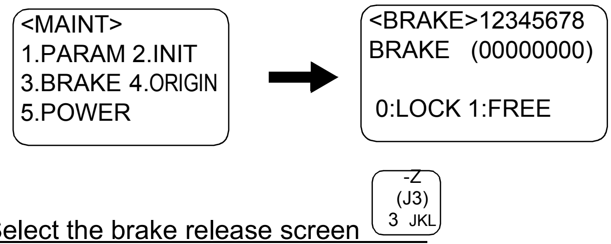


```
<MENU>
1.TEACH  2.RUN
3.FILE     4.MONI
5.MAINT   6.SET
```

```
<MAINT>
1.PARAM 2.INIT
3.BRAKE  4.ORIGIN
5.POWER
```

Select the maintenance screen

1) Press the [5] key, and select the maintenance screen.

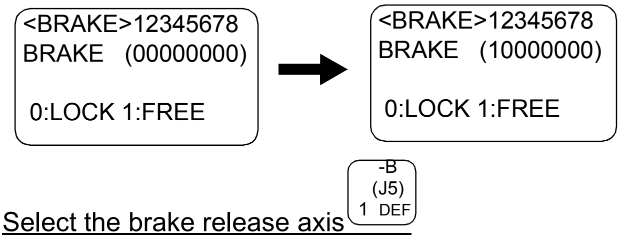


```
<MAINT>
1.PARAM 2.INIT
3.BRAKE 4.ORIGIN
5.POWER
```

```
<BRAKE>12345678
BRAKE   (00000000)

 0:LOCK 1:FREE
```

Select the brake release screen

2) Press the [3] key, and select the brake release screen.

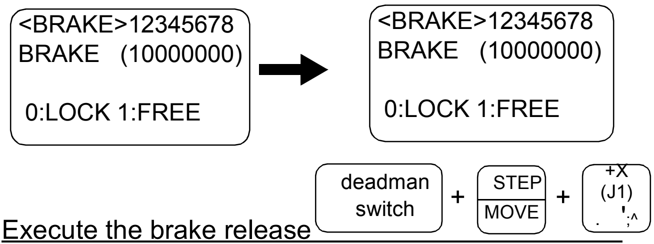


```
<BRAKE>12345678
BRAKE   (00000000)

 0:LOCK 1:FREE
```

```
<BRAKE>12345678
BRAKE   (10000000)

 0:LOCK 1:FREE
```

Select the brake release axis

3) Press the [1] key to change the number corresponding to the axis for which the brakes are to be released to 1. Refer to Table 3-5 and set the axis designation.
In the example on the left, the J1 axis brake release is designated for the RP-1AH.



```
<BRAKE>12345678
BRAKE   (10000000)

 0:LOCK 1:FREE
```

```
<BRAKE>12345678
BRAKE   (10000000)

 0:LOCK 1:FREE
```

Execute the brake release

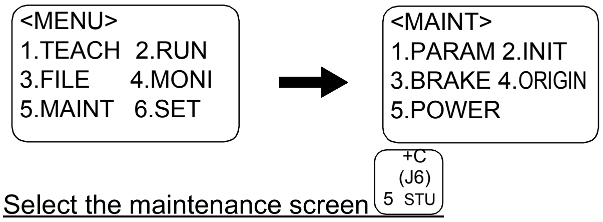
4) Hold the deadman switch (on the back of the T/B). Then hold down the [MOVE] key and press the [+X] key continuously to release the brake of the specified axis only while the keys are pressed.
The brakes will activate when the [MOVE] key or [+X] key is released.

(5) Setting the 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.

(6) Displaying the clock data for maintenance

The controller's cumulative power ON time and remaining battery time are displayed.(Unit:hour)

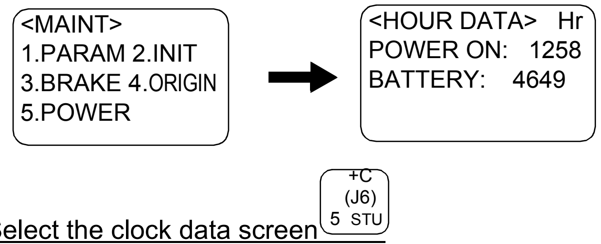


```
<MENU>
1.TEACH  2.RUN
3.FILE     4.MONI
5.MAINT   6.SET
```

```
<MAINT>
1.PARAM 2.INIT
3.BRAKE 4.ORIGIN
5.POWER
```

Select the maintenance screen

1) Press the [5] key, and select the maintenance screen.



```
<MAINT>
1.PARAM 2.INIT
3.BRAKE 4.ORIGIN
5.POWER
```

```
<HOUR DATA>   Hr
POWER ON:   1258
BATTERY:    4649
```

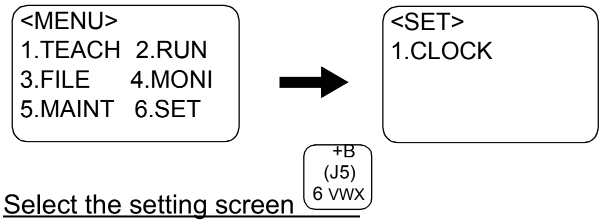
Select the clock data screen

2) When the [5] key is pressed, the clock data for maintenance will appear.

3.14 Operation of the setting screen

(1) Setting the time

A clock function, used when registering the program, and displaying the change time or error time, etc., is provided in the controller. If the times are deviated from the current date and time, change the date and time to the correct values.

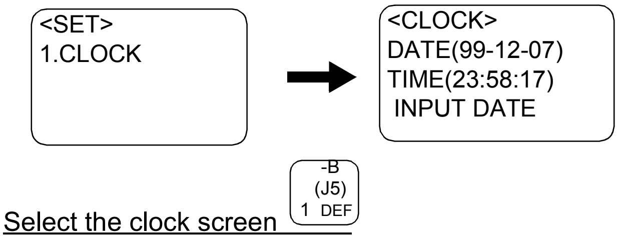


```
<MENU>
1.TEACH  2.RUN
3.FILE   4.MONI
5.MAINT  6.SET
```

```
<SET>
1.CLOCK
```

Select the setting screen

1) Press the [6] key, and select the setting screen.

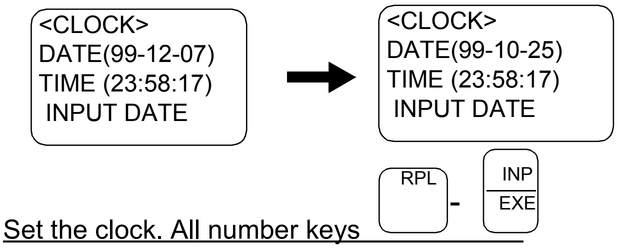


```
<SET>
1.CLOCK
```

```
<CLOCK>
DATE(99-12-07)
TIME(23:58:17)
 INPUT DATE
```

Select the clock screen

2) Press the [1] key, and select the clock screen.

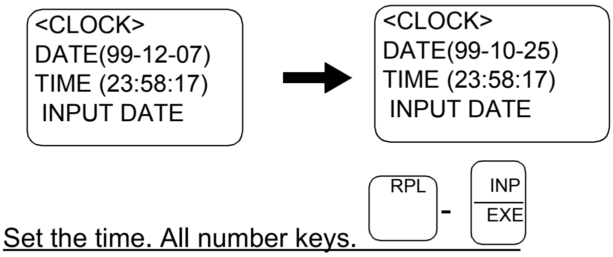


```
<CLOCK>
DATE(99-12-07)
TIME (23:58:17)
 INPUT DATE
```

```
<CLOCK>
DATE(99-10-25)
TIME (23:58:17)
 INPUT DATE
```

Set the clock. All number keys

3) Set the correct date and press the [INP] key. The date will be changed. If the date does not need to be changed, press the [RPL] key. The cursor will move to the time setting section.



```
<CLOCK>
DATE(99-12-07)
TIME (23:58:17)
 INPUT DATE
```

```
<CLOCK>
DATE(99-10-25)
TIME (23:58:17)
 INPUT DATE
```

Set the time. All number keys.

4) Set the time with the same operations as for setting the date.

4 MELFA-BASIC IV

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

4.1 MELFA-BASIC IV functions

The outline of the programming language "MELFA-BASIC IV" 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 118, "4.11 Detailed explanation of command words".

4.1.1 Robot operation control

(1) Joint interpolation movement

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

*Command word

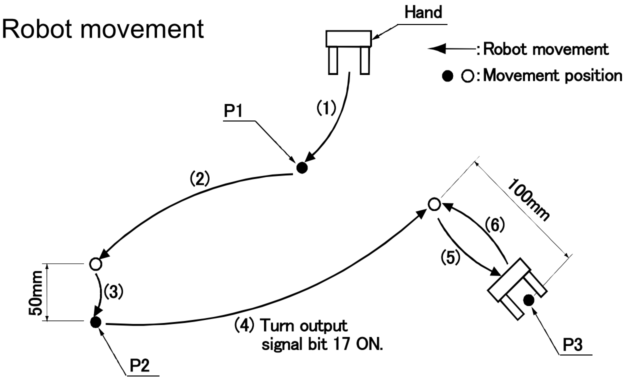
Command word	Explanation
MOV	The robot moves to the designated position with joint interpolation. It is possible to specify the interpolation form using the TYPE instruction. An appended statement WTH or WTHIF can be designated

*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 84.
MOV P1*P2	' Moves to the position relatively converted from P1 to P2. Refer to Page 84.
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 **Specification of forward/backward movement of the hand**

*1) The statement examples and program examples are for a vertical 6-axis robot (e.g., RV-20A).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
10 MOV P1	'(1) Moves to P1.
20 MOV P2, -50 *1)	'(2) Moves from P2 to a position retracted 50mm in the hand direction.
30 MOV P2	'(3) Moves to P2
40 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.
50 MOV P3	'(5) Moves to P3
60 MOV P3, -100 *1)	'(6) Returns from P3 to a position retracted 100mm in the hand direction.
70 END	'Ends the program.

*Related functions

Function	Explanation page
Designate the movement speed..	Page 66, "(5) Acceleration/deceleration time and speed control"
Designate the acceleration/deceleration time.	Page 66, "(5) Acceleration/deceleration time and speed control"
Confirm that the target position is reached.	Page 68, "(6) Confirming that the target position is reached"
Continuously move to next position without stopping at target position..	Page 65, "(4) Continuous movement"
Move linearly.	Page 62, "(2) Linear interpolation movement"
Move while drawing a circle or arc.	Page 63, "(3) Circular interpolation movement"
Add a movement command to the process......................................	Page 221, " WTH (With)"

(2) Linear interpolation movement

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

*Command word

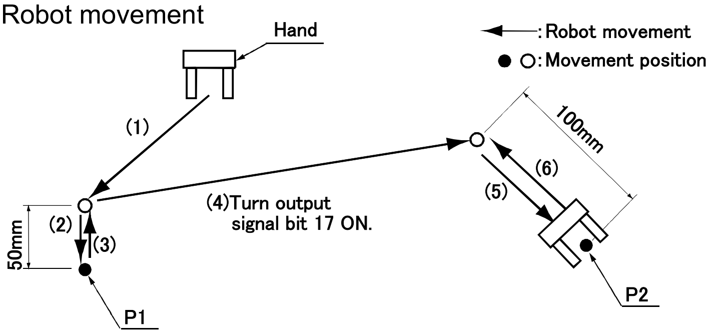
Command word	Explanation
MVS	The robot moves to the designated position with linear interpolation. It is possible to specify the interpolation form using the TYPE instruction. An appended statement WTH or WTHIF can be designated.

*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 84.
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

Robot movement



⚠CAUTION Specification of forward/ backward movement of the hand

*1) The statement examples and program examples are for a vertical 6-axis robot (e.g., RV-20A).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
10 MVS P1, -50 *1)	' (1) Moves with linear interpolation from P1 to a position retracted 50mm in the hand direction.
20 MVS P1	' (2) Moves to P1 with linear interpolation.
30 MVS ,-50 *1)	' (3) Moves with linear interpolation from the current position (P1) to a position retracted 50mm in the hand direction.
40 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.
50 MVS P2	(5) Moves with linear interpolation to P2.
60 MVS , -100 *1)	(6) Moves with linear interpolation from the current position (P2) to a position retracted 50mm in the hand direction.
70 END	'Ends the program.

*Related functions

Function	Explanation page
Designate the movement speed.	Page 66, "(5) Acceleration/deceleration time and speed control"
Designate the acceleration/deceleration time.	Page 66, "(5) Acceleration/deceleration time and speed control"
Confirm that the target position is reached.	Page 68, "(6) Confirming that the target position is reached"
Continuously move to next position without stopping at target position..........	Page 65, "(4) Continuous movement"
Move with joint interpolation..	Page 61, "(1) Joint interpolation movement"
Move while drawing a circle or arc...	Page 63, "(3) Circular interpolation movement"
Add a movement command to the process..	Page 221, " 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.

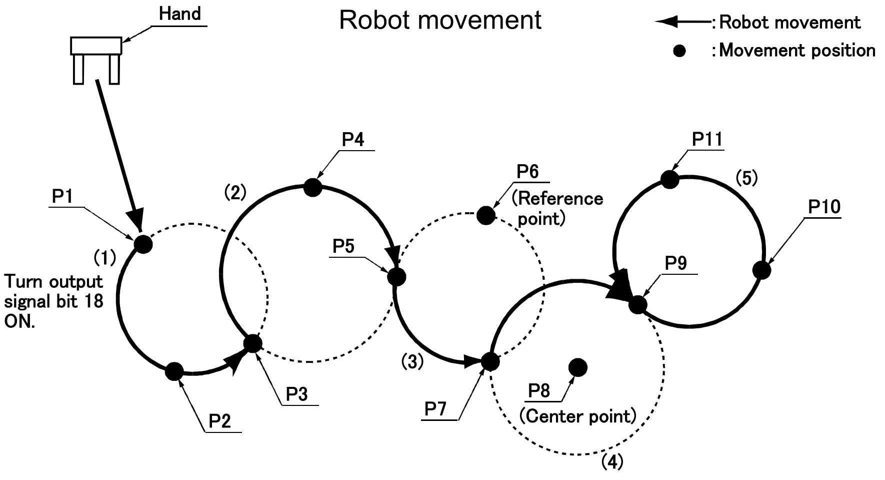
*Command word

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



•Program example

Program	Explanation
10 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.
20 MVR P3, P4, P5	' (2) Moves between P3 - P4 - P5 as an arc.
30 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.
40 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.
50 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)
60 END	' Ends the program.

*Related functions

Function	Explanation page
Designate the movement speed.	Page 66, "(5) Acceleration/deceleration time and speed control"
Designate the acceleration/deceleration time.	Page 66, "(5) Acceleration/deceleration time and speed control"
Confirm that the target position is reached.	Page 68, "(6) Confirming that the target position is reached"
Continuously move to next position without stopping at target position.........	Page 65, "(4) Continuous movement"
Move with joint interpolation...	Page 61, "(1) Joint interpolation movement"
Move linearly.	Page 62, "(2) Linear interpolation movement"
Add a movement command to the process..	Page 221, " 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

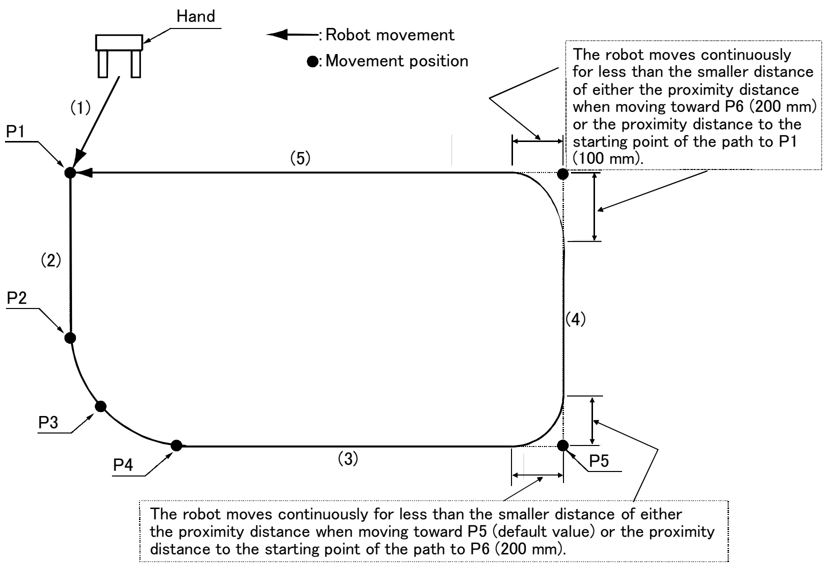
Command word	Explanation
CNT	Designates the start and end of the continuous movement.

*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



The robot moves continuously for less than the smaller distance of either the proximity distance when moving toward P6 (200 mm) or the proximity distance to the starting point of the path to P1 (100 mm).

The robot moves continuously for less than the smaller distance of either the proximity distance when moving toward P5 (default value) or the proximity distance to the starting point of the path to P6 (200 mm).

⚠CAUTION

Specification of forward/backward movement of the hand

*1) The statement examples and program examples are for a vertical 6-axis robot (e.g., RV-20A).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
10 MOV P1	' (1) Moves with joint interpolation to P1.
20 CNT 1	' Validates continuous movement. (Following movement is continuous movement.)
30 MVR P2, P3, P4	' (2) Moves linearly to P2, and continuously moves to P4 with arc movement.
40 MVS P5	' After arc movement, moves linearly to P5.
50 CNT 1, 200, 100	' (3) Sets the continuous movement's start point neighborhood distance to 200mm, and the end point neighborhood distance to 100mm.
60 MVS P6	' (4) After moving to previous P5, moves in succession linearly to P6.
70 MVS P1	' (5) Continuously moves to P1 with linear movement.
80 CNT 0	' Invalidates the continuous movement.
90 END	' Ends the program.

*Related functions

Function	Explanation page
Designate the movement speed.	Page 66, "(5) Acceleration/deceleration time and speed control"
Designate the acceleration/deceleration time.	Page 66, "(5) Acceleration/deceleration time and speed control"
Confirm that the target position is reached.	Page 68, "(6) Confirming that the target position is reached"
Move with joint interpolation.	Page 61, "(1) Joint interpolation movement"
Move linearly.	Page 62, "(2) Linear interpolation movement"
Move while drawing a circle or arc.	Page 63, "(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

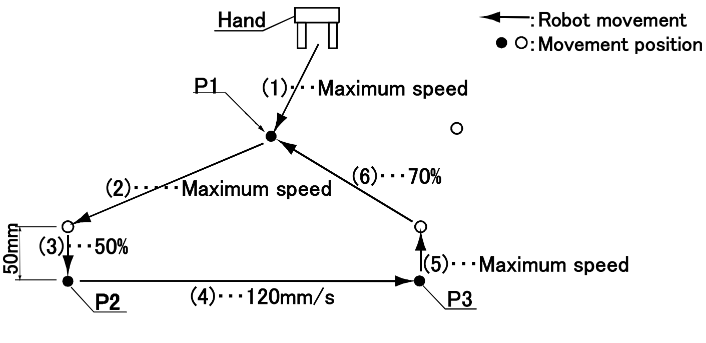
Statement example	Explanation
ACCEL..	Sets both the acceleration and deceleration to 100%.
ACCEL 60, 80..	Sets the acceleration to 60% and the deceleration to 80%. (For maximum acceleration/deceleration is 0.2 sec. acceleration 0.2/0.6=0.33 sec. deceleration 0.2/0.8=0.25 sec.)
OVRD 50	Sets the joint 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 interpolationController (T/B) setting value x OVRD command setting value x JOVRD command setting value.

*Movement speed during linear and circular interpolationController (T/B) setting value x OVRD command setting value x SPD command setting value.

*Program example

Robot movement



⚠CAUTION <u>Specification of forward/ backward movement of the hand</u>

*1) The statement examples and program examples are for a vertical 6-axis robot (e.g., RV-20A).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
10 OVRD 100	' Sets the movement speed applied on the entire program to the maximum speed.
20 MVS P1	' (1) Moves at maximum speed to P1.
30 MVS P2, -50 *1)	' (2) Moves at maximum speed from P2 to position retracted 50mm in hand direction.
40 OVRD 50	' Sets the movement speed applied on the entire program to half of the maximum speed.
50 MVS P2	' (3) Moves linearly to P2 with a speed half of the default speed.
60 SPD 120	' Sets the end speed to 120mm/s. (Since the override is 50%, it actually moves at 60 mm/s.)
70 OVRD 100	' Sets the movement speed percentage to 100% to obtain the actual end speed of 120mm/s.
80 ACCEL 70, 70	' Sets the acceleration and deceleration to 70% of the maximum speed.
90 MVS P3	' (4) Moves linearly to P3 with the end speed 120mm/s.
100 SPD M_NSPD	' Returns the end speed to the default value.
110 JOVRD 70	' Sets the speed for joint interpolation to 70%.
120 ACCEL	' Returns both the acceleration and deceleration to the maximum speed.
130 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.
140 MVS P1	' (6) Moves to P1 at 70% of the maximum speed.
150 END	' Ends the program.

*Related functions

Function	Explanation page
Move with joint interpolation..	Page 61, "(1) Joint interpolation movement"
Move linearly.	Page 62, "(2) Linear interpolation movement"
Move while drawing a circle or arc.	Page 63, "(3) Circular interpolation movement"
Continuously move to next position without stopping at target position..........	Page 65, "(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

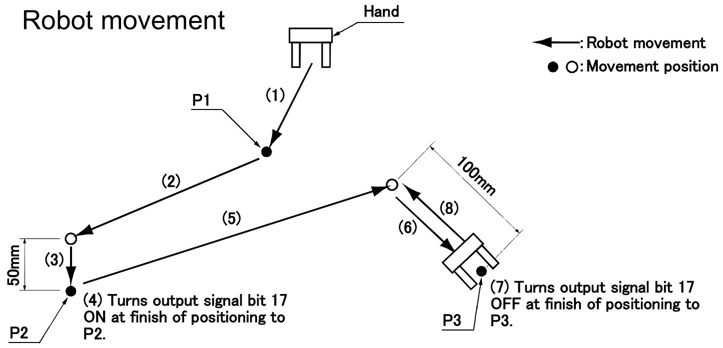
Command word	Explanation
FINE	Designates the positioning finish conditions with a No. of pulses. Specify a small number of pulses to allow more accurate positioning.
MOV and DLY	After the MOV movement command, command the DLY instruction (timer) to complete positioning . (this is effective for belt-driven robots, e.g., RP-1AH, 3AH, and 5AH).

*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. (this is effective for belt-driven robots, e.g., RP-1AH, 3AH, and 5AH).

*Program example

Robot movement


Hand
: Robot movement
● ○: Movement position
P1
(1)
(2)
(5)
(3)
50mm
P2
(4) Turns output signal bit 17 ON at finish of positioning to P2.
100mm
(8)
(6)
(7) Turns output signal bit 17 OFF at finish of positioning to P3.
P3

⚠CAUTION Specification of forward/backward movement of the hand

*1) The statement examples and program examples are for a vertical 6-axis robot (e.g., RV-20A).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
10 CNT 0	' The FINE instruction is valid only when the CNT instruction is OFF.
20 MVS P1	' (1) Moves with joint interpolation to P1.
30 MVS P2, -50 *1)	' (2) Moves with joint interpolation from P2 to position retracted 50mm in hand direction.
40 FINE 50	' Sets positioning finish pulse to 50.
50 MVS P2	' (3) Moves with linear interpolation to P2 (MVS completes if the positioning complete pulse count is 50 or less.)
60 M_OUT(17)=1	' (4) Turns output signal 17 ON when positioning finish pulse reaches 50 pulses.
70 FINE 1000	' Sets positioning finish pulse to 1000.
80 MVS P3, -100 *1)	' (5) Moves linearly from P3 to position retracted 100mm in hand direction.
90 MVS P3	' (6) Moves with linear interpolation to P3.
100 DLY 0.1	' Performs the positioning by the timer.
110 M_OUT(17)=0	' (7) Turns output signal 17 off.
120 MVS , -100 *1)	' (8) Moves linearly from current position (P3) to position retracted 100mm in hand direction.
130 END	' Ends the program.

*Related functions

Function	Explanation page
Move with joint interpolation.	Page 63, "(3) Circular interpolation movement"
Move linearly.	Page 62, "(2) Linear interpolation movement"
Continuously move to next position without stopping at target position.	Page 65, "(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, RV-1A/ 2AJ, RV-2A/ 3AJ, RV-4A/ 5AJ/ 3AL/ 4AJL, RV-20A, RV-3S/ 3SJ/3SB/3SJB, RV-6S/ 6SL/12S/ 12SL and RV-18S series.

*Command word

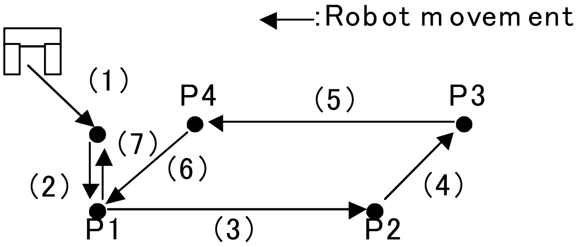
Command word	Explanation
PREC	This instruction specifies whether the high path accuracy mode should be enabled or disabled.

*Statement example

Statement example	Explanation
PRECON	Enables the high path accuracy mode.
PRECOFF	Disables the high path accuracy mode.

*Program example

Robot movement



⚠CAUTION Specification of forward/ backward movement of the hand

*1) The statement examples and program examples are for a vertical 6-axis robot (e.g., RV-20A).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
10 MOV P1, -50 *1)	' (1) Moves with joint interpolation from P1 to position retracted 50mm in hand direction.
20 OVRD 50	' Sets the movement speed to half of the maximum speed.
30 MVS P1	' (2) Moves with linear interpolation to P1.
40 PREC ON	' The high path accuracy mode is enabled.
50 MVS P2	' (3) Moves the robot from P1 to P2 with high path accuracy.
60 MVS P3	' (4) Moves the robot from P2 to P3 with high path accuracy.
70 MVS P4	' (5) Moves the robot from P3 to P4 with high path accuracy.
80 MVS P1	' (6) Moves the robot from P4 to P1 with high path accuracy.
90 PREC OFF	' The high path accuracy mode is ÇÑisableÇÑ.
100 MVS P1, -50	' (7) Returns the robot to the position 50 mm behind P1 in the hand direction using linear interpolation.
110 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

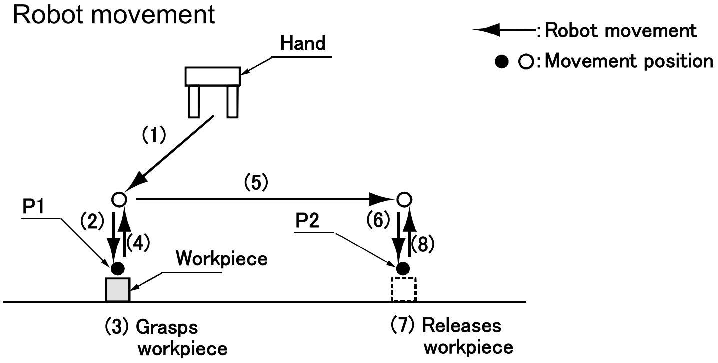
Command word	Explanation
HOPEN	Opens the designated hand.
HCLOSE	Closes the designated hand.
TOOL	Sets the shape of the tool being used, and sets the control point.

*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 <u>Specification of forward/ backward movement of the hand</u>

*1) The statement examples and program examples are for a vertical 6-axis robot (e.g., RV-20A).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
10 TOOL(0, 0, 95, 0, 0, 0)	'Sets the hand length to 95 mm.
20 MVS P1, -50 *1)	'(1) Moves with joint interpolation from P1 to position retracted 50mm in hand direction.
30 OVRD 50	'Sets the movement speed to half of the maximum speed.
40 MVS P1	'(2) Moves with linear interpolation to P1. (Goes to grasp workpiece.)
50 DLY 0.5	' Wait for the 0.5 seconds for the completion of arrival to the target position.
60 HCLOSE 1	'(3) Closes hand 1. (Grasps workpiece.)
70 DLY 0.5	'Waits 0.5 seconds.
80 OVRD 100	'Sets movement speed to maximum speed.
90 MVS , -50 *1)	'(4) Moves linearly from current position (P1) to position retracted 50mm in hand direction. (Lifts up workpiece.)
100 MVS P2, -50 *1)	'(5) Moves with joint interpolation from P2 to position retracted 50mm in hand direction.
110 OVRD 50	'Sets movement speed to half of the maximum speed.
120 MVS P2	'(6) Moves with linear interpolation to P2. (Goes to place workpiece.)
130 DLY 0.5	' Wait for the 0.5 seconds for the completion of arrival to the target position.
140 HOPEN 1	'(7) Opens hand 1. (Releases workpiece.)
150 DLY 0.5	'Waits 0.5 seconds.
160 OVRD 100	' Sets movement speed to maximum speed.
170 MVS , -50 *1)	'(8) Moves linearly from current position (P2) to position retracted 50mm in hand direction. (Separates from workpiece.)
180 END	'Ends the program.

*Related functions

Function	Explanation page
Appended statement	Page 221, " 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.

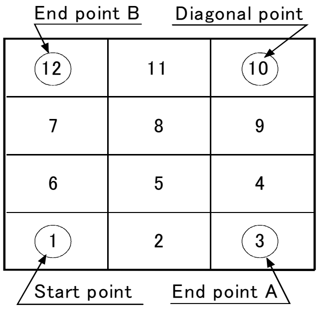
*Command word

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

*Statement example

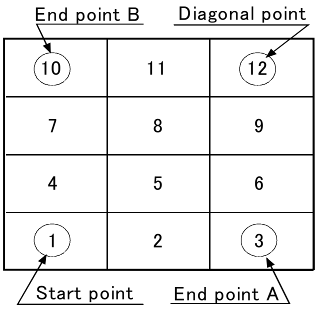
Statement example	Explanation
DEF PLT 1, P1, P2, P3, P4, 4, 3, 1	Defines to operate pallet No. 1 with a start point = P1, end point A = P2, end point B = P3 and diagonal point = P4, a total of 12 work positions (quantity A = 4, quantity B = 3), and an assignment direction = 1(Zigzag).
DEF PLT 2, P1, 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 an assignment direction = 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.

Note 1) The relation of the position designation and assignment direction is shown below



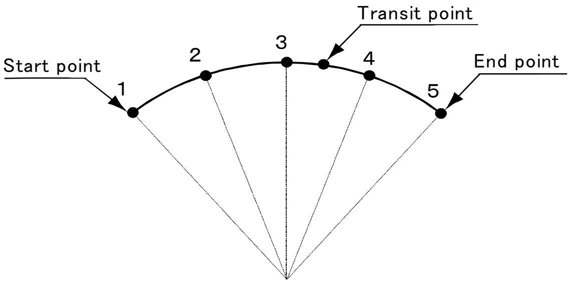
Zigzag

Assignment direction = 1 (zigzag)



Same direction

Assignment direction = 2 (same direction)



Arc pallet

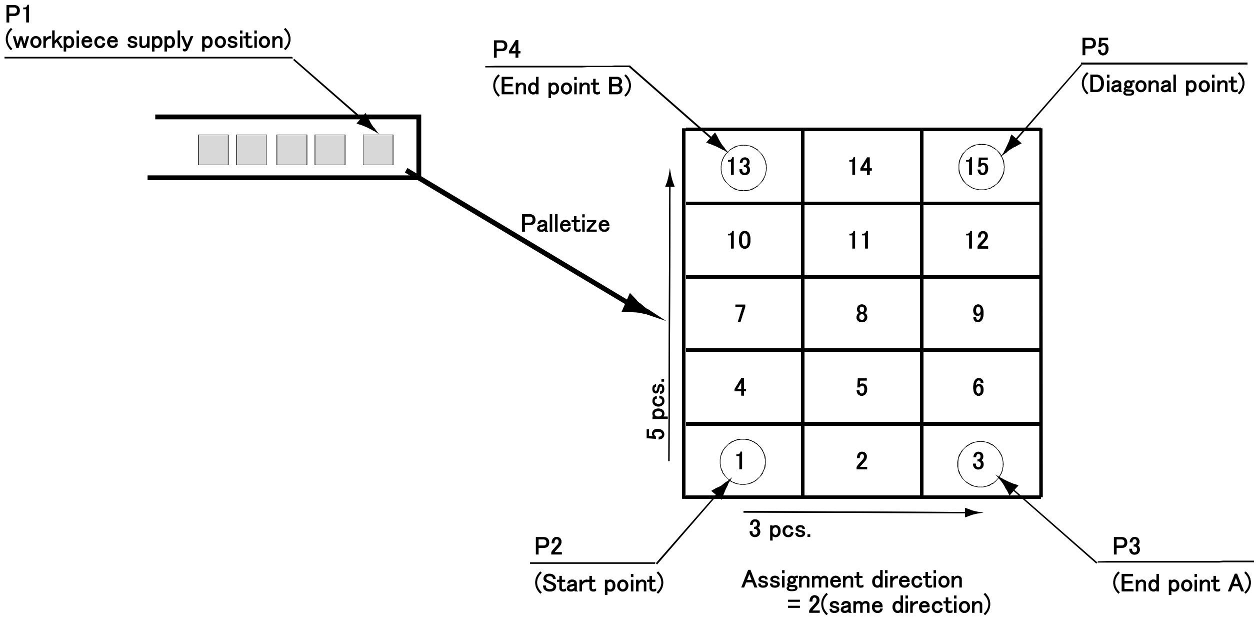
Assignment direction = 3 (arc pallet)

<About the posture of position data defining a pallet>

The signs of the posture data (A, B, and C) at four points, the starting point, endpoint A, endpoint B, and the diagonal point, must match. In the case of a vertical multi-joint robot, if the mechanical interface plane (flange plane) is facing downward, the A, B, and C axis coordinates (especially the A and C axes) may reach 180 degrees. In this case, the sign can be either + or -. Each position of the pallet is calculated from the position data of the starting point and endpoints; if the signs do not match, the hand rotates as a result. +180 and -180 result in the same position. Use + or - consistently when the signs are different, and perform this correction only when the values are exactly 180 degrees.

*Program example

Robot movement



•Program example

Program	Explanation
10 DEF PLT 1, P2, P3, P4, P5, 3, 5, 2	’GDefines 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, assignment direction = 2 (Same direction).
20 M1=1	’GSubstitutes value 1 in numeric variable M1. (M1 is used as a counter.
30 *LOOP	’GDesignates label LOOP at the jump destination.
40 MOV P1, -50 *1)	’GMoves with joint interpolation from P1 to a position retracted 50mm in hand direction.
50 OVRD 50	’GSets movement speed to half of the maximum speed.
60 MVS P1	’GMoves linearly to P1. (Goes to grasp workpiece.)
70 HCLOSE 1	’GCloses hand 1. (Grasps workpiece.)
80 DLY 0.5	’GWaits 0.5 seconds.
90 OVRD 100	’GSets movement speed to maximum speed.
100 MVS , -50 *1)	’GMoves linearly from current position (P1) to a position retracted 50mm in hand direction. (Lifts up workpiece.)
110 P10=(PLT1,M1)	’GOperates the position in pallet No. 1 indicated by the numeric variable M1, and substitutes the results in P10.
120 MOV P10, -50 *1)	’GMoves with joint interpolation from P10 to a position retracted 50mm in hand direction.
130 OVRD 50	’GSets movement speed to half of the maximum speed.
140 MVS P10	’GMoves linearly to P10. (Goes to place workpiece.)
150 HOPEN 1	’GOpens hand 1. (Places workpiece.)
160 DLY 0.5	’GWaits 0.5 seconds.
170 OVRD 100	’GSets movement speed to maximum speed.
180 MVS , -50	’GMoves linearly from current position (P10) to a position retracted 50mm in hand direction. (Separates from workpiece.)
190 M1=M1+1	’GIncrements numeric variable M1 by 1. (Advances the pallet counter.)
200 IF M1<=15 THEN *LOOP	’GIf numeric variable M1 value is less than 15, jumps to label LOOP and repeat process. If more than 15, goes to next step.
210 END	’GEnds the program.

*Related functions

Function	Explanation page
Substitute, operation	Page 82, "4.1.6 Expressions and operations"
Condition branching	Page 73, "(1) Unconditional branching, conditional branching, waiting"

4.1.3 Program control

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

(1) Unconditional branching, conditional branching, waiting

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

*Command word

Command word	Explanation
GOTO	Jumps unconditionally to the designated line.
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 line)	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 line.

IF THEN
ELSE
END IF
(Instructions written in several lines)
	Several lines 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 lines following THEN until the ELSE line are executed. If the conditions are not met, the lines after ELSE until END IF are executed.
	Note) This function is available for controller version G1 or later.

SELECT
CASE
END SELECT
	Jumps according to the designated variable and the designated conditions of that value.
	The value conditions can be designated randomly.
	Multiple types of conditions can be designated per command statement.

WAIT	Waits for the variable to reach the designated value.

*Statement example

Statement example	Explanation
GOTO 200	Jumps unconditionally to line 200.
GOTO *FN	Jumps unconditionally to the label FIN line.
ON M1 GOTO 100, 200, 300	If the numeric variable M1 value is 1, jumps to line 100, if 2 jumps to line 200, and if 3 jumps to line 300. If the value does not correspond, proceeds to next step.
IF M1=1 THEN 100	If the numeric variable M1 value is 1, branches to line 100. If not, proceeds to the next step.
IF M1=1 THEN 100 ELSE 200	If the numeric variable M1 value is 1, branches to line 100. If not, branches to line 200.
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 CASE 10 : BRAKE CASE IS 11 : BRAKE CASE IS <5 : BRAKE CASE 6 TO 9 : BRAKE DEFAULT : BRAKE END SELECT	Branches to the CASE statement corresponding to the value of numeric variable M1. If the value is 10, executes only between CASE 10 and the next CASE 11. If the value is 11, executes only between CASE 11 and the next CASE IS <5. If the value is smaller than 5, executes only between CASE IS <5 and next CASE 6 TO 9. If value is between 6 and 9, executes only between CASE 6 TO 9 and next DEFAULT. If value does not correspond to any of the above, executes only between DEFAULT and next 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 75, "(2) Repetition"
Interrupt	Page 76, "(3) Interrupt"
Subroutine	Page 77, "(4) Subroutine"
External signal input	Page 80, "(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 73, "(1) Unconditional branching, conditional branching, waiting"
Interrupt...	Page 76, "(3) Interrupt"
Input signal wait	Page 80, "(1) Input signals"

(3) Interrupt

Once the designated conditions are established, the command statement being executed can be interrupted and a designated line 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 100	If input signal bit 10 is turned on for interrupt number 1, the subroutine on line 100 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 200, L	If input signal bit 11 is turned on for interrupt number 2, the subroutine on line 200 is defined to be called after the statement currently being executed is completed.
DEF ACT 3, M_IN(12)=1 GOSUB 300, S	If input signal bit 12 is turned on for interrupt number 3, the subroutine on line 300 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 line where the interrupt occurred.
RETURN 1	Returns to the line following the line where the interrupt occurred.

*Related functions

Function	Explanation page
Unconditional branching, branching..	Page 73, "(1) Unconditional branching, conditional branching, waiting"
Subroutine...	Page 77, "(4) Subroutine"
Communication	Page 81, "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 line 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 line following the line called with the GOSUB command.
CALLP	Calls the designated program. The next line 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 line 100.
ON GOSUB	Calls the subroutine from label GET.
ON M1 GOSUB 100, 200, 300	If the numeric variable M1 value is 1, calls the subroutine at line 100, if 2 calls the subroutine at line 200, and if 3 calls the subroutine at line 300. If the value does not correspond, proceeds to next step.
RETURN	Returns to the line following the line 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 76, "(3) Interrupt"
Communication	Page 81, "4.1.5 Communication"
Unconditional branching	Page 73, "(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 80, "(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 line.
END	This instruction defines the end of one cycle of a program. In continuous operation, the program is executed again from the start line 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 221, " 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 106, "4.3.26 Robot status variables" for details on the robot status variables.

*Command word

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

*System variables
M_IN, M_INB, M_INW, M_DIN

*Statement example

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

*Related functions

Function	Explanation page
Signal output	Page 80, "(2) Output signals"
Branching with input signal	Page 73, "(1) Unconditional branching, conditional branching, waiting"
Interrupting with input signal	Page 76, "(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 106, "4.3.26 Robot status variables" for details on the robot status variables.

*Command word

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

*System variables
M_OUT, M_OUTB, M_OUTW, M_DOUT

*Statement example

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

*Related functions

Function	Explanation page
Signal input	Page 80, "(1) Input signals"
Timer	Page 78, "(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 300	Defines to call line 300 subroutine when data is input in communication line COM1.
ON COM(2) GOSUB *RECV	Defines to call subroutine at label RECV line 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 77, "(4) Subroutine"
Interrupt..	Page 76, "(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 200
			IF STS$="OK" THEN 100
				’Branch to line 200 if numeric variable M1 value is 1.
				’Branch to line 100 if character string in character string variable STS$ is OK.

	<> or ><	Compare whether not equal	IF M1<>2 THEN 300
			IF STS$<>"OK" THEN 100
				’Branch to line 300 if numeric variable M1 value is 2.
				’Branch to line 900 if character string in character string variable STS$ is not OK.

	<	Compare whether smaller	IF M1< 10 THEN 300
			IF LEN(STS$)<3 THEN 100
				’Branch to line 300 if numeric variable M1 value is less than 10.
				’Branch to line 100 if No. of characters in character string STS$ variable is less than 3.

	>	Compare whether larger	IF M1>9 THEN 200
			IF LEN(STS$)>2 THEN 300
				’Branch to line 200 if numeric variable M1 value is more than 9.
				’Branch to line 300 if No. of characters in character string variable STS$ is more than 2.

	=< or <=	Compare whether equal to or less than	IF M1<=10 THEN 200
			IF LEN(STS$)<=5 THEN 300
				’Branch to line 200 if numeric variable M1 value is equal to or less than 10.
				’Branch to line 300 if No. of characters in character string variable STS$ is equal to or less then 5.

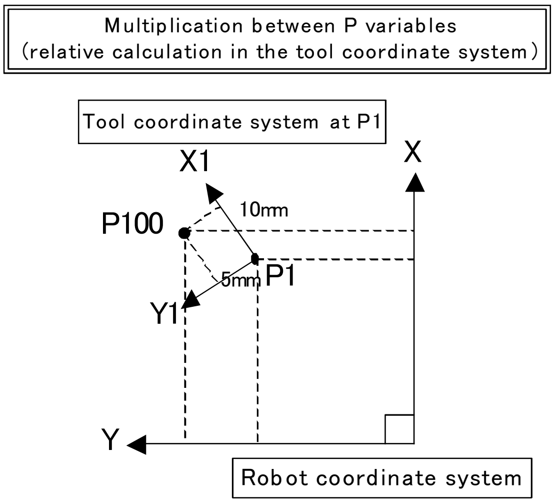
	=> or >=	Compare whether equal to or more than	IF M1=>11 THEN 200
			IF LEN(STS$)>=6 THEN 300
				’Branch to line 200 if numeric variable M1 value is equal to or more than 11.
				’Branch to line 300 if No. of characters in character string variable STS$ is equal to or more than 6.

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 84, "Relative calculation of position data (multiplication)".
Note2) Please refer to Page 84, "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.


Multiplication between P variables
(relative calculation in the tool coordinate system)
Tool coordinate system at P1
X1
X
P100
10mm
5mm
P1
Y1
Y
Robot coordinate system

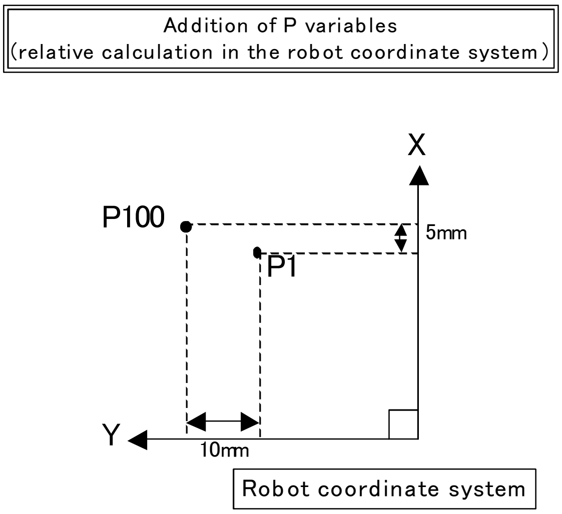
An example of relative calculation (multiplication)

```
10 P2=(10,5,0,0,0,0)(0,0)
20 P100=P1*P2
30 MOV P1
40 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)

```
10 P2=(5,10,0,0,0,0)(0,0)
20 P100=P1+P2
30 MOV P1
40 MVS P100
P1=(200,150,100,0,0,45)(4,0)
```

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

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

CAUTION)

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

4.1.7 Appended statement

A process can be added to a movement command.

*Appended statement

Appended statement	Explanation
WTH	Unconditionally adds a process to the movement command.
WTHIF	Conditionally adds a process to the movement command.

*Statement example

Statement example	Explanation
MOV P1 WTH M_OUT(20)=1..	Turns output signal bit 20 ON simultaneously with the start of movement to P1.
MOV P1 WITHIF M_IN(20)=1, HLT.................................	Stops if the input signal bit 20 turns ON during movement to P1.
MOV P1 WTHIF M_IN(19)=1, SKIP	Stops movement to P1 if the input signal bit 19 turns ON during movement to P1, and then proceeds to the next step.

*Related functions

Function	Explanation page
Joint interpolation movement...	Page 61, "(1) Joint interpolation movement"
Linear interpolation movement...	Page 62, "(2) Linear interpolation movement"
Circular interpolation movement	Page 63, "(3) Circular interpolation movement"
Stopping	Page 79, "(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.

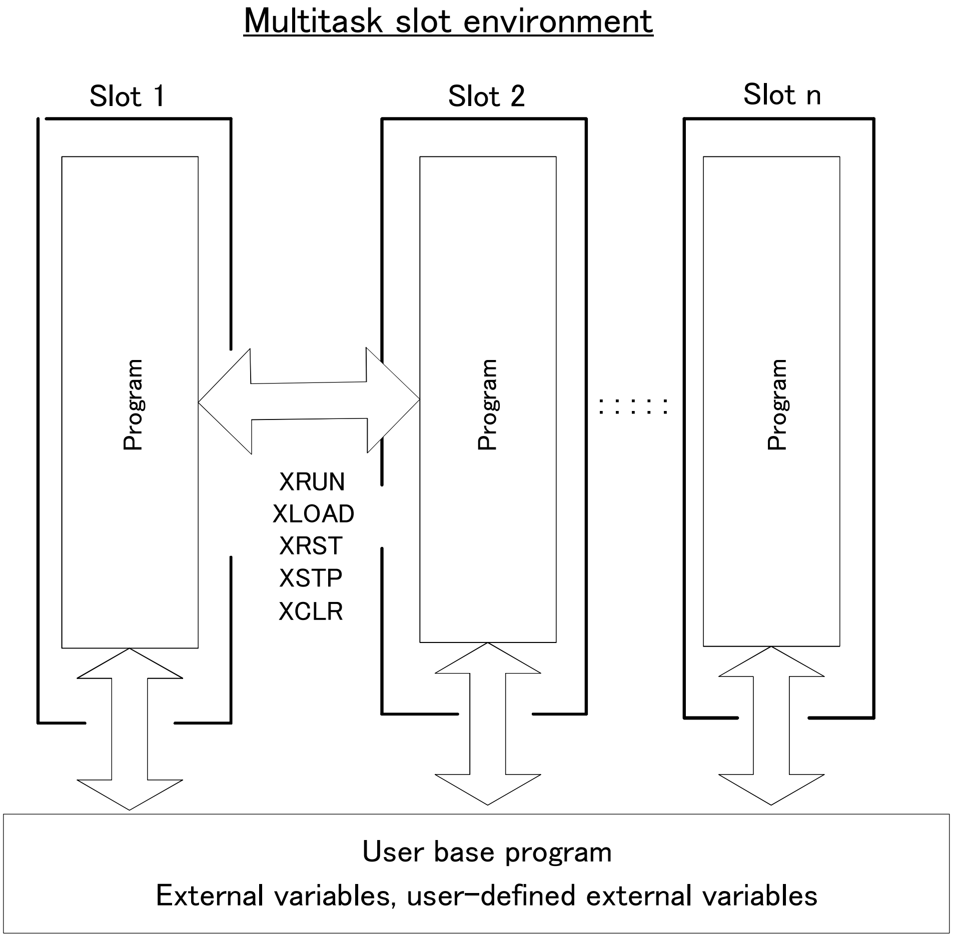


Fig.4-1:Multitask slot environment

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 controller's operation panel), the controller system unconditionally places the program selected with the operation panel in slot 1.

4.2.2 Executing a multitask

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

	Types of execution	Explanation
1	Execution from a program	This method starts parallel operation of the programs from a random position in the program using a MELFA-BASIC IV command. The programs to be run in parallel can be designated, and a program running in parallel can be stopped. This method is effective when selecting the programs to be run in parallel according to the program flow. The related commands include the XLOAD, XRUN, XSTP and XRST commands. Refer to "4.11 Detailed explanation of command words" on page 118 in this manual for details.
2	Execution from controller operation panel or external input/output signa	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 7 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.

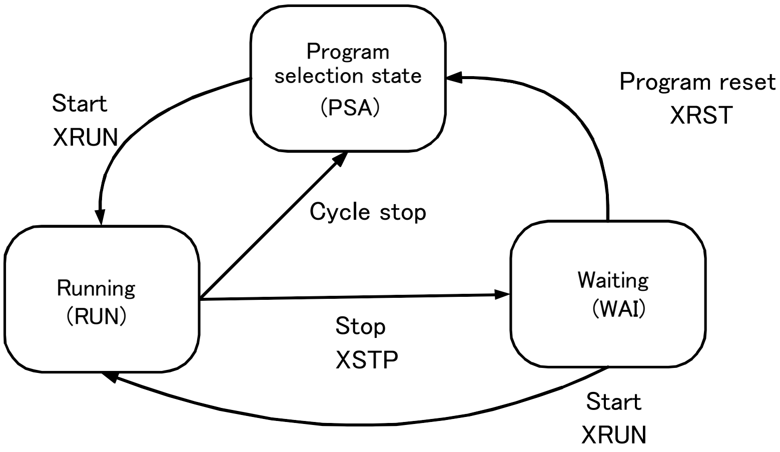


Fig.4-2:Operation state of each slot

<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 "5 Functions set with parameters" on page 306 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 "5 Functions set with parameters" on page 306 for details.
3. Starting conditions	START	START : Execution of a program using the START button on the operation panel or the I/O START signal	Select START when it is desired to start normally. 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. The operation mode (REP/CYC) is ignored.
		ERROR : Execution of a program when the controller is in error status	Specify ERROR when it is desired to execute a program in case an error occurs. It is not possible to execute instructions for moving the robot, such as the MOV instruction. The operation mode (REP/CYC) is one-cycle operation (CYC) regardless of the setting value.
4. Order of priority (number of lines executed in priority)	1	1 to 31: Number of lines executed at one time at multitask operation	If this number is increased, the number of lines executed at one time for the task slot is increased. For example, if 10 is specified for SLT1, 5 for SLT2, and 1 for SLT3, then after 10 lines of the program specified in SLT1 have been executed, five lines of the program specified in SLT2 are executed, and then one line of the program specified in SLT3 is executed. Afterward this cycle will be repeated.

Note1) The start operation conducted from the operation panel or by sending the dedicated input signal START will start the execution of programs of all the task slots whose start conditions are set to "START" at the same time.
To start the program independently, start in slot units with the dedicated input signal (S1 START to S32START). In this case, the line No. is preassigned to the same dedicated input/output parameter. Refer to "6.3 Dedicated input/output" on page 371 in this manual for details on the assignment of the dedicated input/output.

*Setting example
An example of the parameter settings for designating the following items in slot 2 is shown below.
Designation details) Program name : 5
Operation format : Continuous operation
Starting conditions : Always
Order of priority : 10
SLT2=5, REP, ALWAYS, 10

4.2.4 Precautions for creating multitask program

(1) Relationship between number of tasks and processing time

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

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

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

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

Data is passed between programs being executed in multitask operation via program external variables such as M_00 and P_00 (refer to "4.3.22 External variables" on page 104) and the user-defined external variables (refer to "4.3.24 User-defined external variables" on page 105). 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>
   10 M_00=0                          ; Substitute 0 in M_00
   20 IF M_00=0 THEN 20               ; Wait for M_00 value to change from 0.
   30 M_00=0                          ; Substitute 0 in M_00
   40 MOV P1                          ; Proceed with the target work.
   50 MOV P2
         :
   100 GOTO 20                        ; Repeat from line 20.
```

```
<Slot 2> (Program of signals and variables)
   10 IF M_IN(8) <> 1 THEN 30           ; Branch to line 30 if input signal 8 is not ON.
   20 M_00=1                            ; Substitute 1 in M_00
   30 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 "4.3.26 Robot status variables" on page 106 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 "4.11 Detailed explanation of command words" on page 118 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 even if the operation mode is set to CYC. Therefore, in order to perform the initialization processing, they should be programmed in such a way that the initialization processing is not executed more than once by jumping within the program using the GOTO instruction, etc.

Mechanism 1 is assigned to slot 1

In the default state, mechanism 1 (robot arm of standard system) is automatically assigned to slot 1. Because of this, slot 1 can execute the movement command even without acquiring mechanism 1 (without executing GETM command). However, when executing the movement command in a slot other than slot 1, the slot 1 mechanism acquisition state must be released (RELM command executed), and the mechanism must be acquired with the slot that is to execute the movement command (execute the GETM command).

4.2.5 Precautions for using a multitask program

(1) Starting the multitask

When starting from the operation panel or with the dedicated input signal START, the programs in all slots for which the "start request execution" is set in the slot parameter start conditions will start simultaneously.

When starting with the dedicated input signals S1START to S32START, the program can be started in each slot. In this case, the line No. is preassigned to the same dedicated input/output parameter. Refer to "6.3 Dedicated input/output" on page 371 for details on the assignment of the dedicated input/output.

(2) Display of operation status

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

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

For a detailed description of assignment of dedicated input/output, please refer to "6.3 Dedicated input/output" on page 371 of this manual.

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

4.2.6 Example of using multitask

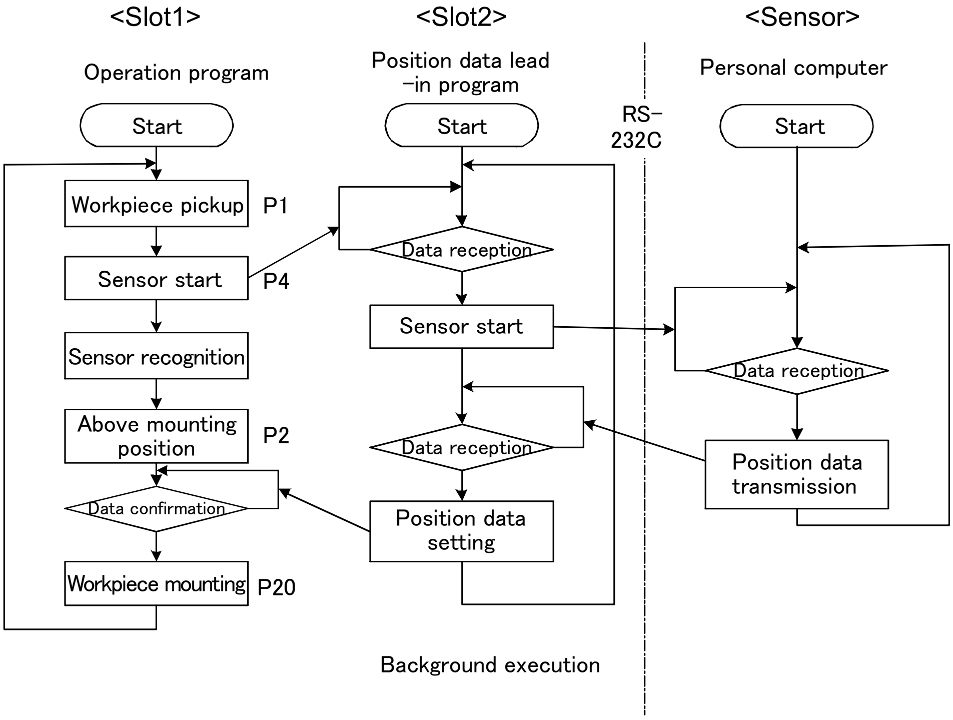
An example of the multitask execution is given in this section.

(1) Robot work details.

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

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

<Process flow>



P1: Workpiece pickup position (Vacuum timer DLY 0.05)

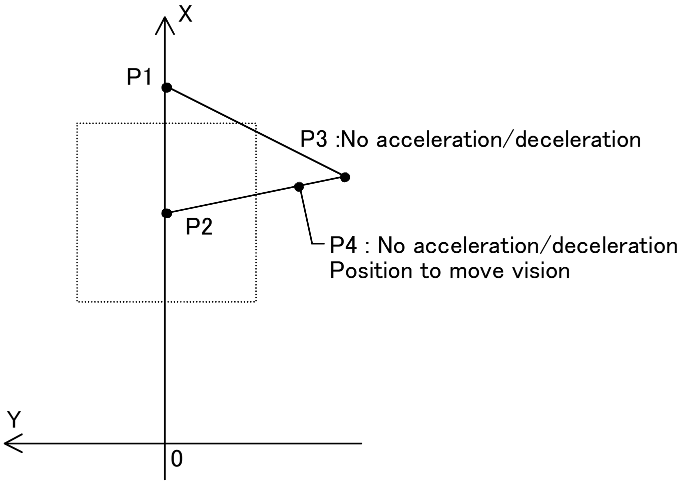
P2: Workpiece placing position (Release timer DLY 0.05)

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

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

P_01: Vision compensation data

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


X
P1
P3 :No acceleration/deceleration
P2
P4 : No acceleration/deceleration
Position to move vision
Y
0

(2) Procedures to multitask execution

*Procedure 1: Program creation

<1> Movement program (Program name: 1)

```
100 CNT 1                        'Validate path connected movement
110 MOV P2,10                    'Move to +10mm above P2
120 MOV P1,10                    'Move to +10mm above P1
130 MOV P1                       'Move to P1 workpiece pickup position
135 M_OUT(10)=0                  'Pickup workpiece
140 DLY 0.05                     'Timer 0.05 second
150 MOV P1,10                    'Move to +10mm above P1
160 MOV P3                       'Move to vision pre-position P3
165 SPD 500                      'Set linear speed to 500mm/sec.
170 MVS P4                       'Start vision lead-in with P4 passage
180 M_02#=0                      'Start data lead-in with background process at interlock variable
                                  (M_01=1/M_02=0)
190 M_01#=1                      'Start data load-in with background process
200 MVS P2,10                    'Move to +10mm above P2
210 IF M_02#=0 THEN GOTO 210     'Wait for interlock variable M_02 to reach 1
220 P20=P2*P_01                  'Add vision compensation P_01 to P20, and move to +10mm above
230 MOV P20,10                   'Move to +10mm above P20
240 MOV P20                      'Go to P20 workpiece placing position
245 M_OUT(10)=1                  'Place workpiece
250 DLY 0.05                     'Timer 0.05 second
260 MOV P20,10                   'Move to +10mm above P20
270 CNT 0                        'Invalidate path connected movement
280 END                          'End one cycle
```

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

```
100 IF M_01#=0 THEN GOTO 100     'Wait for interlock variable M_01 to reach 1
105 OPEN "COM1:" AS #1           'Open RS-232-C line
110 DLY M_03#                    'Hypothetical process timer (0.05 second)
115 PRINT #1,"SENS"              'Transmit character string "SENS" to RS-232-C (vision side)
117 INPUT #1,M1,M2,M3            'Wait to lead-in vision compensation value (relative data)
120 P_01.X=M1                    'Substitute delta X coordinate
130 P_01.Y=M2                    'Substitute delta Y coordinate
140 P_01.Z=0.0                   '
150 P_01.A=0.0                   '
160 P_01.B=0.0                   '
170 P_01.C=RAD(M3)               'Substitute delta C coordinate
175 CLOSE                        'Close RS-232-C line
180 M_01#=0                      'Interlock variable M_01 = 0
190 M_02#=1                      'Interlock variable M_02 = 0
200 END                          'End process
```

*Procedure 2: Setting the slot parameters

Set the slot parameters as shown below.

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

*Procedure 3: Reflecting the slot parameters

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

*Procedure 4: Starting

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

4.3 Detailed specifications of MELFA-BASIC IV

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

(1) Program name

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

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

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

(2) Command statement

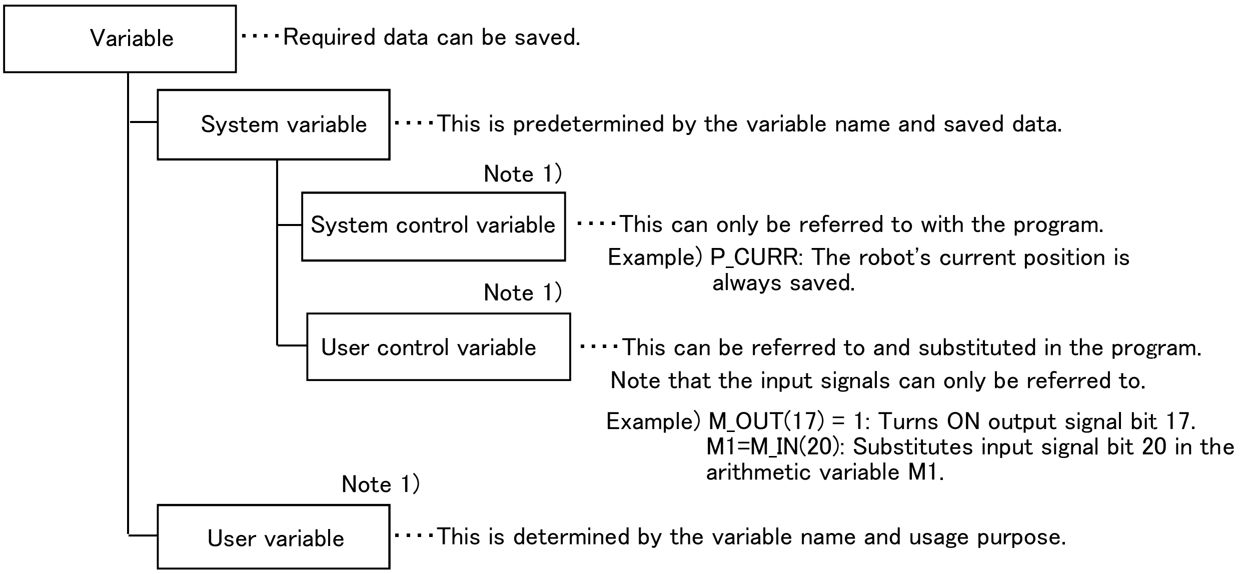
Example of constructing a statement

```
10   MOV   P1    WTH M_OUT(17)=1
1)     2)    3)       4)
```

1) Line No. : Numbers for determining the order of execution within the program. Lines 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.



Variable ····Required data can be saved.

System variable ····This is predetermined by the variable name and saved data.

Note 1) System control variable ····This can only be referred to with the program.
Example) P_CURR: The robot's current position is always saved.

Note 1) User control variable ····This can be referred to and substituted in the program.
Note that the input signals can only be referred to.
Example) M_OUT(17) = 1: Turns ON output signal bit 17.
M1=M_IN(20): Substitutes input signal bit 20 in the arithmetic variable M1.

Note 1) User variable ····This is determined by the variable name and usage purpose.

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

Position type variable ····The robot's orthogonal coordinate value is saved. The variable name starts with ″P″.
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 221, " WTH (With)" or Page 222, " WTHIF (With If)", as well as each of the movement instructions (MOV, MVS, MVR, MVR2, MVR3, MVC, MVA) for detailed descriptions.

4.3.3 Line

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

One line can have up to 127 characters. (This does not include the last character of the line.)

Only one command statement per line

Multiple command statements cannot be separated with a semicolon and described on one line as done with the general BASIC.

4.3.4 Line No.

Line Nos. should be in ascending order, starting from the first line, in order for the program to run properly. When a program is stored in the memory, it is stored in the order of the line Nos.

Line Nos. can be any integer from 1 to 32767.

Direct execution if line No. is not assigned

If an instruction statement is described without a line 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 line No. The head of the label must be an alphabetic character, and the entire label must be within eight characters long. If a label starting with the alphabetic character L is described after the asterisk, an underscore (_) can be used immediately after the character.

* Characters that cannot be used in labels:

•Reserved words (DLY, HOPEN, etc.)

•Any name that begins with a symbol or numeral

•Any name that is already used for a variable name or function name

```
Example)  10   GOTO   *LBL
          100  *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 4-3. 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 4-3: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	@Note1)	@Note 1)
Numerals	0 1 2 3 4 5 6 7 8 9	O	@Note2)	O
Symbols	" ' & ( ) * + - . , / : ; = < > ? @ ` [\ ] ^ { } ~ |	X	X	X
	! # $ %	X	Available for type specification	X
	_(Underscore)	X	@Note3)	@Note4)
Spaces	Space character	X	X	X

Note1) Alphanumerics in variable names and label names in the program are automatically converted into uppercase characters.

Note2) Only alphabetical characters can be used as the first character of the variable name. Numerals can be used as the second and succeeding characters.

Note3) They can be used as the second and succeeding characters. Any variable having an underscore (_) as the second character becomes an external variable.

Note4) If an underscore (_) is used in a label name, start with an asterisk (*) followed by alphabetic character "L."

Refer to Page 93, "(1) Program name" for detail of program names, refer to Page 101, "4.3.15 Variables" for detail of variable names, and refer to Page 95, "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 104, "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 lines. When assigned at the head of a character it is a substitute for the REM statement.

```
Example)  100 MOV P1 'GET          ;GET will be set as the comment.
          150 '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) 200 *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.

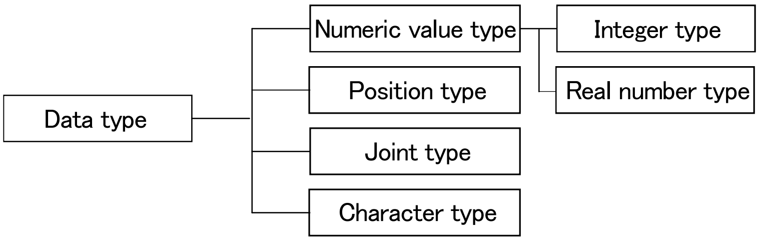
Example) M1 = P2.X   ; Substitute the position variable P2.X coordinate element in numeric variable M1.

(7) Space

The space character, when used as part of a character string constant or within a comments line, is interpreted as a character. The space character is required as a delimiter immediately after a line No. or a command word, and between data items. In the [Format] given in section Page 118, "4.11 Detailed explanation of command words", the space is indicated with a "[] " where required.

4.3.8 Data type

In MELFA BASIC IV 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.

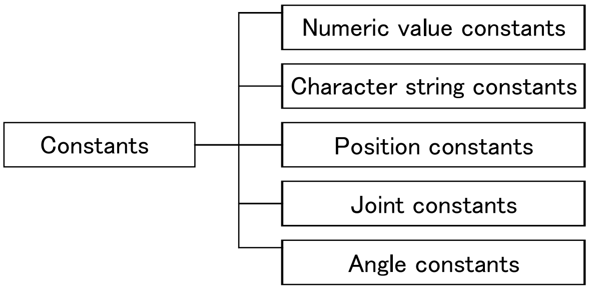


Example)

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 (").

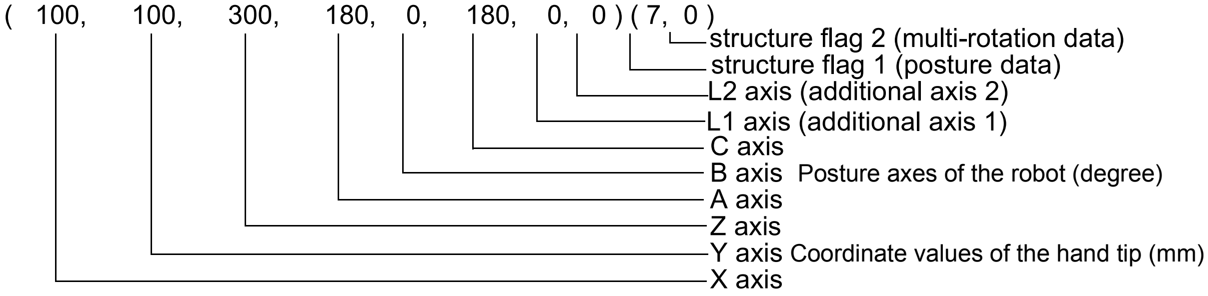
Example) "ABCDEFGHIJKLMN" "123"

Up to 127 characters for character string

The character string can have up to 127 characters, including the line No. and double quotations. Enter two double quotation marks successively in order to include the double quotation mark itself in a character string. For the character string AB"CD, input "AB""CD".

4.3.12 Position constants

The syntax for position constants is as shown below. Variables cannot be described within position constants.


(100, 100, 300, 180, 0, 180, 0, 0) (7, 0)
structure flag 2 (multi-rotation data)
structure flag 1 (posture data)
L2 axis (additional axis 2)
L1 axis (additional axis 1)
C axis
B axis Posture axes of the robot (degree)
A axis
Z axis
Y axis Coordinate values of the hand tip (mm)
X axis

Example)

P1=(300, 100, 400, 180, 0, 180, 0, 0) (7, 0)
P2=(0, 0, -5, 0, 0, 0) (0, 0) [A case where there is no traveling axis data]
P3=(100, 200, 300, 0, 0, 90) (4, 0) [A case of a 4-axis horizontal multi-joint robot]

(1) Coordinate, posture and additional axis data types and meanings

[Format] X, Y, Z, A, B, C , L1, L2

[Meaning] X, Y, Z: coordinate data. The position of the tip of the robot's hand in the XYZ coordinates. (The unit is mm.)

A, B, C: posture data. This is the angle of the posture. (The unit is deg.) Note1)

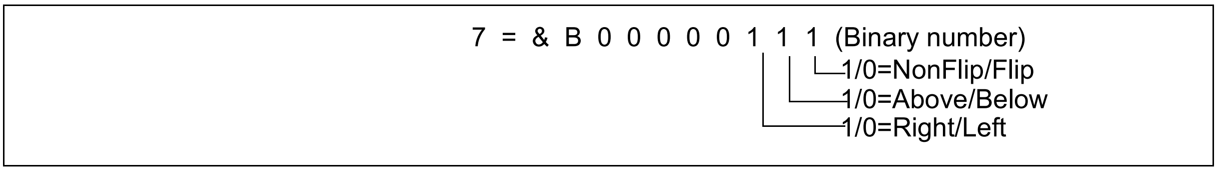
L1, L2: additional axis data. These are the coordinates for additional axis 1 and additional axis 2, respectively. (The unit is mm or deg.)

Note1) The T/B and Personal computer support software display the unit in deg; however, the unit of radian is used for substitution and calculation in the program.

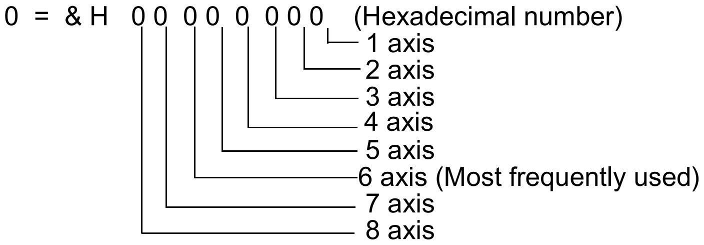
(2) Meaning of structure flag data type and meanings

[Format] FL1, FL2

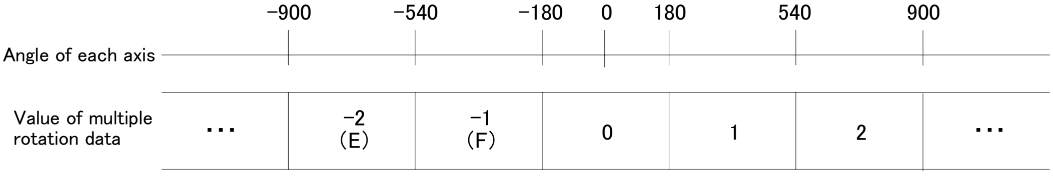
[Meaning] FL1: Posture data



FL2: Multi-rotation data information - Default value = 0 (The range is 0 to +4294967295 ... Information for eight axes is held with a 1-axis 4-bit configuration.)Two types of screens are available for the PC: screens that display the number of rotations for each axis (-8 to 7) in decimal and those that display the number of rotations for each axis in hexadecimal..


0 = & H 0 0 0 0 0 0 0 0 (Hexadecimal number)
1 axis
2 axis
3 axis
4 axis
5 axis
6 axis (Most frequently used)
7 axis
8 axis

Value of multiple rotation data



Designation of axis No.

1. There is no need to describe the coordinate and posture data for all eight axes. However, if omitted, the following axis data will be processed as undefined.
   For a 4-axis robot (X,Y,Z,C axis configuration), describe as (X, Y, Z, , , C) or (X,Y,Z,0,0,C).
2. To omit all axes,insert at least one ","(comma), such as (,).

Use of variables in position element data

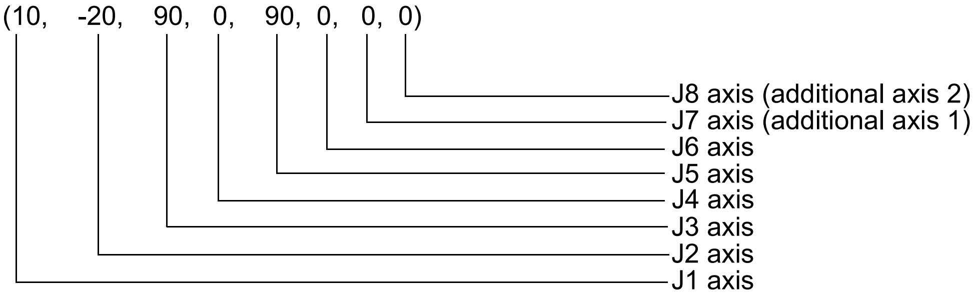
The coordinate, position, additional axis data and structure flag data are called the position element data. A variable cannot be contained in the position element data that configures the position constant.

Omitting the structure flag data

If the structure flag data is omitted, the default value will be applied.((7,0) Varies depending on the machine model.)

4.3.13 Joint constants

The syntax for the joint constants is as shown below



Example)

6 axis robot	J1 = (0, 10, 80, 10, 90, 0)
6 axis + Additional axis	J1 = (0, 10, 80, 10, 90, 0, 10, 10)
5 axis robot	J1 = (0, 10, 80, 0, 90, 0)
5 axis + Additional axis	J1 = (0, 10, 80, 0, 90, 0, 10, 10)
4 axis robot	J1 = (10, 20, 90, 0)
4 axis + Additional axis	J1 = (10, 20, 90, 0, , , 10, 10)

(1) Axis data format and meanings

[Format] J1,J2,J3,J4,J5,J6,J7,J8

[Meaning] J1 to J6: Robot axis data (Unit is mm or deg.)
J7, J8: Additional axis data, and may be omitted (optional).
(Unit is mm or deg. Depending on the parameter setting.
The unit is mm, not degrees, if the J3 axis of a horizontal multi-joint type robot is a direct-driven axis.

Use of variables in joint element data

The axis data is called the joint element data.
A variable cannot be contained in the joint constant data that configures the joint constant.

4.3.14 Angle value

The angle value is used to express the angle in "degrees" and not in "radian".

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

Example) SIN(90DEG)----A 90 degree sine is indicated.

4.3.15 Variables

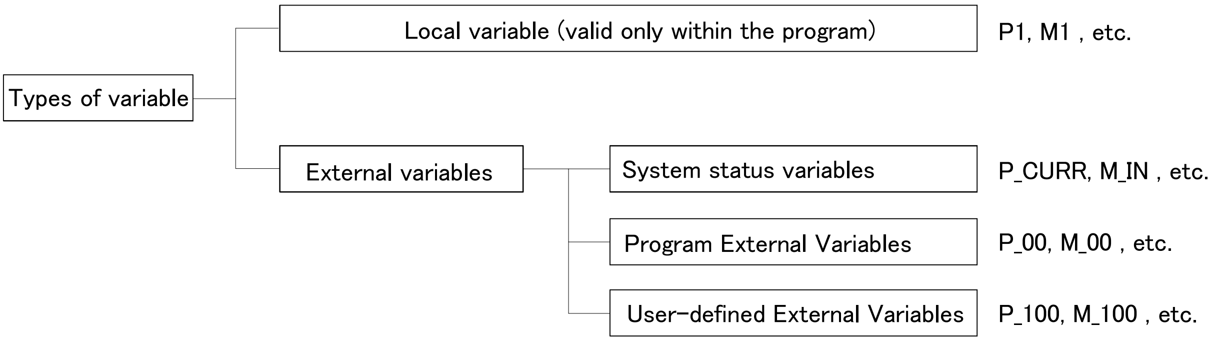
A variable name should be specified using up to eight characters.

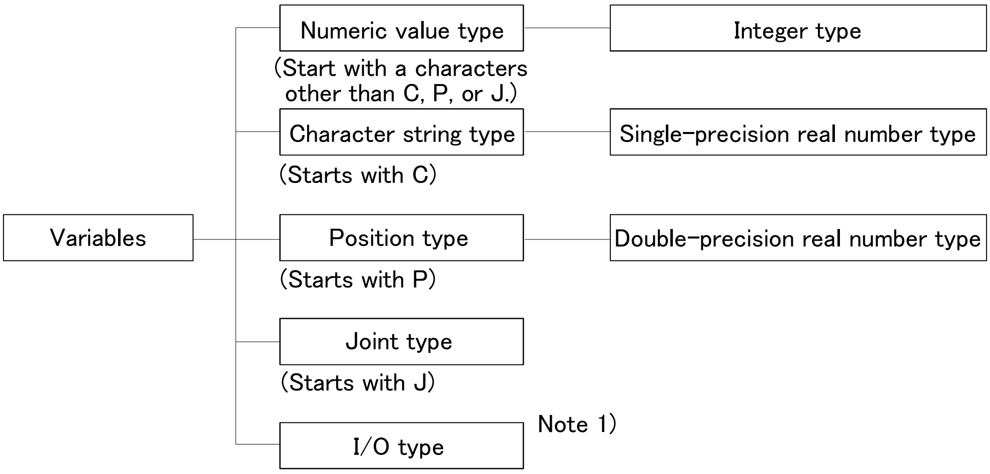
The variable types include the numeric value type, character string type, position type, joint type and I/O type. Each is called a "variable type". The variable type is determined by the head character of the identifier (variable name).

The numeric value type can be further classified as integer type, single-precision real number type, or double-precision real number type.

The following two types of data valid ranges are used.

1. Local variable valid only in one program
2. Robot status variable, program external variable and user-defined external variable valid over programs. (The user-defined external variable has a _ for the second character of the variable name. Refer to Page 104, "4.3.22 External variables" for details.)





Note 1) The identifiers include those determined by the robot status variable (M_IN,M_OUT, etc.), and those declared in the program with the DEFIO command.

Variables are not initialized

The variables will not be cleared to zero when generated, when the program is loaded, or when reset.

4.3.16 Numeric value variables

Variables whose names begin with a character other than P, J, or C are considered numeric value variables. In MELFA-BASIC IV, it is often specified that a variable is an numeric value variable by placing an M at the head. M is the initial letter of mathematics.

```
Example) M1 = 100
         M2! = -1.73E+10
         M3# = 0.123
         ABC = 1
```

1) It is possible to define the type of variable by attaching an numeric value type indicator at the end of the variable name. If it is omitted, the variable type is assumed to be of the single-precision real number type.

Numeric value type suffix	Meaning
%	Integer
!	Single-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 the value is exceeded during a single precision = double precision execution, an error will occur.

Table 4-4:Range of numeric value variable data

Type	Range	
Integer type	-32768 to 32767	
Single-precision real number type	-3.40282347e+38 to 3.40282347e+38	Note) E expresses a power of 10.
Double-precision real number type	-1.7976931348623157e+308 to 1.7976931348623157e+308	

4.3.17 Character string variables

A character string variable should start with C and end with "$." If it is defined by the DEF CHAR instruction, it is possible to specify a name beginning with a character other than C.

```
Example) C1$ = "ABC"
         CS$ = C1$
         DEF CHAR MOJI
         MOJI = "MOJIMOJI"
```

4.3.18 Position variables

Variables whose names begin with character P are considered position variables. If it is defined by the DEF POS instruction, it is possible to specify a name beginning with a character other than P. It is possible to reference individual coordinate data of position variables. In this case, add "." and the name of a coordinate axis, e.g. "X," after the variable name.

P1.X, P1.Y, P1.Z, P1.A, P1.B P1.C, P1.L1, P1.L2

The unit of the angular coordinate axes A, B, and C is radians. Use the DEG function to convert it to degrees.

```
Example) P1 = PORG
         DIM P3(10)
         M1 = P1. X          (Unit : mm)
         M2 = DEG(P1. A)  (Unit : degree)
         DEG  POS  L10
         MOV  L10
```

4.3.19 Joint variables

A character string variable should start with J. If it is defined by the DEF JNT instruction, it is possible to specify a name beginning with a character other than J.
It is possible to reference individual coordinate data of joint variables.
In this case, add "." and the name of a coordinate axis, e.g. "J1," after the variable name.

JDATA.J1, JDATA.J2, JDATA.J3, JDATA.J4, JDATA.J5, JDATA.J6, JDATA.J7, JDATA.J8

The unit of the angular coordinate axes A, B, and C is radians. Use the DEG function to convert it to degrees.

```
Example) JSTARAT = ( 0, 0, 90, 0, 90, 0, 0, 0 )
         JDATA = JSTART
         DIM J3 (10)
         M1 = J1.J1         (Unit : radian)
         M2 = DEG (J1.J2)
         DEF JNT K10
         MOV K 10
```

4.3.20 Input/output variables

The following types of input/output variables are available. They are provided beforehand by the robot status variables.

Input/output variables name	Explanation
M_IN	For referencing input signal bits
M_INB	For referencing input signal bytes (8-bit signals)
M_INW	For referencing input signal words (16-bit signals)
M_OUT	For referencing/assigning output signal bits
M_OUTB	For referencing/assigning output signal bytes (8-bit signals)
M_OUTW	For referencing/assigning output signal words (16-bit signals)
M_DIN	For referencing input registers for CC-Link
M_DOUT	For referencing output registers for CC-Link

Please refer to the robot status variables Page 248, " M_IN/M_INB/M_INW", Page 254, " M_OUT/M_OUTB/M_OUTW", and Page 244, " 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 98, "4.3.8 Data type". The following three types of external variables are available.

Table 4-5: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 4-6 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 4-6:Program external variables

Data type	Variable name	Qty.	Remarks
Position	P_00 to P_19 P_20 to P_39 Note)	20 20	

Position array (No. of elements 10)	P_100( ) to P_104( )
	P_105( ) to P_109( ) Note)
		5
		5
			Use the array element in the first dimensions.

Joint	J_00 to J_19 J_20 to J_39 Note)	20 20	

Joint array (No. of elements 10)	J_100( ) to J_104( )
	J_105( ) to J_109( ) Note)
		5
		5
			Use the array element in the first dimensions.

Numeric value	M_00 to M_19 M_20 to M_39 Note)	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( ) Note)
		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 Note)	20 20	

Character string array
(No. of elements 10)
	C_100( ) to C_104( )
	C_105( ) to C_109( ) Note)
		5
		5
			Use the array element in the first dimensions.

Note) The software version of the controller is J1 or later, the program external variable was extended. 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 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 single variables again.

Example) Example of using user-defined external variables

On the main program (program name 1) side

```
10 DIM P_100(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
```

On the user base program (program name UBP) side

```
10 DEF POS P_900, P_901, P_902, P_903
20 DIM P_100(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 written when user-defined external variables are to be used, but it is not necessary to 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.

4.3.26 Robot status variables

The available robot status variables are shown in Table 4-7. 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 4-7: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	268
2	J_CURR	Mechanism No.(1 to 3)	Current position (joint)	R	Joint type	234
3	J_ECURR	Mechanism No.(1 to 3)	Current encoder pulse position	R	Joint type	236
4	J_FBC	Mechanism No.(1 to 3)	Joint position generated based on the feedback value from the servo	R	Joint type	237
5	J_AMPFBC	Mechanism No.(1 to 3)	Current feedback value	R	Joint type	237
6	P_FBC	Mechanism No.(1 to 3)	XYZ position generated based on the feedback value from the servo	R	Position type	269
7	M_FBD	Mechanism No.(1 to 3)	Distance between commanded position and feedback position	R	Position type	246
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	239
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	241
10	P_TOOL	Mechanism No.(1 to 3)	Currently designated tool conversion data	R	Position type	270
11	P_BASE	Mechanism No.(1 to 3)	Currently designated base conversion data	R	Position type	266
12	P_NTOOL	Mechanism No.(1 to 3)	System default value (tool conversion data)	R	Position type	270
13	P_NBASE	Mechanism No.(1 to 3)	System default value (base conversion data)	R	Position type	266
14	M_TOOL	Mechanism No.(1 to 3)	Tool No. (1 to 4)	RW	Integer type	261
15	J_COLMXL	Mechanism No.(1 to 3)	Difference between estimated torque and actual torque	R	Joint type, %	235
16	M_COLSTS	Mechanism No.(1 to 3)	Impact detection status (1: Colliding, 0: Others)	R	Integer type	242
17	P_COLDIR	Mechanism No.(1 to 3)	Movement direction at collision	R	Position type	267
18	M_OPOVRD	None	Speed override on the operation panel (0 to 100%)	R	Integer type, %	249
19	M_OVRD	Slot No.(1to 32)	Override in currently designated program (0 to 100%)	R	Integer type, %	249
20	M_JOVRD	Slot No.(1to 32)	Currently designated joint override (0 to 100%)	R	Integer type, %	249
21	M_NOVRD	Slot No.(1to 32)	System default value (default value of M_OVRD) (%)	R	Single-precision real number type, %	249
22	M_NJOVRD	Slot No.(1to 32)	System default value (default value of M_JOVRD) (%)	R	Single-precision real number type, %	249
23	M_WUPOV	Mechanism No.(1 to 3)	Warm-up operation override (50 to 100%)	R	Single-precision real number type, %	263
24	M_WUPRT	Mechanism No.(1 to 3)	Time until the warm-up operation status is canceled (sec.)	R	Single-precision real number type, sec	264
25	M_WUPST	Mechanism No.(1 to 3)	Time until the warm-up operation status is set again (sec.)	R	Single-precision real number type, sec	265
26	M_RATIO	Slot No.(1to 32)	Fraction of the current movement left before reaching the target position (%)	R	Integer type, %	255
27	M_RDST	Slot No.(1to 32)	Remaining distance left of the current movement (only the three dimensions of X, Y, and Z are taken into consideration: mm)	R	Single-precision real number type, mm	256

No	Variable name	Array designation Note1)	Details	Attribute Note2)	Data type, Unit	Page
28	M_SPD	Slot No.(1to 32)	Current specified speed (valid only for linear/ circular interpolation)	R	Single-precision real number type, mm/s	259
29	M_NSPD	Slot No.(1to 32)	System default value (default value of M_SPD) (mm/s)	R	Single-precision real number type, mm/s	259
30	M_RSPD	Slot No.(1to 32)	Current directive speed (mm/s)	R	Single-precision real number type,mm/s	259
31	M_ACL	Slot No.(1to 32)	Current specified acceleration rate (%)	R	Single-precision real number type, %	238
32	M_DACL	Slot No.(1to 32)	Current specified deceleration rate (%)	R	Single-precision real number type, %	238
33	M_NACL	Slot No.(1to 32)	System default value (default value of M_ACL) (%)	R	Single-precision real number type, %	238
34	M_NDACL	Slot No.(1to 32)	System default value (default value of M_DACL) (%)	R	Single-precision real number type, %	238
35	M_ACLSTS	Slot No.(1to 32)	Current acceleration/deceleration status 0 = Stopped, 1 = Accelerating, 2 = Constant speed, 3=Decelerating	R	Integer type	238
36	M_SETADL	Axis No.(1 to 8)	Specify the acceleration/deceleration time ratio (%) of each axis. Software version J1 or later	RW	Single-precision real number type, %	257
37	M_LDFACT	Axis No.(1 to 8)	The load factor of the servo motor of each axis. (%) Software version J1 or later	R	Single-precision real number type, %	250
38	M_RUN	Slot No.(1to 32)	Operation status (1: Operating, 0: Not operating)	R	Integer type	256
39	M_WAI	Slot No.(1to 32)	Pause status (1: Pausing, 0: Not pausing)	R	Integer type	262
40	M_PSA	Slot No.(1to 32)	Specifies whether or not the program selection is possible in the specified task slot. (1: Selection possible, 0: Selection not possible, in pause status)	R	Integer type	255
41	M_CYS	Slot No.(1to 32)	Cycle operation status (1: Cycle operation, 0: Non-cycle operation)	R	Integer type	243
42	M_CSTP	None	Cycle stop operation status (1: Cycle stop, 0: Not cycle stop)	R	Integer type	243
43	C_PRG	Slot No.(1to 32)	Execution program name	R	Character string type	232
44	M_LINE	Slot No.(1to 32)	Currently executed line No.	R	Integer type	250
45	M_SKIPCQ	Slot No.(1to 32)	A value of 1 is input if execution of an instruction is skipped as a result of executing the line that includes the last executed SKIP command, otherwise a value of 0 is input.	R	Integer type	257
46	M_BRKCQ	None	Result of the BREAK instruction (1: BREAK, 0: None)	R	Integer type	239
47	M_ERR	None	Error occurring (1: An error has occurred, 0: No errors have occurred)	R	Integer type	245
48	M_ERRLVL	None	Reads an error level. caution/low/high1/high2ÅÅ1/2/3/4	R	Integer type	245
49	M_ERRNO	None	Reads an error number.	R	Integer type	245
50	M_SVO	Mechanism No.(1 to 3)	Servo motor power on (1: Servo power on, 0: Servo power off)	R	Integer type	259
51	M_UAR	Mechanism No.(1 to 3)	Bit data. (1: Within user specified area, 0: Outside user specified area) (Bit 0:area 1 to Bit 7:area 8)	R	Integer type	262

No	Variable name	Array designation Note1)	Details	Attribute Note2)	Data type, Unit	Page
52	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	248
53	M_INB	Input No.(0 to 32767)	Use this variable when inputting external input signals (8-bit units) General-purpose bit device: byte signal input The signal numbers will be 6000s for CC-Link	R	Integer type	248
54	M_INW	Input No.(0 to 32767)	Use this variable when inputting external input signals (16-bit units) General-purpose bit device: word signal input The signal numbers will be 6000s for CC-Link	R	Integer type	248
55	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	254
56	M_OUTB	Output No.(0 to 32767)	Use this variable when outputting external output signals (8-bit units) General-purpose bit device: byte signal input The signal numbers will be 6000s for CC-Link	RW	Integer type	254
57	M_OUTW	Output No.(0 to 32767)	Use this variable when outputting external output signals (16-bit units) General-purpose bit device: word signal input The signal numbers will be 6000s for CC-Link	RW	Integer type	254
58	M_DIN	Input No.(from 6000)	CC-Link's remote register: Input register	R	Integer type	244
59	M_DOUT	Output No.(from 6000)	CC-Link's remote register: output register	RW	Integer type	244
60	M_HNDCQ	Input No.(1 to 8)	Returns a hand check input signal.	R	Integer type	247
61	P_SAFE	Mechanism No.(1 to 3)	Returns an safe point position.	R	Position type	269
62	J_ORIGIN	Mechanism No.(1 to 3)	Returns the joint coordinate data when setting the origin.	R	Joint type	237
63	M_OPEN	File No.(1 to 8)	Returns the open status of the specified file or the communication line.	R	Integer type	253
64	C_MECHA	Slot No.(1 to 32)	Returns the type name of the robot.	R	Character string type	232
65	C_MAKER	None	Shows manufacturer information (a string of up to 64 characters).	R	Character string type	231
66	C_USER	None	Returns the content of the parameter "USERMSG."(a string of up to 64 characters).	R	Character string type	233
67	C_DATE	None	Current date expressed as "year/month/date".	R	Character string type	231
68	C_TIME	None	Current time expressed as "time/minute/second".	R	Character string type	233
69	M_BTIME	None	Returns the remaining battery capacity time (hours).	R	Integer type, Time	239
70	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	260
71	P_ZERO	None	A variable whose position coordinate values (X, Y, Z, A, B, C, FL1, FL2) are all 0	R	Position type	270
72	M_PI	None	Circumference rate (3.1415...)	R	Double-precision real number type	254
73	M_EXP	None	Base of natural logarithm (2.71828...)	R	Double-precision real number type	245
74	M_G	None	Specific gravity constant (9.80665)	R	Double-precision real number type	246
75	M_ON	None	1 is always set	R	Integer type	252

No	Variable name	Array designation Note1)	Details	Attribute Note2)	Data type, Unit	Page
76	M_OFF	None	0 is always set	R	Integer type	252
77	M_MODE	None	Contains the status of the key switch of the operation panel TEACH/AUTO(OP)/AUTO(Ext.)(1/2/3)	R	Integer type	251

Note1) Mechanism No. .... 1 to 3, Specifies a mechanism number corresponding to the multitask processing function.
Slot No.......................... 1 to 32, Specifies a slot number corresponding to the multitask function.
Input No........................ 0 to 32767: (theoretical values). Specifies a bit number of an input signal.
Output No. .... 0 to 32767: (theoretical values). Specifies a bit number of an output signal.
Note2) R .... Only reading is possible.
RW............................... Both reading and writing are possible.

4.4 Logic numbers

Logic numbers indicate the results of such things as comparison and input/output.
If not 0 when evaluated with an Integer, then it is true, and if 0, it is false. When substituted, if true, 1 is assigned. The processes that can use logic numbers are shown in Table 4-8.

Table 4-8:Values corresponding to true or false logic number

Items expressed with logic number "1"	Items expressed with logic number "0"
*Result of cmparison operation (if true)	*Result of comparison operation (if false)
*Result of logic operation (if true)	*Result of logic operation (if false)
*Switch ON	*Switch OFF
*Input/output signal ON	*Input/output signal OFF
*Hand open (supply current to the hand)	*Hand close (do not supply current to the hand)
*Settings for enable/valid such as for interrupts	*Settings for disable/invalid such as for interrupts

4.5 Functions

A function carries out a specific operation for an assigned argument, and returns the result as a numeric value type or character string type. There are built-in functions, that are preassembled, and user-defined functions, defined by the user.

(1) User-defined functions

The function is defined with the DEF FN statement.

Example) DEF 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 4-9.

Table 4-9:List of built-in functions

Class	Function name (format)	Functions	Page	Result
Numeric functions	ABS (<Numeric expression>)	Produces the absolute value	272	Numeric value
	CINT (<Numeric expression>)	Rounds off the decimal value and converts into an integer.	277	
	DEG (<Numeric expression:radian>)	Converts the angle unit from radian (rad) to degree (deg).	280	
	EXP (<Numeric expression>)	Calculates the value of the expression's exponential function	281	
	FIX (<Numeric expression>)	Produces an integer section	282	
	INT (<Numeric expression>)	Produces the largest integer that does not exceed the value in the expression.	284	
	LEN(<Character string expression>)	Produces the length of the character string.	286	
	LN (<Numeric expression>)	Produces the logarithm.	287	
	LOG (<Numeric expression>)	Produces the common logarithm.	287	
	MAX (<Numeric expression>...)	Obtains the max. value from a random number of arguments.	288	
	MIN (<Numeric expression>...)	Obtains the min. value from a random number of arguments.	289	
	RAD (<Numeric expression: deg.>)	Converts the angle unit from radian (rad) to degree (deg).	293	
	SGN (<Numeric expression>)	Checks the sign of the number in the expression	300	
	SQR (<Numeric expression>)	Calculates the square root	301	
	STRPOS(<Character string expression>, <Character string expression>)	Obtains the 2nd argument character string position in the 1st argument character string.	301	
	RND (<Numeric expression>)	Produces the random numbers.	295	
	ASC(<Character string expression>)	Provides a character code for the first character of the character string in the expression.	274	
	CVI(<Character string expression>)	Converts a 2-byte character string into integers.	279	
	CVS(<Character string expression>)	Converts a 4-byte character string into a single-precision real number.	279	
	CVD(<Character string expression>)	Converts an 8-byte character string into a double-precision real number.	280	
	VAL(<Character string expression>)	Converts a character string into a numeric value.	303	
Trigonometric functions	ATN(<Numeric expression>)	Calculates the arc tangent. Unit: radian Definition range: Numeric value, Value range: -PI/2 to +PI/2	274	Numeric value
	ATN2(<Numeric expression>,<Numeric expression>)	Calculates the arc tangent. Unit: radian THETA=ATN2(delta y, deltax) Definition range: Numeric value of delta y or delta x that is not 0 Value range: -PI to +PI	274	

Class	Function name (format)	Functions	Page	Result
Trigonometric functions	COS(<Numeric expression>)	Calculates the cosine Unit: radian Definition range: Numeric value range, Value range: -1 to +1	278	Numeric value
	SIN(<Numeric expression>)	Calculates the sine Unit: radian Definition range: Numeric value range, Value range: -1 to +1	300	
	TAN(<Numeric expression>)	Calculates the tangent. Unit: radian Definition range: Numeric value range, Value range: Range of numeric value	302	
Character string functions	BIN$(<Numeric expression>)	Converts numeric expression value into binary character string.	275	Character string
	CHR$(<Numeric expression>)	Provides character having numeric expression value character code.	277	
	HEX$(<Numeric expression>)	Converts numeric expression value into hexadecimal character string.	284	
	LEFT$(<Character string expression>,<Numeric expression>)	Obtains character string having length designated with 2nd argument from left side of 1st argument character string.	286	
	MID$(<Character string expression>,<Numeric expression> <Numeric expression>)	Obtains character string having length designated with 3rd argument from the position designated with the 2nd argument in the 1st argument character string.	288	
	MIRROR$(<Character string expression>)	Mirror reversal of the character string binary bit is carried out.	289	
	MKI$(<Numeric expression>)	Converts numeric expression value into 2-byte character string.	290	
	MKS$(<Numeric expression>)	Converts numeric expression value into 4-byte character string.	290	
	MKD$(<Numeric expression>)	Converts numeric expression value into 8-byte character string.	291	
	RIGHT$(<Character string expression>,<Numeric expression>)	Obtains character string having length designated with 2nd argument from right side of 1st argument character string.	295	
	STR$(<Numeric expression>)	Converts the numeric expression value into a decimal character string.	302	
	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.	278	Numeric value
Position variables	DIST(<Position>,<Position>)	Obtains the distance between two points.	281	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.	283	
	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.	293	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)	296	
	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).	294	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.	297	
	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.	273	
	INV(<Position>)	Obtains the reverse matrix.	285	Position
	PTOJ(<Position>)	Converts the position data into joint data.	292	Joint
	JTOP(<Position>)	Converts the joint data into position data.	285	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	304	Numeric value

Class	Function name (format)	Functions	Page	Result
Position variables	ZONE2 (<Position 1>,<Position 2>,<Position 3> <Numeric value1>, <Numeric value2>, <Numeric value3>,<Position 4>)	Checks whether position 1 is within the space (cylinder) created by the position 2 and position 3 points. Outside the range=0, Within the range=1 Only the X, Y, and Z coordinate values are considered; the A, B, and C posture data is ignored.	305	Numeric value
	POSCQ(<Position>)	Checks whether <position> is within the movement range.	291	Numeric value
	POSMID (<Position1>,<Position2>, <Numeric value1>, <Numeric value2>)	Calculates the middle position between <position 1> and <position 2>.	292	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>.	276	Numeric value
	SETJNT (<J1 axis>,<J2 axis>,<J3 axis>,<J4 axis> <J5 axis>,<J6 axis>,<J7 axis>,<J8 axis>)	Sets values in joint variables.	298	Joint
	SETPOS (<X axis>,<Y axis>,<Z axis>,<A axis> <B axis>,<C axis>,<L1 axis>,<L2 axis>)	Sets values in position variables.	299	Position

4.6 List of Instructions

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

(1) Instructions related to movement control

Command	Explanation	Page
Page 180, "MOV (Move)"	Joint interpolation	180
Page 190, "MVS (Move S)"	Linear interpolation	190
Page 184, "MVR (Move R)"	Circular interpolation	184
Page 186, "MVR2 (Move R2)"	Circular interpolation 2	186
Page 188, "MVR3 (Move R 3)"	Circular interpolation 3	188
Page 183, "MVC (Move C)"	Circular interpolation	183
Page 181, "MVA (Move Arch)"	Arch motion interpolation	181
Page 199, "OVRD (Override)"	Overall speed specification	199
Page 213, "SPD (Speed)"	Speed specification during linear or circular interpolation movement	213
Page 176, "JOVRD (J Override)"	Speed specification during joint interpolation movement	176
Page 138, "CNT (Continuous)"	Continuous path mode specification	138
Page 119, "ACCEL (Accelerate)"	Acceleration/deceleration rate specification	119
Page 130, "CMP JNT (Comp Joint)"	Specification of compliance in the JOINT coordinate system	130
Page 132, "CMP POS (Composition Posture)"	Specification of compliance in the XYZ coordinate system	132
Page 134, "CMP TOOL (Composition Tool)"	Specification of compliance in the TOOL coordinate system	134
Page 136, "CMP OFF (Composition OFF)"	Compliance setting invalid	136
Page 137, "CMPG (Composition Gain)"	Compliance gain specification	137
Page 193, "OADL (Optimal Acceleration)"	Optimum acceleration/deceleration rate specification	193
Page 179, "LOADSET (Load Set)"	Hand's optional condition specification	179
Page 201, "PREC (Precision)"	High accuracy mode specification	201
Page 218, "TORQ (Torque)"	Torque specification of each axis	218
Page 177, "JRC (Joint Roll Change)"	Enables multiple rotation of the tip axis	177
Page 164, "FINE (Fine)"	Robot's positioning range specification	164
Page 211, "SERVO (Servo)"	Servo motor power ON/OFF	211
Page 214, "SPDOPT (Speed Optimize)"	Optimize the speed during linear interpolation movement.	214
Page 221, "WTH (With)"	Addition instruction of movement instruction	221
Page 222, "WTHIF (With If)"	Additional conditional instruction of movement instruction	222

(2) Instructions related to program control

Command	Explanation	Page
Page 205, "REM (Remarks)"	Comment(')	205
Page 173, "IF...THEN...ELSE...ENDIF (If Then Else)"	Conditional branching	173
Page 209, "SELECT CASE (Select Case)"	Enables multiple branching	209
Page 169, "GOTO (Go To)"	Jump	169
Page 168, "GOSUB (RETURN)(Go Subroutine)"	Subroutine jump	168
Page 206, "RESET ERR (Reset Error)"	Resets an error (use of default is not allowed)	206
Page 125, "CALLP (Call P)"	Program call	125
Page 166, "FPRM (FPRM)"	Program call argument definition	166
Page 160, "DLY (Delay)"	Timer	160
Page 170, "HLT (Halt)"	Suspends a program	170
Page 163, "END (End)"	End a program	163
Page 196, "ON ... GOSUB (ON Go Subroutine)"	Subroutine jump according to the value	196
Page 197, "ON ... GOTO (On Go To)"	Jump according to the value	197

Command	Explanation	Page
Page 165, "FOR - NEXT (For-next)"	Repeat	165
Page 220, "WHILE-WEND (While End)"	Conditional repeat	220
Page 198, "OPEN (Open)"	Opens a file or communication line	198
Page 202, "PRINT (Print)"	Outputs data	202
Page 175, "INPUT (Input)"	Inputs data	175
Page 128, "CLOSE (Close)"	Closes a file or communication line	128
Page 141, "COLCHK (Col Check)"	Enables or disables the impact detection function	141
Page 195, "ON COM GOSUB (ON Communication Go Subroutine)"	Communication interrupt subroutine jump	195
Page 145, "COM ON/COM OFF/COM STOP (Communication ON/OFF/STOP)"	Allows/prohibits/stops communication interrupts	145
Page 171, "HOPEN / HCLOSE (Hand Open/Hand Close)"	Hand's open/close	171
Page 162, "ERROR (error)"	User error	162
Page 212, "SKIP (Skip)"	Skip while moving	212
Page 219, "WAIT (Wait)"	Waiting for conditions	219
Page 129, "CLR (Clear)"	Signal clear	129

(3) Definition instructions

Command	Explanation	Page
Page 159, "DIM (Dim)"	Array variable declaration	159
Page 157, "DEF PLT (Define pallet)"	Pallet declaration	157
Page 200, "PLT (Pallet)"	Pallet position calculation	200
Page 146, "DEF ACT (Define act)"	Interrupt definition	146
Page 121, "ACT (Act)"	Starts or ends interrupt monitoring	121
Page 149, "DEF ARCH (Define arch)"	Definition of arch shape for arch motion	149
Page 156, "DEF JNT (Define Joint)"	Joint type position variable definition	156
Page 158, "DEF POS (Define Position)"	XYZ type position variable definition	158
Page 153, "DEF INTE/DEF FLOAT/DEF DOUBLE (Define Integer/Float/Double)"	Integer or real number variable definition	153
Page 151, "DEF CHAR (Define Character)"	Character variable definition	151
Page 154, "DEF IO (Define IO)"	Signal variable definition	154
Page 152, "DEF FN (Define function)"	User function definition	152
Page 217, "TOOL (Tool)"	Hand length setting	217
Page 123, "BASE (Base)"	Robot base position setting	123
Page 217, "TOOL (Tool)"	Tool length setting	217

(4) Multi-task related

Command	Explanation	Page
Page 224, "XLOAD (X Load)"	Loads a program to another task slot	224
Page 226, "XRUN (X Run)"	Execute the program in another task slot	226
Page 227, "XSTP (X Stop)"	Stop the program in another task slot	227
Page 225, "XRST (X Reset)"	Resets the program in another task slot being suspended	225
Page 223, "XCLR (X Clear)"	Cancels the loading of the program from the specified task slot	223
Page 167, "GETM (Get Mechanism)"	Obtains mechanical control right	167
Page 204, "RELM (Release Mechanism)"	Releases mechanical control right	204
Page 203, "PRIORITY (Priority)"	Changes the task slot priority	203
Page 206, "RESET ERR (Reset Error)"	Resets an error (use of default is not allowed)	206

(5) Others

Command	Explanation	Page
Page 127, "CHRSRCH (Character search)"	Searches the character string out of the character array.	127
GET POS (Get Position)	Reserved.	-

4.7 Operators

The value's real number or integer type do not need to be declared. Instead, the type may be forcibly converted according to the operation type. (Refer to Table 4-10.) 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 4-10: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= Addition + Comparison (Comparison operators)	Character string Character string Integer	- - -	- - -	- - -	- - -
Integer	Addition + Substract - Multiplication * Division / Integer division \ Remainder MOD Exponent ^ Substitution = Comparison (Comparison operators) Logic (Logic operators)	- - - - - - - - - - -	Integer Integer Integer Integer Integer Integer Integer Integer Integer Integer	Real number Real number Real number Real number Integer Integer Integer Integer Integer Integer	- - - - - - - - - -	- - - - - - - - - -
Real number	Addition + Substract - Multiplication * Division / Integer division \ Remainder MOD Exponent ^ Substitution = Comparison (Comparison operators) Logic (Logic operators)	- - - - - - - - - -	Real number Real number Real number Real number Integer Integer Integer Integer Integer Integer	Real number Real number Real number Real number Integer Integer Real number Real number Integer Integer	- - - - - - - - - -	- - - - - - - - - -
Position	Addition + Substract - Multiplication * Division / Integer division \ Remainder MOD Exponent ^ Substitution = Comparison (Comparison operators) Logic (Logic operators)	- - - - - - - - - -	- - Position Position - - - - - -	- - Position Position - - - - - -	Position Position Position Position - - - Position - -	- - - - - - - - - -
Joint	Addition + Substract - Multiplication * Division / Integer division \ Remainder MOD Exponent ^ Substitution = Comparison (Comparison operators) Logic (Logic operators)	- - - - - - - - - -	- - Joint Joint - - - - - -	- - Joint Joint - - - - - -	- - - - - - - - - -	Joint Joint - Joint - - - Joint - -
Right argument only (Single arugument)	eversal - Negate NOT	- -	Integer Integer	Integer Integer	Position -	Joint -

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 interger multiplication/division is an integer type for multiplication, and a real number type for division.
•If the right argument is a 0 divisor (divide by 0), an operation will not be possible.
•During exponential operation, remainder operation or logical operation (including negate), all real numbers will be forcibly converted into integers (rounded off), and operated.

4.8 Priority level of operations

In the event there are many operators within an expression being calculated, the order of operations is as shown in Table 4-11.

Table 4-11:Priority level of operations

Operations, (operators)	Type of operation	Priority level
1) Operations inside parentheses ()		High
2) Functions	Functions	:
3) Exponents	Numeric value operation	:
4) Single argument operator (+, -)	Numeric value operation	:
5) * /	Numeric value operation	:
6) \	Numeric value operation	:
7)MOD	Numeric value operation	:
8) + -	Numeric value operation	:
9)<< >>	Logic operation	:
10) Comparison operator	Comparison operation	:
(=,<>,><,<,<=,=<,>=,=>)		:
11)NOT	Logic operation	:
12)AND	Logic operation	:
13)OR	Logic operation	:
14)XOR	Logic operation	Low

4.9 Depth of program's control structure

When creating a program, the depth of the control structure must be considered.
When using the commands in the table below, the program's level of control structure becomes one level deeper. Each command has a limit to the depth of the control structure. Exceeding these limits will cause an error.

Table 4-12:Limit to control structure depth

	No. of levels	Applicable commands
User stack in program	16 levels	Repeated controls (FOR-NEXT,WHILE-WEND)
	8 levels	Function calling (CALLP)
	800 levels max.	Subroutine calling (GOSUB) The number decreases by usage frequency of FOR-NEXT, WHILE-WEND, and CALLP instructions.

4.10 Reserved words

Reserved words are those that are already used for the system.
A name that is the same as one of the reserved words cannot be used in the program.
Instructions, functions, and system status variables, etc. are considered reserved words.

4.11 Detailed explanation of command words

4.11.1 How to read the described items

[Function] : Indicates the command word functions.
[Format] : Indicates how to input the command word argument.
The argument is shown in <>.
[ ] indicates that the argument can be omitted.
[] indicates that a space is required.
[Terminology] : Indicates the meaning and range, etc. of the argument.
[Reference Program] : Indicates a program example.
[Explanation] : Indicates detailed functions and cautions, etc.
[The available robot type] : Indicates the available robot type.
[Related parameter] : Indicates the related parameter.
[Related system variables] : Indicates the related system variables.
[Related instructions] : Indicates the related instructions.

4.11.2 Explanation of each command word

Each instruction is explained below in alphabetical order.

ACCEL (Accelerate)

[Function]

Designate the robot's acceleration and deceleration speeds as a percentage (%).
It is valid during optimum acceleration/deceleration.

* The acceleration/deceleration time during optimum acceleration/deceleration refers to the optimum time calculated when using an OADL instruction, which takes account of the value of the M_SETADL variable.

[Format]

```
ACCEL[] [<Acceleration rate>] [, <Deceleration rate>]
```

Controller software version G2 or later

```
ACCEL[] [<Acceleration rate>] [, <Deceleration rate>]
         ,[<Acceleration rate when moving upward>], [<Deceleration rate when moving upward>]
         ,[<Acceleration rate when moving downward>], [<Deceleration rate when moving down-
         ward>]
```

[Terminology]

<Acceleration/Deceleration>

1 to 100(%). Designate the acceleration/deceleration to reach the maximum speed from speed 0 as a percentage. This can be described as a constant or variable. A default value of 100 is set if the argument is omitted. A value of 100 corresponds to the maximum rate of acceleration/deceleration. Unit: %

<Acceleration/Deceleration rate when moving upward>

Specify the acceleration/deceleration rate when moving upward in an arch motion due to the MVA instruction.
A default value of 100 is set if the argument is omitted. It is possible to specify the argument either by a constant or variable.

<Acceleration/Deceleration rate when moving downward>

Specify the acceleration/deceleration rate when moving downward in an arch motion due to the MVA instruction.
A default value of 100 is set if the argument is omitted. It is possible to specify the argument either by a constant or variable.

[Reference Program]

```
10 ACCEL 50,100              ' Heavy load designation (when acceleration/deceleration is 0.2 sec-
                               onds, the acceleration will be 0.4, and the deceleration will be 0.2 sec-
                               onds).
20 MOV P1
30 ACCEL 100,100             ' Standard load designation.
40 MOV P2
50 DEF ARCH 1,10,10,25,25,1,0,0
60 ACCEL 100,100,20,20,20,20 ' Specify the override value to 20 when moving upward or downward due
                               to the MVA instruction.
70 MVA P3,1
```

[Explanation]

(1) The maximum acceleration/deceleration is determined according to the robot being used. Set the corresponding percentage(%). The system default value is 100,100.

(2) The acceleration percentage changed with this command is reset to the system default value when the program is reset or the END statement executed.

(3) Although it is possible to describe the acceleration/deceleration time to more than 100%, some models internally set its upper limit to 100%. If the acceleration/deceleration time is set to more than 100%, it may affect the lifespan of the machine. In addition, speed-over errors and overload errors may tend to occur. Therefore, be extra careful when you are setting it to more than 100%.

(4) The smooth operation when CNT is valid will have a different locus according to the acceleration speed or operation speed. To move smoothly at a constant speed, set the acceleration and deceleration to the same value. CNT is invalid in the default state.

(5) It is also valid during optimum acceleration/deceleration control (OADL ON).

[Related instructions]

OADL (Optimal Acceleration), LOADSET (Load Set)

[Related system variables]

M_ACL/M_DACL/M_NACL/M_NDACL/M_ACLSTS, M_SETADL

[Related parameter]

JADL

ACT (Act)

[Function]

This instruction specifies whether to allow or prohibit interrupt processing caused by signals, etc. during operation.

[Format]

```
ACT[]<Priority No.> = <1/0>
```

[Terminology]

<Priority No.>   0: Either enables or disables the entire interrupt.
1 to 8: Designate the priority No. for the interrupt defined in the DEF ACT statement.
When entering the priority No., always leave a space (character) after the ACT command.
If described as ACT1, it will be a variable name declaration statement.

<1/0>   1: Allows interrupts, 0:Prohibits interrupts.

[Reference Program]

(1) When the input signal 1 turns on (set to 1) while moving from P1 to P2, it loops until that signal is set to 0.

```
 10 DEF ACT 1,M_IN(1)=1 GOSUB *INTR      ' Assign input signal 1 to the interrupt 1 condition
 20 MOV P1
 30 ACT 1=1                              ' Enable interrupt 1.
 40 MOV P2
 50 ACT 1=0                              ' Disable interrupt 1.
     :
100 *INTR                                '
110 IF M_IN(1)=1 GOTO 110                ' Loops until the M_IN(1) signal becomes 0.
120 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.

```
10 DEF ACT 1,M_IN(1)=1 GOSUB *INTR      'Assign input signal 1 to the interrupt 1 condition
20 MOV P1
30 ACT 1=1                              ' Enable interrupt 1.
40 MOV P2
 :
100 *INTR
110 ACT 1=0                             ' Disable interrupt 1.
120 M_OUT(10)=1                         ' Turn on the output signal 10
130 RETURN 1                            ' Returns to the next line which interrupted
```

[Explanation]

(1) When the program starts, the status of <Priority No.> 0 is "enabled." When <Priority No.> 0 is "disabled," even if <Priority No.> 1 to 8 are set to "enabled," no interrupt will be enabled.

(2) The statuses of <Priority No.> 1 to 8 are all "disabled" when the program starts.

(3) An interrupt will occur only when all of the following conditions have been satisfied:
*<Priority No.> 0 is set to "enabled."
*The status of the DEF ACT statement has been defined.
*When the <Priority No.> designated by DEF ACT is made valid by an ACT statement.

(4) The return from an interrupt process should be done by describing either RETURN 0 or RETURN 1. However when returning from interruption processing to the next line 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 line, the line may be skipped.

(5) Even if the robot is in the middle of interpolation, an interrupt defined by a DEF ACT statement will be executed.

(6) During an interrupt process, that <Priority No.> will be executed with the status as "disable."

(7) A communications interrupt (COM) has a higher priority than an interrupt defined by a DEF ACT statement.

(8) The relationship of priority rankings is as shown below:
COM>ACT>WTHIF(WTH)>Pulse substitution

[Related instructions] DEF ACT (Define act), RETURN (Return)

BASE (Base)

[Function]

With this instruction, it is possible to move or rotate the robot coordinate system. Specify the base conversion data for this instruction. Pay extra attention when making changes in a program, as it may be mistaken in jog operations, etc.

[Format]

```
BASE[]<Base conversion data>
```

[Terminology]

<Base conversion data>   Specify with position constants or position variables.

[Reference Program]

```
10 BASE (50,100,0,0,0,90)     ' Input the conversion data as a constant.
20 MVS P1
30 BASE P2                    ' Input the conversion data as a variable.
40 MVS P1
50 BASE P_NBASE               ' Reset the conversion data to the default values.
```

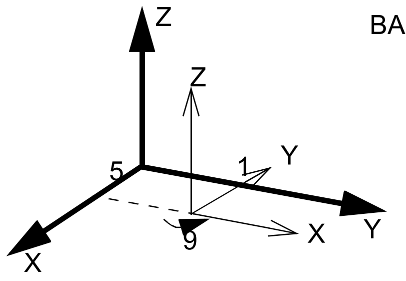
[Explanation]

(1) The X, Y, and Z components of the position data represent the amount of parallel movement from the origin of the world coordinate system to the origin of the base coordinate system. The base conversion data can be changed only with the BASE command. The components A, B, and C of the position data represent the amount that the base coordinate system is tilted in relation to the world coordinate system.

X.............Distance to move parallel to X axis
Y.............Distance to move parallel to Y axis
Z.............Distance to move parallel to Z axis
A............Angle to turn toward the X axis
B............Angle to turn toward the Y axis
C............Angle to turn toward the Z axis
L1..........Movement amount of additional axis 1
L2..........Movement amount of additional axis 2

(2) For A, B and C, the clockwise direction looking from the front of the origin of the coordinate, used as the center of rotation, is the forward rotation direction.
(3) The contents of the structural flag have no meaning.
(4) The system's default value for this data is P_NBASE=(0,0,0,0,0,0) (0,0). This is calculated with the 6-axis three dimensional regardless of the mechanism structure.
(5) The valid axis element of base conversion data is different depending on the type of robot. Set up the appropriate data referring to the Page 327, "Table 5-7: Valid axis elements of the base conversion data depending on the robot model".

Fig.4-3:Conceptual diagram of the base coordinate system



BASE (50,100,0,0,0,90)

[Related parameter]

After it has been changed by the MEXBS BASE instruction, the base coordinate system is stored in the MEXBS parameter and maintained even after the controller's power is turned off. Refer to Page 327, "About Standard Base Coordinates".

[Related system variables]

P_BASE/P_NBASE

CALLP (Call P)

[Function]

This instruction executes the specified program (by calling the program in a manner similar to using GOSUB to call a subroutine). The execution returns to the main program when the END instruction or the final line 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 93, "(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

```
10 M1=0
 20 CALLP "10" ,M1,P1,P2
 30 M1=1
 40 CALLP "10" ,M1,P1,P2
        :
100 CALLP "10", M2,P3,P4
        :
150 END
```

"10" sub program side

```
 10 FPRM M01, P01,P02
 20 IF M01<>0 THEN GOTO *LBL1
 30 MOV P01
 40 *LBL1
 50 MVS P02
 60 END                    'Return to the main program at this point.
```

* When lines 20 and 40 of the main program are executed, M1, P1 and P2 are set in M01, P01 and P02 of the sub program, respectively. When line 100 of the main program is executed, M2, P3 and P4 are set in M01, P01 and P02 of the sub program, respectively.

(2) When not passing the argument to the program to call.

Main program

```
  10 MOV P1
  20 CALLP "20"
  30 MOV P2
  40 CALLP "20"
  50 END
```

"20" sub program side

```
  10 MOV P1                'P1 of the sub program differs from P1 of the main program.
  20 MVS P002
  30 M_OUT(17)=1
  40 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 line of the sub program is reached.

(2) If arguments need to be passed to the sub program, they should be defined using the FPRM instruction at the beginning of the sub program.

(3) If the type or the number of arguments passed to the sub program is different from those defined (by the FPRM instruction) in the sub program, an error occurs at execution.

(4) If a program is reset, the control returns to the beginning of the top main program.

(5) Definition statements (DEF ACT, DEF FN, DEF PLT, and DIM instructions) executed in the main program are invalid in a program called by the CALLP instruction. They become valid when the control is returned to the main program from the program called by the CALLP instruction again.

(6) Speed and tool data are all valid in a sub program. Values of ACCEL and SPD are invalid. The mode of OADL is valid.

(7) Another sub program can be executed by calling CALLP in a sub program. However, a main program or a program that is currently being executed in another task slot cannot be called. In addition, own program cannot be called, either.

(8) Eight levels (in a hierarchy) of sub programs can be executed by calling CALLP in the first main program.

(9) Variable values may be passed from a main program to a sub program using arguments, however, it is not possible to pass the processing result of a sub program to a main program by assigning it in an argument. To use the processing result of a sub program in a main program, pass the values using external variables.

[Related instructions]

FPRM (FPRM)

CHRSRCH (Character search)

[Function]

Searches the character string out of the character array.

[Format]

```
CHRSRCH[]<Character string array variable>,<Character string>,<Search result storage destination>
```

[Terminology]

<Character string array variable> Specify the character string array to be searched.
<Character string> Specify the character string to be searched.
<Search result storage destination> The number of the element for which the character string to be searched is found is set.

[Reference Program]

```
10 DIM C1$(10)
20 C1$(1)="ABCDEFG"
30 C1$(2)="MELFA"
40 C1$(3)="BCDF"
50 C1$(4)="ABD"
60 C1$(5)="XYZ"
70 C1$(6)="MELFA"
80 C1$(7)="CDF"
90 C1$(8)="ROBOT"
100 C1$(9)="FFF"
110 C1$(10)="BCD"
120 CHRSRCH C1$(1), "ROBOT", M1          ' 8 is set in M1.
130 CHRSRCH C1$(1), "MELFA", M2          ' 2 is set in M2.
```

[Explanation]

(1) The specified character string is searched from the character string array variables, and the element number of the completely matched character string array is set in <search result storage destination>. Partially matched character strings are not searched.
Even if CHRSRCH C1$(1), "ROBO", M1 are described in the above statement example, the matched character string is not searched.
(2) If the character string to be searched is not found, 0 is set in <search result storage destination>.
(3) Character string search is performed sequentially beginning with element number 1, and the element number found first is set.
Even if CHRSRCH C1$(3), "MELFA", M2 are described in the above statement example, 2 is set in M2. (The same character string is set in C1$(2) and C1$(6).)
(4) The <character string array variable> that can be searched is the one-dimensional array only. If a two-dimensional or higher array is specified as a variable, an error will occur at the time of execution.

CLOSE (Close)

[Function]

Closes the designated file.(including communication lines)

[Format]

```
CLOSE[] [[#]<File No.>[, [[#]<File No.> ...]
```

[Terminology]

<File No.> Designate the No. of the file to be closed. Only a numerical constant is allowed. If this argument is omitted, all open files are closed.

[Reference Program]

```
 10 OPEN "COM1:" AS #1        ' "Open "COM1:" as file No. 1.
 20 PRINT #1,M1
   :
100 INPUT #1,M2
110 CLOSE #1                  ' Close file No. 1, "COM1:".
   :
200 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 4-13:Processing of each buffer when the file is closed

Buffer types	Processing when the file is closed
Communication line reception buffer	The contents of the buffer are destroyed
Communication line transmission buffer	(No data remains in the transmission buffer since the data in the transmission buffer is sent immediately by executing the PRINT instruction.)
File load buffer	The contents of the buffer are destroyed.
File unload buffer	The contents of the buffer are written into the file, and then the file is closed.

(2) Executing an END statement will also close a file.
(3) If the file number is omitted, all files will be closed.

[Related instructions]

OPEN (Open), PRINT (Print), INPUT (Input)

CLR (Clear)

[Function]

This instruction clears general-purpose output signals, local numerical variables in a program, and numerical external variables.

[Format]

```
CLR[]<Type>
```

[Terminology]

<Type>

It is possible to specify either a constant or a variable.

0 : All steps 1 to 3 below are executed.

1 : The general-purpose output signal is cleared based on the output reset pattern. The output reset pattern is designated with parameters ORST0 to ORST224.

Refer to Page 335, "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.

```
10 CLR 1
```

(2) The local numeric variables and numeric array variables in the program are cleared to 0.

```
10 DIM MA(10)
20 DEF INTE IVAL
30 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

```
10 CLR 3
```

(4) (1) though (3) above are performed simultaneously.

```
10 CLR 0
```

[Related parameter]

ORST0 to ORST224

[Related system variables]

M_IN/M_INB/M_INW, M_OUT/M_OUTB/M_OUTW

CMP JNT (Comp Joint)

[Function]

Start the soft control mode (compliance mode) of the specified axis in the JOINT coordinates system.
Note) The available robot type is limited such as RH-nAH. 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]

```
   10 MOV P1
   20 CMPG 0.0,0.0,1.0,1.0, , , ,     ' Set softness.
   30 CMP JNT,&B11                    ' The J1 and J2 axes are put in the state where they are controlled in a
pliable manner.
   40 MOV P2
   50 HOPEN 1
   60 MOV P1
   70 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-*AH type, specify 0.0 for the horizontal axes J1 and J2 to make the robot behave equivalently to a servo free system (the spring constant is zero). (Note that the vertical axes cannot be made to behave equivalently to a servo free system even if 0.0 is set for them. Also, be careful not to let these axes reach a position beyond the movement limit or where the amount of diversion becomes too large.) Note that 4) and 5) below do not function if this servo-free equivalent behavior is in use.

(3) The soft state is maintained even after the robot program execution is stopped. To cancel the soft status, execute the "CMP OFF" command or turn OFF the power.

(4) When pressing in the soft state, the robot cannot move to positions that exceed the operation limit of each joint axis.

(5) If the amount of difference between the original target position and the actual robot position becomes greater than 200 mm by pushing the hand, etc., the robot will not move any further and the operation shifts to the next line 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.

⚠CAUTION The compliance mode is in effect continuously until the CMP OFF instruction is executed, or the power is turned off.

⚠CAUTION To execute a jog operation after setting the compliance mode with the CMP JNT instruction, use the JOINT jog mode.
If any other jog mode is used, the robot may operate in a direction different from the expected moving direction because the directions of the coordinate systems controlled by the jog operation and the compliance mode differ.

⚠CAUTION When performing the teaching of a position while in the compliance mode, perform servo OFF first.
Be careful that if teaching operation is performed with Servo ON, the original command position is taught, instead of the actual robot position. As a result, the robot may move to a location different from what has been taught.

[Available robot type]

RH-5AH/10AH/15AH series
RH-6SH/12SH/18SH series

[Related system variables]
M_BTIME

[Related instructions]
CMP OFF (Composition OFF), CMPG (Composition Gain), CMP POS (Composition Posture), CMP TOOL (Composition Tool)

CMP POS (Composition Posture)

[Function]

Start the soft control mode (compliance mode) of the specified axis in the XYZ coordinates system.
Note) The available robot type is limited such as RV-4A. Refer to "[Available robot type]".

[Format]

```
CMP[]POS, <Axis designation>
```

[Terminology]

<Axis designation>   Designate axis to move softly with a bit pattern.
1 : Enable, 0 : Disable &B00000000
This corresponds to axis L2L1CBAZYX

[Reference Program]

```
10 MOV P1                          ' Move in front of the part insertion position.
20 CMPG 0.5, 0.5, 1.0, 0.5, 0.5, , ,  ' Set softness
30 CMP POS, &B011011               ' The X, Y, A, and B axes are put in the state where they are con-
                                     trolled in a pliable manner.
40 MVS P2                          ' Moves to the part insertion position.
50 M_OUT(10)=1                     ' Instructs to close the chuck for positioning.
60 DLY 1.0                         ' Waits for the completion of chuck closing.(1 sec.)
70 HOPEN 1                         ' Open the hand.
80 MVS, -100                       ' Retreats 100 mm in the Z direction of the TOOL coordinate sys-
                                     tem.
90 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 line 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.

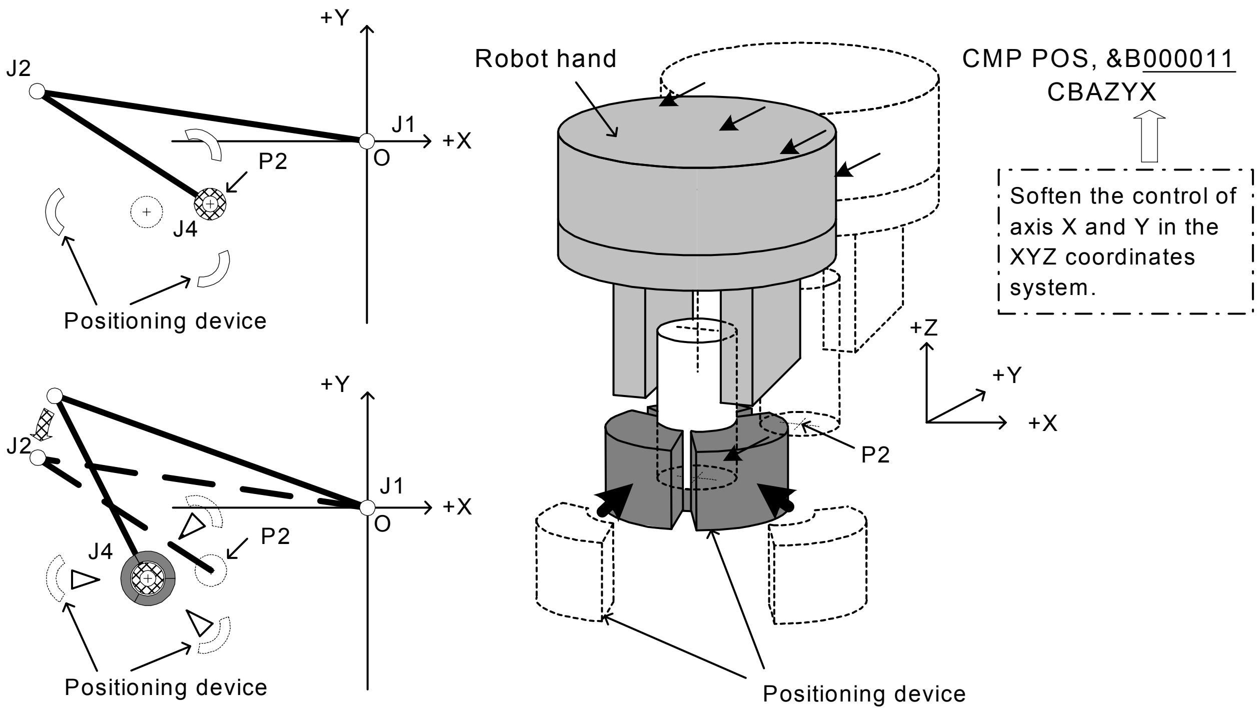


Fig.4-4:The example of compliance mode use

⚠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.

[Available robot type]

RV-1A/2AJ series
RV-2A/3AJ series
RV-4A/5AJ series
RV-20A
RV-3S/3SJ/3SB/3SJB series
RV-6S/6SL/12S/12SL series
RH-5AH/10AH/15AH series
RH-6SH/12SH/18SH series

[Related system variables]
M_BTIME

[Related instructions]
CMP OFF (Composition OFF), CMPG (Composition Gain), CMP TOOL (Composition Tool), CMP JNT (Comp Joint)

CMP TOOL (Composition Tool)

[Function]

Start the soft control mode (compliance mode) of the specified axis in the TOOL coordinates system.
Note) The available robot type is limited such as RV-4A. Refer to "[Available robot type]".

[Format]

```
CMP[]TOOL, <Axis designation>
```

[Terminology]

```
<Axis designation>  Designate axis to move softly with a bit pattern.
                    1 : Enable, 0 : Disable  &B00000000
                                   This corresponds to axis L2L1CBAZYX
```

[Reference Program]

```
10 MOV P1                          ' Moves to in front of the part insertion position.
20 CMPG 0.5, 0.5, 1.0, 0.5, 0.5, , ,  ' Set softness.
30 CMP TOOL, &B011011              ' The X, Y, A, and B axes are put in the state where they are con-
                                     trolled in a pliable manner.
40 MVS P2                          ' Moves to the part insertion position.
50 M_OUT(10)=1                     ' Instructs to close the chuck for positioning.
60 DLY 1.0                         ' Waits for the completion of chuck closing.(1 sec.)
70 HOPEN 1                         ' Open the hand.
80 MVS, -100                       ' Retreats 100 mm in the Z direction of the TOOL coordinate sys-
                                     tem.
90 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 324, "5.6 Standard Tool Coordinates".

(2) For example, when inserting a pin in the tool coordinate Z axis direction, if the X, Y, A and B axes are set to soft operation, the pin can be inserted smoothly.

(3) The degree of softness can be designated with the CMPG command.

(4) The soft state is maintained even after the robot program execution is stopped. To cancel the soft status, execute the "CMP OFF" command or turn OFF the power.

(5) When pressing in the soft state, the robot cannot move to positions that exceed the operation limit of each joint axis.

(6) The deviation of the command position and actual position can be read with M_CMPDST. The success/failure of pin insertion can be checked using this variable.

(7) If the amount of difference between the original target position and the actual robot position becomes greater than 200 mm by pushing the hand, etc., the robot will not move any further and the operation shifts to the next line of the program.

(8) It is not possible to use CMP JNT, POS, and TOOL at the same time. In other words, an error occurs if the CMP POS or CMP TOOL instruction is executed while the CMP JNT instruction is being performed. Cancel the CMP JNT instruction once using the CMP OFF instruction to execute these instructions.

(9) If the servo turns from OFF to ON while this command is functioning, the robot position could change.

(10) It is possible to perform jog operations while the robot is in compliance mode. However, the setting of the compliance mode cannot be canceled by the T/B; in order to do so, execute this instruction in a program or execute it directly via the program edit screen of the T/B.

(11) To change the axis specification, cancel the compliance mode with the CMP OFF instruction first, and then execute the CMP TOOL instruction again.

(12) For vertical 5-axis robots (such as the RV-5AJ), 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.

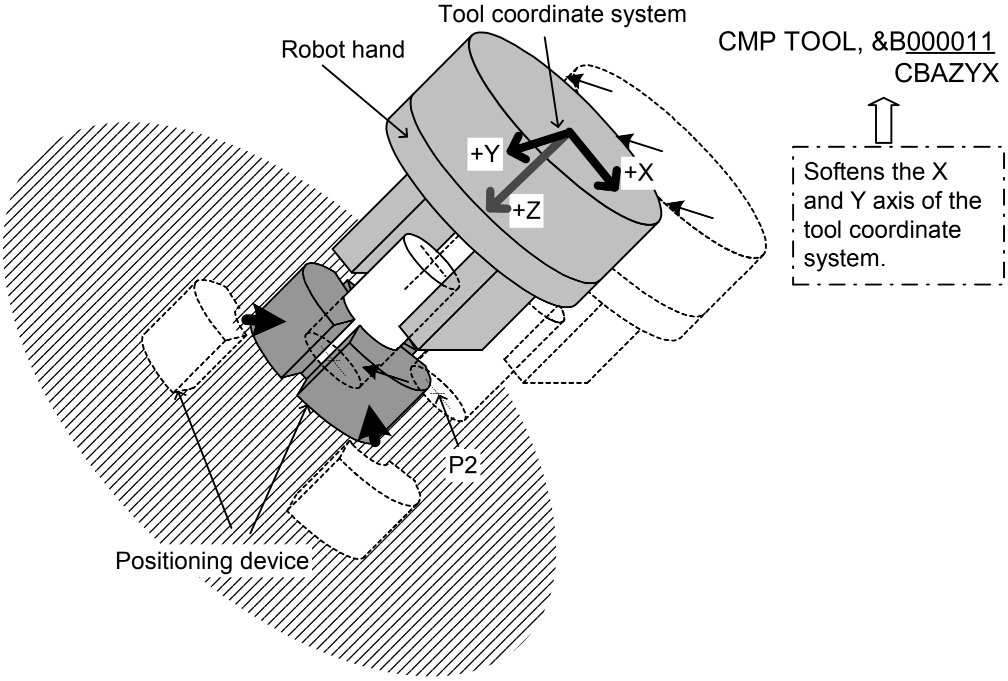


Fig.4-5:The example of using the compliance mode

⚠CAUTION The compliance mode is in effect continuously until the CMP OFF instruction is executed, or the power is turned off. Exercise caution when changing the executable program number or operating the jog.

⚠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.

[Available robot type]

RV-1A/2AJ series
RV-2A/3AJ series
RV-4A/5AJ series
RV-20A
RV-3S/3SJ/3SB/3SJB series
RV-6S/6SL/12S/12SL series
RH-5AH/10AH/15AH series
RH-6SH/12SH/18SH series

[Related system variables]

M_BTIME

[Related instructions]

CMP OFF (Composition OFF), CMPG (Composition Gain), CMP POS (Composition Posture), CMP JNT (Comp Joint)

CMP OFF (Composition OFF)

[Function]

Release the soft control mode (compliance mode).
Note) The available robot type is limited such as RV-4A. Refer to "[Available robot type]".

[Format]

```
CMP[]OFF
```

[Reference Program]

```
10 MOV P1                          ' Moves to in front of the part insertion position.
20 CMPG 0.5, 0.5, 1.0, 0.5, 0.5, , ,  ' Set softness.
30 CMP TOOL, &B011011              ' The X, Y, A, and B axes are put in the state where they are con-
                                     trolled in a pliable manner.
40 MVS P2                          ' Moves to the part insertion position.
50 M_OUT(10)=1                     ' Instructs to close the chuck for positioning.
60 DLY 1.0                         ' Waits for the completion of chuck closing.(1 sec.)
70 HOPEN 1                         ' Open the hand.
80 MVS, -100                       ' Retreats 100 mm in the Z direction of the TOOL coordinate sys-
                                     tem.
90 CMP OFF                         ' Return to normal state.
```

[Explanation]

(1) This instruction cancels the compliance mode started by the CMP TOOL, CMP POS, or CMP JNT instruction.

(2) In order to cancel jog operations in the compliance mode, either execute this instruction in a program or execute it directly via the program edit screen of the T/B.

[Available robot type]

RV-1A/2AJ series
RV-2A/3AJ series
RV-4A/5AJ series
RV-20A
RV-3S/3SJ/3SB/3SJB series
RV-6S/6SL/12S/12SL series
RH-5AH/10AH/15AH series
RH-6SH/12SH/18SH series

[Related instructions]

CMPG (Composition Gain), CMP TOOL (Composition Tool), CMP POS (Composition Posture), CMP JNT (Comp Joint)

CMPG (Composition Gain)

[Function]

Specify the softness of robot control.
Note) The available robot type is limited such as RV-4A. Refer to "[Available robot type]".

[Format]

CMP POS, CMP TOOL

```
CMPG[] [<X axis gain>], [<Y axis gain>], [<Z axis gain>], [<A axis gain>],
              [<B axis gain>], [<C axis gain>], [<L1 axis gain>], [<L2axis gain>],
```

CMP JNT

```
CMPG[] [<J1 axis gain>], [<J2 axis gain>], [<J3 axis gain>], [<J4 axis gain>],
              [<J5 axis gain>], [<J6 axis gain>], [<J7 axis gain>], [<J8 axis gain>],
```

[Terminology]

<X to L2 axis gain>
<J1 to J8 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]

```
10 CMPG  , ,0.5, , , , , ' This statement selects only the Z-axis. For axes that are omitted, keep the corresponding entries blank and just enter commas.
```

[Explanation]

(1) The softness can be designated in each axis units.
(2) The soft state will not be entered unless validated with the CMP POS or CMP TOOL commands.
(3) A spring-like force will be generated in proportion to the deviation of the command position and actual position. CMPG designates that spring constant.
(4) The deviation of the command position and actual position can be read with M_CMPDST. The success/failure of pin insertion can be checked using this variable.
(5) If a small gain is set, and the soft state is entered with the CMP POS, CMP TOOL, and CMP JNT commands, the robot position could drop. Set the softness state gradually while checking it.
(6) The softness can be changed halfway when this command executed under the soft control status.
(7) The gain value of less than 0.2 is invalid. The robot is controlled using the value 0.2.
(However, up to 0.1 can be set for the RV-1A/2AJ, and up to 0.0 for the RH-[]AH.)
Also, two or more decimal positions can be set for gain values.

[Available robot type]

RV-1A/2AJ series
RV-2A/3AJ series
RV-4A/5AJ series
RV-20A
RV-3S/3SJ/3SB/3SJB series
RV-6S/6SL/12S/12SL series
RH-5AH/10AH/15AH series
RH-6SH/12SH/18SH series

CNT (Continuous)

[Function]

Designates continuous movement control for interpolation. Shortening of the operating time can be performed by carrying out continuous movement.

[Format]

```
CNT[] <Continuous movement mode/acceleration/deceleration movement mode>]
          [, <Numeric value 1>] [, <Numeric value 2>]
```

[Terminology]

<1/0> Designate the continuous operation or acceleration/deceleration operation mode.
1 : Continuous movement.
0 : Acceleration/deceleration movement.(default value.)

<Numeric value 1> Specify the maximum proximity distance in mm for starting the next interpolation when changing to a new path segment.
The default value is the position where the acceleration/deceleration is started.

<Numeric value 2> Specify the maximum proximity distance in mm for ending the previous interpolation when changing to a new path segment.
The default value is the position where the acceleration/deceleration is started.

[Reference Program]

When the maximum neighborhood distance is specified when changing a locus.

```
10 CNT 0              ' Invalidate CNT (Continuous movement).
20 MVS P1             ' Operate with acceleration/deceleration
30 CNT 1              ' Validate CNT (Continuous movement).
                       (Operate with continuous movement after this line.)
40 MVS P2             ' The connection with the next interpolation is continuous movement.
50 CNT 1,100,200      ' Continuous operation specification at 100 mm on the starting side and at 200
                        mm on the end side.
60 MVS P3             ' Continuous operation at a specified distance before and after an interpolation.
70 CNT 1,300          ' Continuous operation specification at 300 mm on the starting side and at 300
                        mm on the end side.
80 MOV P4             ' Continuous operation specification at 300 mm on the starting side.
90 CNT 0              ' Invalidate CNT (Continuous movement).
100 MOV P5            ' Operate with acceleration/deceleration
```

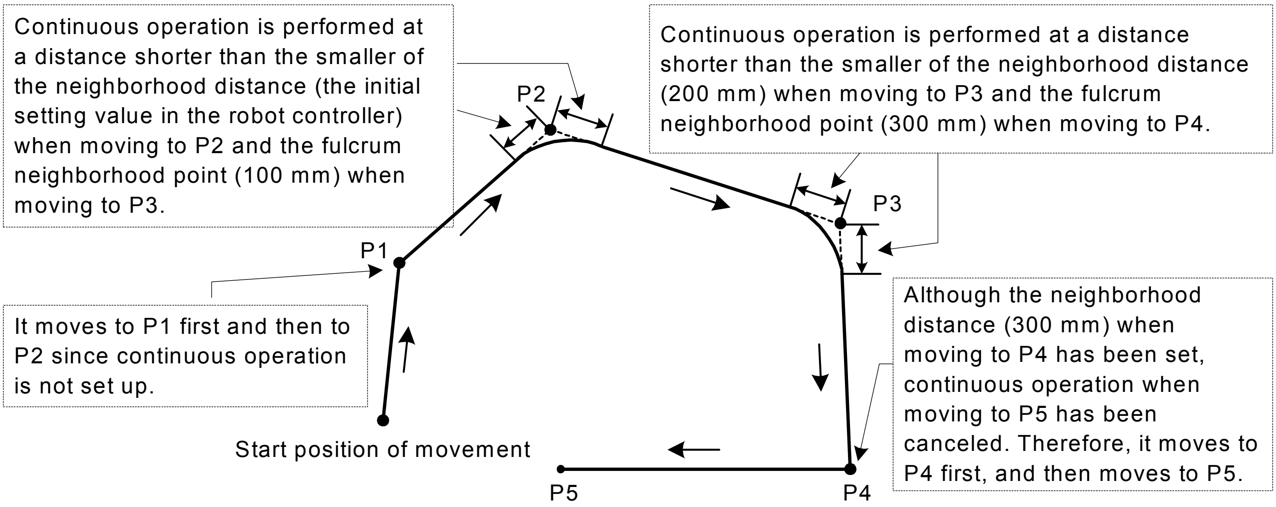

Continuous operation is performed at a distance shorter than the smaller of the neighborhood distance (the initial setting value in the robot controller) when moving to P2 and the fulcrum neighborhood point (100 mm) when moving to P3.
P2
Continuous operation is performed at a distance shorter than the smaller of the neighborhood distance (200 mm) when moving to P3 and the fulcrum neighborhood point (300 mm) when moving to P4.
P3
P1
It moves to P1 first and then to P2 since continuous operation is not set up.
Although the neighborhood distance (300 mm) when moving to P4 has been set, continuous operation when moving to P5 has been canceled. Therefore, it moves to P4 first, and then moves to P5.
Start position of movement
P5
P4

Fig.4-6:Example of continuous path operation

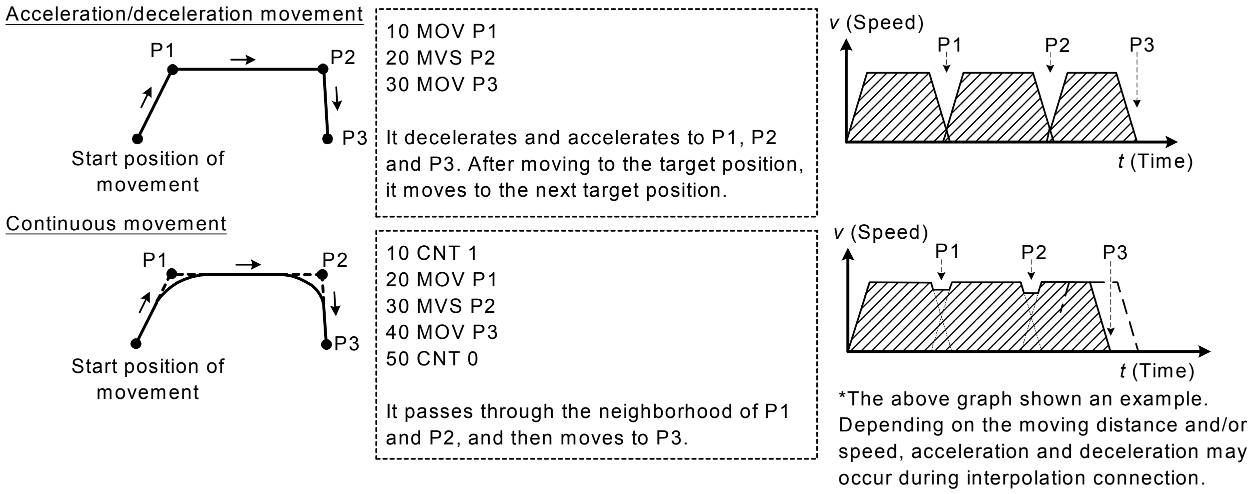
[Explanation]

(1) The interpolation (40 line to 80 line of the example) surrounded by CNT 1 - CNT 0 is set as the target of continuous action.

(2) The system default value is CNT 0 (Acceleration/deceleration movement).

(3) If values 1 and 2 are omitted, the connection with the next path segment is started from the time the deceleration is started.

(4) As shown in Fig. 4-7, in the acceleration and deceleration operating mode, the speed is reduced in front of the target position. After moving to the target position, the speed for moving to the next target position starts to be accelerated. On the other hand, in the continuous operating mode, the speed is reduced in front of the target position, but it does not stop completely. The speed for moving to the next target position starts to be accelerated at that point. Therefore, it does not pass through each target position, but it passes through the neighborhood position.


Acceleration/deceleration movement
P1
P2
P3
Start position of movement
v (Speed)
P1
P2
P3
t (Time)
Continuous movement
P1
P2
P3
Start position of movement
v (Speed)
P1
P2
P3
t (Time)

```
10 MOV P1
20 MVS P2
30 MOV P3
```

It decelerates and accelerates to P1, P2 and P3. After moving to the target position, it moves to the next target position.

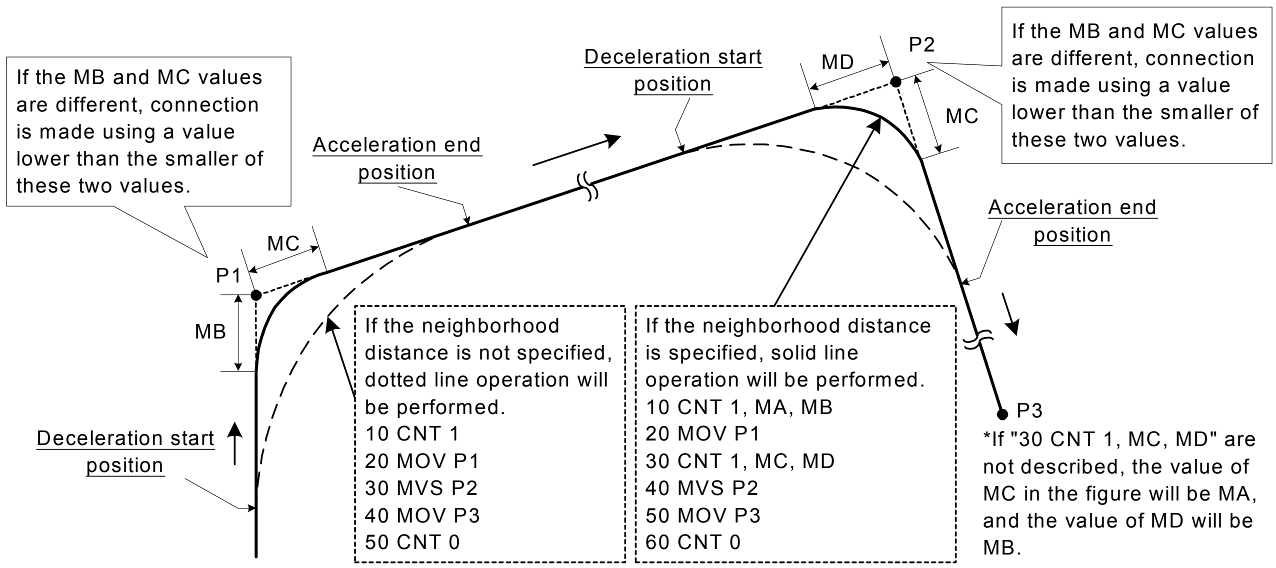
```
10 CNT 1
20 MOV P1
30 MVS P2
40 MOV P3
50 CNT 0
```

It passes through the neighborhood of P1 and P2, and then moves to P3.

*The above graph shown an example. Depending on the moving distance and/or speed, acceleration and deceleration may occur during interpolation connection.

Fig.4-7:Acceleration/deceleration movement and continuous movement

(5) The neighborhood distance denotes the changing distance to the interpolation operation at the next target position. If this neighborhood distance (numerical value 1, numerical value 2) is omitted, the accelerate and deceleration starting position will be the changing position to the next interpolation. In this case, it passes through a location away from the target position, but the operating time will be the shortest. To pass through a location closer to the target position, set this neighborhood distance (numerical value 1, numerical value 2).


Deceleration start position
MD
P2
MC
Acceleration end position
Acceleration end position
MC
P1
MB
Deceleration start position
P3

If the MB and MC values are different, connection is made using a value lower than the smaller of these two values.

If the MB and MC values are different, connection is made using a value lower than the smaller of these two values.

If the neighborhood distance is not specified, dotted line operation will be performed.

```
10 CNT 1
20 MOV P1
30 MVS P2
40 MOV P3
50 CNT 0
```

If the neighborhood distance is specified, solid line operation will be performed.

```
10 CNT 1, MA, MB
20 MOV P1
30 CNT 1, MC, MD
40 MVS P2
50 MOV P3
60 CNT 0
```

*If "30 CNT 1, MC, MD" are not described, the value of MC in the figure will be MA, and the value of MD will be MB.

Fig.4-8:Setting Up the Neighborhood Distance

(6) If the specifications of numerical value 1 and numerical value 2 are different, continuous operation will be performed at the position (distance) that is the smaller of these two.
(7) If numeric value 2 is omitted, the same value as numeric value 1 will be applied.
(8) When continuous operation is specified, the positioning completion specification by the FINE instruction will be invalid.
(9) If the proximity distance (value 1, value 2) is set small, the movement time may become longer than in the status where CNT 0.

COLCHK (Col Check)

[Function]

Set to enable/disable the impact detection function.

The impact detection function quickly stops the robot when the robot's hand and/or arm interferes with peripheral devices so as to minimize damage to and deformation of the robot's tool part or peripheral devices. However, it cannot completely prevent such damage and deformation.

The impact detection function can only be used in certain models (Refer to "[Available robot type]".). This function is available for controller software version J2 or later.

[Format]

```
COLCHK[]ON [, NOERR] / OFF
```

[Terminology]

ON　Enable the impact detection function.
Once an impact is detected, it immediately stops the robot, issues an error numbered in 1010's, and turns OFF the servo.

OFF　Disable the impact detection function

NOERR　Even if an impact is detected, no error is issued. (If omitted, an error will occur.)

[Reference Program 1]

If an error is set in the case of impact

```
10 COLLVL 80,80,80,80,80,80,,       'Specify the allowable level for impact detection.
20 COLCHK ON                        'Enable the impact detection function.
30 MOV P1
40 MOV P2
50 DLY 0.2                          'Wait until the completion of operation
                                     (FINE instruction can also be used).
50 COLCHK OFF                       'Disable the impact detection function.
60 MOV P3
```

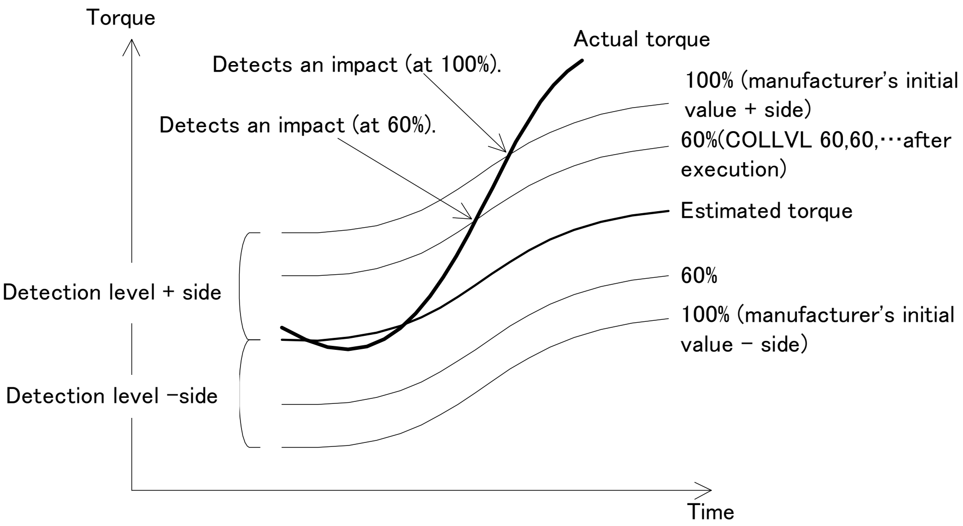
[Reference Program 2]

If interrupt processing is used in the case of impact

```
10 DEF ACT 1,M_COLSTS(1)=1 GOTO *HOME,S'Define the processing to be executed when an impact
                                              is detected using an interrupt.
20 ACT 1=1
30 COLCHK ON,NOERR                  'Enable the impact detection function in the error non-occurrence
                                     mode.
40 MOV P1
50 MOV P2                           'If an impact is detected while executing lines 40 through 70, it
                                     jumps to interrupt processing.
60 MOV P3
70 MOV P4
80 ACT 1=0
     :
1000 *HOME                          'Interrupt processing during impact detection
1010 COLCHK OFF                     'Disable the impact detection function.
1020 SERVO ON                       'Turn the servo on.
1030 PESC=P_COLDIR(1)*(-2)          'Create the amount of movement for escape operation.
1040 PDST=P_FBC(1)+PESC             'Create the escape position.
1050 MVS PDST                       'Move to the escape position.
1060 ERROR 9100                     'Stop operation by generating a user-defined L level error.
```

[Explanation]

(1) The impact detection function estimates the amount of torque that will be applied to the axes during movement executed by a Move instruction. It determines that there has been an impact if the difference between the estimated torque and the actual torque exceeds the tolerance, and immediately stops the robot.



(2) Immediately after power ON, the impact detection function is disabled. Enable the COL parameter before using.

(3) The detection level can be adjusted by a COLLVL instruction. The initial value of the detection level is the setting value of the COLLVL parameter.

(4) After the impact detection function is enabled by this instruction, that state is maintained continuously until it is disabled by the COLCHK OFF instruction, the program is reset, an END instruction is executed or the power is turned OFF.

(5) Even if the impact detection function is disabled by this instruction, the impact tolerance level set by a COLLVL instruction is retained.

(6) When the continuity function is enabled, the previous impact detection setting state is restored at next power ON even if the power is turned OFF.

(7) Error 3950 occurs if an interrupt by the M_COLSTS status variable (an interrupt with the interrupt condition of M_COLSTS(*)=1 and * denotes a machine number) is not enabled when specifying NOERR (error non-occurrence mode). See [Syntax Example 2]. Error 3960 also occurs if this interrupt processing is disabled while in the error non-occurrence mode.

(8) If an impact is detected while in the error non-occurrence mode, the robot turns OFF the servo and stops. Therefore, no error occurs and operation also continues. However, it is recorded in the error log that an impact was detected. (The recording into the log is done only if no other errors occur simultaneously.)

(9) If an attempt is made to execute COLCHK ON and COLCHK ON,NOERR on a robot that cannot use the impact detection function, low level error 3970 occurs. In the case of COLCHK OFF, neither error occurs nor processing is performed.

(10) The impact detection function cannot be enabled while compliance is being enabled by a CMP instruction or the torque limit is being enabled by a TORQ instruction. In this case, error 3940 will occur if an attempt is made to enable the impact detection function. Conversely, error 3930 will occur if an attempt is made to enable a CMP or TORQ instruction while impact detection is being enabled.

(11) If COLCHK OFF is described immediately after an operation instruction, impact detection may not work near the last stop position of a given operation. As shown in syntax example 1, execute COLCHK OFF upon completion of positioning by a DLY or FINE instruction between an operation instruction and a COLCHK OFF instruction.

(12) Erroneous detection may occur if the hand weight (HNDDATn parameter) and workpiece weight (WRKDATn parameter) are not set correctly. Be sure to set these parameters correctly before using.

[Related instructions and variables]

COLLVL (Col Level), M_COLSTS, J_COLMXL, P_COLDIR

[Related parameter]

COL, COLLVL, COLLVLJG

[Available robot type]

RV-3S/3SJ/3SB/3SJB series
RV-6S/6SL/12S/12SL series
RH-6SH/12SH/18SH series

COLLVL (Col Level)

[Function]

Set the detection level of the impact detection function.
The impact detection function can only be used in certain models (Refer to "[Available robot type]".). This function is available for controller software version J2 or later.

[Format]

```
COLLVL[] [<J1 axis>],[<J2 axis>],[<J3 axis>],[<J4 axis>],[<J5 axis>],[<J6 axis>],[<J7 axis>],[<J8 axis>]
```

[Terminology]

<J1 to J8 axis>   Specify the detection level in a range between 1 and 500%.
If omitted, the previously set value is retained.
Currently, the J7 and J8 axes do not function.
The initial value is the setting value of the COLLVL parameter.

[Reference Program]

```
10 COLLVL 80,80,80,80,80,80,,  'Specify the allowable level for impact detection.
20 COLCHK ON                   'Enable the impact detection function.
30 MOV P1
40 COLLVL ,50,50,,,,,          'Change the allowable level of the J2 and J3 axes for impact detection.
50 MOV P2
60 DLY 0.2                     'After arriving at P2, disable impact detection.
70 COLCHK OFF                  'Disable the impact detection function.
80 MOV P3
```

[Explanation]

(1) Set the allowable level of each axis for the impact detection function during program operation.
(2) Normally, the setting value of the allowable level immediately after power ON is the setting value of the COLLVL parameter.
(3) "All axes 100%" is set as the initial value of the COLLVL parameter.
(4) If this value is increased, the detection level (sensitivity) lowers; if this value is lowered, the detection level increases.
(5) Please do not increase the detection level too much, as it increases the possibility of erroneous detection. Erroneous detection may also occur even with the initial value depending on the posture and operating speed. In such a case, lower the sensitivity level.
(6) Erroneous detection may occur if the hand weight (HNDDATn parameter) and workpiece weight (WRKDATn parameter) are not set correctly. Be sure to set these parameters correctly before using.
(7) When the continuity function is enabled, the previously set value is restored at next power ON even if the power is turned OFF.
(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 impact detection function, the instruction is ignored and thus no error occurs.
(10) Currently, the impact detection function does not work even if the J7 and J8 axes are set. They are reserved for future expansion.

[Related instructions and variables]

COLCHK (Col Check), M_COLSTS, J_COLMXL, P_COLDIR

[Related parameter]

COL,COLLVL

[Available robot type]

RV-3S/3SJ/3SB/3SJB series
RV-6S/6SL/12S/12SL series
RH-6SH/12SH/18SH series

COM ON/COM OFF/COM STOP (Communication ON/OFF/STOP)

[Function]

COM ON	:Allows interrupts from a communication line.
COM OFF	:Prohibits interrupts from a communication line.
COM STOP	:Prevents interrupts from a communication line temporarily (data is received). Jump immediately to the interrupt routine the next time the COM ON instruction is executed.

[Format]

```
COM[(<Communication Line No.>)][]ON
```

```
COM[(<Communication Line No.>)][]OFF
```

```
COM[(<Communication Line No.>)][]STOP
```

[Terminology]

<Communication Line No.> Describes numbers 1 to 3 assigned to the communication line. (If the argument is omitted, 1 is set as the default value.)

[Reference Program]

Refer to Page 195, "ON COM GOSUB (ON Communication Go Subroutine)".

[Explanation]

(1) When COMMON 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 198, "OPEN (Open)".

(3) After COM STOP is executed, even if communication is attempted, the interrupt will not be generated. Note that the receiving data and the fact of the interrupt will be recorded, and be executed the next time the line is reopened.

DEF ACT (Define act)

[Function]

This instruction defines the interrupt conditions for monitoring signals concurrently and performing interrupt processing during program execution, as well as the processing that will take place when an interrupt occurs.

[Format]

DEF[]ACT[]<Priority No.>, <Expression>[]<Process> [, <Type>]

[Terminology]

<Priority No.>　This is the priority No. of the interrupt. It can be set with constant Nos. 1 to 8.

<Expression>　For the interrupt status, use the formats described below: (Refer to the syntax diagram)
<Numeric type data> <Comparison operator> <Numeric type data> or
<Numeric type data> <Logical operator> <Numeric type data>
<Numeric type data> refers to the following:
<Numeric type constant>| <Numeric variable>|<Numeric array variable>|
<Component data>

<Process>　Refers to a GOTO statement or a GOSUB statement used to process an interrupt when it occurs.

<Type>　When omitted: Stop type 1
The robot stops at the stop position, assuming 100% execution of the external override. If the external override is small, the time required for the robot to stop becomes longer, but it will always stop at the same position.
S : Stop type 2 (only for software version E3 or later)
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 line being executed is completed).

[Reference Program]

```
10 DEF ACT 1,M_IN(17)=1 GOSUB 100          ' Defines the subroutine at line 100 to be the one to be
                                             called up when the status for the general purpose input
                                             signal No. 17 is ON.
20 DEF ACT 2,MFG1 AND MFG2 GOTO 200        ' Defines the line 200 as the one to jump to when  the
                                             logic operation of AND applied to  MFG1 or MFG2
                                             results in "true."
30 DEF ACT 3,M_TIMER(1)>10.5 GOSUB *LBL    ' When 10.5 seconds pass, the program jumps to the
                                             line 300 subroutine.
    :
100 M_TIMER(1)=0                           ' Sets the timer to zero.
110 ACT 3=1                                ' Enables ACT 3.
120 RETURN 0
200 MOV P_SAFE
210 END
300 *LBL
310 M_TIMER(1)=0.0                         ' Resets the timer to zero.
320 ACT 3=0                                ' Disables ACT 3.
320 RETURN 0
```

[Explanation]

(1) The priority level for the interrupts is decided by the <Priority No.>, and the priority level, from the highest ranges from 1 to 8.

(2) There can be up to 8 settings for the interrupts. Use the <Priority No.> to differentiate them.

(3) An <expression> should be either a simple logical operation or a comparison operation (one operator). Parentheses cannot be used either.

(4) If two DEF ACT instructions with the same priority number are included in a program, the latter one defined becomes valid.

(5) Since DEF ACT defines only the interrupt, always use the ACT command to designate the enable/disable status of the interrupt.

(6) The communications interrupt (COM) has a higher priority level than any of the interrupts defined by DEF ACT.

(7) DEF ACT definitions are valid only in the programs where they are defined. These are invalid when called up in a program by CALLP. If necessary, the data in a sub program may need to be redefined.

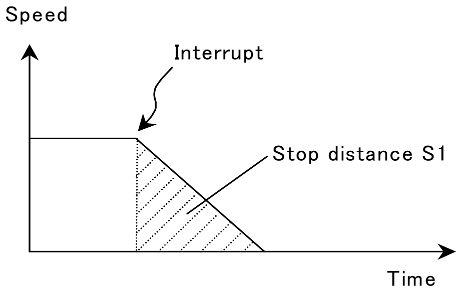
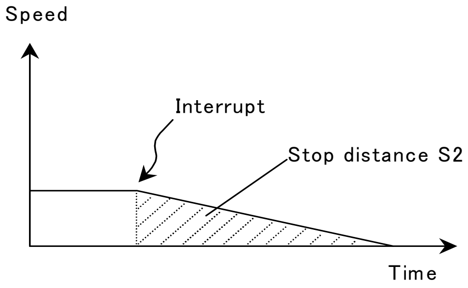
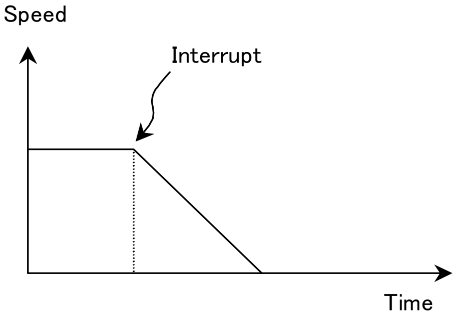
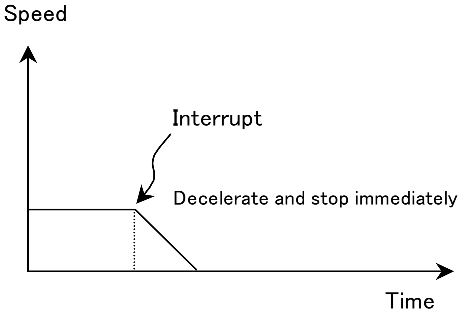
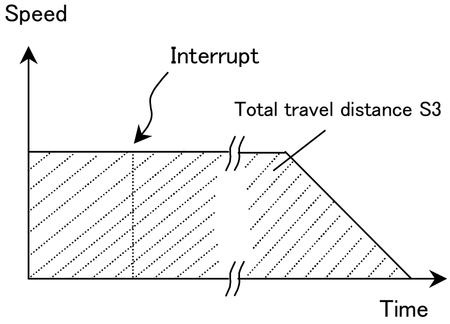
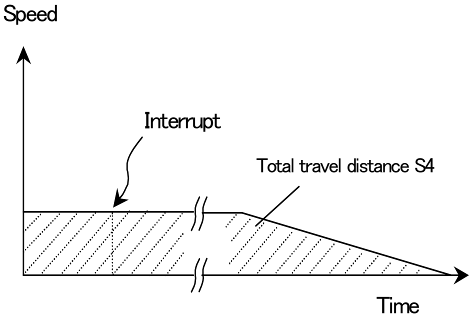
(8) If an interrupt is generated when a GOTO command is designated by <Process> for a DEF ACT command, during execution of the remaining program, the interrupt in progress will remain, and only interrupts of a higher level will be accepted. The interrupt in progress for a GOTO statement can be canceled with the execution of an END statement.

(9) Expressions containing conditional expressions combined with logical operations, such as (M1 AND &H001) = 1, are not allowed.

⚠CAUTION Specify the proper interrupt stop type according to the purpose. Specify "S" for the stop type if it is desired to stop the robot in the shortest time and distance possible by an interrupt while the robot is executing a movement instruction.

The following conceptual diagrams illustrate the effects of the three types of execution program stop commands when the interrupt conditions are met while the robot is moving according to a movement instruction.

Table 4-14: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]

This function is available for controller software version G2 or later.

```
DEF[]ARCH[]<Arch number>, [<upward movement increment>][<downward movement increment >],
                          [<Upward evasion increment>], [<downward evasion increment>],
                          [<interpolation type>], [<interpolation type 1>, <interpolation type 2> ]
```

[Terminology]

<Arch number> Arch motion movement pattern number. Specify a number from 1 to 4 using a constant or a variable.

<Upward movement increment>

<Downward movement increment >

Refer to figure at right. It is possible to specify either a constant or a variable.

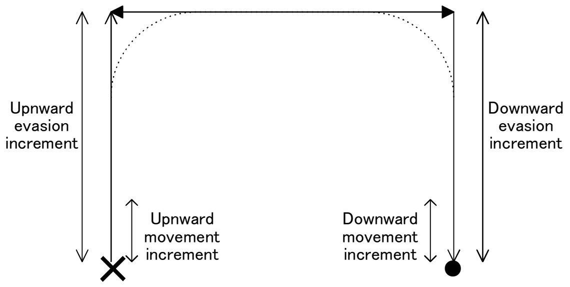
<Upward evasion increment>

<Downward evasion increment>

<Interpolation type> Interpolation type for upward and downward movements.
Linear/joint = 1/0

<Interpolation type 1> Detour/short cut = 1/0,

<Interpolation type 2> 3-axis XYZ/Equivalent rotation = 1/0


Upnward evasion increment
Downward evasion increment
Upnward movement increment
Downward movement increment

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-1A/2AJ, RV-4A/5AJ, etc.)

Parameter name	Arch number	Interpolation type	Interpolation type 1	Interpolation type 2
ARCH1T	1	1	0	0
ARCH2T	2	1	0	0
ARCH3T	3	1	0	0
ARCH4T	4	1	0	0

Horizontal multi-joint robot(RH-5AH **, etc.)

Parameter name	Arch number	Interpolation type	Interpolation type 1	Interpolation type 2
ARCH1T	1	0	0	0
ARCH2T	2	0	0	0
ARCH3T	3	0	0	0
ARCH4T	4	0	0	0

[Reference Program]

```
10 DEF ARCH 1,5,5,20,20
20 MVA P1,1             'Performs the arch motion movement defined in the shape definition in line 10.
30 MVA P2,2             'The robot moves according to the default values specified by the parameters.
```

[Explanation]

(1) If the MVA instruction is executed without the DEF ARCH instruction, the robot moves according to the arch shape specified by the parameters.

(2) Used to change the increments in a program, etc.

[Related instructions]

MVA (Move Arch), ACCEL (Accelerate), OVRD (Override), MVS (Move S)(Used as a reference for interpolation types 1 and 2)

DEF CHAR (Define Character)

[Function]

Declares a character string variable. It is used when using a variable with a name that begins with a character other than "C." It is not necessary to declare variables whose names begin with the character "C" using the DEF CHAR instruction.

[Format]

```
DEF[]CHAR[]<Character string variable name>
                 [, <Character string variable name>...
```

[Terminology]

<Character string variable name> Designate a variable name.

[Reference Program]

```
10 DEF CHAR MESSAGE            ' Declare "MESSAGE" as a character string variable.
20 MESSAGE = "WORKSET"         ' Substitute "WORKSET" in the MESSAGE variable.
30 CMSG = "ABC"                ' Substitute "ABC" for variable CMSG. For variables starting with
                                 C, the definition of "DEF CHAR" is not required.
```

[Explanation]

(1) The variable name can have up to eight characters. Refer to the Page 96, "4.3.6 Types of characters that can be used in program" for the characters that can be used.

(2) When designating multiple variable names, the maximum value (127 characters including command) can be set on one line.

(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 105, "4.3.24 User-defined external variables" for details.

DEF FN (Define function)

[Function]

Defines a function and gives it name.

[Format]

```
DEF[]FN <Identification  character><Name> [(<Dummy Argument> [, <Dummy Argument>]...)]
                    = <Function Definition Expression>
```

[Terminology]

<Identification character> The identification character has the following four type.
Numeric value type:M
Character string type:C
Position type:P
Joint type:J

<Name> Describe a user-selected character string. (5 is the maximum)

<Dummy argument> When a function has been called up, it is transferred to the function.
It is possible to describe all the variables, and up to 16 variables can be used.

<Function Definition Expression>
Describe the expression for what operation to use as a function.

[Reference Program]

```
10 DEF FNMAVE(MA,MB)=(MA+MB)/2        ' Define FNMAVE to obtain the average of two numeric values.
20 MDATA1=20
30 MDATA2=30
40 MAVE=FNMAVE(MDATA1,MDATA2)         ' Substitute average value 25 of 20 and 30 in numeric variable MAVE.
50 DEF FNPADD(PA,PB)=PA+PB            ' Position type addition.
60 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 line can be described.

(3) Built-in functions and user-defined functions that have already been defined can be used in the function definition expression. In this case, up to 16 levels of user-defined functions can be written.

(4) If the variables used in <Function Definition Expression> are not located in <Dummy Argument>, then the value that the variable has at that time will be used. Also, an error will occur if during execution, the number or argument type (numeric value or character string) of arguments differs from the number or type declared.

(5) A user-defined function is valid only in the program where it is defined. It cannot be used by a CALLP designation program.

DEF INTE/DEF FLOAT/DEF DOUBLE (Define Integer/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[] 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) The definition of the integer type variable.

```
10 DEF INTE WORK1, WORK2' Declare WORK 1 and WORK 2 as an numeric value variable name.
20 WORK1 = 100           ' Substitute the value 100 in WORK 1.
30 WORK2 = 10.562        ' Numerical "11" is set to WORK2.
40 WORK2 = 10.12         ' Numerical "10" is set to WORK2.
```

(2) The definition of the single precision type real number variable.

```
10 DEF FLOAT WORK3
20 WORK3 = 123.468       ' Numerical "123.468000" is set to WORK3.
```

(3) The definition of the double precision type real number variable.

```
10 DEF DOUBLE WORK4
20 WORK4 = 100/3         ' Numerical "33.333332061767599" is set to WORK4.
```

[Explanation]

(1) The variable name can have up to eight characters. Refer to the Page 96, "4.3.6 Types of characters that can be used in program" for the characters that can be used.
(2) When designating multiple variable names, the maximum value (123 characters including command) can be set on one line.
(3) The variable declared with INTE will be an integer type.(-32768 to +32767)
(4) The variable declared with FLOAT will be a single-precision type.(+/-1.70141E+38)
(5) The variable declared with DOUBLE will be a double-precision type.(+/-1.701411834604692E+308)

DEF IO (Define IO)

[Function]

Declares an input/output variable. Use this instruction to specify bit widths. M_IN and M_OUT variables are used for normal single-bit signals, M_INB and M_OUTB are used in the case of 8-bit bytes, and M_INW and M_OUTW are used in the case of 16-bit words.

Be aware that it is not allowed to reference output signals with variables declared using this instruction.

[Format]

```
DEF[]IO[]<Input/output variable name> = <Type designation>, <Input/output bit No.>
                                                         [, <Mask information>]
```

[Terminology]

<Input/output variable name>	Designate the variable name.
<Type designation>	Designate BIT(1bit), BYTE(8bit), WORD(16bit) or INTEGER.
<Input/output bit No.>	Designate the input(When referencing) or output(When assigning) bit No.
<Mask information>	Designate when only a specific signal is to be validated.

[Reference Program]

(1) Assign the input variable named PORT1 to input/output signal number 6 in bit type.

```
 10 DEF IO PORT1 = BIT,6
   :
100 PORT1 = 1      ' Output signal number 6 turns on.
   :
200 PORT1 = 2      ' Output signal number 6 turns off.(Because the lowest bit of the numerical value 2 is
                     0.)
210 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.

```
 10 DEF IO PORT2 = BYTE, 5, &H0F
   :
100 PORT2 = &HFF        ' Output signal number 5 to 8 turns on.
   :
200 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.

```
 10 DEF IO PORT3 = WORD, 8, &H0FFF
   :
100 PORT3 = 9                    ' Output signal number 8 and 11 turns on.
   :
200 M3 = PORT3                   ' Substitute the value of the input signals 8 to 19 for the variable
                                   M3.
```

[Explanation]

(1) An input signal is read when referencing this variable.

(2) An output signal is written when assigning a value to this variable.

(3) It is not allowed to reference an output signal by this variable. Use the M_OUT variable in order to reference an output signal.

(4) The variable name can have up to eight characters. Refer to the Page 96, "4.3.6 Types of characters that can be used in program" for the characters that can be used.

(5) When mask information is designated, only the specified signal will be validated.

Example) In the above example on the 20th line, the input/output data with a bit width of eight is masked by 0F in hexadecimal. Thus, if PORT 2 is used thereafter,

•When used as an input signal (M1 = PORT 2):

Numbers 5 to 8 are used for input, and numbers 9 to 12 are always treated as 0.

No. 12 No.5 (Input/output bit No.)

0000 1111

Invalid Valid

•When used as an output signal (PORT 2 = M1):

Data to be output this time is output to numbers 5 to 8, and the status currently being output is retained at numbers 9 to 12.

No. 12 No.5 (Input/output bit No.)

**** 1111

| |

Retains the current output status Output data of this time

DEF JNT (Define Joint)

[Function]

This instruction declares joint type position variables. It is used when using a variable with a name that begins with a character other than "J." It is not necessary to declare variables whose names begin with the character "J" using the DEF JNT instruction.

[Format]

```
DEF[]JNT[] <Joint variable name> [, <Joint variable name>]...
```

[Terminology]

<Joint variable name>   Designate a variable name.

[Reference Program]

```
10 DEF JNT SAFE                    ' Declare "SAFE" as a joint variable.
20 MOV J1                          ' For joint type position variables starting with J, the definition of
                                     "DEF JNT" is not required.
30 SAFE = (-50,120,30,300,0,0,0,0)
40 MOV SAFE                        ' Move to SAFE.
```

[Explanation]

(1) Use this instruction to define a joint position variable by a name beginning with a character other than J.

(2) The variable name can have up to eight characters. Refer to the Page 96, "4.3.6 Types of characters that can be used in program" for the characters that can be used. When designating multiple variable names, the maximum value (127 characters including command) can be set on one line.

(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 105, "4.3.24 User-defined external variables" for details.

DEF PLT (Define pallet)

[Function]

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

[Format]

```
DEF[]PLT[] <Pallet No.>, <Start Point>, <End Point A>, <End Point B>, [<Diagonal Point>],
                              <Quantity A>, <Quantity B>, <Assignment Direction>
```

[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.

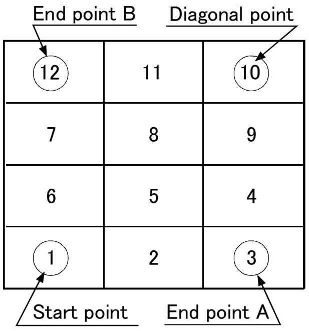
<Diagonal Point> The diagonal point from the pallet's start point. Insignificant for arc pallet.

<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.

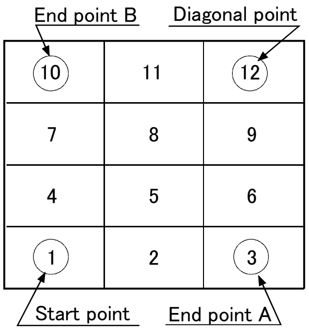
<Quantity B> The No. of workpieces from the pallet's start point to the end point B. Insignificant for an arc pallet. (1, etc., must be designated.)

<Assignment Direction> Describes the direction of the number assignment when numbering divided grid points.

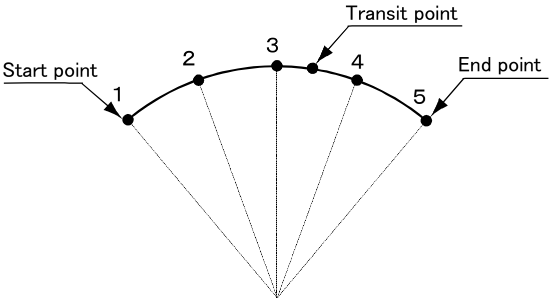
1 : Zigzag 2 : Same direction 3 : Arc pallet



Zigzag



Same direction



Arc pallet

[Reference Program]

```
10 DEF PLT 1,P1,P2,P3, ,4,3,1       ' Define a 3-point pallet.
20 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. The diagonal point will be insignificant even when designated. Set a dummy value.

(6) If the hand is facing downward, the signs of the A, B, and C axis coordinates at the starting point, end-point A, and endpoint B 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.

Please refer to the illustrations in 4.1.2Pallet 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]

```
10 DEF POS WORKSET                        ' Declare "WORKSET" as the XYZ type position variable.
20 MOV P1                                 ' For XYZ type position variables starting with P, the defini-
                                            tion of "DEF POS" is not required.
30 WORKSET=(250,460,100,0,0,-90,0,0)(0,0)
40 MOV WORKSET                            ' Move to WORKSET.
```

[Explanation]

(1) Use this instruction to define a XYZ type position variable by a name beginning with a character other than P.

(2) The variable name can have up to eight characters. Refer to the Page 96, "4.3.6 Types of characters that can be used in program" for the characters that can be used.

(3) When designating multiple variable names, the maximum value (127 characters including command) can be set on one line.

(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 105, "4.3.24 User-defined external variables" for details.

DIM (Dim)

[Function]

Declares the quantity of elements in the array variable. (Arrays up to the third dimension are possible.)

[Format]

```
DIM[]<Variable name> (<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.

[Reference Program]

```
10 DIM PDATA(10)            ' Define the position array variable PDATA having ten elements.
20 DIM MDATA#(5)            ' Define double-precision type array variable MDATA# having the five
                              elements.
30 DIM M1%(6)               ' Define integer-type array variable M1% having the six elements.
40 DIM M2!(4)               ' Define single-precision real number type array variable M2! having the
                              four elements.
50 DIM CMOJI(7)             ' Define the character-string type variable CMOJI having the seven ele-
                              ments.
60 DIM MD6(2,3), PD1(5,5)   ' Define the 2-dimensional single precision real number type array vari-
                              able MDATA having the element of 2x3.
                            ' Define the 2-dimensional position array variable PD 1 having the ele-
                              ment 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 ele-
                ments.
```

(3) Eelement number start from 1 when actually referencing array variables. For PDATA on line 10 of the statement example, the element number will be 1 to 10.

(4) <Eelement Value> can be described with numeric constants from 1 to 999. It is not allowed to use a numerical value operation expression.
If the number of elements is specified using a real number, an integer with rounded decimal part will be assumed. Depending on the system memory's free space, arrays may not be allocated for the number of specified elements. In this case, an error will occur when lines are registered.

(5) If an element number larger than the number of defined elements is specified, an error will occur at the time of execution.

(6) At the point when array variables are defined, variable values are indeterminate.

(7) To use array variables, it is necessary to define them using the DIM instruction.

(8) The arrays defined by the DIM instruction are valid only in the program where they are defined. To use these arrays by a sub program called by the CALLP instruction, it is necessary to define them again.

(9) Array variables can be used similar to normal variables. However, note that variables of which variable names and/or the number of characters for specifying element numbers exceed eight characters cannot be used on the monitor variable screen and position edit screen of the teaching pendant.

(10) If a variable name whose second character is underlined "_" is registered in a user program, a user defined external variable (a variable common among programs) will be assumed..
Refer to Page 105, "4.3.24 User-defined external variables" for details.

DLY (Delay)

[Function]

1) When used as a single command:
   At a designated time, it causes a wait. It is used for positioning the robot and timing input/output signals.
2) When used as an additional pulse output:
   Designates an output time for a pulse.

[Format]

1) When used as a single command

```
DLY[]<Time>
```

2) When used as an additional pulse output

```
Example) M_OUT(1) = 1 DLY[]<Time>
```

[Terminology]

<Time> Describes the waiting time or the output time for the pulse output, in terms of a numeric operation expression. Unit: [Seconds]
The minimum value that can be set is 0.01 seconds. It is allowed to specify 0.00 as well.

[Reference Program]

(1) Waiting for time

```
10 DLY 30                    ' Wait for 30 seconds
```

(2) Pulse output of the signal

```
20 M_OUT(17)=1 DLY 0.5       ' Send the signal output to the general-purpose output signal 17
                               for 0.5 seconds.
30 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.

```
10 MOV P1                    ' Moves to P1.
20 DLY 0.1                   ' Positions to 1.
```

(4) Wait for completion of hand opening. (closing)

```
10 HOPEN 1                   ' Open the hand 1.
20 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 line 20 in [statement example] above).
(2) The pulse output will be executed simultaneously as the next command in the lines 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 line 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 line 20 in the following program, the output signal state will be held, and the execution is stopped.

```
10 M_OUT(17)=1
20 DLY 10
30 M_OUT(17)=0
```

Note2) If a pulse output by the M_OUTB (8-bit signal) or the M_OUTW (1 6-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.

ERROR (error)

[Function]

This instruction makes a program generate an error (9000s number).

[Format]

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

[Terminology]

<Error No.>   Either a constant or numeric operation expression can be set. Designate the No. within the range of 9000 to 9299.

[Reference Program]

(1) Generate the error 9000.

```
 100 ERROR 9000
```

(2) Change the error number to generate corresponding to the value of M1.

```
 40 IF M1 <> 0 THEN *LERR     ' When M1 is not 0, branches to "*LERR".
  :
140 *LERR
150   MERR=9000+M1*10         ' Calculate the error number according to the value of M1.

160   ERROR MERR              ' The calculated error number is generated.
170 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.
Lines after the ERROR instruction are not executed. A CAUTION error does not pause a program; the next line and onward are executed. The action of system by error number is shown in the Table 4-15.

(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 4-15 is specified.

Table 4-15: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

END (End)

[Function]

This instruction defines the final line 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]

```
10 MOV P1
20 GOSUB *ABC
30 END                    ' End the program.
  :
100 *ABC
110   M1=1
120 RETURN
```

[Explanation]

(1) This instruction defines the final line 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)

FINE (Fine)

[Function]

This instruction specifies completion conditions of the robot's positioning. It is invalid during the smooth movement control (CNT 1).

Depending on the type of robot (RP series), positioning using the DLY instruction may be more effective than using the FINE instruction.

[Format]

```
FINE[]<No. of pulses> [, <Axis No.>]
```

[Terminology]

<No. of pulses>   Specify the positioning pulses number.
This will be invalid to when set to 0. The default value is 0.

<Axis No.>   Designate the axis No. to which the positioning pulses are to be designated. The positioning pulses will be applied on all axes when omitted.

[Reference Program]

```
10 FINE 300       ' Designate 300 for the positioning pulses.
20 MOV P1
30 FINE 100,2     ' Change the 2nd axis positioning pulses to 100.
40 MOV P2
50 FINE 0         ' Invalidate the positioning pulse designation.
60 MOV P3
70 FINE 100       ' Designate 100 for the positioning pulses.
80 MOV P4
```

[Explanation]

(1) The FINE instruction does not complete movement instructions such as MOV by giving commands to the servo; rather, it completes positioning by determining whether or not the feedback pulse value from the servo is within the specified range. It is thus possible to confirm positioning more accurately.

(2) There are cases when the DLY instruction (timer) is used for positioning instead of the FINE instruction. This instruction is easier to specify.

```
10 MOV P1
20 DLY 0.1
```

(3) FINE is invalid in the program until the FINE command is executed. Once FINE is validated, it remains valid until invalidated.

(4) FINE is invalidated at the end of the program (Execution of the END instruction, program reset after pausing).

(5) When the continuous movement control valid state (CNT 1) is entered, the FINE command will be ignored even if it is valid (i.e., it will be treated as invalid, but the status will be kept).

(6) To the addition axis (general-purpose servo axis), although the valid/invalid change of FINE is possible, specification of the pulse number cannot be performed. The value registered in the "INP" parameter on the servo amplifier side is used. Thus, when the integers other than zero are specified, the FINE becomes effective by the parameter set value of servo amplifier, and the FINE becomes invalid when 0 is specified.

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

```
10 MSUM=0                ' Initialize the total MSUM.
20 FOR M1=1 TO 10        ' Increase the counter by 1 from 1 to 10 for the numeric variable M1.
30  MSUM=MSUM+M1         ' Add M1 value to numeric variable MSUM.
40 NEXT M1               ' Return to line 20.
```

(2) A program that puts the result of a product of two numbers into a 2-dimensional array variable

```
10 DIM MBOX(10,10)       ' Reserve space for a 10Å~10 array.
20 FOR M1=1 TO 10 STEP 1 ' Increase the counter by 1 from 1 to 10 for the numeric variable M1.
30   FOR M2=1 TO 10 STEP 1 ' Increase the counter by 1 from 1 to 10 for the numeric variable M2.
40   MBOX(M1,M2)=M1*M2   ' Substitute the value of M1*M2 for the array variable MBOX (M1, M2).
50   NEXT M2             ' Return to line 30.
60 NEXT M1               ' Return to line 20.
```

[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 line 50 and "M1" in line 60 can be omitted. The processing speed will be slightly faster to omit the counter variable.

FPRM (FPRM)

[Function]

Defines the order of the arguments, the type, and number for the main program that uses arguments in a sub program (i.e., when the host program uses another program with CALL P).

[Format]

```
FPRM[]<Dummy Argument> [,<Dummy Argument>] ...
```

[Terminology]

<Dummy Argument>   The variable in the sub program that is transferred to the main statement when executed. All variables can be used. Up to 16 variables may be used.

[Reference Program]

```
<Main program>
10 M1=1
20 P2=P_CURR
30 P3=P100
40 CALLP "100",M1,P2,P3        ' It can be described like "CALLP "100", 1, P_CURR, P100" also.

<Sub program "100">
10 FPRM M1,P1,P2
20 IF M1=1 THEN GOTO 40
30 MOV P1
40 MVS P2
50 END                          ' Return to the main program.
```

[Explanation]

(1) FPRM is unnecessary if there are no arguments in the sub program that is called up.

(2) An error occur when the type or number is different between the argument of CALLP and the dummy argument that defined by FPRM.

(3) It is not possible to pass the processing result of a sub program to a main program by assigning it in an argument.
To use the processing result of a sub program in a main program, pass the values using external variables.

[Related instructions]

CALLP (Call P)

GETM (Get Mechanism)

[Function]

This instruction is used to control the robot by a program other than the slot 1 program when a multi-task is used, or to control a multi-mechanism by setting an additional axis as a user-defined mechanism.

Control right is acquired by specifying the mechanism number of the robot to be controlled. To release control right, use the RELM instruction.

[Format]

```
GETM[]<Mechanism No.>
```

[Terminology]

<Mechanism No.> 1 to 3, Specify this argument using a numerical or a variable.
The standard system's robot arm is assigned to mechanism 1.

[Reference Program]

(1) Start the task slot 2 from the task slot 1, and control the mechanism 1 in the task slot 2.

```
Task slot 1.
10 RELM                 ' Releases the mechanism in order to control mechanism 1 using slot 2.
20 XRUN  2,"10"         ' Start the program 10 in slot 2.
30 WAIT M_RUN(2)=1      ' Wait for the starting confirmation of the slot 2.
  :

Task slot 2. (Program "10")
10 GETM 1          ' Get the control of mechanism 1.
20 SERVO ON        ' Turn on the servo of mechanism 1.
30 MOV P1
40 MVS P2
50 P3=P_CURR       ' Substitute P3 in mechanism 1 current position.
60 SERVO OFF       ' Turn mechanism 1 servo OFF.
70 RELM            ' Releases the control right of mechanism 1.
80 END
```

[Explanation]

(1) Normally (in single task operation), mechanism 1 is obtained in the initial status; it is not necessary to use the GETM instruction.

(2) Because the control right of the same mechanism cannot be acquired simultaneously by multiple tasks, the following procedure is required in order to operate the robot by other than slot 1:
First, release control right using the RELM instruction by the slot 1 program. Next, acquire control right using the GETM instruction by the slot program that operates the robot. An error will be generated if the GETM instruction is executed again using a slot that has already acquired control right.

(3) The instructions requiring control right include the motor power ON/OFF instruction, the interpolation instruction, the speed acceleration deceleration specification instruction, and the TOOL/BASE instruction.

(4) If the argument is omitted from the system status variable requiring the mechanism designation, the currently acquired mechanism will be designated.

(5) If the program is stopped, RELM will be executed automatically by the system. When the program is restarted, GETM will be executed automatically.

(6) This instruction cannot be used in a constantly executed program.

[Related instructions]

RELM (Release Mechanism)

GOSUB (RETURN)(Go Subroutine)

[Function]

Calls up the subroutine at the designated line No. or line label. Be sure to return from the jump destination using the RETURN instruction.

[Format]

```
GOSUB[]<Call Destination>
```

[Terminology]

<Call Destination>   Describe the line No. or label name.

[Reference Program]

<For a line number>

```
100  GOSUB 1000
110  END

1000 MOV P1
1010 RETURN              ' Be sure to use the RETURN instruction to return.
```

<For a label>

```
100  GOSUB *LBL
110  END

1000 *LBL
1010 MOV P1
1020 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) The line number or label can be specified as a jump destination.
When the line 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 line number or label line unconditionally.

[Format]

```
GOTO[]<Branch Destination>
```

[Terminology]

<Branch Destination>   Describe the line No. or label name.

[Reference Program]

(1) Specify the line number.

```
 10 MOV P1
  :
100  GOTO 10            ' Returns to the line number 10.
110  END
```

(2) Specify the label.

```
 100  GOTO *LBL         ' Branches to the label *LBL.
   :
1000 *LBL
1010 MOV P1
```

[Explanation]

(1) A line number or label can be specified as a branch destination.
(2) If a branch destination or label does not exist, an error will occur during execution.

HLT (Halt)

[Function]

Interrupts the execution of the program and movement of the robot, and stops. The program which was being executed at this time becomes standby status.

[Format]

```
HLT
```

[Reference Program]

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

```
150 HLT                                ' Stop the program without condition.
```

(2) Stop the robot on some conditions.

```
100 IF M_IN(18)=1 THEN HLT             ' Stop the program execution when the input signal 18 turns on.

200 MOV P1 WTHIF M_IN(17)=1, HLT ' When the input signal 17 turns on during moving to P1, the pro-
                                   gram execution is stopped.
```

[Explanation]

(1) Interrupts the execution of a program and decelerates the robot to a stop. The system will enter the waiting state.

(2) If the HLT instruction is used in multitask operation, only the task slot that executed the HLT instruction is paused.

(3) To restart, start the O/P or issue the start signal from an external source. The program will be restarted at the next line after the HLT statement. Note that if the HLT statement is an appended statement, the operation will restart from the same line of the program where it was interrupted.

[Related instructions]

END (End)

HOPEN / HCLOSE (Hand Open/Hand Close)

[Function]

Commands the hand to open or close.

[Format]

```
HOPEN[]<Hand No.> [, <Starting grasp force>, <Holding grasp force>,
                         <Starting grasp force holding time>]
HCLOSE[]<Hand No.>
```

[Terminology]

<Hand No.> Select a numeric value between 1 and 8. Specify this argument using a constant or a variable.

<Starting grasp forcer> This parameter is valid for the motorized hand, and invalid for any other type of hand.
Set the required grasping force for starting the hand open/close.
Set the grasping force as a step between 0 and 63 (63 = 3.5kg).
The default value is 63. When omitted, the previous setting value will be applied.

<Holding grasp force> This parameter is valid for the motorized hand, and invalid for any other type of hand.
Set the required grasping force for holding the hand open/close.
Set the grasping force as a step between 0 and 63 (63 = 3.5kg).
The default value is 63. When omitted, the previous setting value will be applied.

<Starting grasp force holding timer>This parameter is valid for the motorized hand.
Set the time to hold the starting grasping force as a value from 0.00 (sec.).
The default value is 0.3 sec.

[Reference Program]

```
10 HOPEN 1        ' Open hand 1.
20 DLY 0.2        ' Set the timer to 0.2 sec. (Wait for the hand to open securely.)
30 HCLOSE 1       ' Close hand 1.
40 DLY 0.2        ' Set the timer to 0.2 sec. (Wait for the hand to close securely.)
50 MOV PUP        '
```

[Explanation]

(1) The operation (single/double) of each hand is set with parameter HANDTYPE.

(2) If the hand type is set to double solenoid, hands 1 to 4 can be supported. If the hand type is set to single solenoid, hands 1 to 8 can be supported.

(3) The status of the hand output signal when the power is turned ON is set with parameter HANDINIT.

(4) The hand input signal can be confirmed with the robot status variable M_HNDCQ ("Hand input number"). The signal can also be confirmed with the input signals No. 900 to 907 (when there is one mechanism).

```
10 HCLOSE 1
20 IF M_HNDCQ(1)<>1 THEN GOTO 20
30 MOV P1
```

(5) There are related parameters. Refer to Page 330, "5.10 Automatic return setting after jog feed at pause" and, Page 334, "5.13 About default hand status" of this manual.

[Related system variables]
M_IN/M_INB/M_INW (900s number), M_OUT/M_OUTB/M_OUTW (900s number), M_HNDCQ

[Related instructions]
LOADSET (Load Set), OADL (Optimal Acceleration)

[Related parameter]
HANDTYPE, HANDINIT
Refer to Page 330, "5.10 Automatic return setting after jog feed at pause"and, Page 334, "5.13 About default hand status".

IF...THEN...ELSE...ENDIF (If Then Else)

[Function]

A process is selected and executed according to the results of an expression.

[Format]

```
IF[]<Expression>[]THEN[]<Process>[][ELSE <Process>]
```

This function is available for controller software version G1 or later.

This BREAK command is available for controller software version J1 or later.

```
IF[]<Expression>[]THEN
   <Process>
   <Process>
   BREAK
     :
[ELSE]
   <Process>
   <Process>
   BREAK
     :
ENDIF
```

[Terminology]

<Expression> Describe the expression targeted for comparison as a comparison operation expression or logic operation expression.

<Process> Describe the process following THEN for when the comparison results are true, and the process following ELSE for when the comparison results are false.

[Reference Program]

(1) The software version earlier than G1 edition.

```
100 IF M1>10 THEN 1000                        ' When M1 is larger than 10, jump to the line
                                                number 1000.
110 IF M1>10 THEN GOTO 200 ELSE GOTO 300      ' If M1 is larger than 10, it jumps to line number
                                                200; if smaller than 10, it jumps to line num-
                                                ber 300.
                                                The "GOTO" after" THEN" or "ELSE" can be
                                                omitted.
  :
200 M1=10
210 MOV P1
220 GOTO 400
300 M1=-10
310 MOV P2
320 GOTO 400
```

(2) The software version is G1 edition or later.

```
100 IF M1>10 THEN
110  M1=10
120   MOV P1
130 ELSE
140  M1=-10
150   MOV P2
160 ENDIF
```

* The description method of earlier than G1 edition is also possible.

```
250 IF M2=0 THEN GOSUB *SUB1 ELSE GOSUB *SUB2
```

(3) When a IF statement is described inside THEN or ELSE (allowed in revision G1 and later)

```
300 IF M1>10 THEN
310   IF M2 > 20 THEN
320     M1 = 10
330     M2 = 10
340   ELSE
350     M1 = 0
360     M2 = 0
370   ENDIF
380 ELSE
390   M1 = -10
400   M2 = -10
410 ENDIF
```

(4) In the THEN or the ELSE, it can escape to the next line of ENDIF by BREAK.(Version J1 or later,)

```
300 IF M1>10 THEN
310   IF M2 > 20 THEN BREAK        ' If the conditions are met, branches to line 390
320   M1 = 10
330   M2 = 10
340 ELSE
350   M1 = -10
360   IF M2>20 THEN BREAK          ' If the conditions are met, branches to line 390
370   M2 = -10
380 ENDIF
390 IF M_BRKCQ=1 THEN HLT
400 MOV P1
```

[Explanation]

(1) The IF .. THEN .. ELSE .. statements should be contained in one line.
(2) It is allowed to split an IF .. THEN .. ELSE .. ENDIF block over several lines.
(3) ELSE can be omitted.
(4) Make sure to include the ENDIF statement in the IF .. THEN .. ELSE .. ENDIF block.
(5) If the GOTO instruction is used to jump out from inside an IF .. THEN .. ELSE .. ENDIF block, an error will occur when the memory for control structure (stack memory) becomes insufficient.
(6) For IF .. THEN .. ELSE .. ENDIF, it is possible to describe IF .. THEN .. ELSE .. ENDIF inside THEN or ELSE. (UP to eight levels of nesting is allowed.)
(7) GOTO following THEN or ELSE may be omitted.

Example) IF M1 > 10 THEN 200 ELSE 300
Also, only when THEN is followed by GOTO, either one of THEN or GOTO may be omitted.
ELSE cannot be omitted.

Example) IF M1 > 10 THEN GOTO 200 (The program at left can be rewritten as shown below.)
--- IF M1 > 10 THEN 200
--- IF M1 > 10 GOTO 200

(8) In the THEN or the ELSE, it can escape to the next line of ENDIF by BREAK. That is, process of IF THEN ENDIF can be skipped..(Version J1 or later,)

INPUT (Input)

[Function]

Inputs data into a file (including communication lines). Only ASCII character data can be received.
Please refer to Page 337, "5.15 About the communication setting", which lists related parameters.

[Format]

```
INPUT[]#<File No.>, <Input data name> [, <Input data name>] ...
```

[Terminology]

<File No.> Describe a number between 1 and 8.
This corresponds to the file No. assigned with the OPEN command.

<Input data name> Describe the variable name for saving the input data. All variables can be described.

[Reference Program]

```
10 OPEN "COM1:" AS #1  ' Assign RS-232-C to file No. 1.
20 INPUT #1, M1        ' The value will be set to the numerical variable M1 if data are inputted from the
                         keyboard.
30 INPUT #1, CABC$     '
 :
100 CLOSE #1
```

[Explanation]

(1) Data is input from file having the file No. opened with the OPEN statement, and is substituted in the variable. If the OPEN statement has not been executed, an error will occur.
(2) The type of data input and the type of variable that is substituting it must be the same.
(3) When describing multiple variable names, use a comma (,) between variable names as delimiters.
(4) When the INPUT statement is executed, the status will be "standby for input. "The input data will be substituted for the variables at the same time as the carriage return (CR and LF) are input.
(5) If the protocol (in the case of the standard port: the "CRPC232" parameter is 0) of the specified port is for PC support (non procedure), it is necessary to attach "PRN" at the head of any data sent from a PC. Normally, the standard port is connected to a PC and used for transferring and debugging robot programs. Therefore, it is recommended to use the optional expansion serial interface if a data link is used.
(6) If the number of elements input is greater than the number of arguments in the INPUT statement, they will be read and discarded.
When the END or CLOSE statement is executed, the data saved in the buffer will be erased.
Example) To input both a character string, numeric value and position.

```
10 INPUT #1,C1$,M1,P1
```

Data sent from the PC side
(when received by the standard port of the robot: the "CRPS232" parameter is 0)

```
PRNMELFA,125.75,(130.5,-117.2,55.1,16.2,0,0)(1,0) CR
```

MELFA is substituted in C1$, 125.75 in M1, and (130.5, -117.2,55.1,16.2,0,0)(1,0) in P1.

[Related instructions]

OPEN (Open), CLOSE (Close), PRINT (Print)

JOVRD (J Override)

[Function]

Designates the override that is valid only during the robot's joint movements.

[Format]

```
JOVRD[]<Designated override>
```

[Terminology]

<Designated override>	Describe the override as a real number. A numeric operation expression can also be described. Unit: [%] (Recommended range: 1 to 100.0)

[Reference Program]

```
10 JOVRD 50
20 MOV P1
30 JOVRD M_NJOVRD    ' Set the default value.
```

[Explanation]

(1) The JOVRD command is valid only during joint interpolation.

(2) The actual override is = (Operation panel (T/B) override setting value) x (Program override (OVRD command)) x (Joint override (JOVRD command)). The JOVRD command changes only the override for the joint interpolation movement.

(3) The 100% <Designate override> is the maximum capacity of the robot. Normally, the system default value (M_NOVRD) is set to 100%. The value is reset to the default value when the END statement is executed or the program is reset.

[Related instructions]

OVRD (Override), SPD (Speed)

[Related system variables]

M_JOVRD/M_NJOVRD/M_OPOVRD/M_OVRD/M_NOVRD

(M_NJOVRD:System default value, M_JOVRD:Currently specified joint override)

JRC (Joint Roll Change)

[Function]

- This instruction rewrites the current coordinate values by adding +/-360 degrees to the current joint coordinate values of the applicable axis (refer to <Axis No> in [Terminology]) of the robot arm.
- User-defined axis (additional axis, user defined mechanism)
  This instruction rewrites the current coordinate values by adding/subtracting the value specified by a parameter to/from the current joint coordinate values of the specified axis. This instruction can be used for both rotating and linear axes. The origin can also be reset at the current position.

[Format]

```
JRC < [+] 1 / -1 / 0 > [, < Axis No>]
```

<Numeric Value> can be used in the controller's software version J1 or later.

```
JRC < [+] <Numeric Value> / -<Numeric Value> / 0 > [, < Axis No>]
```

[Terminology]

<+1> The current joint angle of the designated axis is incremented by the amount designated in parameter JRCQTT(The sign can be omitted.). For the priority axes of the robot arm, it is fixed at 360 degrees.

<-1> The current joint angle of the designated axis is decremented by the amount designated in parameter JRCQTT. For the priority axes of the robot arm, it is fixed at 360 degrees.

<0> The origin for the designated axis is reset at the value designated in parameter JRCORG. This can be used only for the user-defined axis.

<Axis No> The target axis is specified with the number. The priority axes are used if omitted.
[Applicable Models and Applicable Axes]
(1)Applicable models and priority axes

RV−A series: J6 axis
RH−A series: J4 axis
RV−S series: J6 axis
RH−S series: J4 axis
RP−A series: J4 axis

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

<u>The software version J1 or later.</u>

<Numeric Value> Specify an incremental/decremental number (a multiple of 360 degrees). Description by the constant or the variable is possible (J1 edition or later is possible).
Example) +3: Increases the applicable axis angle by 1080 degrees.
-2: Decreases the angle by 720 degrees..

[Reference Program]

```
10 MOV P1       ' Moves to P1.
20 JRC 1        ' Add 360 degrees to the current coordinate values of the applicable axis.
30 MOV P1       ' Moves to P1.
```

<u>The software version J1 or later.</u>

```
10 MOV P1       ' Moves to P1.(The movement to which the J6 axis moves in the minus direction)
20 JRC +1       ' Add 360 degrees to the current coordinate values of the applicable axis.
30 MOV P1       ’ Moves to P1.
40 JRC +1       ' Add 360 degrees to the current coordinate values of the applicable axis.
50 MOV P1       ' Moves to P1.
60 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 instruction cannot be used in a constantly executed program.

[Related parameter]

JRCEXE

Set whether to enable/disable the JRC execution.
Execution disabled = 0 (default value)/execution enabled = 1

JRCQTT

Designate the amount to move (1 deg./1mm unit) when incrementing or decrementing with the JRC command in additional axis or user-defined mechanism.
For the JRC's applicable axis on the robot arm side, it is fixed at 360 degrees regardless of this setting.

JRCORG

Designate the origin for executing JRC 0. in additional axis or user-defined mechanism.
Refer to Page 306, "5 Functions set with parameters" for detail.

[Target mechanism and target axis]

- RV-1A/2AJ, RV-4A/5AJ and related models, RV-20A J6 axis
- User-defined additional axis for all mechanisms
- All user-defined mechanism axes

LOADSET (Load Set)

[Function]

This instruction specifies the condition of the hand/workpiece at execution of the OADL instruction.

[Format]

```
LOADSET[]<Hand condition No.>, <Workpiece condition No.>
```

[Terminology]

<Hand condition No.> 1 to 8.Designate the hand condition (HNDDAT 1 to 8) No. for which the weight and size are designated. In the RV-S/RH-S series, 0 (HNDDAT0) can also be set.

<Workpiece condition No.> 1 to 8. Designate the hand condition (WRKDAT 1 to 8) No. for which the weight and size are designated. In the RV-S/RH-S series, 0 (WRKDAT0) can also be set.

[Reference Program]

```
10 OADL ON
20 LOADSET 1,1    ' Hand 1(HNDDAT1) and workpiece 1(WRKDAT1) conditions.
30 MOV P1
40 MOV P2
50 LOADSET 1,2    ' Hand 1(HNDDAT1) and workpiece 2(WRKDAT1) conditions.
60 MOV P1
70 MOV P2
80 OADL OFF
```

For RV-S/RH-S series

```
10 OADL ON
20 LOADSET 1,1    ' Hand 1(HNDDAT1) and workpiece 1(WRKDAT1) conditions.
30 MOV P1
40 LOADSET 0,0    ' Hand 0(HNDDAT0) and workpiece 0(WRKDAT0) conditions.
50 MOV P2
60 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) Regarding the system initial values, HNDDAT0, WRKDAT0 and HNDHOLD0 can be changed in the RV-S/RH-S series.

(7) Refer to Page 340, "5.16 Hand and Workpiece Conditions (optimum acceleration/deceleration settings)" for details on the optimum acceleration/deceleration.

[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 340, "5.16 Hand and Workpiece Conditions (optimum acceleration/deceleration settings)".

Refer to Page 314, "Table 5-2: List Signal parameter" for the ACCMODE.

MOV (Move)

[Function]

Using joint interpolation operation, moves from the current position to the destination position.

[Format]

```
MOV[]<Target Position> [, <Close Distance>] [[]TYPE[]<Constants 1>, <Constants 2>][]
        [<Appended conditions>]
```

[Terminology]

<Movement Target Position>This is the final position for interpolation operation. This position may be specified using a position type variable and constant, or a joint variable.

<Close Distance> If this value is designated, the actual movement target position will be a position separated by the designated distance in the tool coordinate system Z axis direction (+/- direction).

<Constants 1> 1/0 : Detour/short cut. The default value is 1(detour).

<Constants 2> Invalid (Specify 0).

<Appended conditions> The WTH and WHTIF statements can be used.

[Reference Program]

```
10 MOV P1 TYPE 1,0
20 MOV J1
30 MOV (PLT 1,10),100.0 WTH M_OUT(17)=1
40 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 330, "5.10 Automatic return setting after jog feed at pause")

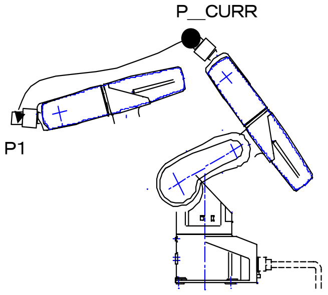


Fig.4-9:Example of joint interpolation motion path

MVA (Move Arch)

[Function]

This instruction moves the robot from the current position to the target position with an arch movement (arch interpolation).

[Format]

This function is available for controller software version G2 or later.

```
MVA[]<Target Position> [, <Arch number>]
```

[Terminology]

<Target Position>　Final position of interpolation movement. This position may be specified using a position type variable and constant, or a joint variable.

<Arch number>　A number defined by the DEF ARCH instruction (1 to 4).
If the argument is omitted, 1 is set as the default value.

[Reference Program]

```
10 DEF ARCH 1,5,5,20,20       ' Defines the arch shape configuration.
20 OVRD 100,20,20             ' Specifies override.
30 ACCEL 100,100,50,50,50,50  ' Specifies acceleration/deceleration rate.
20 MVA P1,1                   ' Performs the arch motion movement according to the shape configura-
                                tion defined in line 10.
30 MVA P2,2                   ' Moves the robot according to the default values registered in the
                                parameters.
```

[Explanation]

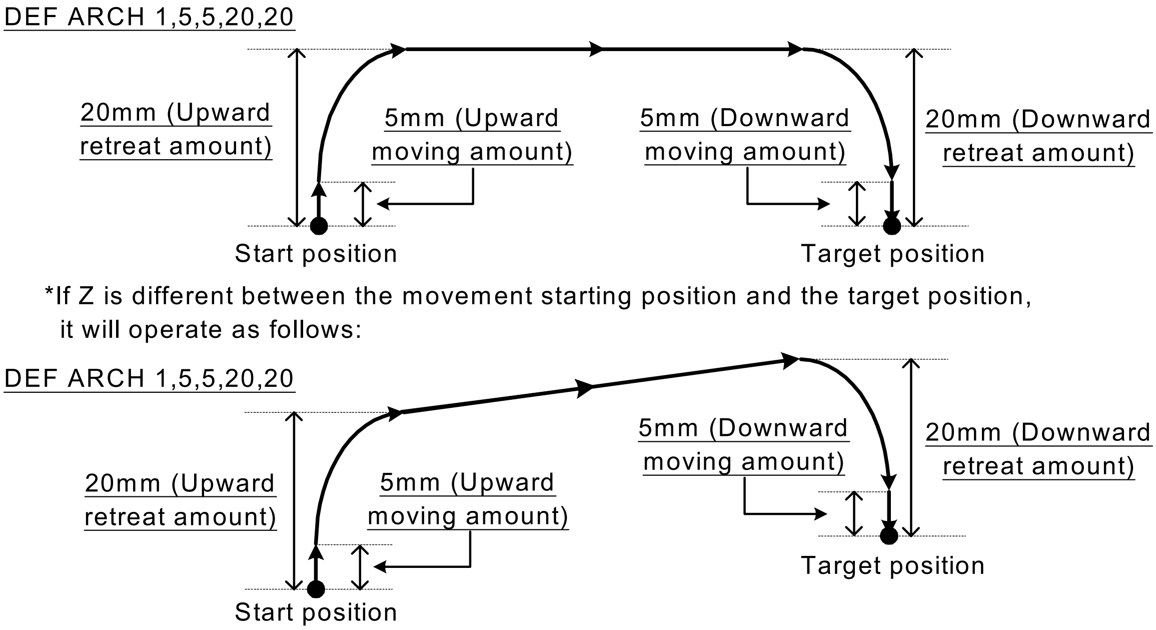
(1) The robot moves upward along the Z-axis direction from the current position, then moves to a position above the target position, and finally moves downward, reaching the target position. This so-called arch motion movement is performed with one instruction.

(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 149, " 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 149, " DEF ARCH (Define arch)".

(4) This instruction cannot be used in a constantly executed program.

(5) If paused during execution of a MVA instruction and restarted after jog feed, the robot returns to the interrupted position and restarts the MVA instruction. (this can be changed by the "RETPATH" parameter). The interpolation method (JOINT interpolation / XYZ interpolation) which returns to the interrupted position can be changed by the "RETPATH" parameter. (Refer to Page 330, "5.10 Automatic return setting after jog feed at pause")


DEF ARCH 1,5,5,20,20
20mm (Upward retreat amount)
5mm (Upward moving amount)
5mm (Downward moving amount)
20mm (Downward retreat amount)
Start position
Target position
DEF ARCH 1,5,5,20,20
20mm (Upward retreat amount)
5mm (Upward moving amount)
5mm (Downward moving amount)
20mm (Downward retreat amount)
Target position
Start position

*If Z is different between the movement starting position and the target position, it will operate as follows:

Fig.4-10:Example of arch interpolation motion path (seen from the side)

[Related instructions]

DEF ARCH (Define arch), ACCEL (Accelerate), OVRD (Override)

MVC (Move C)

[Function]

Carries out 3D circular interpolation in the order of start point, transit point 1, transit point 2 and start point.

[Format]

```
MVC[]<Start point>,<Transit point 1>,<Transit point 2>[][<Additional condition>]
```

[Terminology]

<Start point>	The start point and end point for a circle. Describe a position operation expression or joint operation expression.
<Transit point 1>	Transit point 1 for a circular arc. Describe a position operation expression or joint operation expression.
<Transit point 2>	Transit point 2 for a circular arc. Describe a position operation expression or joint operation expression.
<Additional condition>	Describe a WTH conjunction or a WTHIF conjunction

[Reference Program]

```
10 MVC P1,P2,P3
20 MVC P1,J2,P3
30 MVC P1,P2,P3 WTH M_OUT(17)=1
40 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 330, "5.10 Automatic return setting after jog feed at pause")

(5) This instruction cannot be used in a constantly executed program.

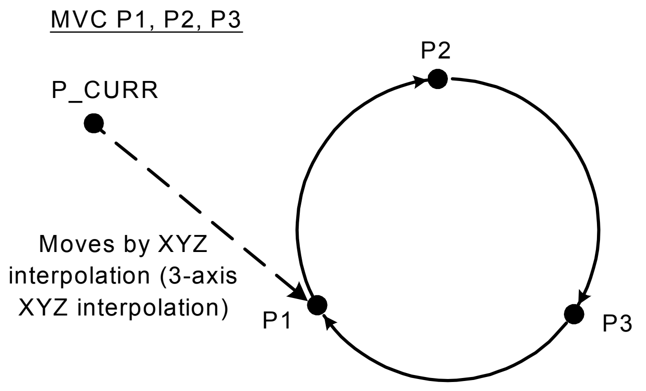


Fig.4-11:Example of circle interpolation motion path

MVR (Move R)

[Function]

Carries out 3-dimensional circular interpolation movement from the start point to the end point via transit points.

[Format]

```
MVR[]<Start Point>, <Transit Point>, <End Point>
        [[]TYPE[]<Constants 1>, <Constants 2>][] [<Appended Condition>]
```

[Terminology]

<Start Point> Start point for the arc. Describe a position operation expression or joint operation expression.

<Transit Point> Transit point for the arc. Describe a position operation expression or joint operation expression.

<End Point> End point for the arc. Describe a position operation expression or joint operation expression.

<Constants 1> Detour/short cut = 1/0, The default value is 0.

<Constants 2> 3-axis XYZ/Equivalent rotation = 1/0, The default value is 0.

<Appended conditions> The WTH and WTHIF statements can be used.

[Reference Program]

```
10 MVR P1,P2,P3
20 MVR P1,J2,P3
30 MVR P1,P2,P3 WTH M_OUT(17)=1
40 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 330, "5.10 Automatic return setting after jog feed at pause")

(5) If the start point and end point structure flags differ for an interpolation method other than 3-axis XYZ interpolation, 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 numeric constant 2, the numeric constant 1 will be invalidated, and the robot will move with the taught posture.

(8) Numeric constant 2 designates the posture interpolation type. 3-axis XYZ is used when carrying out interpolation on the (X, Y, Z, J4, J5, J6) coordinate system, and the robot is to move near a particular point.

(9) This instruction cannot be used in a constantly executed program.

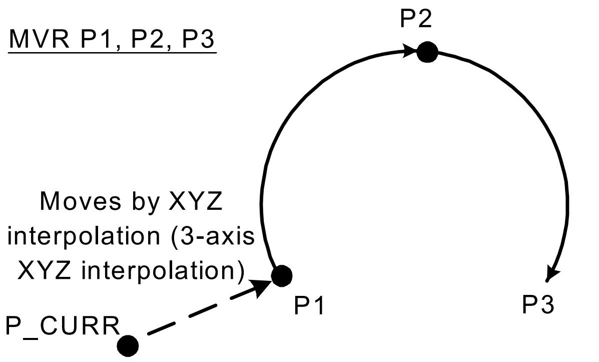


Fig.4-12:Example of circular interpolation motion path 1

MVR2 (Move R2)

[Function]

Carries out 3-dimensional circular interpolation motion from the start point to the end point on the arc composed of the start point, end point, and reference points.

The direction of movement is in a direction that does not pass through the reference points.

[Format]

```
MVR2[]<Start Point>, <End Point>, <Reference point>
[[]TYPE[]<Constants 1>, <Constants 2>][][<Appended Condition>]
```

[Terminology]

<Start Point> Start point for the arc. This position may be specified using a position type variable and constant, or a joint variable.

<End Point> End point for the arc. This position may be specified using a position type variable and constant, or a joint variable.

<Reference point> Reference point for a circular arc. This position may be specified using a position type variable and constant, or a joint variable.

<Constants 1> Detour/short cut = 1/0, The default value is 0.

<Constants 2> 3-axis XYZ/Equivalent rotation = 1/0, The default value is 0.

<Appended conditions> The WTH and WTHIF statements can be used.

[Reference Program]

```
10 MVR2 P1,P2,P3
20 MVR2 P1,J2,P3
30 MVR2 P1,P2,P3 WTH M_OUT(17)=1
40 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 330, "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 for an interpolation method other than 3-axis XYZ interpolation, 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 numeric constant 2, the numeric constant 1 will be invalidated, and the robot will move with the taught posture.

(9) Numeric 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.

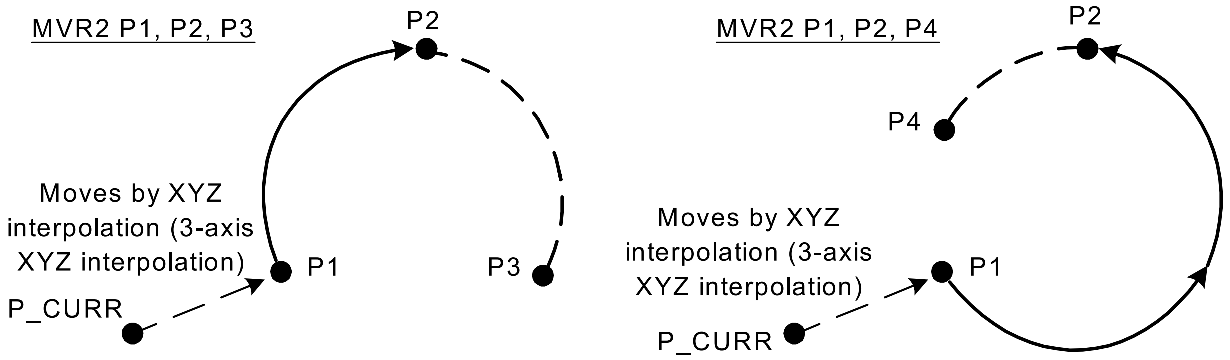

MVR2 P1, P2, P3
P2
Moves by XYZ interpolation (3-axis XYZ interpolation)
P1
P3
P_CURR
MVR2 P1, P2, P4
P2
P4
Moves by XYZ interpolation (3-axis XYZ interpolation)
P1
P_CURR

Fig.4-13:Example of circular interpolation motion path 2

MVR3 (Move R 3)

[Function]

Carries out 3-dimensional circular interpolation movement from the start point to the end point on the arc composed of the center point, start point and end point.

[Format]

```
MVR3[]<Start Point>, <End Point>, <Center Point>
          [[]TYPE[]<Constants 1>ÅC<Constants 2>][] [<Appended Condition>]
```

[Terminology]

<Start Point> Start point for the arc. This position may be specified using a position type variable and constant, or a joint variable.

<End Point> End point for the arc. This position may be specified using a position type variable and constant, or a joint variable.

<Center Point> Center point for the arc. This position may be specified using a position type variable and constant, or a joint variable.

<Constants 1> Detour/short cut = 1/0, The default value is 0.

<Constants 2> 3-axis XYZ/Equivalent rotation = 1/0, The default value is 0.

<Appended conditions> The WTH and WTHIF statements can be used.

[Reference Program]

```
10 MVR3 P1,P2,P3
20 MVR3 P1,J2,P3
30 MVR3 P1,P2,P3 WTH M_OUT(17)=1
40 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 330, "5.10 Automatic return setting after jog feed at pause")
(5) If the start point and end point structure flags differ for an interpolation method other than 3-axis XYZ interpolation, an error will occur at the execution.
(6) 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.
(7) Numeric constant 2 designates the posture interpolation type. 3-axis XYZ is used when carrying out interpolation on the (X, Y, Z, J4, J5, J6) coordinate system, and the robot is to move near a particular point.
(8) The fan angle from the start point to the end point is 0 < fan angle < 180 deg.
(9) Designate the positions so that the difference from the center point to the end point and the center point to the distance is within 0.01mm.
(10) If the three points are on the same line, or if the start point and center point, or end point and center point are the same, an error will occur.
(11) If the start point and end point are the same or if three points are the same, an error will not occur, and the next command will be executed. Note that if the posture changes at this time, only the posture will be interpolated.
(12) This instruction cannot be used in a constantly executed program.

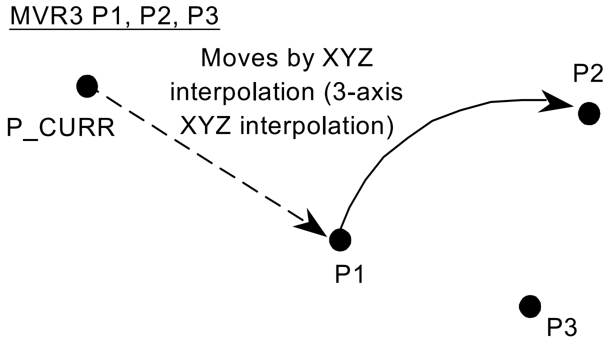


Fig.4-14:Example of circular interpolation motion path 3

MVS (Move S)

[Function]

Carries out linear interpolation movement from the current position to the movement target position.

[Format 1]

```
MVS[]<Movement Target Position> [, <Close Distance>]
        [[]TYPE <Constants 1>,<Constants 2>][][<Appended Condition>]
```

[Format 2]

```
MVS[], <Separation Distance> [][<Interpolation Type>]
```

[Terminology]

<Movement Target Position> The final position for the linear interpolation. This position may be specified using a position type variable and constant, or a joint variable.

<Close Distance> If this value is designated, the actual movement target position will be a position separated by the designated distance in the tool coordinate system Z axis direction (+/- direction).

<Constants 1> Detour/short cut = 1/0, The default value is 0(detour).

<Constants 2> 3-axis XYZ/Equivalent rotation = 1/0, The default value is 0(equivalent rotation).

<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.

```
10 MVS P1
```

(2)Turns on the output signal 17 at the same time if it moves to the target position P1 by linear interpolation.

```
10 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.

```
20 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.

```
30 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 324, "5.6 Standard Tool Coordinates" for detail. The "Fig.4-15:Example of movement at linear interpolation" is the example of RV-1A movement.

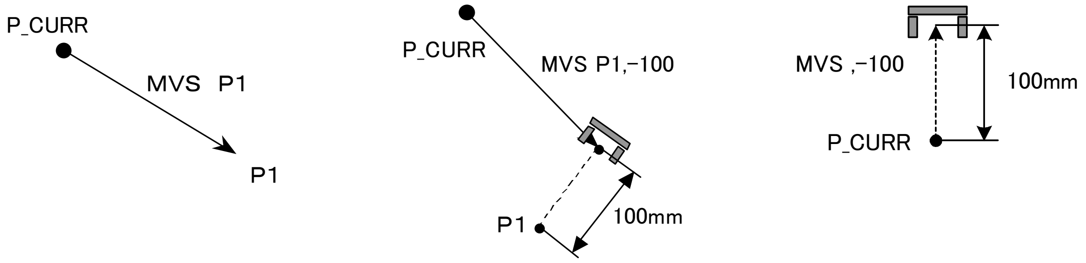


Fig.4-15:Example of movement at linear interpolation

(4) If paused during execution of a MVS instruction and restarted after jog feed, the robot returns to the interrupted position and restarts the MVS instruction. This can be changed by the "RETPATH" parameter, and also the interpolation method (JOINT interpolation / XYZ interpolation) which returns to the interrupted position can be changed by same parameter. Some robots for liquid crystal transportation have different default values of this parameter. Refer to Page 330, "5.10 Automatic return setting after jog feed at pause".

(5) This instruction cannot be used in a constantly executed program.

(6) If the start point and end point structure flags differ for an interpolation method other than 3-axis XYZ interpolation, an error will occur at the execution.

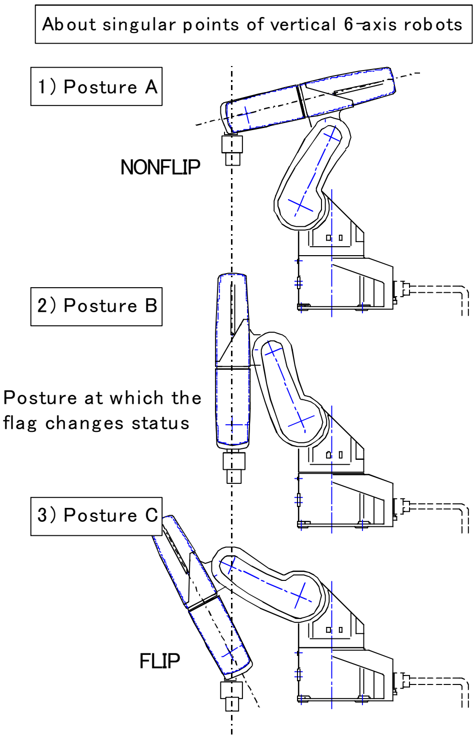
(7) If 3-axis XYZ is designated for the numeric constant 2, the numeric constant 1 will be invalidated, and the robot will move with the taught posture.

(8) Numeric constant 2 specifies the type of posture interpolation. Three-axis XYZ operation is used to pass through near a singular point in order to perform interpolation in the coordinate system of (X, Y, Z, J4, J5, J6).

(9) 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).


About singular points of vertical 6-axis robots
1) Posture A
NONFLIP
2) Posture B
Posture at which the flag changes status
3) Posture C
FLIP

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.

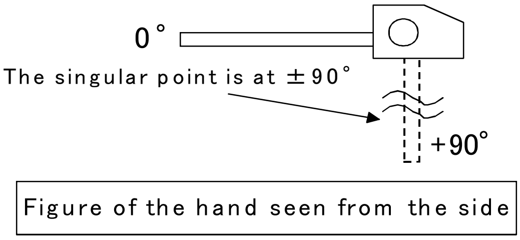
The 3-axis XYZ (TYPE 0, 1) method in the command option of MVS should be used if it is desired to perform linear interpolation based on such posture coordinates. Note that, strictly speaking, this 3-axis XYZ method does not maintain the postures as it evenly interpolates the joint angle of axes J4, J5, and J6 at posture A and C. Therefore, it is expected that the robot hand's posture may move front and back 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 origin and the wrist is facing upward. In this case, J1 and J6 are located on the same axis and it is not possible to calculate the robot position.

Fig.4-16:Singular point 1

<In the case of a 6-axis robot for liquid crystal transportation>


0°
The singular point is at ±90°
+90°
Figure of the hand seen from the side

The singular points are when the wrist axis J5 is at +90 degrees, and when the center of axis J6 is located at the origin and the wrist is facing upward. In these cases, axes J1 and J5 are located on the same axis and it is not possible to calculate the robot position.

Fig.4-17:Singular point 2

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]

```
10 OADL ON
20 MOV P1              ' Move with maximum load.
30 LOADSET 1ÅC1        ' Set hand 1 and workpiece 1.
40 MOV P2              ' Move with hand 1 + workpiece 1 load.
50 HOPEN 1             '
60 MOV P3              ' Move with hand 1 load.
70 HCLOSE 1            '
80 MOV P4              ' Move with hand 1 + workpiece 1 load.
90 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 314, "Table 5-2: List Signal parameter")

(4) Once OADL is ON, it is valid until OADL OFF is executed or until the program END is executed.

(5) Depending on the conditions of the hand and/or workpiece, the motion time may become longer than usual.

(6) It is possible to perform the optimum acceleration/deceleration operation by using the LOADSET and OADL instructions, and by setting the HNDDAT1(0) through 8 and WRKDAT1(0) through 8 parameters to appropriate values. (Refer to Page 340, "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 JADL parameter.

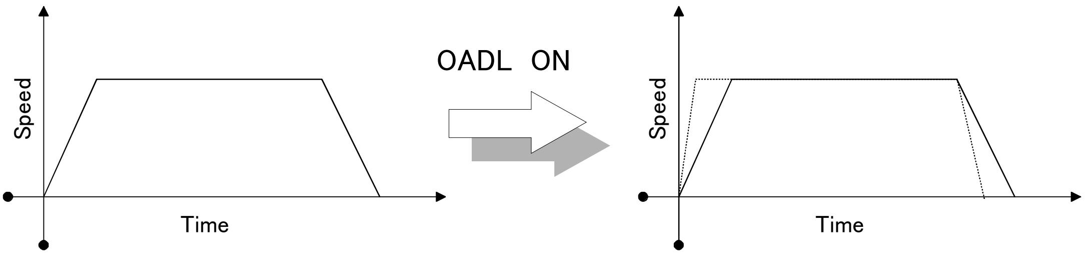


Fig.4-18:Acceleration/deceleration pattern at light load

[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.

```
 10 OPEN "COM1:" AS #1        ' Communication line opening.
 20 ON COM(1) GOSUB *RECV' The definition of interruption.
 30 COM(1) ON                 ' Enable interrupt from file No. 1 communication line.
 40                           '
100 ' <<If the communicative interrupt occurs here, it will branch to label *RECV.>>
110 '
120 MOV P1
130 COM(1) STOP               ' Suspend the interrupt during movement only from P1 to P2.
140 MOV P2
150 COM(1) ON                 ' If there are some communications during movement from P1 to P2, the
                                interrupt occurs here.
160                           '
170 ' <<If the communicative interrupt occurs here, it will branch to label *RECV.>>
260                           '
270 COM(1) OFF                ' Disable interrupt from file No. 1 communication line.
280 CLOSE #1
290 END
  :
  :
3000 *RECV                    ' Communication interruption processing.
3010   INPUT #1, M0001        ' Set the received information as M0001 and P0001.
3020   INPUT #1, P0001
 :
3100 RETURN 1                 ' Returns control to the next line of interrupted line.
```

[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 a communication interrupt is generated while the robot is moving, the robot will stop.
It is possible to use COM STOP to stop the interrupt, and prevent the robot from stopping.

(4) Interrupts are prohibited in the initial state. To enable interrupts, execute the COM ON instruction after this instruction.

(5) Make sure to return from a subroutine using the RETURN instruction. An error occurs if the GOTO instruction is used to return, because the free memory available for control structure (stack memory) decreases and eventually becomes insufficient.

[Related instructions]

COM ON/COM OFF/COM STOP (Communication ON/OFF/STOP), RETURN (Return), OPEN (Open), INPUT (Input), PRINT (Print), CLOSE (Close)

ON ... GOSUB (ON Go Subroutine)

[Function]

Calls up the subroutine at the line No. or label corresponding to the value.

[Format]

```
ON[]<Terminology>[]GOSUB[][<Expression>] [, [<Call Destination>]] ...
```

[Terminology]

<Terminology> Designate the line No. or label on the line to branch to with a numeric operation expression.
<Call Destination> Describe the line No. or the label No. The maximum number is 32.

[Reference Program]

Sets the value equivalent to three bits of input signal 16 in M1, and branches according to the value of M1 (1 through 7).

(Calls line 1000 if M1 is 1, label LSUB if M1 is 2, line 2000 if M1 is 3, 4 or 5, and label L67 if M1 is 6 or 7.)

```
10 M1 = M_INB(16) AND &H7
20 ON M1 GOSUB 1000,*LSUB,2000,2000,2000,*L67,*L67

1000 ' Describes processing when M1=1.
1010 '
1200 RETURN                    ' Be sure to return by using RETURN.

1210 *LSUB
1220                           ' Describes processing when M1=2.
1300 RETURN                    ' Be sure to return by using RETURN.

1700 *L67
1710  ' Describes processing when M1=6 or M1=7.
1720 RETURN                    ' Be sure to return by using RETURN.

2000 ' Describes processing when M1=3, M1=4, or M1=5.
2010 '
2020 RETURN                    ' Be sure to return by using RETURN.
```

[Explanation]

(1) The value of <Expression> determines which line No. or label subroutine to call.
For example, if the value of <Expression> is 2, the line No. or 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 line. For example, the program control jumps to the next line if the value of <expression> is 5 and there are only three <destinations called up>.

(3) When a line No. or label that is called up does not exist, or when there are two definitions, an error will occur.

(4) Make sure to return from a subroutine using the RETURN instruction. An error occurs if the GOTO instruction is used to return, because the free memory available for control structure (stack memory) decreases and eventually becomes insufficient.

Value of <Expression>	Process <Control>
Real number	Value is converted to an integer by rounding it off, and then branching is executed.
When 0, or when the value exceeds the number of line Nos. or labels	Control proceeds to the next line
Negative number or 32767 is exceeded	Execution error

ON ... GOTO (On Go To)

[Function]

Branches to the line with the line No. or label that corresponds to the designated value.

[Format]

```
ON[]<Expression>[]GOTO[][<Branch Destination>] [, [<Branch Destination>]] ...
```

[Terminology]

<Expression> Designate the line No. or label on the line to branch to with a numeric operation expression.

<Call Destination> Describe the line No. or the label No. The maximum number is 32.

[Reference Program]

Branches based on the value (1-7) of the numerical variable M1.

(Branches to line 1000 if M1 is 1, to label LJMP if M1 is 2, to line 2000 if M1 is 3, 4 or 5, and to label L67 if M1 is 6 or 7.)

```
 100 ON M1 GOTO 1000,*LJMP,2000,2000,2000,*L67,*L67
 110 ' Control is passed to this line when M1 is other than 1 through 7 (i.e., 0, or 8 or larger).

1000 ' Describes processing when M1=1.
1010 '    :

1110 *LJMP            ' When M1=2.
1120 '  Describes processing when M1=2.
1130 '    :

1700 *L67
1710 '  Describes processing when M1=6 or M1=7.
1720 '    :

2000 ' Describes processing when M1=3, M1=4, or M1=5.
2010 '    :
```

[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 line. For example, the program control jumps to the next line if the value of <expression> is 5 and there are only three <destinations called up>.

(3) When a line 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 line Nos. or labels	Control proceeds to the next line
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,PRINT,APPEND
Communi-cation line	COM1: Standard RS-232C(default value) COM2:The setting in the "COMDEV" parameter. : COM8:The setting in the "COMDEV" parameter.	Omitted = random mode only

<Mode> Designate the method to access a file.
*Omitted = random mode. This can be omitted when using a communication line.
*ÅEINPUT = input mode. Inputs from an existing file.
*OUTPUT = output mode (new file). Creates a new file and outputs it there.
*APPEND = Output mode (existing file). Appends output to the end of an existing file.

<File No.> Specify a constant from 1 to 8.
To interrupt from communication line: 1 to 3.

[Reference Program]

(1) Communication line.

```
10 OPEN "COM1:" AS #1        ' Open standard RS-232C line as file No. 1.20 MOV P_01
20 MOV P_01
30 PRINT #1,P_CURR           ' Output current position to external source.
                               "(100.00,200.00,300.00,400.00)(7.0)" format
40 INPUT #1,M1,M2,M3         ' Receive from external source with "101.00,202.00,303.00" ASCII for-
                               mat.
50 P_01.X=M1
60 P_01.Y=M2
70 P_01.C=RAD(M3)            ' Copy to global data.
80 CLOSE                     ' Close all opened files.
90 END
```

(2) File operation. (Create the file "temp.txt" to the controller and write "abc")

```
10 OPEN "temp.txt" FOR APPEND AS #1
20 PRINT #1, "abc"
30 CLOSE #1
```

[Explanation]

(1) Opens the file specified in <File name> using the file number.
Use this file No. when reading from or writing to the file.

(2) A communication line is handled as a file.

[Related instructions]

CLOSE (Close), PRINT (Print), INPUT (Input)

[Related parameter]

COMDEV

OVRD (Override)

[Function]

This instruction specifies the speed of the robot movement as a value in the range from 1 to 100%. This is the override applied to the entire program.

[Format]

```
OVRD[]<Override>
```

This function is available for controller software version G2 or later.

```
OVRD[]<Override> [, <Override when moving upward>] [, <Override when moving downward>]
```

[Terminology]

<Override> Designate the override with a real number. The default value is 100.
Unit: [%] (Recommended range: 0.1 to 100.0)
A numeric operation expression can also be described. If 0 or a value over 100 is set, an error will occur.

<Override when moving upward/downward>
Sets the override value when moving upward/downward by the arch motion instruction (MVA).

[Reference Program]

```
10 OVRD 50
20 MOV P1
30 MVS P2
40 OVRD M_NOVRD       ' Set default value.
50 MOV P1
60 OVRD 30,10,10      ' Sets the override when moving upward/downward by the arch motion
                        instruction to 10.
70 MVA P3,3
```

[Explanation]

(1) The OVRD command is valid regardless of the interpolation type.

(2) The actual override is as follows:

*During joint interpolation: Operation panel (T/B) override setting value) x (Program override (OVRD command)) x (Joint override (JOVRD command)).

*During linear interpolation: Operation panel (T/B) override setting value) x (Program override (OVRD command)) x (Linear designated speed (SPD command)).

(3) The OVRD command changes only the program override. 100% is the maximum capacity of the robot. Normally, the system default value (M_NOVRD) is set to 100%. The designated override is the system default value until the OVRD command is executed in the program.

(4) Once the OVRD command has been executed, the designated override is applied until the next OVRD command is executed, the program END is executed or until the program is reset. The value will return to the default value when the END statement is executed or the program is reset.

[Related instructions]

JOVRD (J Override) (For joint interpolation), SPD (Speed)(For linear/circular interpolation)

[Related system variables]

M_JOVRD/M_NJOVRD/M_OPOVRD/M_OVRD/M_NOVRD
(M_NOVRD (System default value), M_OVRD (Current designated speed))

PLT (Pallet)

[Function]

Calculates the position of grid in the pallet.

[Format]

```
PLT[]<Pallet No.> , <Grid No.>
```

[Terminology]

<Pallet No.> Select a pallet No. between 1 and 8 that has already been defined with a DEF PLT command. Specify this argument using a constant or a variable.

<Grid No.> The position number to calculate in the palette. Specify this argument using a constant or a variable.

[Reference Program]

```
100 DEF PLT 1,P1,P2,P3,P4,4,3,1' The definition of the four-point pallet. (P1,P2,P3,P4)
110                           '
120 M1=1                      ' Initialize the counter M1.
130 *LOOP
140   MOV PICK, 50            ' Moves 50 mm above the work unload position.
150   OVRD 50
160   MVS PICK
170   HCLOSE 1                ' Close the hand.
180   DLY 0.5                 ' Wait for the hand to close securely (0.5 sec.)
190   OVRD 100
200   MVS,50                  ' Moves 50 mm above the current position.
210   PLACE = PLT 1, M1       ' Calculates the M1th position
220   MOV PLACE, 50           ' Moves 50 mm above the pallet top mount position.
230   OVRD 50
240   MVS PLACE
250   HOPEN 1                 ' Open the hand.
260   DLY 0.5
270   OVRD 100
280   MVS,50                  ' Moves 50 mm above the current position.
290   M1=M1+1                 ' Add the counter.
300   IF M1 <=12 THEN *LOOP   ' If the counter is within the limits, repeats from *LOOP.
310   MOV PICK,50
320 END
```

[Explanation]

(1) The position of grid of a pallet defined by the DEF PLT statement is operated.

(2) The pallet Nos. are from 1 to 8, and up to 8 can be defined at once.

(3) Note that the position of the grid may vary because of the designated direction in the pallet definition.

(4) If a grid No. is designated that exceeds the largest grid No. defined in the pallet definition statement, an error will occur during execution.

(5) When using the pallet grid point as the target position of the movement command, an error will occur if the point is not enclosed in parentheses as shown above. Refer to Page 71, "4.1.2 Pallet operation" for detail.

[Related instructions]

DEF PLT (Define pallet)

PREC (Precision)

[Function]

This instruction is used to improve the motion path tracking. It switches between enabling and disabling the high accuracy mode.

Note) The available robot type is limited such as RV-4A. Refer to "[Available robot type]".

[Format]

This function is available for controller software version D1 or later.

```
PREC[]<ON / OFF>
```

[Terminology]

<ON / OFF>	ON : When enabling the high accuracy mode. OFF : When disabling the high accuracy mode.

[Reference Program]

```
10 PREC ON                ' Enables the high accuracy mode.
20 MVS P1
30 MVS P2
40 PREC OFF               ' Disables the high accuracy mode.
50 MOV P1
```

[Explanation]

(1) The high accuracy mode is enabled using the PREC ON instruction if it is desired to perform interpolation movement with increased path accuracy.

(2) When this instruction is used, the path accuracy is improved but the program execution time (tact time) may become longer because the acceleration/deceleration times are changed internally.

(3) The enabling/disabling of the high accuracy mode is activated from the first interpolation instruction after the execution of this instruction.

(4) The high accuracy mode is disabled if the PREC OFF or END instruction is executed, or a program reset operation is performed.

(5) The high accuracy mode is disabled immediately after turning the power on.

(6) The high accuracy mode is always disabled in jog movement.

[Available robot type]

RV-1A/2AJ series
RV-2A/3AJ series
RV-4A/5AJ series
RV-20A
RV-3S/3SJ/3SB/3SJB series
RV-6S/6SL/12S/12SL series
RH-6SH/12SH/18SH series

PRINT (Print)

[Function]

Outputs data into a file (including communication lines). All data uses the ASCII format.

[Format]

```
PRINT[]#<File No.>[] [,  [<Expression> ; ] ...[<Expression>[ ; ]]]
```

[Terminology]

<File No.> Described with numbers 1 to 8.
Corresponds to the control No. assigned by the OPEN command.

<Expression> Describes numeric operation expressions, position operation expressions and character string expressions.

[Reference Program]

```
10 OPEN "COM1" AS #1              ' Open standard RS-232-C line as file No. 1.20 MOV P_01.
20 MDATA=150                      ' Substitute 150 for the numeric variable MDATA.
30 PRINT #1,"***PRINT TEST***"    ' Outputs the character string "***PRINT TEST****."
40 PRINT #1                       ' Issue a carriage return
50 PRINT #1,"MDATA=",MDATA        ' Output the character string "MDATA" followed by the value of
                                    MDATA, (150).
60 PRINT #1                       ' Issue a carriage return.
40 PRINT #1,"****************"    ' Outputs the character string "**************."
50 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)

```
[10 M1=123.5
 20 P1=(130.5,-117.2,55.1,16.2,0.0,0.0)(1,0) ]
1)[30 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)[30 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)[30 PRINT# 1,"OUTPUT TEST",
   40 PRINT# 1,M1;
   50 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]

This function is available for controller software version C2 or later.

```
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
10 PRIORITY 3            ' Sets the number of executed lines for the current slot to 3.

Slot 2
10 PRIORITY 4            ' Sets the number of executed lines for this slot to 4.
```

[Explanation]

(1) Programs of other slots are not executed until the specified number of lines 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 lines of the slot 1 program are executed first, then four lines 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 line has been executed.

(3) An error occurs if there is no program corresponding to the specified task slot.

(4) It is possible to change the priority even while the program of the specified task slot is being executed.

RELM (Release Mechanism)

[Function]

This instruction is used in connection with control of a mechanism via task slots during multitask operation. It is used to release the mechanism obtained by the GETM instruction.

[Format]

```
RELM
```

[Reference Program]

(1) Start the task slot 2 from the task slot 1, and control the mechanism 1 in the task slot 2.

```
Task slot 1
10 RELM                  ' Releases the mechanism in order to control mechanism 1 using slot 2.
20 XRUN  2,"10"          ' Start the program 10 in slot 2.
30 WAIT M_RUN(2)=1       ' Wait for the starting confirmation of the slot 2.
 :

Task slot 2. (Program "10")
10 GETM 1                ' Get the control of mechanism 1.
20 SERVO ON              ' Turn on the servo of mechanism 1.
30 MOV P1
40 MVS P2
50 SERVO OFF             ' Turn off the servo of mechanism 1.
60 RELM                  ' Releases the control right of mechanism 1.
70 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 lines.

[Reference Program]

```
10 REM ***MAIN PROGRAM***
20 ' ***MAIN PROGRAM***
30 MOV P1                              '  Move to P1.
```

[Explanation]

(1) REM can be abbreviated to be a single quotation mark (') .

(2) It can be described after the instruction like an 30 line in reference program.

RESET ERR (Reset Error)

[Function]

This instruction resets an error generated in the robot controller. It is not allowed to use this instruction in the initial status. If an error other than warnings occurs, normal programs other than constantly executed programs cannot be operated. This instruction is effective if used in constantly executed programs.

[Format]

This function is available for controller software version B1 or later.

```
RESET ERR
```

[Reference Program]

Example of execution in a constantly executed program

```
10 IF M_ERR=1 THEN RESET ERR             'Resets an error when an error occurs in the controller.
```

[Explanation]

(1) This instruction is used in a program whose start condition is set to constant execution (ALWAYS) by the "SLT*" parameter when it is desired to reset system errors of the robot.

(2) It becomes enabled when the controller's power is turned on again after changing the value of the "ALWENA" parameter from 0 to 7.

[Related parameter]

ALWENA

[Related system variables]

M_ERR/M_ERRLVL/M_ERRNO

RETURN (Return)

[Function]

(1) When returning from a normal subroutine returns to the next line after the GOSUB.

(2) When returning from an interrupt processing subroutine, returns either to the line where the interrupt was generated, or to the next line.

[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 line 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 .

```
10 ' ***MAIN PROGRAM***
20 GOSUB *SUB_INIT          ' Subroutine jumps to label SUB_INIT.
30 MOV P1
 :
1000 ' ***SUB INIT***       ' Subroutine
1010 *SUB_INIT
1020   PSTART=P1
1030   M100=123
1040 RETURN                 ' Returns to the line immediately following the line where the subroutine
                              was called from.
```

(2) The example of RETURN from the subroutine for interruption processing. Calls the subroutine on line 100 when the input signal of general-purpose input signal number 17 is turned on.

```
10 DEF ACT 1,M_IN(17)=1 GOSUB 100' Definition of interrupt of ACT 1.
20 ACT 1=1                       ' Enable the ACT 1.
 :
100                              ' The subroutine for interrupt of ACT 1.
110   ACT 1=0                    ' Disable the interrupt.
120   M_TIMER(1)=0               ' Set the timer to zero.
130   MOV P2                     ' Move to P2.
140   WAIT M_IN(17)=0            ' Wait until the input signal 17 turns off.
150   ACT 1=1                    ' Set up interrupt again.
160 RETURN 0                     ' Returns control to the interrupted line.
```

[Explanation]

(1) Writes the RETURN instruction at the end of the jump destination processing called up by the GOSUB instruction.

(2) An error occurs if the RETURN instruction is executed without being called by the GOSUB instruction.

(3) Always use the RETURN instruction to return from a subroutine when called by the GOSUB instruction. An error occurs if the GOTO instruction is used to return, because the free memory available for control structure (stack memory) decreases and eventually becomes insufficient.

(4) When there is a RETURN command in a normal subroutine with a return-to designation number, and when there is a RETURN command in an interrupt-processing subroutine with no return-to destination number, an error will occur.

(5) when returning from interruption processing to the next line 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 line, the line may be skipped. Please refer to Page 146, "DEF ACT (Define act)" for the interrupt processing.

[Related instructions]

GOSUB (RETURN)(Go Subroutine), ON ... GOSUB (ON Go Subroutine), ON COM GOSUB (ON Communication Go Subroutine), DEF ACT (Define act)

SELECT CASE (Select Case)

[Function]

Executes one of multiple statement blocks according to the condition expression value.

[Format]

```
SELECT[] <Condition>
      CASE[]<Expression>
             [<Process>]
             BREAK
      CASE[]<Expression>
             [<Process>]
             BREAK
                :
      DEFAULT
             [<Process>]
             BREAK
END[]SELECT
```

[Terminology]

<Condition>	Describe a numeric operation expression.
<Expression>	Describe a numeric operation expression. The type must be the same as the condition expression.
<Process>	Writes any instruction (other than the GOTO instruction) provided by MELFA-BASIC IV.

[Reference Program]

```
10 SELECT MCNT
20  M1=10                ' This line is not executed
30 CASE IS <= 10        ' MCNT <= 10
40  MOV P1
50  BREAK
60 CASE 11              'MCNT=11
70  MOV P2
80  BREAK
90 CASE 13 TO 18        '13 <= MCNTÅ <= 18
100  MOV P4
110  BREAK
120 DEFAULT             ' Other than the above.
130  M_OUT(10)=1
140  BREAK
150 END SELECT
```

[Explanation]

(1) If the condition matches one of the CASE items, the process will be executed until the next CASE, DEFAULT or ENDSELECT. If the case does not match with any of the CASE items but DEFAULT is described, that block will be executed.

(2) If there is no DEFAULT, the program will jump to the line after ENDSELECT without processing.

(3) The SELECT CASE and END SELECT statements must alway 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) In the case of controller software version G1 or later, it is possible to write a SELECT CASE block within another SELECT CASE block (up to eight nesting levels are 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>".

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]

```
10 SERVO ON              ' Servo ON.
20 IF M_SVO<>1 GOTO 20   ' Wait for servo ON.
30 SPD M_NSPD
40 MOV P1
50 SERVO OFF
```

[Explanation]

(1) The robot arm controls the servo power for all axes.
(2) If additional axes are attached, the servo power supply for the additional axes is also affected.
(3) If used in a program that is executed constantly, this instruction is enabled by changing the value of the "ALWENA" parameter from 0 to 7 and then turning the controller's power on again.

[Related system variables]

M_SVO (1 : ON, 0 : OFF)

[Related parameter]

ALWENA

SKIP (Skip)

[Function]

Transfers control of the program to the next line.

[Format]

```
SKIP
```

[Reference Program]

```
10 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 line.
20 IF M_SKIPCQ=1 THEN HLT         ' Pauses the program if the execution is skipped.
```

[Explanation]

(1) This command is described with the WHT or WTHIF statements. In this case, the execution of that line is interrupted, and control is automatically transferred to the next line. Execution of skip can be seen with the M_SKIPCQ information.

[Related system variables]

M_SETADL (1: Skipped, 0: Not skipped)

SPD (Speed)

[Function]

Designates the speed for the robot's linear and circular movements. This instruction also specifies the optimum speed control mode.

[Format]

```
SPD[]<Designated Speed

SPD[]M_NSPD (Optimum speed control mode)
```

[Terminology]

<Designated Speed>	Designate the speed as a real number. Unit: [mm/s]

[Reference Program]

```
10 SPD 100
20 MVS P1
30 SPD M_NSPD          ' Set the default value.(The optimal speed-control mode .)
40 MOV P2
50 MOV P3
60 OVRD 80             ' Countermeasure against an excessive speed error in the optimal speed mode
70 MOV P4
80 OVRD 100
```

[Explanation]

(1) The SPD command is valid only for the robot's linear and circular movements.

(2) The actual designated override is (Operation panel (T/B) override setting value) x (Program override (OVRD command)) x (Linear designated speed (SPD command)).

(3) The SPD command changes only the linear/circular designated speed.

(4) When M_NSPD (The default value is 10000) is designated for the designated speed, the robot will always move at the maximum possible speed, so the line speed will not be constant(optimum speed control).

(5) An error may occur depending on the posture of the robot despite of the optimal speed control. If an excessive speed error occurs, insert an OVRD instruction in front of the error causing operation instruction in order to lower the speed only in that segment.

(6) The system default value is applied for the designated speed until the SPD command is executed in the program. Once the SPD command is executed, that designated speed is held until the next SPD command.

(7) The designated speed will return to the system default value when the program END statement is executed.

[Related system variables]

M_SPD/M_NSPD/M_RSPD

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 OP (X=Y=0: one of the robot's singular points).

Note) This command is limited to the corresponding models such as the RH-1000G series. Refer to "[available robot type]".

[Format]

This function is available for controller software version H7 or later

```
SPDOPT[] <ON/OFF>
```

[Terminology]

<ON/OFF>	ON: Enable the speed-optimization function. OFF: Disable the speed-optimization function.

[Reference Program]

```
10 MOV P1
20 SPDOPT ON            ' Enable the speed-optimization function.
30 MVS P2
40 MVS P3
50 SPDOPT OFF           ' Disable the speed-optimization function.
60 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. 4-19, 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. 4-20 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. 4-20.

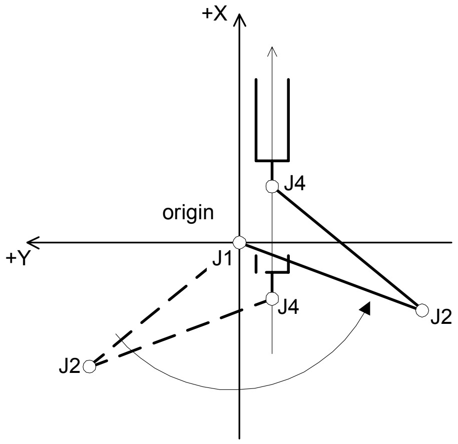


Fig.4-19:When passing through near the origin point by linear interpolation

(2) This instruction functions only for XYZ interpolation. It does not function for JOINT interpolation and CIRCULAR interpolation. Also, it does not function for linear interpolation by which the J4 axis does not pass through the speed adjustment area and the singular point as shown in Page 215 "Fig. 4-21 Speed-optimization area and singular point area.".

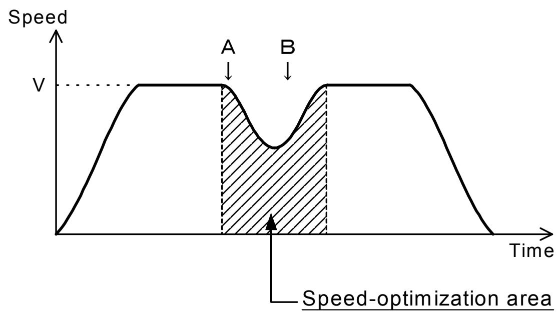


Fig.4-20:The situation of the speed at speed-optimization.

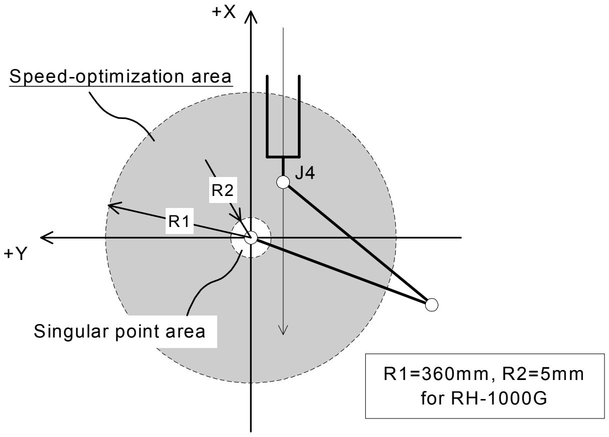


Fig.4-21:Speed-optimization area and singular point area.

(3) The initial state of the speed adjustment function immediately after the power is turned on can be changed by the SPDOPT parameter. This parameter also limits the applicable models including the RH-1000G series. Please check that it is listed in the "Command List" in the Additional Handbook/Standard Specifications before using it.
The initial value on an applicable model is SPDOPT=1 (speed adjustment enabled).
(4) 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.
(5) 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. 4-21 is executed, and the operation is then suspended.
(6) Even if this instruction is described in a program, it is ignored on models other than the applicable models.
(7) 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.
(8) 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 instruction).

[The available robot type]

RH-1000G series

[Related parameter]
SPDOPT

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.

This command is available for controller software version J1 or later.

[Format]

```
TITLE[]"<Character String>"
```

[Terminology]

<Character String> Message for title

[Reference Program]

```
10 TITLE "ROBOT Loader program"
20 MOV P1
30 MVS P2
```

[Explanation]

(1) Although characters can be registered up to the maximum allowed for each line 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.

```
10 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.
20 MVS P1
30 TOOL P_NTOOL               ' Returns the control position to the initial value. (mechanical inter-
                                face 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).)

```
10 TOOL PTL01
20 MVS P1
```

[Explanation]

(1) The TOOL instruction is used to specify the control points at the tip of each hand in a system using double hands. If both hands are of the same type, the control point should be set by the "MEXTL" parameter instead of by the TOOL instruction.

(2) The tool conversion data changed with the TOOL command is saved in parameter MEXTL, and is saved even after the controller power is turned OFF.

(3) The system default value (P_NTOOL) is applied until the TOOl command is executed.
Once the TOOL command is executed, the designated tool conversion data is applied until the next TOOL command is executed. This is operated with 6-axis three-dimension regardless of the mechanism structure.

(4) If different tool conversion data are used at teaching and automatic operation, the robot may move to an unexpected position. Make sure that the settings at operation and teaching match.
The valid axis element of tool conversion data is different depending on the type of robot.
Set up the appropriate data referring to the Page 326, "Table 5-6: Valid axis elements of the tool conversion data depending on the robot model".

(5) Using the M_TOOL variable, it is possible to set the MEXTL1 to 4 parameters as tool data.

[Related parameter]

MEXTL, MEXTL 1 to 4 Refer to Page 324, "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]

```
10 DEF ACT 1,M_FBD>10 GOTO *SUB1,S
                                   ' Generate an interrupt when the difference between the com-
                                     mand position and the feedback position reaches 10 mm or
                                     more.
20 ACT 1=1                         ' Enable the interrupt 1
30 TORQ 3,10                       ' Set the torque limit of the three axes to 10% of the normal torque
                                     using the torque instruction.
40 MVS P1                          ' Moves
50 MOV P2
:
100 *SUB1
110 MOV P_FBC                      ' Align the command position with the feedback position.
120 M_OUT(10)=1                    ' Signal No. 10 output
130 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 110th line 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

```
10 WAIT M_IN(1)=1         ' The same meaning as "10 IF M_IN(1)=0 THEN GOTO 10".
20 WAIT M_IN(3)=0
```

(2) Task slot state

```
30 WAIT M_RUN(2)=1
```

(3) Variable state

```
40 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 line once the specified condition is met.

(3) In case the WAIT instruction is executed in several tasks at one time in the multitask execution status, the processing time (tact time) may become longer and affect the system. In such cases, use the IF-THEN instruction instead of the WAIT instruction.

Example) 50 WAIT M_ABC=0 .... 50 IF M_ABC<>0 THEN GOTO 50

WHILE-WEND (While End)

[Function]

The program between the WHILE statement and WEND statement is repeated until the loop conditions are satisfied.

[Format]

```
WHILE[]<Loop Condition>
    :
WEND
```

[Terminology]

<Loop Condition>   Describe a numeric operation expression. (Refer to the syntax diagram)

[Reference Program]

Repeat the process while the numeric variable M1 value is between -5 and +5, and transfer control to line after WEND statement if range is exceeded.

```
10 WHILE (M1>=-5) AND (M1<=5)      ' Repeat the process while the value of numeric variable M1 is
                                     between -5 and +5.
20   M1=-(M1+1)                    ' Add 1 to M1, and reverse the sign.
30   PRINT# 1, M1                  ' Output the M1 value.
40 WEND                            ' Return to the WHILE statement (line 10)
50 END                             ' End the program.
```

[Explanation]

(1) The program between the WHILE statement and WEND statement is repeated.

(2) If the result of <Expression> is true (not 0), the control moves to the line 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 line following the WEND statement.

(4) If a GOTO instruction forces the program to jump out from between a WHILE statement and a WEND statement, the free memory available for control structure (stack memory) decreases. Thus, if a program is executed continuously, an error will eventually occur. Write a program in such a way that the loop exits when the condition of the WHILE statement is met.

WTH (With)

[Function]

A process is added to the interpolation motion.

[Format]

```
Example) MOV P1  WTH[]<Process>
```

[Terminology]

<Process>   Describe the process to be added. The commands that can be described are as follow.
1. <Numeric type data B> <Substitution operator><Numeric type data A> [Substitute, signal modifier command (refer to syntax diagram)]

[Reference Program]

```
10 MOV P1 WTH M_OUT(17)=1 DLY M1+2   ' Simultaneously with the start of movement to P1, the out-
                                       put signal No. 17 will turn ON for the value indicated with
                                       the numeric variable M1 + two seconds.
```

[Explanation]

(1) This command can only be used to describe the additional condition for the movement command.
(2) An error will occur if the WTH command is used alone.
(3) The process will be executed simultaneously with the start of movement.
(4) The relationship between the interrupts regarding the priority order is shown below.
    COM > ACT > WTHIF(WTH) > Pulse substitution

WTHIF (With If)

[Function]

A process is conditionally added to the interpolation motion command.

[Format]

```
WTHIF[]<Additional Condition>, <Process>
```

[Terminology]

<Additional Condition> Describe the condition for adding the process. (Same as ACT condition expression)

<Process> Describe the process to be added when the additional conditions are established. (Same as WTH)

The commands that can be described as a process are as follow. (Refer to syntax diagram.)

1. <Numeric type data B> <Substitution operator><Numeric type data A>
   Example) M_OUT(1)=1, P1=P2
2. HLT statement
3. SKIP statement

[Reference Program]

(1) If the input signal 17 turns on, the robot will stop.

```
10 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.

```
20 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.

```
30 MVS P3 WTHIF M_RATIO>15, M_OUT(1)=1
```

[Explanation]

(1) This command can only be used to describe the additional conditions to the movement command.

(2) Monitoring of the condition will start simultaneously with the start of movement.

(3) It is not allowed to write the DLY instruction at the processing part.

XCLR (X Clear)

[Function]

This instruction cancels the program selection status of the specified task slot from within a program. It is used during multitask operation.

[Format]

```
XCLR[]<Slot No.>
```

[Terminology]

<Slot No.>   Designate the slot number.

[Reference Program]

```
10 XRUN 2,"1"            ' Executes the first program in task slot 2.
  :
100 XSTP 2               ' Pauses the program of task slot 2.
110 WAIT M_WAI(2)=1      ' Waits until the program of task slot 2 pauses.
150 XRST 2               ' Cancels the pause status of the program of task slot 2.
  :
200 XCLR 2               ' Cancels the program selection status of task slot 2.
210 END
```

[Explanation]

(1) An error occurs at execution if the specified slot does not select the program.
(2) If the designated program is being operating, an error will occur at execution.
(3) If the designated program is being pausing, an error will occur at execution.
(4) If this instruction is used within a constantly executed program, it becomes enabled by changing the value of the "ALWENA" parameter from 0 to 7 and turning the controller's power off and on again.

[Related instructions]

XLOAD (X Load), XRST (X Reset), XRUN (X Run), XSTP (X Stop)

[Related parameter]

ALWENA

XLOAD (X Load)

[Function]

This instruction commands the specified program to be loaded into the specified task slot from within a program.
It is used during multitask operation.

[Format]

```
XLOAD[]<Slot No.> <Program Name>
```

[Terminology]

<Slot No.>	Designate the slot number.
<Program Name>	Designate the program name.

[Reference Program]

```
10 IF M_PSA(2)=0 THEN 60             ' Checks whether slot 2 is in the program selectable state.
20 XLOAD  2,"10"                     ' Select program 10 for slot 2.
30 IF C_PRG(2)<>"10" THEN GOTO 30 ' Waits for a while until the program is loaded.
40 XRUN 2                            ' Start slot 2.
50 WAIT M_RUN(2)=1                   ' Wait to confirm starting of slot 2.
60 '
70 ' When the slot 2 is already operating, execute from here.
```

[Explanation]

(1) An error occurs at execution if the specified program does not exist.
(2) If the designated program is already selected for another slot, an error will occur at execution.
(3) If the designated program is being edited, an error will occur at execution.
(4) If the designated program is being executed, an error will occur at execution.
(5) Designate the program name in double quotations.
(6) If used in a program that is executed constantly, this instruction is enabled by changing the value of the "ALWENA" parameter from 0 to 7 and then turning the controller's power on again.
(7) If XRUN is executed immediately after executing XLOAD, an error may occur while loading a program. If necessary, perform a load completion check as shown on the 30th line of the statement example.

[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 line 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.> Designate the slot number.

[Reference Program]

```
10 XRUN 2               ' Start.
20 WAIT M_RUN(2)=1      ' Wait to confirm starting.
  :
100 XSTP 2              ' Stop.
110 WAIT M_WAI(2)=1     ' Wait for stop to complete.
  :
150 XRST  2             ' Set program execution start line to head line.
160 WAIT M_PSA(2)=1     ' Wait for program reset to complete.
  :
200 XRUN  2             ' Restart.
210 WAIT M_RUN(2)=1     ' Wait for restart to complete.
```

[Explanation]

(1) This is valid only when the slot is in the stopped state.

(2) If used in a program that is executed constantly, this instruction is enabled by changing the value of the "ALWENA" parameter from 0 to 7 and then turning the controller's power on again.

[Related instructions]

XCLR (X Clear), XLOAD (X Load), XRUN (X Run), XSTP (X Stop)

[Related parameter]

ALWENA

[Related system variables]

M_PSA (Slot number) (1: Program selection is possible, 0: Program selection is impossible)
M_RUN (Slot number) (1: Executing, 0: Not executing)
M_WAI (Slot number) (1: Stopping, 0: Not stopping)

XRUN (X Run)

[Function]

This instruction executes concurrently the specified programs from within a program.It is used during multitask operation.

[Format]

```
XRUN[]<Slot No.> [, "<Program Name>" [, <Operation Mode>] ]
```

[Terminology]

<Slot No.>　If the argument is omitted, the current operation mode is used.

<Program Name>　Designate the program name.

<Operation Mode>　0 = Continuous operation,
1 = Cycle stop operation. If the operation mode is omitted, the current operation mode will be used. Specify this argument using a constant or a variable.

[Reference Program]

(1) When the program of execution is specified by XRUN command (continuous executing).

```
10 XRUN 2,"1"                      ' Start the program 1 with slot 2.
20 WAIT M_RUN(2)=1                 ' Wait to have started.
```

(2) When the program of execution is specified by XRUN command (cycle operation)

```
10 XRUN 3,"2",1                    ' Start the program 2 with slot 3 in the cycle operation mode
20 WAIT M_RUN(3)=1                 ' Wait to have started.
```

(3) When the program of execution is specified by XLOAD command (continuous executing).

```
10 XLOAD 2, "1"                    ' Select the program 1 as the slot 2.
20 IF C_PRG(2)<>"1" THEN GOTO 20 ' Wait for load complete.
30 XRUN 2                          ' Start the slot 2.
```

(4) When the program of execution is specified by XLOAD command (cycle operation)

```
10 XLOAD 3, "2"                    ' Select the program 2 as the slot 3.
20 IF C_PRG(2)<>"1" THEN GOTO 20 ' Wait for load complete
30 XRUN 3, ,1                      ' Start the program 1 with cycle operation.
```

[Explanation]

(1) An error occurs at execution if the specified program does not exist.
(2) If the designated slot No. is already in use, an error will occur at execution.
(3) If a program has not been loaded into a task slot, this instruction will load it. It is thus possible to operate the program without executing the XLOAD instruction.
(4) If XRUN is executed in the "Pausing" state with the program stopped midway, continuous execution will start.
(5) Designate the program name in double quotations.
(6) If the operation mode is omitted, the current operation mode will be used.
(7) If it is used in programs that are constantly executed, change the value from 0 to 7 in the ALWENA parameter, and power ON the controller again.
(8) If XRUN is executed immediately after executing XLOAD, an error may occur while loading a program. If necessary, perform a load completion check as shown on the 20th line of both statement examples [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.>   Designate the slot No.

[Reference Program]

```
10 XRUN 2                  ' Execute.
  :
100 XSTP 2                 ' Stop.
110 WAIT M_WAI(2)=1        ' Wait for stop to complete.
  :
200 XRUN  2                ' Restart.
```

[Explanation]

(1) If the program is already stopped, an error will not occur.

(2) XSTP can also stop the constant execution attribute program.

(3) If used in a program that is executed constantly, this instruction is enabled by changing the value of the "ALWENA" parameter from 0 to 7 and then turning the controller's power on again.

[Related instructions]

XCLR (X Clear), XLOAD (X Load), XRST (X Reset), XRUN (X Run)

[Related parameter]

ALWENA

[Related system variables]

M_WAI (Slot number) (1: Stopping, 0: Not stopping)

Substitute

[Function]

The results of an operation are substituted in a variable or array variable.

[Format]

```
<Variable Name> = <Expression 1>
```

For pulse substitution

```
<Variable Name> = <Expression 1> DLY <Expression 2>
```

[Terminology]

<Variable Name>	Designate the variable name of the value is to be substituted. (Refer to the syntax diagram for the types of variables.)
<Expression 1>	Substitution value. Describe an numeric value operation expression.
<Expression 2>	Pulse timer. Describe an numeric value operation expression.

[Reference Program]

(1) Substitution of the variable operation result .

```
10 P100=P1+P2*2
```

(2) Output of the signal.

```
20 M_OUT(10)=1                ' Turn on the output signal 10.
```

(3) Pulse output of the signal.

```
30 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 lines.

(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 line 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>
```

The controller software version J1 or later

```
*<Label Name> [:<Command line>]
```

[Terminology]

<Label Name> Describe a character string that starts with an alphabetic character.
Up to 8 characters can be used. (Up to 9 characters including *.)

<Command line> The command line can be described after the colon after the label (:).

[Reference Program]

```
100 *SUB1
200 IF M1=1 THEN GOTO *SUB1
```

The controller software version J1 or later

```
300 *LBL1 : IF M_IN(19)=0 THEN GOTO *LBL1' Wait by the 300 lines until the input signal of No. 10
                                              turns on.
```

[Explanation]

(1) An error will not occur even if this is not referred to during the program.

(2) If the same label is defined several times in the same program, an error will occur at the execution.

(3) The reserved words can't be used for the label.

(4) If the underscore is used for the label name, the 1st character is "L." only. If the characters except "L" are used (ex. *A_LABEL), an error occurs.

Ex.) The correct example of the label with using the underscore. (The 1st character is "L")

*L_ABC, *L12_345, *LABEL_1

The mistake example of the label with using the underscore.

*H_ABC, *ABC_123, *NG_, *_LABEL

(5) The software J1 or later, the command line can be described after the colon after the label (:). However, after the command line, the colon cannot be described and the command line cannot be described again.

4.12 Detailed explanation of Robot Status Variable

4.12.1 How to Read Described items

[Function]	: This indicates a function of a variable.
[Format]	: This indicates how to enter arguments of an instruction. [ ] means that arguments may be omitted. System status variables can be used in conditional expressions, as well as in reference and assignment statements. In the format example, only reference and assignment statements are given to make the description simple.
[Reference Program]	: An example program using variables is shown.
[Terminology]	: This indicates the meaning and range of an argument.
[Explanation]	: This indicates detailed functions and precautions.
[Reference]	: This indicates related items.

4.12.2 Explanation of Each Robot Status Variable

Each variable is explained below in alphabetical order.

C_DATE

[Function]

This variable returns the current date in the format of year/month/date.

[Format]

```
Example) <Character String Variable >=C_DATE
```

[Reference Program]

```
    10 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]

```
    10 C1$=C_MAKER             ' "COPYRIGHT1999......." is assigned to C1$.
```

[Explanation]

(1) This variable returns information on the manufacturer of the robot controller.
(2) This variable only reads the data.

[Reference]

C_MECHA

C_MECHA

[Function]

This variable returns the name of the mechanism to be used.

[Format]

```
Example) <Character String Variable >=C_MECHA[(<Mechanism Number>)]
```

[Terminology]

<Character String Variable > Specify a character string variable to be assigned.

<Mechanism Number> Enter the mechanism number. 1 to 3, If the argument is omitted, 1 is set as the default value.

[Reference Program]

```
10 C1$=C_MECHA(1)      ' "RV-4A" is assigned to C1$. (If the robot type name is RV-4A)
```

[Explanation]

(1) This variable returns the name of the mechanism to be used.
(2) This variable only reads the data.

C_PRG

[Function]

This variable returns the selected program number (name).

[Format]

```
Example) <Character String Variable >=C_PRG [(<Numeric>)]
```

[Terminology]

<Character String Variable > Specify a character string variable to be assigned.

<Numeric> 1 to 32, Enter the task slot number. If the argument is omitted, 1 is set as the default value.

[Reference Program]

```
10 C1$=C_PRG(1)        ' "10" is assigned to C1$. (if the program number is 10.)
```

[Explanation]

(1) The program number (name) set (loaded) into the specified task slot is assigned.
(2) If this variable is used in single task operation, the task slot number becomes 1.
(3) If it is set in the operation panel, that number is set.
(4) This variable only reads the data.
(5) If a task slot for which a program is not loaded is specified, an error occurs at execution.

C_TIME

[Function]

This variable returns the current time in the format of time:minute:second (24 hours notation).

[Format]

```
Example) <Character String Variable >=C_TME
```

[Reference Program]

```
10 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]

```
10 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]

```
10 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 impact detection function is being enabled.

The impact detection function can only be used in certain models (Refer to "[Available robot type]".). This function is available for controller software version J2 or later.

[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.)

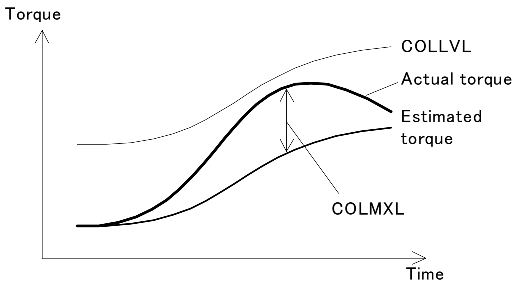
<Mechanism Number> Enter the mechanism number. 1 to 3, If the argument is omitted, 1 is set as the default value.

[Reference Program]

```
10 M1=100                    'Set the initial value of the allowable impact level of each axis.
20 M2=100
30 M3=100
40 M4=100
50 M5=100
60 M6=100
70 COLLVL M1,M2,M3,M4,M5,M6,,'Set the allowable impact level of each axis.
80 COLCHK ON                 'Enable the impact detection function.
                              (Start the calculation of the maximum value of torque error.)
90 MOV P1
  :
  :
500 COLCHK OFF               'Disable the impact detection function.
                              (End the calculation of the maximum value of torque error.)
510 M1=J_COLMXL(1).J1+10     'For each axis, the allowable impact level with a margin of 10% is calcu-
                              lated.
520 M2=J_COLMXL(1).J2+10
530 M3=J_COLMXL(1).J3+10
540 M4=J_COLMXL(1).J4+10
550 M5=J_COLMXL(1).J5+10
560 M6=J_COLMXL(1).J6+10
570 GOTO 70
```

[Explanation]

(1) Keep the maximum value of the error of the estimated torque and actual torque of each axis while impact detection function is valid.



(2) When this value is 100%, it indicates that the maximum error value is the same as the manufacturer's initial value of the allowable impact level.

(3) For robots that prohibit the use of impact detection, 0.0 is always returned for all axes.

(4) The maximum error value is initialized to 0.0 when the servo is turned ON during the execution of a COLCHK ON or COLLVL instruction.

(5) Because they are joint-type variables, it will be conversion values from rad to deg if they are read as joint variables. Therefore, substitute each axis element by a numeric variable as shown in the syntax example when using these joint-type variables.

[Reference]

COLCHK (Col Check), COLLVL (Col Level), M_COLSTS, P_COLDIR

[Available robot type]

RV-3S/3SJ/3SB/3SJB series
RV-6S/6SL/12S/12SL series
RH-6SH/12SH/18SH series

J_ECURR

[Function]

Returns the current encoder pulse value.

[Format]

```
Example) <Joint Type Variable>=J_ECURR [(<Mechanism Number>)]
```

[Terminology]

<Joint Type Variable> Specify a joint type variable to be assigned.

<Mechanism Number> Enter the mechanism number. 1 to 3, If the argument is omitted, 1 is set as the default value.

[Reference Program]

```
10 J1=J_ECURR(1)        ' JA will contain the encoder pulse value of mechanism 1.
20 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]

```
10 J1=J_FBC           ' J1 will contain the current position of the joint that has been generated by
                        servo feedback.
20 J1=J_AMPFBC        ' The present current feedback value is entered in J2.
```

[Explanation]

(1) J_FBC returns the present position of the joint type generated by the feedback of the encoder.
(2) J_FBC can check the difference between the command value to the servo and the delay in the actual servo.
(3) J_FBC can also check if there is a difference as a result of executing a CMP JNT instruction.
(4) This variable only reads the data.

[Reference]

P_FBC

J_ORIGIN

[Function]

Returns the joint data when the origin has been set.

[Format]

```
Example) <Joint Type Variable>=J_ORIGIN [(<Mechanism Number>)]
```

[Terminology]

<Joint Type Variable> Specify a joint type variable to be assigned.

<Mechanism Number> Enter the mechanism number. 1 to 3, If the argument is omitted, 1 is set as the default value.

[Reference Program]

```
10 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]

```
10 M1=M_ACL          ' M1 will contain the ratio of acceleration time set for task slot 1.
20 M1=M_DACL(2)      ' M1 will contain the ratio of deceleration time set for task slot 2.
30 M1=M_NACL         ' M1 will contain the ratio of initial acceleration time value set for task slot 1.
40 M1=M_NDACL(2)     ' M1 will contain the ratio of initial deceleration time value set for task slot 2.
50 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_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]

```
10 WHILE M1<>0
20   IF M2=0 THEN BREAK        ' The remaining battery capacity time is assigned to M1.
30 WEND
40 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_BTME
```

[Terminology]

<Numeric Variable> Specifies the numerical variable to assign.

[Reference Program]

```
10 M1=M_BTIME              ' The remaining battery capacity time is assigned to M1.
```

[Explanation]

(1) Returns the remaining hours the battery can last from now.
(2) As for the battery life, 14,600 hours are stored as the initial value.
(3) After summing the total amount of time the power of robot controller has been off, this value will be subtracted from 14,600 and the result is returned.
(4) This variable only reads the data.

M_CMPDST

[Function]

Returns the amount of difference (in mm) between the command value and the actual value from the robot when executing the compliance function.

[Format]

```
Example)<Numeric Variable>=M_CMPDST [(<Mechanism Number>)]
```

[Terminology]

<Numeric Variable>   Specifies the numerical variable to assign.

<Mechanism Number>   Enter the mechanism number. 1 to 3, If the argument is omitted, 1 is set as the default value.

[Reference Program]

```
10 MOV P1
20 CMPG 0.5,0.5,1.0,0.5,0.5, , ,  ' Set softness.
30 CMP POS, &B00011011           ' Enter soft state.
40 MVS P2
50 M_OUT(10)=1
60 MVS P1
70 M1=M_CMPDST(1)                ' M1 will contain the difference between the position specified by the
                                   operation command and the actual current position.
80 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]

```
10 DEF ACT 1, M_CMPLMT(1)=1 GOTO *LMT' Define the conditions of interrupt 1.
20 '
30 '
100 MOV P1
110 CMPG 1,1,0,1,1,1,1,1
120 CMP POS, &B100                 ' Enable compliance mode.
130 ACT 1=1                        ' Enable interrupt 1.
140 MVS P2                         '
150                                '
160                                '
1000 *LMT
1010 MVS P1                        ' Movement to P2 is interrupted and returns to P1.
1020 RESET ERR                     ' Reset the error.
1030 HLT                           ' Execution is stopped.
```

[Explanation]

(1) This is used to recover from the error status by using interrupt processing if an error has occurred while the command value in the compliance mode attempted to exceed a limit.

(2) For various limits, the joint operation range and operation speed of the command value in the compliance mode, and the dislocation between the commanded position and the actual position are checked.

(3) 0 is set if the servo power is off, or the compliance mode is disabled.

(4) This is a read only variable.

M_COLSTS

[Function]

Return the impact detection status..

1: Detecting an impact

0: No impact has been detected

The impact detection function can only be used in certain models (Refer to "[Available robot type]".). This function is available for controller software version J2 or later.

[Format]

```
Example) DEF ACT 1, M_COLSTS [(<Mechanism Number>)]=1 GOTO *LCOL
```

[Terminology]

<Mechanism Number>   Specify the mechanism number 1 to 3. The default value is 1.

[Reference Program]

```
10 DEF ACT 1,M_COLSTS(1)=1 GOTO *HOME,S'Define the processing to be executed when an impact
                                        is detected using an interrupt.
20 ACT 1=1
30 COLCHK ON,NOERR         'Enable the impact detection function in the error non-occurrence mode.
40 MOV P1
50 MOV P2                  'If an impact is detected while executing lines 40 through 70, it jumps to
                            interrupt processing.
60 MOV P3
70 MOV P4
80 ACT 1=0
  :
  :
1000 *HOME                 'Interrupt processing during impact detection.
1010 COLCHK OFF            'Disable the impact detection function.
1020 SERVO ON              'Turn the servo on.
1030 PESC=P_COLDIR(1)*(-2) 'Create the amount of movement for escape operation
1040 PDST=P_FBC(1)+PESC    'Create the escape position.
1050 MVS PDST              'Move to the escape position.
1060 ERROR 9100            'Stop operation by generating a user-defined L level error.
```

[Explanation]

(1) When an impact is detected, it is set to 1. When the impact state is canceled, it is set to 0.

(2) It is used as an interrupt condition in the DEF ACT instruction when used in the NOERR mode.

[Available robot type]

RV-3S/3SJ/3SB/3SJB series
RV-6S/6SL/12S/12SL series
RH-6SH/12SH/18SH series

M_CSTP

[Function]

Returns the status of whether or not a program is on cycle stop
1: Cycle stop is entered, and cycle stop operation is in effect.
(The input of the END key on the operation panel, or the input of a cycle stop signal)
0: Other than above

[Format]

```
Example)<Numeric Variable>=M_CSTP
```

[Terminology]

<Numeric Variable> Specifies the numerical variable to assign.

[Reference Program]

```
10 M1=M_CSTP             ' 1 is assigned to M1. (When under a cycle stop)
```

[Explanation]

(1) When the END key on the operation panel is pressed while the program is under continuous execution, the system enters a cycle operation state. The status at this time is returned as 1.
(2) This variable only reads the data.

M_CYS

[Function]

Returns the status of whether or not a program is on cycle operation
1: In cycle operation (operating mode set by the slot parameter SLT* to ...)
0: Other than above.

[Format]

```
Example)<Numerical variable> = M_CYS
```

[Terminology]

<Numerical variable>   Specify the numerical variable to substitute.

[Reference Program]

```
10 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).
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]

```
10 M1=M_DIN(6000)      ' M1 will contain the CC-Link input register value.
                       ' (If CC-Link station number is 1.)
20 M1=M_DOUT(6000)     ' M1 will contain the CC-Link output register value.
30 M_DOUT(6000)=100    ' Writes 100 to the CC-Link output register.
```

[Explanation]

(1) For details, refer to the "CC-Link Interface Instruction Manual."
(2) Signal numbers in 6,000's will be used for CC-Link.
(3) M_DIN is read-only.

M_ERR/M_ERRLVL/M_ERRNO

[Function]

Returns information regarding the error generated from the robot.
M_ERR : Returns whether an error has been generated. (1: Error has been generated, 0: No error)
M_ERRLVL : Returns the level of the generated error. (Caution/Low/High1/High2 = 1/2/3/4)
M_ERRNO : Returns the error number of the generated error.

[Format]

```
Example) <Numeric Variable>=M_ERR
Example) <Numeric Variable>=M_ERRLVL
Example) <Numeric Variable>=M_ERRNO
```

[Terminology]

<Numeric Variable>	Specifies the numerical variable to assign.

[Reference Program]

```
10 IF M_ERR=0 THEN 10 ' Waits until an error is generated.
20 M2=M_ERRLVL          ' M2 will contain the error level
30 M3=M_ERRNO           ' M3 will contain the error number.
```

[Explanation]

(1) Normal programs will pause when an error (other than cautions) is generated. The error status of the controller may be monitored using this variable for programs whose startup condition is set to ALWAYS by the SLT* parameter. The program set to ALWAYS will not stop even when an error is generated from other programs.

(2) Level 1 errors are warnings, level 2 errors pause programs. Level 3 errors pause programs and turn the servo power OFF, but error reset can be performed. Level 4 errors pause programs, turn the servo power OFF, and error reset cannot be performed. Thus, when a level 4 error occurs, it is necessary to turn the controller power OFF.

(3) This variable only reads the data.

[Related instructions]

RESET ERR (Reset Error)

M_EXP

[Function]

Returns the base of natural logarithm (2.718281828459045).

[Format]

```
Example) <Numeric Variable>=M_EXP
```

[Terminology]

<Numeric Variable> Specifies the numerical variable to assign.

[Reference Program]

```
10 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.
This variable is available for controller software version J1 or later.

[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]

```
10 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.
20 ACT 1=1                           ' An interrupt takes effect.
30 TORQ 3,10                         ' Set the torque limit of the three axes to 10% or less using
                                       the torque instruction.
40 MVS P1                            ' Moves.
50 END

100 *SUB1
110 MOV P_FBC                        ' Align the command position with the feedback position.
120 M_OUT(10)=1                      ' Signal No. 10 output
130 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]

```
10 M1=M_G             ' Gravitational constant (9.80665) is assigned to M1.
```

[Explanation]

(1) This is used to perform calculation related to gravity.
(2) This variable only reads the data.

M_HNDCQ

[Function]

Returns the hand check input signal value.

[Format]

```
Example) <Numeric Variable>=M_HNDCQ [(<Equation>)]
```

[Terminology]

<Numeric Variable>   Specifies the numerical variable to assign.
<Equation>   Enter the hand input signal number.
1 to 8, (Corresponds to input signals 900 to 907.)

[Reference Program]

```
10 M1=M_HNDCQ(1)      ' M1 will contain the status of hand 1.
```

[Explanation]

(1) Returns one bit of the hand check input signal status (such as a sensor).
(2) M_HNDCQ(1) corresponds to input signal number 900. Same result will be obtained using M_IN (900).
(3) This variable only reads the data.

M_IN/M_INB/M_INW

[Function]

Returns the value of the input signal.
M_IN : Returns a bit.
M_INB : Returns a byte (8 bits).
M_INW : Returns a word (16 bits).

[Format]

```
Example) <Numeric Variable>=M_IN(<Equation>)

Example) <Numeric Variable>=M_INB(<Equation>)

Example) <Numeric Variable>=M_INW(<Equation>)
```

[Terminology]

<Numeric Variable>	Specifies the numerical variable to assign.
<Equation>	Enter the input signal number. 0 to 32767 (Theoretical value) 0 to 255 : Standard remote inputs (Normally 32 points. 0 to 31) 900 to 907 : Hand input. 2000 to 5071 : Input signal of PROFIBUS. 6000 to 8047 : Remote input for CC-Link.

[Reference Program]

```
10 M1=M_IN(0)              ' M1 will contain the value of the input signal 0 (1 or 0).
20 M2=M_INB(0)             ' M2 will contain the 8-bit information starting from input signal 0.
30 M3=M_INB(3) AND &H7     ' M3 will contain the 3-bit information starting from input signal 3.
40 M4=M_INW(5)             ' M4 will contain the 16-bit information starting from input signal 5.
```

[Explanation]

(1) Returns the status of the input signal.
(2) M_INB and M_INW will return 8- or 16-bit information starting from the specified number.
(3) Although the signal number can be as large as 32767, only the signal numbers with corresponding hardware will return a valid value. Value for a signal number without corresponding hardware is set as undefined.
(4) This variable only reads the data.

M_JOVRD/M_NJOVRD/M_OPOVRD/M_OVRD/M_NOVRD

[Function]

Returns override value.
M_JOVRD : Value specified by the override JOVRD instruction for joint interpolation.
M_NJOVRD : Initial override value (100%) for joint interpolation.
M_OPOVRD : Override value of the operation panel.
M_OVRD : Current override value, value specified by the OVRD instruction.
M_NOVRD : Initial override value (100%).

[Format]

```
Example)<Numeric Variable>=M_JOVRD [(i<Equation>)]

Example)<Numeric Variable>=M_NJOVRD[(i<Equation>)]

Example)<Numeric Variable>=M_OPOVRD

Example)<Numeric Variable>=M_OVRD[(<Equation>)]

Example)<Numeric Variable>=M_NOVRD[(<Equation>)]
```

[Terminology]

<Numeric Variable>	Specifies the numerical variable to assign.
<Equation>	1 to 32, Enter the task slot number. If this parameter is omitted, the current slot will be used as the default.

[Reference Program]

```
10 M1=M_OVRD        ' M1 will contain the current override value.
20 M2=M_NOVRD       ' M2 will contain the initial override value (100%).
30 M3=M_JOVRD       ' M3 will contain the current joint override value.
40 M4=M_NJOVRD      ' M4 will contain the initial joint override value.
50 M5=M_OPOVRD      ' M5 will contain the current OP (operation panel) override value.
60 M6=M_OVRD(2)     ' M6 will contain the current override value for slot 2.
```

[Explanation]

(1) If the argument is omitted, the current slot status will be returned.
(2) This variable only reads the data.

M_LDFACT

[Function]

The load ratio for each joint axis can be referenced.
This variable is available for controller software version J1 or later.

[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]

```
10 ACCEL 100,100           ' Lower the overall deceleration time to 50%.
20 MOV P1
30 MOV P2
40 IF M_LDFACT(2)>90 THEN
50   ACCEL 50,50           ' Lower the acceleration/deceleration ratio to 50%.
60   M_SETADL(2)=50        ' Furthermore, lower the acceleration/deceleration ratio of the J2 axis to
                             50%. (In actuality, 50% x 50% = 25%)
70ELSE
80   ACCELL 100.,100       ' Return the acceleration/deceleration time.
90 ENDIF
100 GOTO 20
```

[Explanation]

(1) The load ratio of each axis can be referenced.
(2) The load ratio is derived from the current that flows to each axis motor and its flow time.
(3) The load ratio rises when the robot is operated with a heavy load in a severe posture for a long period of time.
(4) When the load ratio reaches 100%, an overload error occurs. In the above example statement, once the load ratio exceeds 90%, the k acceleration/deceleration time is lowered to 50%.
(5) To lower the load ratio, measures, such as decreasing the acceleration/deceleration time, having the robot standing by in natural posture, or shutting down the servo power supply, are effective.

[Related instructions]

ACCEL (Accelerate), OVRD (Override), M_SETADL

M_LINE

[Function]
Returns the line number that is being executed.

[Format]

```
Example)<Numeric Variable>=M_LINE [(<Equation>)]
```

[Terminology]

<Numeric Variable>	Specifies the numerical variable to assign.
<Equation>	1 to 32, Specifies the task slot number. If this parameter is omitted, the current slot will be used as the default.

[Reference Program]

```
10 M1=M_LINE(2)          ' M1 will contain the line number being executed by slot 2.
```

[Explanation]
(1) This can be used to monitor the line being executed by other tasks during multitask operation.
(2) This variable only reads the data.

M_MODE

[Function]
Returns the key switch mode of the operation panel.
1 : TEACH
2 : AUTO(OP)
3 : AUTO(Ext.)

[Format]

```
Example)<Numeric Variable>=M_MODE
```

[Terminology]

<Numeric Variable>	Specifies the numerical variable to assign.

[Reference Program]

```
10 M1=M_MODE             ' M1 will contain the key switch status.
```

[Explanation]
(1) This can be used in programs set to ALWAYS (constantly executed) during multitask operation.
(2) This variable only reads the data.

M_ON/M_OFF

[Function]

Always returns 1 (M_ON) or 0 (M_OFF).

[Format]

```
Example)<Numeric Variable>=M_ON

Example)<Numeric Variable>=M_OFF
```

[Terminology]

<Numeric Variable> Specifies the numerical variable to assign.

[Reference Program]

```
10 M1=M_ON              ' 1 is assigned to M1.
20 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]

RetReturns the status indicating whether or not a file is opened.
Returns the status of other end of the RS-232C cable.

[Format]

This function is available for controller software version H7 or later

```
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]

```
100 OPEN "COM2:" AS #1              ' Open the communication line COM2 as the file number 1.
110 IF M_OPEN(1)<>1 THEN GOTO 110   ' 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 RS-232C	*Returns the status of other end of the RS-232C cable. Returns the status of the CTS signal input as is. (This can be used only when the RTS signal of other end is enabled using the Mitsubishi genuine cable specification.)	1: Already connected (CTS signal is ON) 0: Unconnected (CTS signal is OFF) -1: Undefined file number (not opened)

*Refer to separate manual "Ethernet Interface INSTRUCTION MANUAL" when using the ethernet.

[Related instructions]

OPEN (Open)

[Related parameter]

COMDEV

M_OUT/M_OUTB/M_OUTW

[Function]

Writes or references external output signal.
M_OUT:Output signal bit.
M_OUTB:Output signal byte (8 bits).
M_OUTW:Output signal word (16 bits).

[Format]

```
Example)M_OUT(<Equation>)=<Numeric Variable>

Example)M_OUTB(<Equation>)=<Numeric Variable>

Example)M_OUTW(<Equation>)=<Numeric Variable>
```

[Terminology]

<Numeric Variable>	Specifies the numerical variable to assign.
<Equation>	Specify the output signal number. 0 to 255 : Standard remote outputs. 900 to 907 : Hand output. 2000 to 5071 : Output signal of PROFIBUS. 6000 to 8047 : Remote output for CC-Link.

[Reference Program]

```
10 M_OUT(2)=1                ' Turn ON output signal 2 (1 bit).
20 M_OUTB(2)=&HFF            ' Turns ON 8-bits starting from the output signal 2.
30 M_OUTW(2)=&HFFFF          ' Turns ON  16-bits starting from the output signal 2.
40 M4=M_OUTB(2) AND &H0F     ' M4 will contain the 4-bit information starting from output signal 2.
```

[Explanation]

(1) This is used when writing or referencing external output signals.
(2) Numbers in 900's will be used as I/O signals for the hand.
(3) Numbers 6000 and beyond will be referenced/assigned to the CC-Link (optional).

M_PI

[Function]

Returns pi (3.14159265358979).

[Format]

```
Example)<Numeric Variable>=M_PI
```

[Terminology]

<Numeric Variable>	Specifies the numerical variable to assign.

[Reference Program]

```
10 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]

```
10 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]

```
10 MOV P1 WTHIF M_RATIO>80, M_OUT(1)=1' The output signal 1 will turn ON when the robot has
                                        moved 80% of the distance until the target position is
                                        reached while moving toward P1.
```

[Explanation]

(1) This is used, for instance, when performing a procedure at a specific position while the robot is moving.
(2) This variable only reads the data.

M_RDST

[Function]

Returns the remaining distance to the target position (in mm) while the robot is moving.

[Format]

```
Example)<Numeric Variable>=M_RDST [(<Equation>)]
```

[Terminology]

<Numeric Variable> Specifies the numerical variable to assign.

<Equation> 1 to 32, Specifies the task slot number. If this parameter is omitted, the current slot will be used as the default.

[Reference Program]

```
10 MOV P1 WTHIF M_RDST<10 M_OUT(10)=1' The output signal 1 will turn ON when the remaining dis-
                                       tance 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]

```
10 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.

This status variable can only be used in certain models (Refer to "[Available robot type]".). This function is available for controller software version J2 or later.

[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]

```
10 ACCEL 100,50           ' Set the overall acceleration/deceleration distribution rate to 50%.
20 IF M_LDFACT(2)>90 THEN ' If the load rate of the J2 axis exceeds 90%,
30   M?SETADL(2)=70       ' set the acceleration/deceleration time distribution rate of the J2
                            axis to 70%.
40 ENDIF                  ' Acceleration 70% (= 100% x 70%), deceleration 35% (= 50% x
                            70%)
50 MOV P1
60 MOV P2
70 M_SETADL(2)=100        ' Return the acceleration/deceleration time distribution rate of the
                            J2 axis to 100%.
80 MOV P3                 ' Acceleration 100%, deceleration 50%
90 ACCEL 100,100          ' Return the overall deceleration distribution rate to 100%.
100 MOV P4
```

[Explanation]

(1) The acceleration/deceleration time distribution rate when optimum acceleration/deceleration is enabled can be set in units of axes. If 100% is specified, the acceleration/deceleration time becomes the shortest.

(2) Using this status variable, the acceleration/deceleration time can be set so as to reduce the load on axes where overload and overheat errors occur.

(3) The setting of this status variable is applied to both the acceleration time and deceleration time.

(4) When this status variable is used together with an ACCEL instruction, the specification of the acceleration/deceleration distribution rate of the ACCEL instruction is also applied to the acceleration/deceleration time calculated using the optimum acceleration/deceleration speed.

(5) With the ACCEL instruction, the acceleration/deceleration time changes at the specified rate. Because this status variable is set independently for each axis and also the acceleration/deceleration time that takes account of the motor load is calculated, the change in the acceleration/deceleration time may show a slightly different value than the specified rate.

[Reference]

ACCEL (Accelerate),OVRD (Override),SPD (Speed),M_LDFACT

[Available robot type]

RV-3S/3SJ/3SB/3SJB series
RV-6S/6SL/12S/12SL series

M_SKIPCQ

[Function]

Returns the result of executing the line containing the last executed SKIP command.
1 : SKIP has been executed.
0 : SKIP has not been executed.

[Format]

```
Example)<Numeric Variable>=M_SKIPCQ [(<Equation>)]
```

[Terminology]

<Numeric Variable> Specifies the numerical variable to assign.

<Equation> 1 to 32, Specifies the task slot number. If this parameter is omitted, the current slot will be used as the default.

[Reference Program]

```
10 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.
20 IF M_SKIPCQ=1 THEN GOTO 1000      ' If SKIP instruction has been executed, jump to line 1000.
ÅE
1000 END
```

[Explanation]

(1) Checks if a SKIP instruction has been executed.
(2) This variable only reads the data.
(3) If the M_SKIPCQ variable is referenced even once, the SKIP status is cleared. (The value is set to zero.) Therefore, to preserve the status, save it by substituting it into a numeric variable.

M_SPD/M_NSPD/M_RSPD

[Function]

Returns the speed information during XYZ and JOINT interpolation.
M_SPD : Currently set speed.
M_NSPD : Initial value (optimum speed control).
M_RSPD : Directive speed.

[Format]

```
Example)<Numeric Variable>=M_SPD [(<Equation>)]

Example)<Numeric Variable>=M_NSPD [(<Equation>)]

Example)<Numeric Variable>=M_RSPD [(<Equation>)]
```

[Terminology]

<Numeric Variable>	Specifies the numerical variable to assign.
<Equation>	1 to 32, Specifies the task slot number. If this parameter is omitted, the current slot will be used as the default.

[Reference Program]

```
10 M1=M_SPD         ' M1 will contain the currently set speed.
20 SPD M_NSPD       ' Reverts the speed to the optimum speed control mode.
```

[Explanation]

(1) M_RSPD returns the directive speed at which the robot is operating.
(2) This can be used in M_RSPD multitask programs or with WTH and WTHIF statements.
(3) This variable only reads the data.

M_SVO

[Function]

Returns the current status of the servo power supply.
1 : Servo power ON
0 : Servo power OFF

[Format]

```
Example)<Numeric Variable>=M_SVO [(<Mechanism Number>)]
```

[Terminology]

<Numeric Variable>	Specifies the numerical variable to assign.
<Mechanism Number>	Enter the mechanism number. 1 to 3, If the argument is omitted, 1 is set as the default value.

[Reference Program]

```
10 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_TMER (<Equation>)
```

[Terminology]

<Numeric Variable>	Specifies the numerical variable to assign.
<Equation>	Enter the number to 8 from 1. Parentheses are required.

[Reference Program]

```
10 M_TIMER(1)=0
20 MOV P1
30 MOV P2
40 M1=M_TIMER(1)        ' M1 will contain the amount of time required to move from P1 to P2 (in ms).
                          Example) If the time is 5.346 sec. the value of M1 is 5346.
50 M_TIMER(1)           ' Set to 1.5 sec.
```

[Explanation]

(1) A value may be assigned. The unit is seconds when set to M_TIMER.

(2) Since measurement can be made in milliseconds (ms), precise execution time measurement is possible.

M_TOOL

[Function]

In addition to using the tool data (MEXTL1 to 4) of the specified number as the current tool data, it is also set in the MEXTL parameter.

The current tool number can also be read.. This function is available for controller software version J1 or later

[Format]

```
Example)<Numeric Variable>=M_TOOL [(<Mechanism Number>)]'Referencing the Current Tool Number

Example)M_TOOL [(<Mechanism Number>)] = [(<Equation>)]  'Set a tool number.
```

[Terminology]

<Numeric Variable>	Specifies the numerical variable to assign.
<Mechanism Number>	Enter the mechanism number to 3 from 1. If the argument is omitted, 1 is set as the default value.
<Equation>	Enter the tool number to 4 from 1.

[Reference Program]

Setting Tool Data

```
10 TOOL (0,0,100,0,0,0)         ' Specify tool data (0,0,100,0,0,0), and write it into MEXTL.
20 MOV P1
30 M_TOOL=2                     ' Change the tool data to the value of tool number 2 (MEXTL2).
40 MOV P2
```

Referencing the Tool Number

```
10 IF M_IN(900)=1 THEN          ' Change the tool data by a hand input signal.
20   M_TOOL=1                   ' Set tool 1 in tool data.
30 ELSE
40   M_TOOL=2                   ' Set tool 2 in tool data.
50 ENDIF
60 MOV P1
```

[Explanation]

(1) The values set in the MEXTL1, MEXTL2, MEXTL3 and MEXTL4 tool parameters are reflected in the tool data. It is also written into the MEXTL parameter.
(2) Tool numbers 1 to 4 correspond to MEXTL1 to 4.
(3) While referencing, the currently set tool number is read.
(4) If the reading value is 0, it indicates that tool data other than MEXTL1 to 4 is set as the current tool data.
(5) The same setting can be performed on the Tool Setup screen of the teaching pendant. For more information, see Page 19, "3.2.8 Switching Tool Data".

[Reference]

TOOL (Tool), MEXTL, MEXTL1, MEXTL2, MEXTL3, MEXTL4

M_UAR

[Function]

Returns whether the robot is in the user-defined area.
Bits 0 through 7 correspond to areas 1 through 8.
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]

```
10 M1=M_UAR(1)          ' M1 indicates whether the robot is within or outside the user-defined area.
                         The value 4 indicates that the robot is in the user-defined area 3.
```

[Explanation]

(1) For details on how to use user-defined areas, refer to Page 328, "About user-defined area".
(2) This variable only reads the data.

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]

```
10 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.This status variable can be used in the controller's software version J8 or later.

Note: For more information about the warm-up operation mode, see Page 355, "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]

```
10 M1=M_WUPOV(1)        ' The value of a warm-up operation override is entered in M1.
```

[Explanation]

(1) This is used to confirm the value of an override (warm-up operation override) to be applied to the command speed in order to reduce the operation speed when the robot is in the warm-up operation status (the status in which operation is performed by automatically reducing the speed).

(2) If the warm-up operation mode is disabled, the MODE switch on the front of the controller is set to "TEACH," or the machine is being locked, the value is always 100.

(3) If the normal status changes to the warm-up operation status, or the warm-up operation status is set immediately after power on, the value specified in the first element (the initial value of a warm-up operation override) of the WUPOVRD parameter is set as the initial value, and the value of M_WUPOV increases according to the operation of the robot. And when the warm-up operation status is canceled, the value of M_WUPOV is set to 100.

(4) The actual override in the warm-up operation status is as follows:

During joint interpolation operation = (operation panel (T/B) override setting value) x (program override (OVRD instruction)) x (joint override (JOVRD instruction)) x warm-up operation override

During linear interpolation operation = (operation panel (T/B) override setting value) x (program override (OVRD instruction)) x (linear specification speed (SPD instruction)) x warm-up operation override

(5) This variable only reads the data.

M_WUPRT

[Function]

Returns the time (sec) during which a target axis must operate to cancel the warm-up operation status. This status variable can be used in the controller's software version J8 or later.

Note: For more information about the warm-up operation mode, see Page 355, "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]

```
10 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.
This status variable can be used in the controller's software version J8 or later.
Note: For more information about the warm-up operation mode, see Page 355, "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]

```
10 M1=M_WUPST(1)     ' The time until the warm-up operation status is set again is entered in M1.
```

[Explanation]

(1) This is used to confirm when the warm-up operation status is set again after how long more the joint axis specified in the WUPAXIS parameter (warm-up operation mode target axis) continues to stop operating while the robot’s warm-up operation status (the status in which operation is performed by automatically reducing the speed) is canceled.

(2) If the warm-up operation mode is disabled, the time specified in the second element (warm-up operation mode resume time) of the WUPTIME parameter is returned.

(3) If a target axis operates while the warm-up operation status is canceled, the time specified in the second element (warm-up operation mode resume time) of the WUPTIME parameter is set as the initial value, and the value of M_WUPST decreases while the target axis is stopping. And when the value is set to 0, the warm-up operation status is set.

(4) If a multiple number of target axes exist, the value of the axis that has been stopped the longest among them is returned.

(5) This variable only reads the data.

P_BASE/P_NBASE

[Function]

Returns information related to the base conversion data.
P_BASE : Returns the base conversion data that is currently being set.
P_NBASE : Returns the initial value (0, 0, 0, 0, 0, 0) (0, 0).

[Format]

```
Example)<Position Variables>=P_BASE [(<Mechanism Number>)]

Example)<Position Variables>=P_NBASE
```

[Terminology]

<Position Variables> Specifies the position variable to assign.

<Mechanism Number> Enter the mechanism number. 1 to 3, If the argument is omitted, 1 is set as the default value.

[Reference Program]

```
10 P1=P_BASE           ' P1 will contain the base conversion data that is currently being set.
20 BASE P_NBASE        ' Resets the base conversion data to the initial value.
```

[Explanation]

(1) P_NBASE will contain (0, 0, 0, 0, 0, 0) (0, 0).
(2) Be careful when using base conversion since it may affect the teaching data.
(3) Use the BASE instruction when changing the base position.
(4) This variable only reads the data.

P_COLDIR

[Function]

Return the operation direction of the robot when an impact is detected.
The impact detection function can only be used in certain models (Refer to "[Available robot type]".). This function is available for controller software version J2 or later.

[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 141, " [Reference Program 2]" for "COLCHK (Col Check)".

[Explanation]

(1) This is used to verify the operation direction of the robot in automatic restoration operation after impact detection.
(2) The operation direction of the robot at the very moment of impact detection is expressed as a ratio using the maximum travel axis as @1.0. Example: If the robot was being operated at a ratio of (X-axis direction:Y-axis direction) = (2:-1)...P_COLDIR = (1,-0.5,0,0,0,0)(0,0)
(3) The posture axis and structural flag are always (*.*.*.0,0,0,0,0)(0,0).
(4) A value is calculated when an impact is detected, and then that value is retained until the next impact is detected.
(5) If an impact is detected when an external object hits the robot in the stationary state, all axes are set to 0.0.
(6) Because this variable calculates the operation direction based on the target position of an operation instruction, all elements may be set to 0.0 if an impact occurs at a position near the target position.
(7) This is read only.
(8) For robots that prohibit the use of impact detection, 0.0 is always returned for all axes.

[Reference]

COLCHK (Col Check), COLLVL (Col Level), M_COLSTS, J_COLMXL

[Available robot type]

RV-3S/3SJ/3SB/3SJB series
RV-6S/6SL/12S/12SL series
RH-6SH/12SH/18SH series

P_CURR

[Function]

Returns the current position (X, Y, Z, A, B, C,L1,L2) (FL1, FL2).

[Format]

Example)<Position Variables>=P_CURR [(<Mechanism Number>)]

[Terminology]

<Position Variables>   Specifies the position variable to assign.

<Mechanism Number>   Enter the mechanism number. 1 to 3, If the argument is omitted, 1 is set as the default value.

[Reference Program]

```
10 DEF ACT 1,M_IN(10)=1 GOTO 1000       ' Defines interrupt.
20 ACT 1=1                              ' Enables interrupt.
30 MOV P1
40 MOV P2
50 ACT 1=0                              ' Disables interrupt.

1000 P100=P_CURR                        ' Loads the current position when an interrupt signal is
                                          received.
1010 MOV P100,-100                      ' Moves 100 mm above P100 (i.e, -100 mm in the Z direc-
                                          tion of the tool).
1020 END                                ' Ends the program.
```

[Explanation]

(1) This can be used to identify the current position.

(2) This variable only reads the data.

[Reference]

J_CURR

P_FBC

[Function]

Returns the current position (X,Y,Z,A,B,C,L1,L2)(FL1,FL2) based on the feedback values from the servo.

[Format]

```
Example)<Position Variables>=P_FBC [(<Mechanism Number>)]
```

[Terminology]

<Position Variables> Specifies the position variable to assign.

<Mechanism Number> Enter the mechanism number. 1 to 3, If the argument is omitted, 1 is set as the default value.

[Reference Program]

```
10 P1=P_FBC          ' P1 will contain the current position based on the feedback.
```

[Explanation]

(1) Returns the current position based on the feedback values from the servo.
(2) This variable only reads the data.

[Reference]

TORQ (Torque),J_FBC/J_AMPFBC,M_FBD

P_SAFE

[Function]

Returns the safe point (XYZ position of the JSAFE parameter).

[Format]

```
Example)<Position Variables>=P_SAFE [(<Mechanism Number>)]
```

[Terminology]

<Position Variables> Specifies the position variable to assign.

<Mechanism Number> Enter the mechanism number. 1 to 3, If the argument is omitted, 1 is set as the default value.

[Reference Program]

```
10 P1=P_SAFE         ' P1 will contain the set safe point being set.
```

[Explanation]

(1) Returns the XYZ position, which has been converted from the joint position registered in parameter JSAFE.
(2) This variable only reads the data.

P_TOOL/P_NTOOL

[Function]

Returns tool conversion data.
P_TOOL: Returns the tool conversion data that is currently being set.
P_NTOOL: Returns the initial value (0,0,0,0,0,0,0,0)(0,0).

[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]

```
10 P1=P_TOOL      ' P1 will contain the tool conversion data.
```

[Explanation]

(1) P_TOOL returns the tool conversion data set by the TOOL instruction or the MEXTL parameter.
(2) Use the TOOL instruction when changing tool data.
(3) This variable only reads the data.

P_ZERO

[Function]

Always returns (0,0,0,0,0,0,0,0)(0,0).

[Format]

```
Example)<Position Variables>=P_ZERO
```

[Terminology]

<Position Variables>   Specifies the position variable to assign.

[Reference Program]

```
10 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.

4.13 Detailed Explanation of Functions

4.13.1 How to Read Described items

[Function]	: This indicates a function of a function.
[Format]	: This indicates how to input the function argument.
[Reference Program]	: An example program using function is shown.
[Terminology]	: This indicates the meaning and range of an argument.
[Explanation]	: This indicates detailed functions and precautions.
[Reference]	: This indicates related function.

4.13.2 Explanation of Each Function

Each variable is explained below in alphabetical order.

ABS

[Function]

Returns the absolute value of a given value.

[Format]

```
<Numeric Variable>=ABS(<Equation>)
```

[Reference Program]

```
10 P2.C=ABS(P1.C)        ' P2.C will contain the value of P1.C without the sign.
20 MOV P2
30 M2=-100
40 M1=ABS(M2)            ' 100 is assigned to M1.
```

[Explanation]

(1) Returns the absolute value (Value with the positive sign) of a given value.

[Reference]

SGN

ALIGN

[Function]

Positional posture axes (A, B, and C axes) are converted to the closest XYZ postures (0, +/-90, and +/-180). ALIGN outputs numerical values only. The actual operation will involve movement instructions such as the MOV instruction.

[Format]

```
<Position Variables>=ALIGN(<Position>)
```

[Reference Program]

```
10 P1=P_CURR
20 P2=ALIGN(P1)
30 MOV P2
```

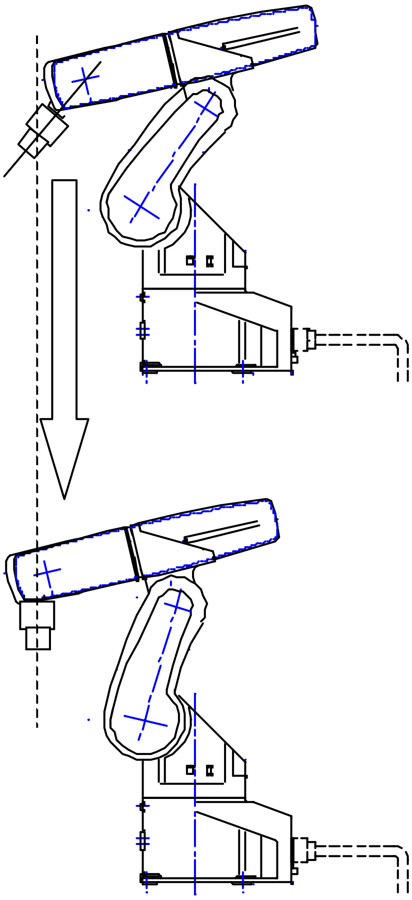
[Explanation]

(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]

```
    10 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

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]

```
   10 M1=ATN(100/100)           'PI/4 is assigned to M1.
   20 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]

```
10 M1=&B11111111
20 C1$=BIN$(M1)          ' C1$ will contain the character string of "11111111".
```

[Explanation]

(1) Value is converted to a binary string.
(2) If the equation does not evaluate to an integer, the integral value obtained by rounding the fraction will be converted to a binary string.
(3) VAL is a command that performs the opposite of this function.

[Reference]

HEX$, STR$, VAL

CALARC

[Function]

Provides information regarding the arc that contains the three specified points.

[Format]

```
<Numeric Variable 4> = CALARC(<Position 1>, <Position 2>, <Position 3>,
                        <Numeric Variable 1>, <Numeric Variable 2>, <Numeric Variable 3>,
                          <Position Variables 1>)
```

[Terminology]

<Position 1> Specifies the starting point of the arc.

<Position 2> Specifies the passing point of the arc. Same as the three points in the MVR instruction.

<Position 3> Specifies the endpoint of the arc.

<Numeric Variable 1> Radius of the specified arc (in mm) will be calculated and returned.

<Numeric Variable 2> Central angle of the specified arc (in radians) will be calculated and returned.

<Numeric Variable 3> Length of the specified arc (in mm) will be calculated and returned.

<Position Variables 1> The center coordinates of the specified arc (in mm) will be calculated and returned (as a position data type, ABC are all zeros).

<Numeric Variable 4> Return value

1 : Calculation was performed normally.

-1 : Of positions 1, 2, and 3, either two points had the exact same position or all three points were on a straight line.

-2 : All three points are at approximately the same position.

[Reference Program]

```
10 M1=CALARC(P1,P2,P3,M10,M20,M30,P10)
20 IF M1<>1 THEN END   ' Ends if an error occurs.
30 MR=M10              ' Radius.
40 MRD=M20             ' Circular arc angle.
50 MARCLEN=M30         ' Circular arc length.
60 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]

```
10 M1=&H40
20 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]

```
10 M1=CINT(1.5)    ' 2 is assigned to M1.
20 M2=CINT(1.4)    ' 1 is assigned to M2.
30 M3=CINT(-1.4)   ' -1 is assigned to M3.
40 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]

```
10 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]

```
10 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]

```
10 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]

```
10 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]

```
    10 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]

```
    10 P1=P_CURR
    20 IF DEG(P1.C) < 170 OR DEG(P1.C) > -150 THEN *NOERR
    30 ERROR(9100)
  40 *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]

```
10 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]

10 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]

```
10 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]

```
10 BASE P_NBASE
20 P100=FRAM(P1,P2,P3)     ' Create P100 coordinate system based on P1, P2 and P3 positions.
30 P10=INV(P10)
40 P10. X=P1. X
50 P10. Y=P1. Y
60 P10. Z=P1. Z
70 BASE P10                ' Position of P100 will be used as the origin for robot.
        :
```

[Explanation]

(1) This can be used to define the base coordinate system.

(2) This creates a plane from the three coordinates X, Y, and Z for the three positions to calculate the position of the origin and the inclination of the plane, and returns the result as a position variable. The X, Y, and Z coordinates of the position data will be identical to that of position variable 1, while A, B, and C will be the inclination of the plane to be specified by the three positions.

(3) Since the return value is a position data, an error will be generated if a joint variable is used in the left-hand side.

(4) It is not possible to describe a function that contains an argument in <position 1>, <position 2> and <position 3>. If such a function is described, an error will be generated during execution.
NG exampleP10=FRAM(FPRM(P01,P02,P03), P04, P05)

[Reference]

Relative conversion (* operator). Refer to Page 328, "5.8 About user-defined area".

HEX$

[Function]

Converts the value of an equation (Between -32768 to 32767) into hexadecimal string.

[Format]

```
<Character String Variable >=HEX$(<Equation> [, <Number of output characters>])
```

[Reference Program]

```
10 C1$=HEX$(&H41FF)        ' "41FF" is assigned to C1$.
20 C2$=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]

```
10 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]

```
10 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]

```
10 P1=JTOP(J1)                ' The position that expresses the J1 (joint type) position using the XYZ
                                type will be assigned to P1.
```

[Explanation]

(1) Converts the joint data into the position data.
(2) Position variables cannot be used as the argument. When a position variable is used, an error will be generated.
(3) Since the return value is a position data, an error will be generated if a joint variable is used in the left-hand side.

[Reference]

PTOJ

LEFT$

[Function]

Obtains a string of the specified length starting from the left end.

[Format]

```
<Character String Variable >=LEFT$(<Character String>, <Equation>)
```

[Reference Program]

```
10 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]

```
10 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]

```
    10 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]

```
    10 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]

```
10 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]

```
10 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]

```
10 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]

```
10 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]

```
10 C1$=MKI$(20299)          ' "OK" is assigned to C1$.
20 M1=CVI(C1$)              ' 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]

```
10 C1$=MKS$(100.1)          '
20 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]

```
10 C1$=MKD$(10000.1)         '
20 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]

```
10 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]

```
10 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]

```
10 J1=PTOJ(P1)             ' J1 will contain the value of P1 (XYZ position variable) that has been con-
                             verted into joint data type.
```

[Explanation]

(1) Converts the position data into the joint data.
(2) Joint variables(J variable) cannot be used as the argument. When a joint variable is used, an error will be generated.
(3) Since the return value is a joint data, an error will be generated if a position variable is used in the left-hand side.

[Reference]

JTOP

RAD

[Function]

Converts the unit of angle measurement from degrees (deg) into radians (rad).

[Format]

```
<Numeric Variable>=RAD(<Equation>)
```

[Reference Program]

```
10 P1=P_CURR
20 P1.C=RAD(90)
30 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]

```
10 P1=(100,0,100,180,0,180)(7,0)' Since the structure flag 7 (&B111) is RAN,
20 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]

```
10 P1=(100,0,100,180,0,180)(7,&H00100000)'
20 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]

```
10 DIM MRND(10)
20 C1=RIGHT$(C_TIME,2)       ' Initializes random numbers using the clock.
30 MRNDBS=CVI(C1))           ' in order to obtain different sequence of numbers.
40 MRND(1)=RND(MRNDBS)       ' Sets the initial value of random numbers and extracts the first random
                               number.
50 FOR M1=2 TO 10            ' Obtain other nine random numbers.
60   MRND(M1)=RND(0)
70 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]

```
10 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]

```
10 MOV P1
20 P2=SETFL1(P1,"LBF")
30 MOV P2
```

[Explanation]

(1) Returns the position data obtained by changing the structure flags in the position data specified by argument 1 to flag values specified by argument 2.

(2) This function changes information from the FL1 element of position data. The content of the position data given by the argument will remain unchanged.

(3) The structure flag will be specified starting from the last character in the string. Therefore, for instance, if the string "LR" is specified, the resulting structure flag will be "L".

(4) If the flags are changed using a numerical value, set P1.FL1=7.

(5) Structure flags may have different meanings depending on the robot model. For details, please refer to "ROBOT ARM SETUP & MAINTENANCE" for each robot.

The structure flag corresponds to 7 in the position constant (100, 0, 300, 180, 0, 180) (7, 0). The actual position is a bit pattern.

```
7 = &B 0 0 0 0  0  1  1  1
                   |  |  └──1/0=N/F
                   |  └─────1/0=A/B
                   └────────1/0=R/L
```

(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).

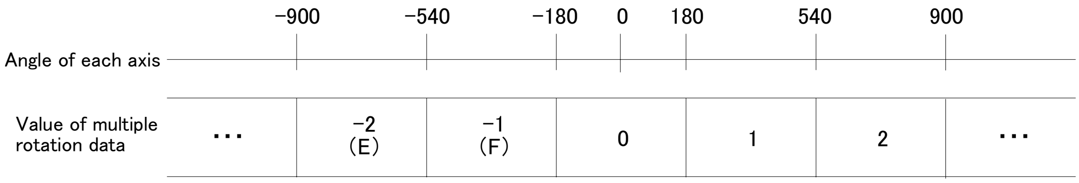
[Reference Program]

```
10 MOV P1
20 P2=SETFL2(P1,6,1)
30 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.

Value of multiple rotation data



(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.
This function is available for controller software version J2 or later.

[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]

```
10  J1=J_CURR
20  FOR M1=0 to 60 SETP 10
30    M2=J1.J3+RAD(M1)
40    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.
50    MOV J2
60  NEXT M1
70  M0=RAD(0)
80  M90=RAD(90)
70  J3=SETJNT(M0,M0,M90,M0,M90,M0)
100 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
This function is available for controller software version J2 or later.

[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]

```
10 P1=P_CURR
20 FOR M1=0 to 100 SETP 10
30   M2=P1.Z+M1
40   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.
50   MOV J2
60 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]

```
10 M1=-12
20 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]

```
10 M1=SIN(RAD(60))          ' 0.866025 is assigned to M1.
```

[Explanation]

(1) Calculates the sine to which the given equation evaluates.
(2) The range of values will be the entire range that numerical values can take.
(3) The range of the return value will be from -1 to 1.
(4) The unit of arguments is in radians.

[Reference]

COS, TAN, ATN/ATN2

SQR

[Function]

Calculates the square root of an equation value.

[Format]

```
<Numeric Variable>=SQR(<Equation>)
```

[Reference Program]

```
10 M1=SQR(2)                ' 1.414214 is assigned to M1.
```

[Explanation]

(1) Calculates the square root of the value to which the given equation evaluates.
(2) An error will be generated if the equation given by the argument evaluates to a negative value.

STRPOS

[Function]

Searches for a specified string in a string.

[Format]

```
<Numeric Variable>=STRPOS(<Character String 1>, <Character String 2>)
```

[Reference Program]

```
10 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]

```
10 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]

```
10 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]

```
10 M1=VAL("15")
20 M2=VAL("&B1111")
30 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]

```
  10 M1=ZONE(P1,P2,P3)
20 IF M1=1 THEN MOV P_SAFE ELSE END
```

[Explanation]

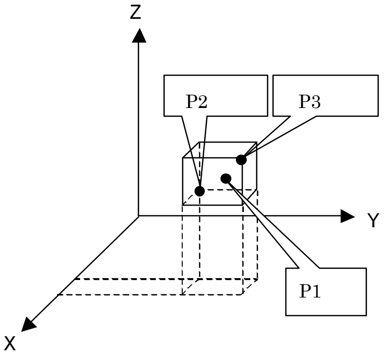
(1) This will check if position 1 is inside the rectangular solid defined by the two points, position 2 and position 3. (The two points will become the diagonal points of the rectangular solid.) If the point is inside the rectangular solid, 1 is returned; otherwise, 0 is returned.

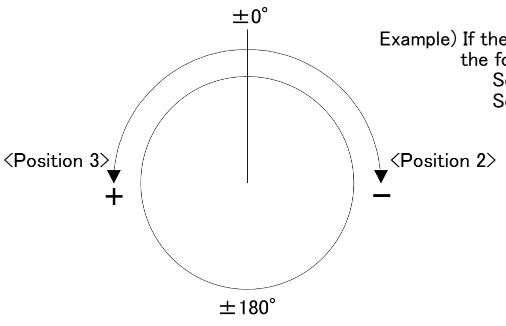
(2) To check whether position 1 is inside that area, each element of position 1 (X, Y, Z, A, B, C, L1 and L2) will be checked if it is between the values for position 2 and position 3.

(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.)

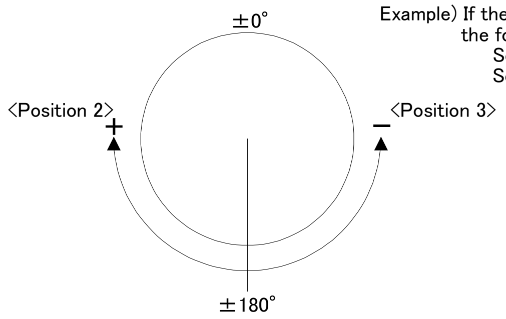
(4) For components that are not checked or do not exist, if the unit is in degrees, position 2 will be set to -360 and position 3 will be set to 360. If the unit is in millimeters, position 2 will be set to -10000 and position 3 will be set to 10000.

(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]

This function is available for controller software version G5 or later

```
<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.

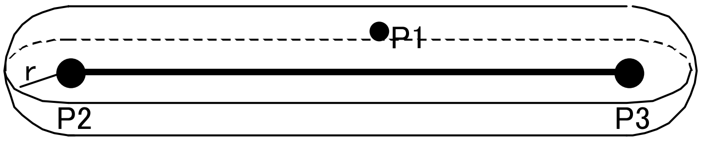
[Reference Program]

```
10 M1=ZONE2(P1,P2,P3,50)
20 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.

5 Functions set with parameters

This controller has various parameters listed in Table 5-2. It is possible to change various functions and default settings by changing the parameter settings.

No.	Classification	Content	Reference
1	Movement parameter	These parameters set the movement range, coordinate system and the items pertaining to the hand of the robot.	Page 306
2	Signal parameter	These parameters set the items pertaining to signals.	Page 314
3	Operation parameter	These parameters set the items pertaining to the operations of the controller, T/B and so forth.	Page 315
4	Command parameter	These parameters set the items pertaining to the robot language.	Page 318
5	Communication parameter	These parameters set the items pertaining to communications.	Page 322

For the parameters regarding dedicated I/O signals, refer to Page 371, "6.3 Dedicated input/output". After changing the parameters, make sure to turn the robot controller's power OFF and then turn ON. Parameter settings will not be in effect until the power is turned on again. For detailed operating method for parameters, refer to Page 54, "(1) Setting the parameters".

⚠CAUTION When changing parameters, check thoroughly the function and setting values first. Otherwise, the robot may move unexpectedly, which could result in personal injury or property damage.

5.1 Movement parameter

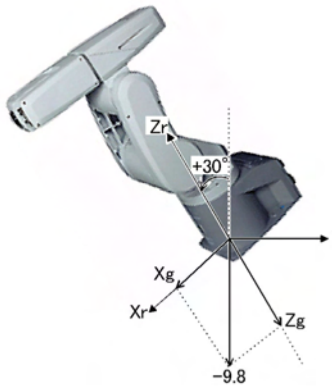
These parameters set the movement range, coordinate system and the items pertaining to the hand of the robot.

Table 5-1:List Movement parameter

Parameter	Parameter name	No. of arrays No. of characters	Details explanation	Factory setting
Joint movement range	MEJAR	Real value 16	Set the overrun limit value for the joint coordinate system. Sets the movement range for each axis. Expanding of the movement range is not recommended, since there is possibility that the robot may strike the mechanical stopper. Set the minus and plus directions. (-J1,+J1,-J2,+J2,......-J8,+J8) Unit:deg	Setting value for each mechanism
XYZ movement range	MEPAR	Real value 6	Set the overrun limit value for the XYZ coordinate system. The movement range of the robot will be limited based on XYZ coordinate system. This can be used to prevent the robot from striking peripheral devices during manual operation when the robot is installed within the device. Set the minus and plus directions. (-X,+X,-Y,+Y,-Z,+Z) Unit:mm	(-X,+X,-Y,+Y,-Z,+Z)= -10000,10000, -10000,10000, -10000,10000
Standard tool coordinates Refer to "5.6Standard 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 Refer to "M_TOOL"	MEXTL1	Real value 6	If the M_TOOL variable is substituted by 1, the tool data can be switched using this parameter value.	(X,Y,Z,A,B,C) = 0.0,0.0,0.0,0.0,0.0,0.0
Tool coordinate 2 Refer to "M_TOOL"	MEXTL2	Real value 6	If the M_TOOL variable is substituted by 2, the tool data can be switched using this parameter value.	(X,Y,Z,A,B,C) = 0.0,0.0,0.0,0.0,0.0,0.0
Tool coordinate 3 Refer to "M_TOOL"	MEXTL3	Real value 6	If the M_TOOL variable is substituted by 3, the tool data can be switched using this parameter value.	(X,Y,Z,A,B,C) = 0.0,0.0,0.0,0.0,0.0,0.0

Parameter	Parameter name	No. of arrays No. of characters	Details explanation	Factory setting
Tool coordinate 4 Refer to "M_TOOL"	MEXTL4	Real value 6	If the M_TOOL variable is substituted by 4, the tool data can be switched using this parameter value.	(X,Y,Z,A,B,C) = 0.0,0.0,0.0,0.0,0.0,0.0
Tool base coordinates Refer to "5.6Standard Tool Coordinates"	MEXBS	Real value 6	Sets the positional relationship between the base coordinate system and the robot coordinate system. The factory default setting is set so that the base coordinate system and the robot coordinate system are identical. This will be set when the coordinate system for the whole device is changed. This parameter does not need to be changed very often. This is set when the coordinate system for the whole device is to be identical. (X, Y, Z, A, B, C) Unit: mm, ABC deg.	(X,Y,Z,A,B,C) = 0.0,0.0,0.0,0.0,0.0,0.0
User area Refer to "5.8About user-defined area"			Designate an area (rectangle defined with two XYZ coordinate points. A signal will be output if that area is outside the movement area (interference), or if the robot's current position is within that area. Up to eight limits can be set using the following four types of parameters. For components that are not checked or do not exist, if the unit is in degrees, set AREA*P1 to -360 and AREA*P2 to 360. If the unit is in mm, set AREA*P1 to -10000 and AREA*P2 to 10000 as the corresponding component.	
	AREA*P1 * is 1 to 8	Real value 8	Designate the first point of the area. (X, Y, Z, A, B, C) Unit: mm, ABC deg.	(X,Y,Z,A,B,C)= 0.0,0.0,0.0,-360.0,-360.0,-360.0
	AREA*P2 * is 1 to 8	Real value 8	Designate the secand point of the area. (X, Y, Z, A, B, C) Unit: mm, ABC deg.	(X,Y,Z,A,B,C)= 0.0,0.0,0.0,+360.0,+360.0,+360.0
	AREA*ME * is 1 to 8	Integer 1	Designate the mechanism No. for which the user-defined area is to be validated. The mechanism No. is 1 to 4, but normally 1 is set.	0
	AREA*AT * is 1 to 8	Integer 1	Specify the behavior of the robot when the robot enters the user definition area. (Invalid / In-zone signal output/Error output=0/1/2) Invalid:This function will be invalid. In-zone signal output:The dedicated output signal USRAREA will turn ON. Error output:An error is generated. Posture data will be ignored.	0(Invalid)
	USRAREA		Defines the number of the signal that outputs the status. Refer to Page 371, "6.3 Dedicated input/output"	-1,-1
Free plane limit Refer to "5.9Free plane limit"			This is the overrun limit set on a free plane. Create a plane with three coordinate points, and set the area that does not include the origin as the outside-movement area. Up to eight limits can be set using the following three types of parameters.	
*The function [-1: The operable area is the side where the robot coordinate origin does not exist] can be used in the controller's software version J1 or later.	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, 0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0
	SFC*ME * is 1 to 8	Integer 1	Designate the mechanism No. for which the free plane limit is to be validated. The mechanism No. is 1 to 3, but normally 1 is set.	0
	SFC*AT * is 1 to 8	Integer 1	Designate the valid/Invalid of the set free plane limit. 0:Invalid 1: Valid (The operable area is the robot coordinate origin side.) -1: Valid (The operable area is the side where the robot coordinate origin does not exist.)	0(Invalid)
Safe point position	JSAFE	Real value 8	Specifies the safe point position. Robot moves to the safe point position if the robot program executes MOV P_SAFE instruction or receives input of the SAFEPOS signal, which is an external signal. (J1,J2,J3,J4,J5,J6,J7,J8) Unit:deg	Not allowed
User-designated origin	USERORG	Real value 8	Designate the user-designated origin position. This normally does not need to be set. (J1,J2,J3,J4,J5,J6,J7,J8) Unit:deg	(J1,J2,J3,J4,J5,J6, J7,J8)= 0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0

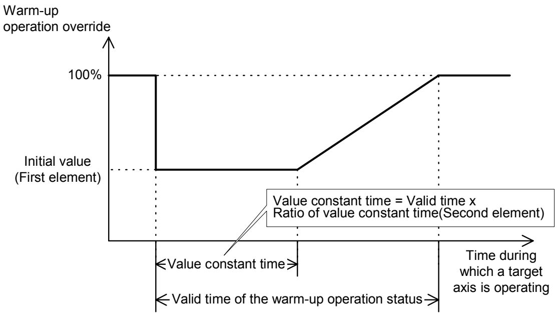
Parameter	Parameter name	No. of arrays No. of characters	Details explanation	Factory setting
User-designated origin	USERORG	Real value 8	Designate the user-designated origin position. This normally does not need to be set. (J1,J2,J3,J4,J5,J6,J7,J8) Unit:deg	(J1,J2,J3,J4,J5,J6, J7,J8)= 0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0
Select the function of singular point adjacent alarm Refer to Page 344, "5.17 About the singular point adjacent alarm" **This parameter can be used for controller software version G8 or later.	MESNGLSW	Integer 1	Designate the valid/invalid of the singular point adjacent alarm. (Invalid/Valid=0/1) When this parameter is set up "VALID", this warning sound is buzzing even if parameter: BZR (buzzer ON/OFF) is set up "OFF".	1(Valid)
Jog setting	JOGJSP	Real value 3	Designate the joint jog and step operation speed. (Inching H, inching L, maximum override.) Inching H: Feed amount when jog speed is set to High Unit: deg. Inching L: Feed amount when jog speed is set to Low Unit: deg. Maximum override: Operates at OP override x maximum override.	Setting value for each mechanism
	JOGPSP	Real value 3	Designate the XYZ jog and step operation speed. (Inching H, inching L, maximum override.) Inching H: Feed amount when jog speed is set to High Unit: deg. Inching L: Feed amount when jog speed is set to Low Unit: deg. Maximum override: Operates at OP override x maximum override. Operation exceeding the maximum speed 250 mm/s cannot be performed.	Setting value for each mechanism
Jog speed limit value	JOGSPMX	Real value 1	Limit the robot movement speed during the teach mode. Unit: mm/s Even if a value larger than 250 is set, the maximum value will be limited to 250.	250.0
Automatic return setting after jog feed at pause Refer to "5.10Automatic return setting after jog feed at pause" *The "2: Return by XYZ interpolation" is available for controller software version H4 or later.	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.	The RH-1000G and 1500G series and RH-15UHC are 2.. The type other than the above are 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
*This parameter can be used for controller software version H4 or later.
Note) This function can be used in RV-1A/2AJ, RV-4A/5AJ series, and RV-20A.</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/second2).
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</td><td>(2.0, 0.0, 0.0, 0.0)</td></tr>
<tr><td>Optional posture[*1]</td><td>(3.0, ***, ***, ***)</td></tr>
</table>
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) or CC-Link (6000’S) (specifying a signal other than one in 900’S by the HANDTYPE parameter), do not use this HANDINIT parameter, but use the ORST* parameter.
The value set by the ORST* parameter becomes the initial value of signals at power ON.</td><td>1,0,1,0,1,0,1,0</td></tr>
<tr><td>Hand type
Refer to "5.12About the hand type"</td><td>HANDTYPE</td><td>Character string 8</td><td>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</td><td>D900,D902,D904,D906,,,,</td></tr>
</table>

Parameter	Parameter name	No. of arrays No. of characters	Details explanation	Factory setting
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.	
	HNDDAT0	Real value 7	Set the initial condition of the hand. (Designate with the tool coordinate system.) Immediately after power ON, this setting value is used. To use the impact detection function during jog operation, set the actual hand condition before using. If it is not set, erroneous detection may occur. (Weight, size X, size Y, Size Z, center of gravity X, center of gravity Y, center of gravity Z) Unit: Kg, mm	RV-3S/3SJ/3SB/3SJB 3.50, 284.00, 284.00, 286.00, 0.00, 0.00, 75.00 RV-6S/6SL 6.00, 213.00, 213.00, 17.00, 0.00, 0.00, 130.00 RV-12S/12SL 12.00, 265.00, 265.00, 22.00, 0.00, 0.00, 66.00 RH-6SH 6.00, 99.00, 99.00, 76.00, 0.00, 0.00, 38.00 RH-12SH 12.00, 225.00, 225.00, 30.00, 0.00, 0.00, 15.00 RH-18SH 18.00, 258.00, 258.00, 34.00, 0.00, 0.00, 17.00 Other type are secret.
	HNDDAT* * is 1 to 8	Real value 7	Set the initial condition of the hand. (Designate with the tool coordinate system.) (Weight, size X, size Y, Size Z, center of gravity X, center of gravity Y, center of gravity Z) Unit: Kg, mm	Standard load ,0.0,0.0,0.0,0.0,0.0, 0.0
	WRKDAT0	Real value 7	Set the work conditions. (Designate with the tool coordinate system.) Immediately after power ON, this setting value is used. (Weight, size X, size Y, Size Z, center of gravity X, center of gravity Y, center of gravity Z) Unit: Kg, mm	RV-S/RH-S series 0.0,0.0,0.0,0.0,0.0,0.0 ,0.0 Other type are secret.
	WRKDAT* * is 1 to 8	Real value 7	Set the work conditions. (Designate with the tool coordinate system.) (Weight, size X, size Y, Size Z, center of gravity X, center of gravity Y, center of gravity Z) Unit: Kg, mm	0.0,0.0,0.0,0.0,0.0,0 .0,0.0
	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)" *This parameter can be used for controller software version G1 or later.	ACCMODE	Integer 1	Sets the initial value and enables/disables the optimum acceleration/deceleration mode. (Invalid/Valid=0/1)	RH-A/RH-S/RV-S series and RV-100TH/150TH/100THL/150THL ....1 Except the above..............0

Parameter	Parameter name	No. of arrays No. of characters	Details explanation	Factory setting
Optimum acceleration/ deceleration adjustment rate *This parameter can be used for controller software version J2 or later.	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-S 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-S 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-S series.	RV-3S/3SJ/3SB/ 3SJB series 100,100,100,100, 100,100,100,100 (%) RV-6S/12S 50,50,50,50, 50,50,50,50(%) RV-6SL/12SL 35,35,35,35, 35,35,35,35(%)
Speed optimization interpolation functional switch *This parameter can be used for controller software version H7 or later.	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) -1: Disable the SPDOPT command 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. If the value is -1, the speed adjustment interpolation function is always disabled even if the SPDOPT instruction is used in a program. Note)This function is supported by limited models of RH-1000G systems, etc.	Only RH-1000G series is 1. Other type are -1.
Impact Detection *This parameter can be used for controller software version J2 or later.	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 occurred errors when an impact is detected. Note) This function is valid only in the RV-S/RH-S series.	RV-S series is 0,0,1 RH-S series is 1,0,1
Detection level *This parameter can be used for controller software version J2 or later.	COLLVL	Integer 8	Set the initial value of the detection level of each axis during program operation. Setting range: 1 to 500, unit: % * If a value exceeding the setting range is specified, the closest value allowed within the range is used instead. Note) This function is valid only in the RV-S/RH-S series.	200,200,200,200, 200,200,200,200
Detection level during jog operation	COLLVLJG	Real value 8	Set the detection level during jog operation. Unit (%) To increase detection sensitivity, reduce the numeric value. If an impact error occurs even when no impact occurs during jog operation, increase the numeric value. Note) This function is valid only in the RV-S/RH-S series.	The standard is 200,200,200,200,200,200,200,200, but it varies with models.

Parameter	Parameter name	No. of arrays No. of characters	Details explanation	Factory setting
Selection of wrist rotation angle (axis A) coordinate system	RCD	Integer 1	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. 0/1/3 = General angle method of the E series/joint angle method/ old general angle method These options are prepared for the compatibility with programs (position data) created for older models (e.g., RV-E3J, RV-E5NJ). To use programs (position data) created for older models, change the parameter value to the same value as the RCD value specified for the given older model. Note that these methods are not mutually compatible; the postures of the hand in the middle of movement and at the registered position may be different for two different values of this parameter, even if the robot is moving toward the same position data. Make sure to set the same method as when the position data was registered in order to execute the program.	2 (general angle method)
Warm-up operation mode setting *This parameter can be used for controller software version J8 or later.	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 *This parameter can be used for controller software version J8 or later.	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.	RV-6S/12S serires :00111000 (J4, J5, J6 axis) RV-3S/3SB :00001110 RV-3SJ/3SJB : 00000110 The type other than the above are 0
Warm-up operation mode control time *This parameter can be used for controller software version J8 or later.	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

Parameter	Parameter name	No. of arrays No. of characters	Details explanation	Factory setting
Warm-up operation override *This parameter can be used for controller software version J8 or later.	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
Functional setting of compliance error *This parameter can be used for controller software version H6 or later.	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)

5.2 Signal parameter

These parameters set the items pertaining to signals

Table 5-2:List Signal parameter

Parameter	Parameter name	No. of arrays No. of characters	Details explanation	Factory setting
Dedicated I/O signal			For the parameters of the dedicated I/O signal, refer to Page 371, "6.3 Dedicated input/output".	
Stop input B contact designation	INB	Integer 1	Change the dedicated input (stop) between the A contact and B contact. (A contact/B contact = 0/1)	0(A contact)
Reads the program number from the numerical input when the start signal is input.	PST	Integer 1	To select a program from the normal external input signal, set the numerical input signal (IODATA) to the program number, establish the number with the program select signal (PRGSEL), and start with the START signal. If this function is enabled, the program select signal becomes unnecessary, and when the START signal turns ON, the program number is read from the numerical input signal (IODATA). (Function invalid/Valid=0/1)	0(Invalid)
CC-Link error release permission. *This parameter can be used for controller software version H7 or later.	E7730	Integer 1	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.	0 (disable error cancellation)
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/*)	
	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	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)

5.3 Operation parameter

These parameters set the items pertaining to the operations of the controller, T/B and so forth.

Table 5-3:List Operation parameter

Parameter	Parameter name	No. of arrays No. of characters	Details explanation	Factory setting
Buzzer ON/ OFF	BZR	Integer 1	Specifies the on/off of the buzzer sound that can be heard when an error occurs in the robot controller. (OFF/ON=0/ÇP)	1(ON)
Program reset operation rights	PRSTENA	Integer 1	Whether or not the operation right is required for program reset operation (Required/Not required = 0/1) If operation rights are abandoned, program can be reset from anywhere. However, this is not possible under the teaching mode for safety reasons.	0(Required)
Program reset when key switch is switched	MDRST	Integer 1	Program paused status is canceled when the key switch is operated. (Invalid/Valid=0/1)	0(Invalid)
Operation panel display mode . This parameter is available for controller software version J1 or later.	OPDISP	Integer 1	Setting of the 5-digit LED display when the key switch is changed over 0: Override display takes effect when the key switch is changed over (initial value). 1: The current display mode is maintained even after the key switch is changed over.	
Program selection rights setting	OPPSL	Integer 1	Specifies the program selection operation rights when the key switch of the operation panel is in AUTO (OP) mode. (External/OP=0/1)	1(OP)
	RMTPSL	Integer 1	Designate the program selection operation rights for the automatic (Ext) mode. (External/OP=0/1)	0(External)
TB override operation rights	OVRDTB	Integer 1	Specifies whether the operation rights are required when changing override from T/B. (Not required/Required = 0/1)	0(Not required)
Speed setting during mode change	OVRDMD	Integer 2	Override is set automatically when the mode is changed. First element..........override value when the mode is automatically changed from teaching mode Second element.....Override value when the mode is changed from Auto to Teaching. Current status is maintained if changed to 0.	0,0
Override change operation rights	OVRDENA	Integer 1	Specifies whether operation rights is required to change override. (Not required/Required = 0/1) If this is set to "Not required," override change can be set from anywhere.	0(Required)
This parameter switches the access target of a program. Refer to Page 345, "5.18 About ROM operation/high-speed RAM operation function". *This parameter can be used for controller software version H7 or later.	ROMDRV	Integer 1	The access target of a program can be switched between RAM and ROM. 0: RAM mode. (Standard mode.) 1: ROM mode. (Special mode.) 2: High-speed RAM mode (DRAM memory is used; can be used in software version J1 or later)	0 (RAM mode.)

Parameter	Parameter name	No. of arrays No. of characters	Details explanation	Factory setting
Copy the information on the RAM to the ROM Refer to Page 345, "5.18 About ROM operation/high-speed RAM operation function". *This parameter can be used for controller software version H7 or later.	BACKUP	Character string 1	Copy the program, the parameter, the common variable, and the error log to the ROM from the RAM. Do not change this parameter.	SRAM->FLROM (unchangeable)
Restore the information on the ROM to the RAM. Refer to Page 345, "5.18 About ROM operation/high-speed RAM operation function". *This parameter can be used for controller software version H7 or later.	RESTORE	Character string 1	Restore the program, the parameter, the common variable, and the error log to the RAM from the ROM. Do not change this parameter.	FLROM->SRAM (unchangeable)
Maintenance forecast * The parameters pertaining to maintenance forecast can be used in the controller's software version J1 or later.	MFENA	Integer 1	This sets whether maintenance forecast is enabled or disabled. 1: Enable 0: Disable Note) This function is limited to the RV-S/RH-S series. This parameter does not take effect on models that do not support the maintenance forecast function.	RV-S/RH-S series ....1 Except the above...........0
Maintenance forecast execution interval	MFINTVL	Integer 2	This sets the interval of collecting data for maintenance forecast. 1st element: Data collection level -- 1 (lowest) to 5 (highest) 2nd element: Forecast check execution interval (unit: hours)	1(lowest),6(hour)
Maintenance forecast announcement method	MFEPRO	Integer 2	This sets the maintenance forecast announcement method. Set 0 in order to stop a warning or signal output. 1st element: 1: Generates a warning, 0: Does not generate a warning 2nd element: 1: Outputs a dedicated signal, 0: Does not output a dedicated signal	0 (Does not generate a warning) , 0 (Does not output a signal)

Parameter	Parameter name	No. of arrays No. of characters	Details explanation	Factory setting
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 (version E1 or later). However, if a personal computer cannot be readied, the accumulated data can be reset by entering this parameter from the teaching pendant instead.	0: Reset all axes. 1 to 8: Reset the specification axis.
	MFBRST	Integer 1	Reset the accumulated data relating to grease in the maintenance forecast function. * When axes generated a warning (numbered in 7530's) that prompts the replacement of belt in the maintenance forecast function and, as a result, the belt was replaced, the data relating to the belt accumulated on the controller must be reset. Generally, a reset operation is performed on the Maintenance Forecast screen in Personal Computer Support software (version E1 or later). 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 *This parameter can be used for controller software version J2 or later.	DJNT	Real value 8	The OP correction data obtained by the Position Restoration Support tool is input. Do not change it with any tool other than the Position Restoration Support tool. It can only be referenced on a dedicated parameter screen in the Personal Computer Support software.	It varies with models.
	MEXDTL	Real value 6	The standard tool correction data obtained by the Position Restoration Support tool is input. Do not change it with any tool other than the Position Restoration Support tool.	(X,Y,Z,A,B,C) = 0.0, 0.0, 0.0, 0.0, 0.0, 0.0
	MEXDTL1	Real value 6	The correction data for tool number 1 obtained by the Position Restoration Support tool is input. Do not change it with any tool other than the Position Restoration Support tool.	(X,Y,Z,A,B,C) = 0.0, 0.0, 0.0, 0.0, 0.0, 0.0
	MEXDTL2	Real value 6	The correction data for tool number 2 obtained by the Position Restoration Support tool is input. Do not change it with any tool other than the Position Restoration Support tool.	(X,Y,Z,A,B,C) = 0.0, 0.0, 0.0, 0.0, 0.0, 0.0
	MEXDTL3	Real value 6	The correction data for tool number 3 obtained by the Position Restoration Support tool is input. Do not change it with any tool other than the Position Restoration Support tool.	(X,Y,Z,A,B,C) = 0.0, 0.0, 0.0, 0.0, 0.0, 0.0
	MEXDTL4	Real value 6	The correction data for tool number obtained by the Position Restoration Support tool is input. Do not change it with any tool other than the Position Restoration Support tool.	(X,Y,Z,A,B,C) = 0.0, 0.0, 0.0, 0.0, 0.0, 0.0
	MEXDBS	Real value 6	The correction data for the base obtained by the Position Restoration Support tool is input. Do not change it with any tool other than the Position Restoration Support tool.	(X,Y,Z,A,B,C) = 0.0, 0.0, 0.0, 0.0, 0.0, 0.0

5.4 Command parameter

This parameter sets the items pertaining to the program execution and robot language.

Table 5-4: List Program Execution Related Parameter

Parameter	Parameter name	No. of arrays No. of characters	Details explanation	Factory setting
No. of multi-tasks	TASKMAX	Integer 1	Designate the number of programs to be executed simultaneously.	8
Slot table (Set during multitask operation.)			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 button on the operation panel or by the start signal. ALWAYS:This is executed immediately after the controller's power is turned on. This program does not affect the status such as startup. To edit a program whose attribute is set to ALWAYS, first cancel the ALWAYS attribute. A program with the ALWAYS attribute is being executed continuously and therefore cannot be edited. Change ALWAYS to START and turn on the controller's power again to stop the constant execution. ERROR:This is executed when an error is generated. This program does not affect the status such as startup. 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(Valid)
Setting that allows the execution of X** instructions and SERVO instruction in an ALWAYS program. Refer to "5.11Automatic execution of program at power up"	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). Furthermore, the XRUN instruction can directly be executed from the T/B's edit screen or support software in the controller's software version J1 or later. Enable/Disable = 1/0	0(Not allowed)
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)

Parameter	Parameter name	No. of arrays No. of characters	Details explanation	Factory setting
Continue function	CTN	Integer 1	For only the program execution slot 1, the state when the power is turned OFF is held, and the operation can be continued from the saved state when the power is turned ON next. The saved data is the program execution environment (override, execution step line, program variables, etc.), and the output signal state. When this function is valid, if the robot is operated when the power is turned OFF, the robot will start in the standby state when the power is turned ON next. To continue operation, turn the servo power ON, and input start. (Function invalid/ Valid=0/1) <Precautions> (1) This function stores the status of execution memory at the time of power off using a part of program memory area. Therefore, this function cannot be used unless there is free space of at least 100K bytes in the program memory area. Please follow the precautions listed below when using this function. The program may become destroyed if the precautions are not followed. 1) Back up all programs in a PC and delete all programs in the controller. 2) Set the "CTN" parameter to 1 and turn the power ON again. 3) Reload only the necessary programs onto the controller. If there is not sufficient empty space at this point, this function cannot be used. If there is not enough space, revert the parameters. (2) Do not change the parameters while the program is still left on, because the program will be destroyed. Although there are 210K bytes of standard memory, available space will be 110K bytes when this function is enabled. Reduction of 1100 points in position count conversion Reduction of 2200 steps in step count conversion With the standard specifications, the memory will drop by 56%. Number of points = 2500 -> 1400 points/Prg. Number of steps = 5000 -> 28000 steps/Prg. (3)As for robots with axes without brakes (J4 and J6 axes of RV-1A/ 2AJ, or J4 axis of RH-5AH), the arm may lower due to gravitational weight or rotate itself when the power is turned off. Thus, extra care is necessary when using this function. (4)Program that can continue using the Continuity function is the one loaded in task slot 1. Programs in task slot 2 or subsequent slots will not continue but will restart in program reset state. (5)The following parameters cannot be changed after this function is enabled. Be sure to change them, if necessary, prior to enabling this function. SLTn, SLOTON, TASKMAX. (6)If parameters in the slot table (SLT*) are changed after enabling this function, the changes are not reflected in the slot table. Disable the continue function once, turn the power supply off and then on, and then change parameters in the slot table.	0(Invalid)
JRC command (Multiple rotation function of axes) *This parameter can be used for controller software version E4 or later.			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	JSet the change amount to increment or decrement with the JRC command in the order of J1, J2, J3 to J8 axes from the head element. The setting is valid only for the user-defined axis, so the J7 and J8 axes will be valid for the robot's additional axis, and a random axis for the mechanism's additional axis. 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
Setting of additional axis	AXUNT	Integer 16	Set the unit system for the additional axis. Angle(degree)/Length(mm) = 0/1	0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0

<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 error setting</td><td>UER1
to UER20</td><td>Integer 1,
Character string 3</td><td>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 ("").
Example)9000,"Time Out","No Signal","Check Button"</td><td>9900,"message","cause","treat"</td></tr>
<tr><td>Unit setting for the rotational element of position data</td><td>PRGMDEG</td><td>Integer 1</td><td>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.</td><td>0(RAD)</td></tr>
<tr><td>Set the delay time of the GC/GO command and the moving command.

*This parameter is valid for using the MOVEMASTER command only.

*This parameter can be used for controller software version H7 or later.</td><td>HANDDLY</td><td>Integer 1</td><td>The delay time of hand open/close in MOVEMASTER command is the time specified by GP command. (Default value is 0.3 sec.)
The delay time of hand open/close can be specified by this parameter.
<table>
<tr><th>Parameter value</th><th>Motorized hand</th><th>Pneumatic hand</th></tr>
<tr><td>-1</td><td>When the status of the hand changes, the delay timer specified by GP command is taken.</td><td>The delay time specified by the GP instruction is stored in this parameter when opening/closing the hand, regardless of whether or not the hand status has changed.</td></tr>
<tr><td>0</td><td>No delay</td><td>No delay</td></tr>
<tr><td>Value
Unit(msec)</td><td colspan="2">When the status of the hand changes, the delay timer specified with this parameter is taken.</td></tr>
</table>
The units of the delay time specified by GP command are 1 / 10 seconds.
The units of the delay time specified with this parameter are 1 / 1000 seconds (=msec).</td><td>-1</td></tr>
<tr><td>Robot language setting</td><td>RLING</td><td>Integer 1</td><td>Select the robot language
1:MELFA-BASIC IV
0:MOVEMASTER COMMAND
Note) This function is only available for certain types of robots, such as RV-1A. Please verify that the type of robot that you are using is listed in the "Command List" of "Separate Volume: Standard Specification" before using this instruction.</td><td>1</td></tr>
<tr><td>Display language Note1)</td><td>LNG</td><td>Character string 1</td><td>Set up the display language.
"JPN":Japanese
"ENG":English

The following language is changed.
(1)The display LCD of teaching pendant.
(2) Personal computer support software.
*alarm message of the robot.
*Parameter explanation list.
(3)Alarm message that read from the robot with external communication. (Standard RS232C, Extended serial I/F, Ethernet I/F)</td><td>The "JPN" is Japanese specification.
The "ENG" is English specification.</td></tr>
<tr><td>Extension of external variable</td><td>PRGGBL</td><td>-</td><td>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.</td><td>0</td></tr>
</table>

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 5-5:List Communication parameter

Parameter	Parameter name	No. of arrays No. of characters	Details explanation	Factory setting
Communication setting Refer to "5.15About the communication setting"			Communication environment is set for RS-232C in the front of the robot controller. However, since this is used by the PC support software, this normally does not need to be changed. When connecting vision sensors, etc., use of optional expansion serial interface is recommended.	
	COMDEV	Character string 8	This configures which lines will be assigned to COM1 and COM2 when using communication lines in the OPEN instruction in MELFA BASIC IV. 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. To use expansion serial interface as COM, if CH1 is installed in option slot 1, "OPT11" should be entered as the value for the parameter. If CH2 (RS232) is installed in slot 1, "OPT12" should be entered as the value for the parameter. If CH2 (RS422) is installed in slot 1, "OPT13" should be entered as the value for the parameter. If CH1 is installed in slot 2, "OPT21" should be entered as the value for the parameter. If CH2 (RS232) is installed in slot 2, "OPT22" should be entered as the value for the parameter. If CH2 (RS422) is installed in slot 2, "OPT23" should be entered as the value for the parameter. For instance, if CH1 is used in slot 1 and is to be assigned to COM2, use '"RS232", "OPT11", , , , ,'. If CH2 is used in slot 2 and is to be assigned to COM3, use '"RS232", , "OPT22", , , , ,'. If Ethernet interface is to be used as COM and is installed in option slot 1 with NETPORT1, enter "OPT11" for the parameter value. NETPORT2, enter "OPT12" for the parameter value. NETPORT3, enter "OPT13" for the parameter value. NETPORT4, enter "OPT14" for the parameter value. NETPORT5, enter "OPT15" for the parameter value. NETPORT6, enter "OPT16" for the parameter value. NETPORT7, enter "OPT17" for the parameter value. NETPORT8, enter "OPT18" for the parameter value. NETPORT9, enter "OPT19" for the parameter value. COM1 is already assigned the standard RS232C in the front of the controller. The following restrictions apply when optional cards are installed. Option slot 1: Additional axis, expansion serial interface, Ethernet Option slot 2: Additional axis, expansion serial interface, CC-Link Option slot 3: Additional axis Additional axis cards are for CR1-only cases. For optional expansion serial interface, refer to "Expansion Serial Interface Instruction Manual". For optional expansion serial interface, refer to "Ethernet Interface Instruction Manual ".	"RS232", , , , , , ,
	CBAU232	Integer 1	Baud rate(9600,19200) For optional expansion serial interface, refer to "Expansion Serial Interface Instruction Manual".	9600
	CPRTY232	Integer 1	Parity bit(0: None, 1: Odd, 2: Even) For optional expansion serial interface, refer to "Expansion Serial Interface Instruction Manual".	2 (Even)
	CSTOP232	Integer 1	Stop bit(1,2) For optional expansion serial interface, refer to "Expansion Serial Interface Instruction Manual".	2 (Stop bit)

Parameter	Parameter name	No. of arrays No. of characters	Details explanation	Factory setting
	CTERM232	Integer 1	End code(0:CR 1:CR+LF) For optional expansion serial interface, refer to "Expansion Serial Interface Instruction Manual".	0 (CR)
	CPRC232	Integer 1	Communication method(protocol) 0: For support software (Non-procedure) If data link is to be performed (OPEN, PRINT and INPUT instructions are executed from the program) under this setting, the external device must attach three characters "PRN" at the beginning when transmitting data. 1: For support software (With procedure) PC side must also be changed. 2: For data link with the robot program (used when communicating with vision sensors, etc.) Be advised that under this setting, connection with the support software cannot be made. Use the optional expansion serial interface. For optional expansion serial interface, refer to "Expansion Serial Interface Instruction Manual".	0

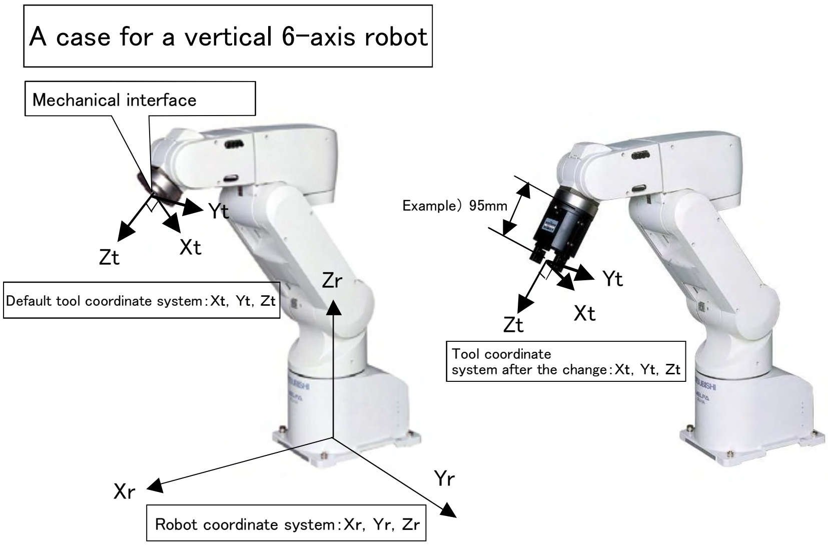
5.6 Standard Tool Coordinates

Tools data must be set if the robot's control point is to be set at the hand tip when the hand is installed on the robot. The setting can be done in the following three manners.

1) Set in the MEXTL parameter.
2) Set in the robot program using the TOOL instruction.
3) Set a tool number in the M_TOOL variable. (Allowed in the controller's software version J1 or later.)
   The values set by the MEXTL1 to 4 parameters are used as tool data.
   Refer to Page 261, " M_TOOL".

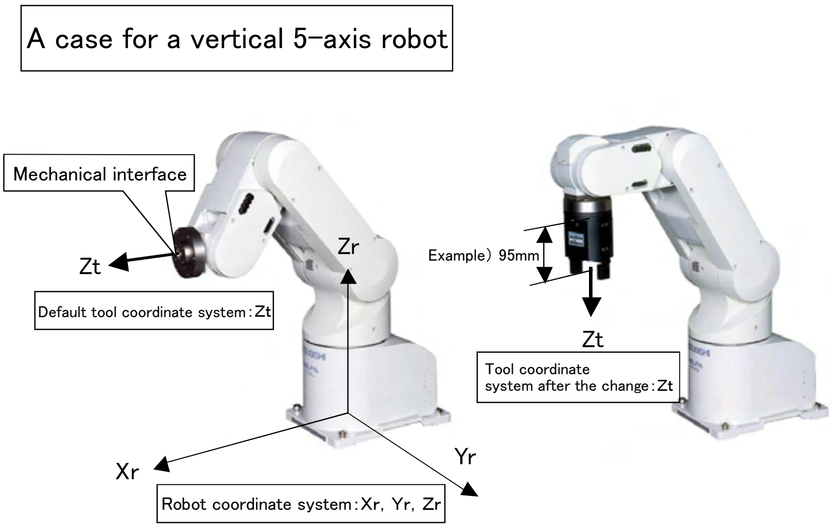
The default value at the factory default setting is set to zero, where the control point is set to the mechanical interface (flange plane).

Structure of tools data : X, Y, Z, A, B, C
X, Y and Z axis:Shift from the mechanical interface in the tool coordinate system
A axis :X-axis rotation in the tool coordinate system
B axis :Y-axis rotation in the tool coordinate system
C axis :Z-axis rotation in the tool coordinate system


A case for a vertical 6-axis robot
Mechanical interface
Yt
Zt
Xt
Default tool coordinate system:Xt, Yt, Zt
Zr
Example) 95mm
Yt
Zt
Xt
Tool coordinate
system after the change:Xt, Yt, Zt
Xr
Yr
Robot coordinate system:Xr, Yr, Zr

<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
10 TOOL (0,0,95,0,0,0)

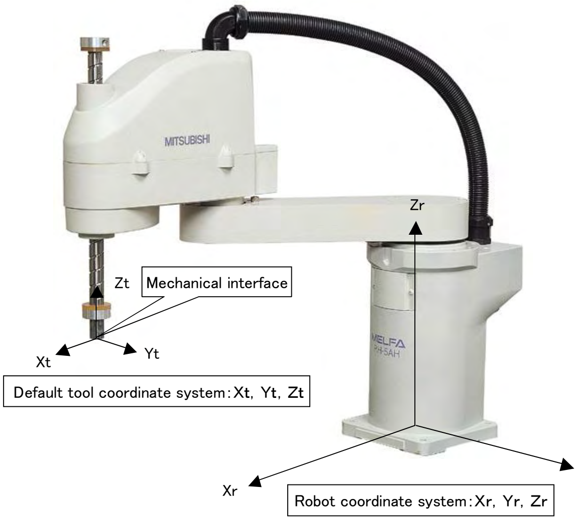
A 6-axis robot can take various postures within the movement range.


A case for a vertical 5-axis robot
Mechanical interface
Zt
Default tool coordinate system:Zt
Zr
Example) 95mm
Zt
Tool coordinate
system after the change:Zt
Xr
Yr
Robot coordinate system:Xr, Yr, Zr

<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
10 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.

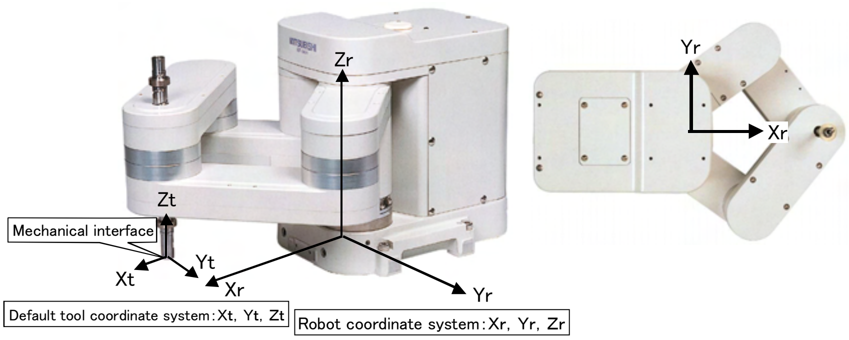
A case for a horizontal 4–axis robot


Zr
Zt
Mechanical interface
Xt
Yt
Default tool coordinate system:Xt, Yt, Zt
Xr
Yr
Robot coordinate system:Xr, Yr, Zr

<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
10 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.

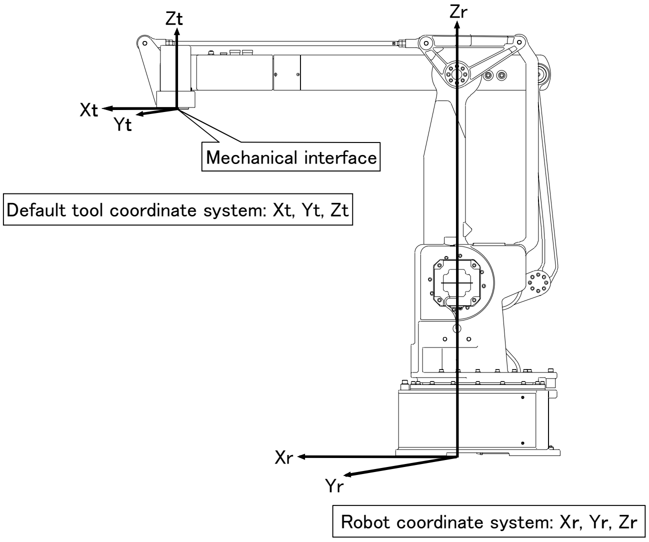
A case for a RP series robot


Zr
Yr
Xr
Zt
Mechanical interface
Yt
Xt
Xr
Yr
Default tool coordinate system:Xt, Yt, Zt
Robot coordinate system:Xr, Yr, Zr

<A case for a RP series robot>
1) Sample parameter setting
Parameter name: MEXTL
Value: 0, 0, -10, 0, 0, 0
2) Sample TOOL instruction setting
10 TOOL (0,0,-10,0,0,0)

RP-series robots use the same coordinate system as the horizontal 4-axis robots. Note that the orientation of the tool coordinate system is set up differently from that of vertical robots.

A case for a palletizing robot


Zt
Zr
Xt
Yt
Mechanical interface
Default tool coordinate system: Xt, Yt, Zt
Xr
Yr
Robot coordinate system: Xr, Yr, Zr

<A case for a palletizing robot>
1) Sample parameter setting
Parameter name: MEXTL
Value: 0, 0, +10, 0, 0, 0
2) Sample TOOL instruction setting
10 TOOL (0,0,+10,0,0,0)

<Other robots>
The following models basically use the same coordinate system as the horizontal 4-axis robots.

Liquid crystal glass transportation robot: RH-1000G***, RH-1500G***, RC-1000GW***,
Palletizing robot : RV-100TH*, RV-150TH

An axis element of the tool conversion data may or may not be valid depending on the robot model. See Table 5-6 to set the appropriate data.

Table 5-6: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
RP-1AH/3AH/5AH	4	O	O	O	X	X	O
RV-1A, RV-2A, RV-4A, RV-3AL , RV-3S, RV-3SB, RV-6S, RV-12S/SL, RV-18S	6	O	O	O	O	O	O
RV-2AJ, RV-3AJ, RV-5AJ, RV-4AJL, RV-3SJ, RV-3SJB	5	@	@	O	@	@	@
RH-5AH/10AH/15AH RH-6SH/12SH/18SH	4	O	O	O	@	@	O
RH-15UHC		O	O	O	X	X	O
RH-1000GH Series	4	O	O	O	X	X	O
RH-1000GJ Series	5	O	O	O	X	X	O
RH-1500GJ Series		O	O	O	X	X	O
RH-1500G Series	6	O	O	O	O	O	O
RC-1000G Series	4	O	O	O	X	X	O[Note2)]
RV-60TH RV-100TH/150TH RV-100THL/150THL	4	O	O	O	X	X	O
RS-30FG, RS-30AG	4	O	O	O	O	X	X[Note2)]
RV-15AJ/15AP	5/10	O	@	O	@	@	@

Note1) O: Valid, @: Invalid. This is meaningless and ignored if set., X: The setting value is fixed to 0.
If a value other than 0 is set, operation may be adversely affected.
Note2) All elements were added (This is not relative calculation)

5.7 About Standard Base Coordinates

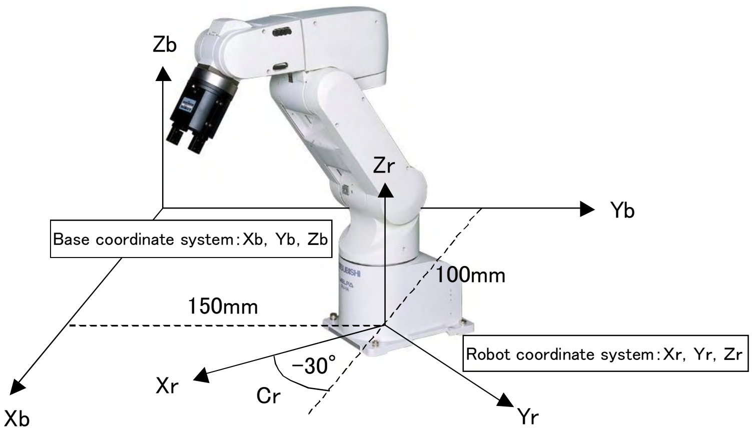
When shifting the robot origin to a position other than the center position of the J1 axis of the robot, the conversion is performed using the base coordinate system. The setting will be done from the following two points. When base data is changed, the coordinates of teaching positions will be values based on the base coordinate system.

1) Set in the MEXTL parameter.
2) Set in the robot program using the BASE instruction.

The factory default setting value is set to zero at the base coordinate system position, which is identical to the robot origin.

Structure of base coordinate system data: X, Y, Z, A, B, and C

X, Y and Z axis : The position of robot coordinate system from the base coordinate system origin
A axis : X-axis rotation in the base coordinate system
B axis : Y-axis rotation in the base coordinate system
C axis : Z-axis rotation in the base coordinate system


Zb
Zr
Yb
Base coordinate system: Xb, Yb, Zb
100mm
150mm
Robot coordinate system: Xr, Yr, Zr
-30°
Xr
Cr
Xb
Yr

(Example)
1) Sample parameter setting
Parameter name: MEXBS
ValueÅF100,150,0,0,0,-30
2) Sample BASE instruction setting
10 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.

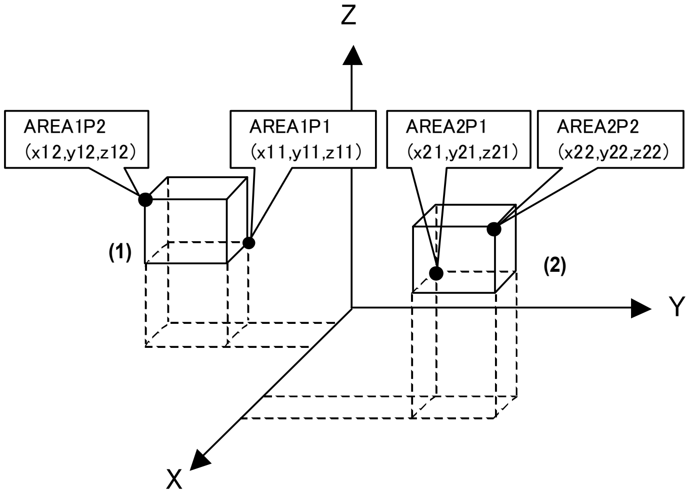
An axis element of the base conversion data may or may not be valid depending on the robot model. See Table 5-7 to set the appropriate data.

Table 5-7: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
RP-1AH/3AH/5AH	4	O	O	O	X	X	O
RV-1A, RV-2A, RV-4A, RV-3AL , RV-3S, RV-3SB, RV-6S, RV-12S/SL, RV-18S	6	O	O	O	O	O	O
RV-2AJ, RV-3AJ, RV-5AJ, RV-4AJL, RV-3SJ, RV-3SJB	5	O	O	O	@	@	@
RH-5AH/10AH/15AH RH-6SH/12SH/18SH	4	O	O	O	@	@	O
RH-15UHC		O	O	O	X	X	O
RH-1000GH Series	4	O	O	O	X	X	O
RH-1000GJ Series	5	O	O	O	X	X	O
RH-1500GJ Series		O	O	O	X	X	O
RH-1500G Series	6	O	O	O	O	O	O
RC-1000G Series	4	O	O	O	X	X	O Note2)
RV-60TH RV-100TH/150TH RV-100THL/150THL	4	O	O	O	X	X	O
RS-30AG, RS-30AG	4	O	O	O	O	X	X Note2)
RV-15AJ/15AP	5/10	@	@	@	@	@	O Note3)

Note1) O: Valid, @: Invalid. This is meaningless and ignored if set., X: The setting value is fixed to 0.
Note2) All elements were added (This is not relative calculation)
Note3) The setting is made in units of 45 degrees

5.8 About user-defined area



When operation is performed together with peripheral devices, work area may have to be shared. Under such circumstances, one device must let the other know when it is within the shared area. For this purpose, a robot can be configured to output a signal while it is in a certain area by setting parameters.

For instance, in the diagram to the left, the following parameter setting will output the signal 10 when operating in area (1) and output the signal 11 when operating in area (2).

Similar confirmation is possible using the M_UAR variable if checking within the program. Refer to Page 262, "M_UAR".

Parameter name	Meaning of the value	Value
AREA1P1	Position data for the first point: X,Y,Z,A,B,C,L1,L2	x11, y11, z11, -360, -360, -360,0,0
AREA1P2	Position data for the second point: X,Y,Z,A,B,C,L1,L2	x12, y12, z12, 360, 360, 360,0,0
AREA1ME	Target mechanism number: Usually 1	1
AREA1AT	Invalid/Output signal/Error: 0/1/2	1
AREA2P1	Position data for the first point: X,Y,Z,A,B,C,L1,L2	x21, y21, z21, -360, -360, -360,0,0
AREA2P2	Position data for the second point: X,Y,Z,A,B,C,L1,L2	x22, y22, z22, 360, 360, 360,0,0
AREA2ME	Target mechanism number: Usually 1	1
AREA2AT	Invalid/Output signal/Error: 0/1/2	1
USRAREA	Output signal: starting number, end number	10, 11 (Information regarding whether or not the robot is in AREA1 is output to the signal 10, and whether or not the robot is in AREA2 is output to the signal 11.) Set 10,10 in the case of one area.

*1 Enter the coordinates (x, y, and z) for x11 to z22.

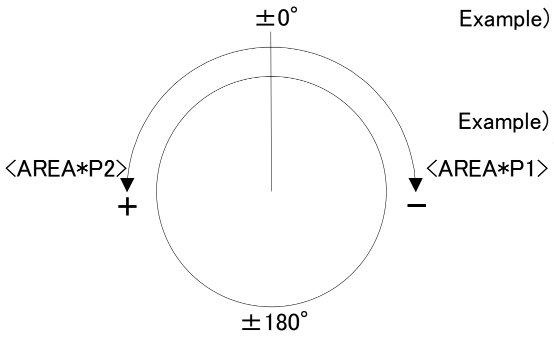
*2 In the setting sample above, since the posture data (A, B, and C) are ignored, a signal will be output regardless of the posture. To set the posture data (A, B, and C), set the values in the AREA*P1 to AREA*P2 direction. (Example: In the case of -10 -> +30, evaluation as in-area occurs in the range from -10 to + 30, but in the case of +30 -> -10 evaluation as in-area occurs in the range from +30 to -10.)

*3 For a non-existent axis of the posture data (for instance the A- and B-axes of horizontal multi-joint type robots), make sure that AREA*P1 is set to -360 and AREA*P2 is set to +360.

*4 AREA*ME specifies to which mechanism will the area checking configuration apply. Under the standard configuration (one unit is connected), this is set to 1.

*5 AREA*AT specifies the type of area checking. The meaning of the value is shown below.
0 = Does not check, 1 = Outputs a signal, 2 = Generates an error.
If 2, posture data, L1 and L2 are ignored.

*6 If additional axes are used, set an area for axes L1 and L2 respectively.

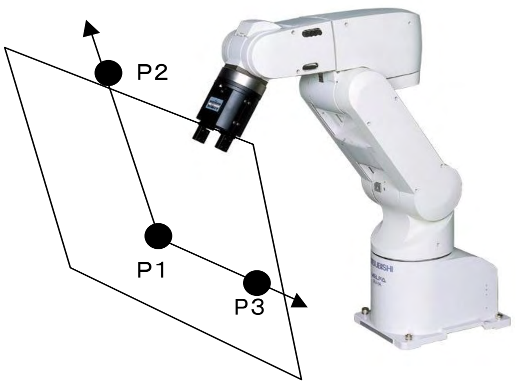


Example) If the value passes through 0 from −100 to +100,
the following setting is necessity.
Sets the negative value to ABC of <AREA*P1>.
Sets the positive value to ABC of <AREA*P2>.

Example) If the value passes through 180 from −100 to +100,
the following setting is necessity.
Sets the positive value to ABC of <AREA*P1>.
Sets the negative value to ABC of <AREA*P2>.

5.9 Free plane limit

Defines any plane in the robot coordinate system, determines the front or back of the plane, and generates a free plane limit error.



As can be seen in the diagram to the left, any plane can be defined by three points (P1, P2, and P3), after which an evaluation of which side of the plane it is in (the side that includes the robot origin or the other side) can be performed.
This function can be used to prevent collision with the floor or interference with peripheral devices.
Maximum of eight planes can be monitored.
There is no limit to the plane.

Parameter and value	Explanation
SFCnP(n=1 to 8)	Specifies the 3 points that define the plane. P1 coordinates X1, Y1, and Z1: The origin of the plane P2 coordinates X2,Y2,Z2: A position on the X axis of the plane P3 coordinates X3,Y3,Z3: A position in the positive Y direction of the X-Y plane in the plane
SFCnME(n=1 to 3)	Specifies the mechanism number to which the free plane limit applies. Usually, set up 1. In the case of multiple mechanisms, the mechanism numbers are specified.
SFCnAT(n=1 to 8)	Designate the valid/Invalid of the set free plane limit. 0:Invalid 1: Valid (The operable area is the robot coordinate origin side.) -1: Valid (The operable area is the side where the robot coordinate origin does not exist. The controller software version J1 or later.)

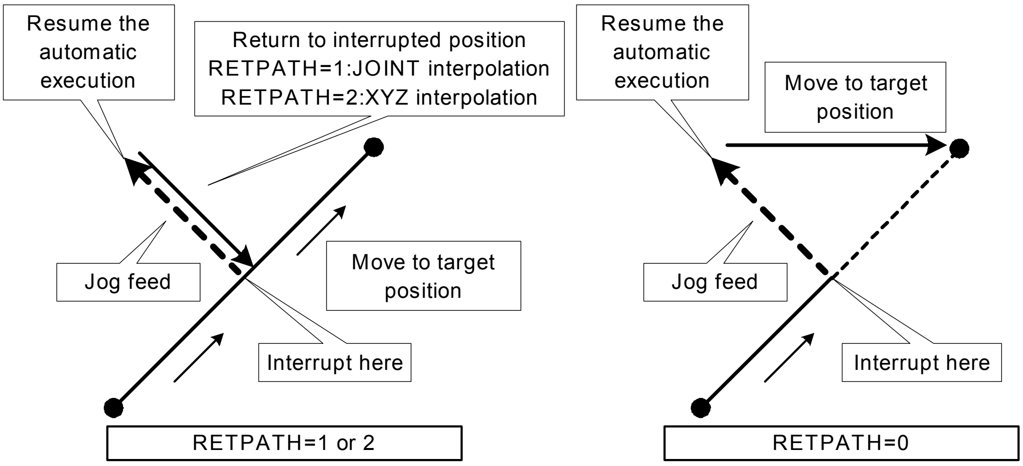
After setting the parameters above, turn the controller's power ON again. This will allow the generation of free plane limit error when it crosses the plane.

5.10 Automatic return setting after jog feed at pause

This specifies the path behavior that takes place when the robot is paused during automatic operation or during step feed operation, moved to a different position using a jog feed with T/B, and the automatic operation is resumed or the step feed operation is executed again. See the following diagram.

Parameter and value	Description of the operation
RETPATH=1 (Default)	1) Returns to the original position where the pause took place using joint interpolation. 2) Resumes from the line that was paused.
RETPATH=0	Resumes from the line that was paused from the position resulting after the jog operation. Therefore, movement will take place using the interpolation method of the instruction under execution from the current position to the next target position.
RETPATH=2 *1	1) Return by XYZ interpolation to the interrupted position. 2) Resume the interrupted line.

*1: The "RETPATH=2" is available for controller software version H4 or later.


Resume the automatic execution
Return to interrupted position
RETPATH=1:JOINT interpolation
RETPATH=2:XYZ interpolation
Jog feed
Move to target position
Interrupt here
RETPATH=1 or 2
Resume the automatic execution
Move to target position
Jog feed
Interrupt here
RETPATH=0

The table below lists the values that can be specified for each interpolation instruction and controller software version.

Table 5-8:Interpolation instruction and RETPATH setting value

Interpolation command	Is before than H4 edition Note1)	Is H4 edition or later
MOV MVS	0 1	0 1 2
MVC MVR MVR2 MVR3	0 1	0 (The same operation as 1) Note2) 1 2
MVA	0 (The same operation as 1) Note3) 1	0 (The same operation as 1) Note3) 1 2

Note1) If "RETPATH=2" is set for versions earlier than H4, the same operation as when "RETPATH=1" is set is obtained. (Returns to interrupted position by JOINT interpolation)

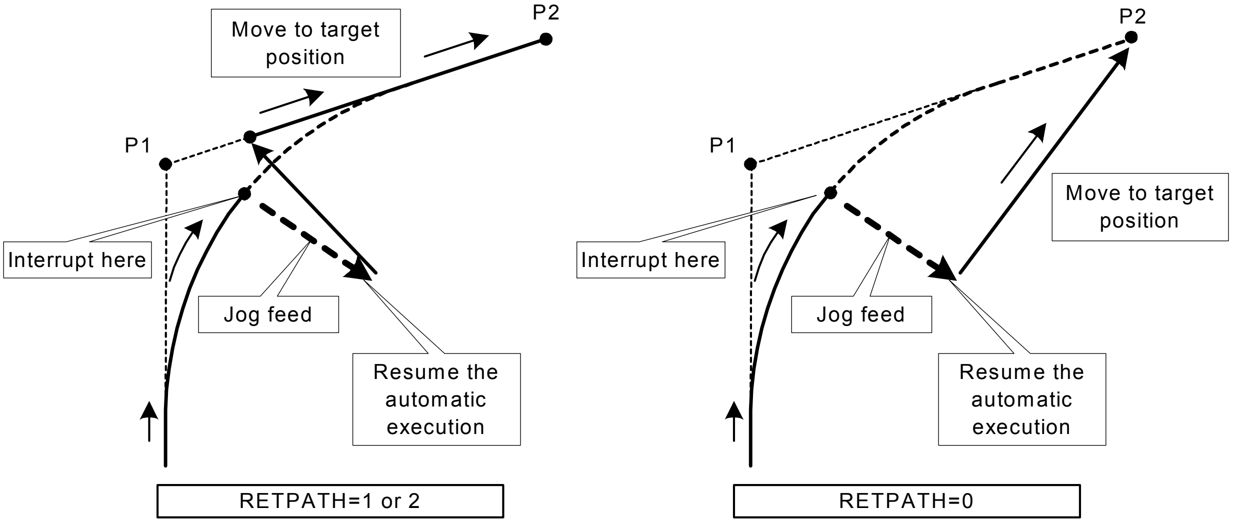
Note2) Note that the operation when RETPATH=0 is set depends on the software version at circular interpolation (MVC, MVR, MVR2, MVR3). For versions later than H4, the same operation as when RETPATH=1 is set is obtained even if RETPATH=0 is set. (Returns to interrupted position by JOINT interpolation)

Note3) In the MVA command, even if set up with RETPATH=0, the operation is same as RETPATH=1. (Returns to interrupted position by JOINT interpolation)

[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->7 In the ALWAYS program, it is possible to execute multitask-related instructions such as XRUN and XLOAD, and also the SERVO instruction.

(1) First, create an ALWAYS program and an operating program.

<Program #2, ALWAYS program>

```
100 ' Auto Start Sample Program
110 '
120 ' Execute Program #1 if the key switch is AUTO (Ext.).
130 ' Stop the program and return the execution line to the beginning of the program if the key switch is not AUTO
(Ext.).
140 '
150 IF M_MODE<>3 AND (M_RUN(1)=1 OR  M_WAI(1)=1) THEN GOSUB *MTSTOP
160 IF M_MODE=3  AND  M_RUN(1)=0 AND M_WAI(1)=0  THEN GOSUB *MTSTART
165 IF M_MODE=2 THEN HLT ' for DEBUG
170 END
180 '
190 *MTSTART
200 XRUN 1,"1"
210 RETURN
220 '
230 *MTSTOP
240 XSTP 1
250 XRST 1
260 RETURN
```

< Program #1, operating program > (this can be any program)

```
100 'Main Program (Position data is for RV-2AJ.)
110 SERVO ON
120 M_OUT(8)=0
130 MOV P1
140 M_OUT(8)=1
150 MOV P2
160 END
P1=(+300.00,-200.00,+200.00,+0.00,+180.00,+0.00)(6,0)
P2=(+300.00,+200.00,+200.00,+0.00,+180.00,+0.00)(6,0)
```

(2) Set the parameter.

Parameter and value	Description of the operation
SLT2	SLT2=2,ALWAYS,REP ’Execute program #2 in ALWAYS mode.
ALWENA	0->7 In the ALWAYS program, it is possible to execute multitask-related instructions such as XRUN and XLOAD, and also the SERVO instruction.

After the setting is complete, turn the controller's power OFF.

(3) Turn the power ON.

In the sample above, after the controller's power is turned on, when the key switch is turned to AUTO (Ext.), program #1 is executed and the robot starts its operation.

5.12 About the hand type

The factory default setting assumes that the double-solenoid type hand will be used. If the single-solenoid type is used or if a general-purpose signal is to be used to control the robot, the HANDTYPE parameter must be set as described below.

Table 5-9:Factory default parameter settings

Parameter name	Value
HANDTYPE	D900,D902,D904,D906, , , ,

Note) The default settings are D224, D226, D192 and D194 in the case of the RC-1300G series.

From the left, the values correspond to hand #1, #2, and so on. The default value is shown below.
Hand 1 = accesses signals #900 and #901
Hand 2 = accesses signals #902 and #903
Hand 3 = accesses signals #904 and #905
Hand 4 = accesses signals #906 and #907
The hand numbers 1 through 4 (or 8) will be used as the argument in the hand open/close instructions (HOPEN or HCLOSE).

<Setting method>
When a double-solenoid type is used, 'D' must be added in front of the signal number to specify the number.
In the case of double-solenoid type, hand number will be from 1 to 4.
When a single-solenoid type is used, 'S' must be added in front of the signal number to specify the number.
In the case of single-solenoid type, hand number will be from 1 to 8.

<Example>

1) To assign two hands of the double-solenoid type from the general-purpose signal #10
   HANDTYPE=D10,D12, , , , ,
2) To assign three hands of the double-solenoid type from the general-purpose signal #10
   HANDTYPE=S10,S,11,S12, , , , ,
3) To assign hand 1 to the general-purpose signal #10 as the single-solenoid type while assigning hand 2 to the general-purpose signal #12 as the single-solenoid type
   HANDTYPE=D10,S12, , , , ,

5.13 About default hand status

The factory default setting is shown below.

Hand type	Status	Status of output signal number		
		Mechanism #1	Mechanism #2	Mechanism #3
When pneumatic hand interface is installed (double-solenoid is assumed)	Hand 1 = Open Hand 2 =Open Hand 3 =Open Hand 4 =Open	900=1 901=0 902=1 903=0 904=1 905=0 906=1 907=0	910=1 911=0 912=1 913=0 914=1 915=0 916=1 917=0	920=1 921=0 922=1 923=0 924=1 925=0 926=1 927=0
When electric-powered hand interface is installed	Hand open	M_OUT (9*0) through M_OUT (9*7) are used by the system and therefore unavailable to the user. If used, normal opening and closing of the hand will not be possible.		
I/F before interface installation	-	M_OUT (9*0) through M_OUT (9*7) do not function.		

A single controller can control multiple robots. If pneumatic hand interface is to be used for each robot, the hand output signal number is assigned in the following manner.
Mechanism #1 = #900 to #907 (This will be the case for standard configuration with one unit connected.)
Mechanism #2 = #910 to #917
Mechanism #3 = #920 to #927
When electric-powered hand interface is used, the system will use the 900's using special controls. The users should not access the 900's directly but instead use the hand control instructions or the hand operation from T/B only. If you access the 900's, normal opening and closing of the hand will not be possible.

The default parameters are set as shown below so that all hands start as "Open" immediately after power up.

Parameter name	Signal number	Value
HANDINIT	900, 901, 902, 903, 904, 905, 906, 907	1, 0, 1, 0, 1, 0, 1, 0

The above describes the situation for standard configuration (one unit is connected). When multiple mechanisms are used, specify the mechanism number to set the HANDINIT parameter.

If for instance hand 1 alone needs to be closed when the power is turned ON, the following should be set. Similarly, in the case of electric-powered hand (hand number is fixed to 1), the hand will be closed when the power is turned on if the following configuration is applied.

Parameter name	Signal number	Value
HANDINIT	900, 901, 902, 903, 904, 905, 906, 907	0, 1, 1, 0, 1, 0, 1, 0, 1

[Caution1] If you set the initial hand status to "Open," note that the workpiece may be dropped when the power is turned ON.

[Caution2] This parameter specifies the initial value when turning ON the power to the dedicated hand signals (900’s) at the robot's tip.
To set the initial status at power ON when controlling the hand using general-purpose I/Os (other than 900’s) or CC-Link (6000’s) (specifying a signal other than one in 900Åfs by the HANDTYPE parameter), do not use this HANDINIT parameter, but use the ORST* parameter.
The value set by the ORST* parameter becomes the initial value of signals at power ON.

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
	:	:
	:	:
	:	:
	ORST4984	00000000, 00000000, 00000000, 00000000
	ORST5016	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

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 the general-purpose signals 10, 11 and 12 of the standard I/O and 32, 33 and 40 of the expansion I/O, the robot should be set to the configuration shown below.

Parameter name	Value
ORST0	00000000, 00111000, 00000000, 00000000
ORST32	11000000, 10000000, 00000000, 00000000

In addition to the above, to make 20, 21 and 22 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
ORST0	00000000, 00111000, 0000***0, 00000000
ORST32	11000000, 10000000, 00000000, 00000000

In the case above, general-purpose signals 20, 21 and 22 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.

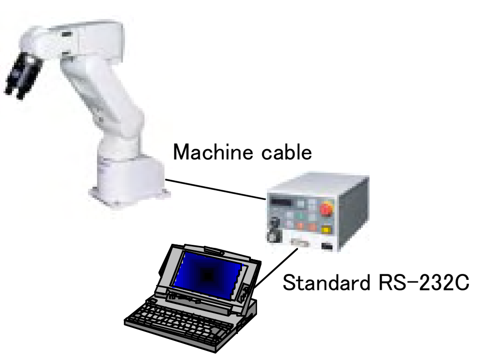
5.15 About the communication setting

(1) Overview

The controller for the CRn series can support 1 standard RS-232C, 2 optional expansion serial cards (2 ports per card), totaling 5 ports. The optional expansion serial interface card has two ports, where one is RS-232C only and the other can select either RS-232C or RS-422. Up to two expansion serial cards can be used.

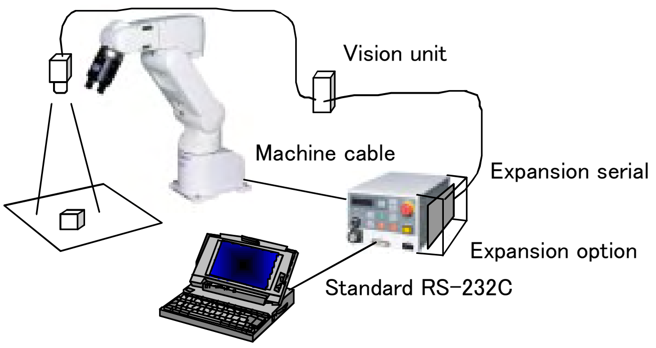
Standard RS-232C port normally connects to a PC for robot program transferring and debugging done with the PC support software. Optional cards can link with vision sensor and external devices for data communication. Communication is performed using the communication instructions (OPEN, CLOSE, PRINT, INPUT, etc.) in the robot program. This is referred to as data link. The controller cannot be controlled from external devices such as a PC (i.e., automatic execution or status monitoring). If this is necessary, contact the dealer or branch from where the robot has been purchased for further consultation.

Example of standard RS-232C port usage



Use of PC with the support software

Example of expansion RS-232C port usage



Use of PC with the support software

Caution)The example above is for robot controller CR1. Controllers other than the CR1 do not require expansion option box.

(2) Performing data link with the outside using RS-232C

Although the standard RS-232C can be used as the data link port, it is recommended that the standard RS-232C be reserved for robot program correction and monitoring at the time of start up and an optional expansion serial interface be provided as a dedicated data link port. The following sample program shows how to use an expansion serial interface.

<Parameter setting> The communication setting of the expansion serial interface is described below.

Parameter name	Default value	After change	Meaning
COMDEV	RS232, , , , , , ,	RS232,OPT11, , , , , ,	When using CH1 in option slot 1
CBAUE11	9600	<-	Baud rate = 9600bps
CPRTYE11	2	<-	Parity = Even
CSTOPE11	2	<-	Stop bit = 2
CTERME11	0(CR)	1(CRLF)	Termination code = CRLF (carriage return and line feed)
CPRCE11	0	2	Switch to data link This setting is for communicating with the outside using OPEN, PRINT, INPUT and CLOSE instructions in the robot program.
Communication example		OPEN "COM2:" AS #1 PRINT #1, "ABC" M1 = 10 PRINT #1, M1 INPUT #1, C1$ INPUT #1, M1	Robot External "ABC"CRLF-> "10"CRLF-> <-"RUN"CRLF <-"20"CRLF

For expansion serial interface, refer to "Expansion Serial Interface Instruction Manual."

<Sample robot program>
The following is a sample program which moves to the camera position, obtains correction data from RS-232C, moves to the position above the corrected position, moves to the target position, closes hand, and breaks away.

Command	Comment
10 OPEN "COM2:" AS #1	'Opens RS232C communication.
20 MOV P1,-50	'Moves to camera position.
30 M_OUT(10)=1 DLY 0.5	'Outputs camera start signal (start up the sensor).
40 PX=P1	'Obtains the camera origin.
50 INPUT #1,MX,MY,MC	'Inputs correction data.
60 PX.X=PX.X+MX	'Applies correction data in the X direction.
70 PX.Y=PX.Y+MY	'Applies correction data in the Y direction.
80 PX.C=PX.C+RAD(MC)	'Applies correction data in the C axis.
90 MOV PX,-50	'Moves to a position above the workpiece.
100 OVRD 30	'Slows down.
110 MVS PX	'Moves to the target position.
120 DLY 0.3	'Waits for positioning.
130 HCLOSE	'Closes hand.
140 DLY 0.3	'Waits for hand closing to complete.
150 MVS ,-50	'Breaks away.
: :	:

In this example, the camera origin position must be stored in robot coordinates through teaching in advance. The Z-axis also must allow the grasping of the workpiece.

<Communication data> (Data string sent by the external devices such as the sensor.)
Data sent to the robot is sent in the order of X, Y, and then C. CR (carriage return = 0d in hexadecimal) is sent as the termination code. The units are mm, mm, and deg (degree), respectively. In the following example, data sent is X = 12.34 mm, Y = 0.39 mm, and C = 56.78degree.

```
"12.34,0.39,56.78(CR)"
```

(3) Notes on using both the PC support software and the data link software on standard port
If the standard port on the front face of the controller is to be used to connect to both the PC support software at power up and to an external device (such as a PC that is running specialized software) via the data link during operation, the data transmitted to the controller from external devices must have "PRN" at the beginning of each data string.

```
Example) "PRN12.34,0.39,56.78(CR)"
```

These letters are required to distinguish between PC support software protocol and data link communication. After "PRN", data should follow immediately; do not insert a space.

Therefore, if the communication protocol from the external device cannot be changed, then the device cannot be used under the default standard port setting. Although it is possible to set a parameter (CPRC232) so that the "PRN" is not required, changing of this parameter disallows the use of PC support software. Therefore, if "PRN" cannot be added to the transmission data from the external device, use the optional expansion serial interface.

Vision sensor AS50VS from Mitsubishi comes with the standard functions to support this "PRN", which allows them to be supported at the standard port. However, with only the standard port, PC support software and vision sensor cannot be used at the same time since there is only one port.

(4) How to ensure stable communication between the personal computer support software and the robot controller.

When communicating with the robot controller (hereinafter referred to as the R/C) using the Personal Computer Support Software (hereinafter referred to as the Software), depending on the personal computer model and settings the communication may become unstable in a batch backup or program upload/download

where large amounts of data are transmitted. To ensure stable communication with the R/C, change the communication settings (communication protocol) of the R/C and the Software as shown below. If the communication settings of the R/C and the Software do not match, normal communication cannot be established. Be sure to change the settings on both the R/C and Software sides.

<The Robot Controller>

As for the R/C communication settings, change the following parameters:

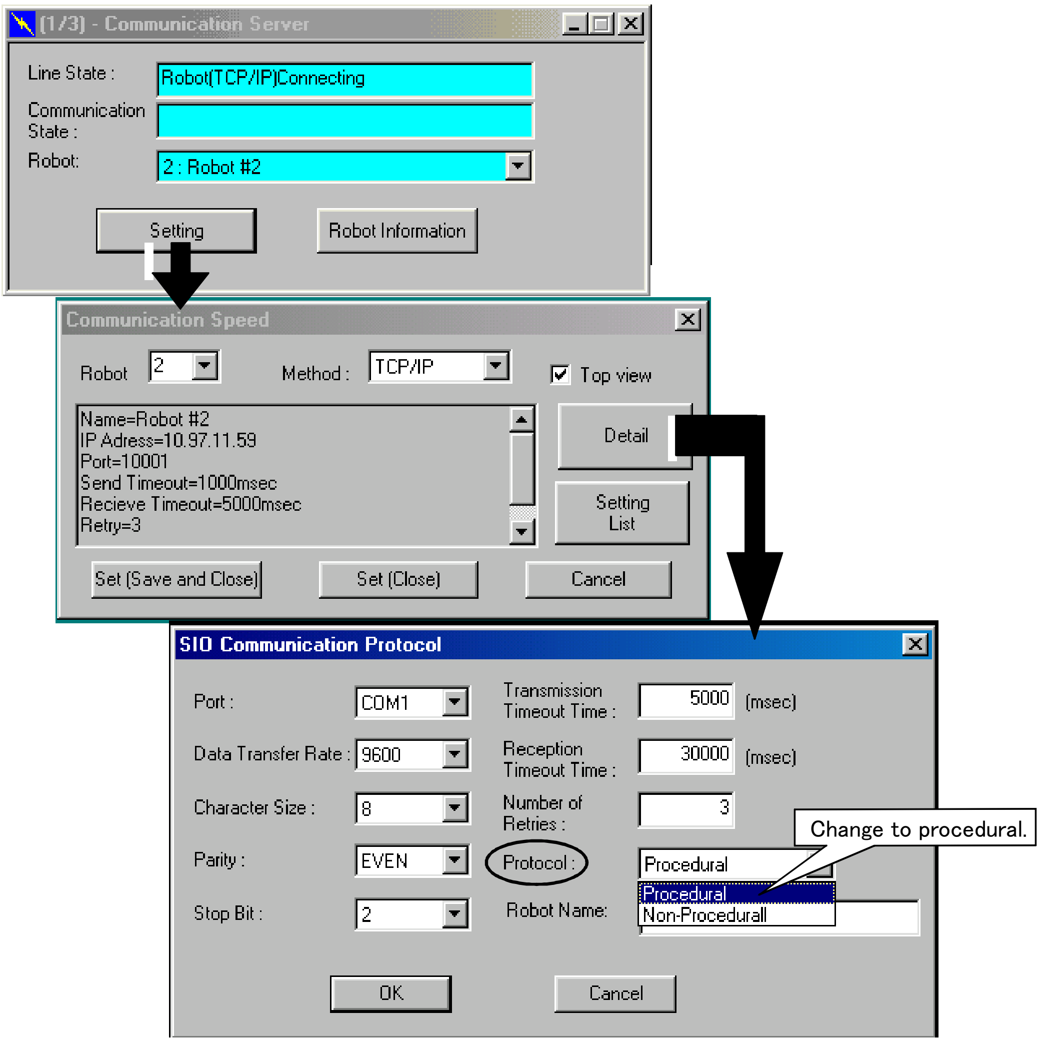
Parameter	Default	Change to
Communication protocol (CPRC232)	0 (Non-Procedural)	1 (Procedural)

To communicate with the personal computer via an extended RS-232C port by using an extension serial interface board, change the parameters of the extended RC-232C port.

<The Personal Computer Support Software>

Change the communication settings of the personal computer support software through the "communication server."

Parameter		Default
Protocol	Non-Procedural	Procedural



(1/3) - Communication Server

Line State : Robot(TCP/IP)Connecting

Communication State :

Robot: 2 : Robot #2

Communication Speed

Robot 2 Method : TCP/IP ☑ Top view

Name=Robot #2
IP Adress=10.97.11.59
Port=10001
Send Timeout=1000msec
Recieve Timeout=5000msec
Retry=3

SIO Communication Protocol

Port : COM1

Data Transfer Rate : 9600

Character Size : 8

Parity : EVEN

Stop Bit : 2

Transmission Timeout Time : 5000 (msec)

Reception Timeout Time : 30000 (msec)

Number of Retries : 3

Protocol : Procedural

Robot Name:

Change to procedural.

When communicating with the R/C using your custom software, reset the R/C protocol setting to "Non-Procedural" in advance.

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-S/RH-S 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. (It is released only with the RV-S/RH-S series.)
	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. (It is released only with the RV-S/RH-S series.)
	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

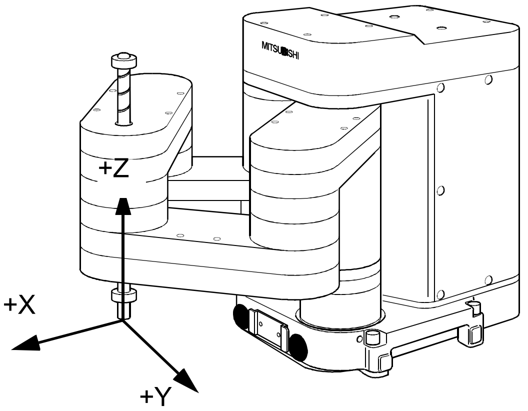
"1" = Set to grasping

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.

*RP-A series



Definitions of Coordinate Axes

In the coordinate system with the tip of the J4 axis as the origin:

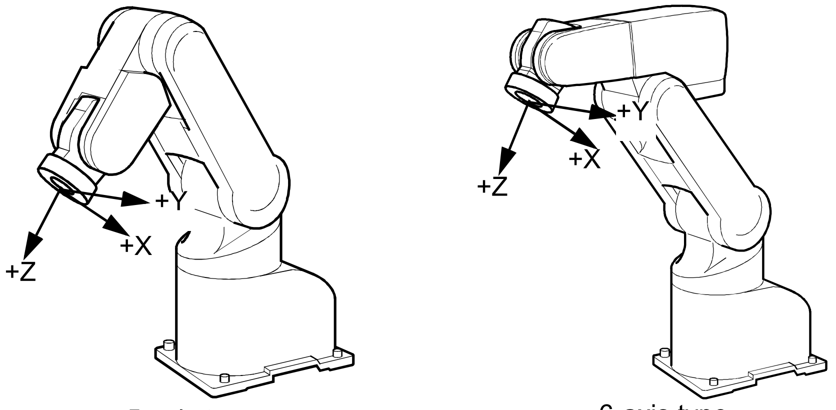
Z axis: The upward direction is positive.

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 and Y elements of the center of gravity and the X and Y elements of the size must be set.

*RV-A, RV-S series



*Example of RV-1A/2AJ

5-axis type

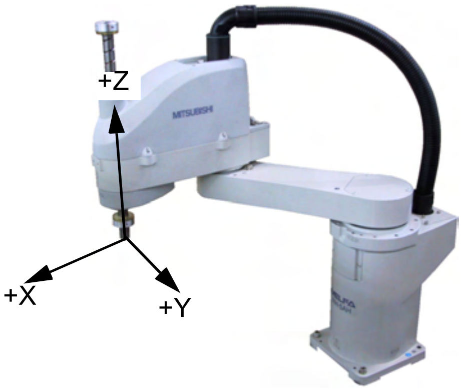
6-axis type

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.

*RH-A, RH-S series



Definitions of Coordinate Axes

In the coordinate system with the tip of the J4 axis as the origin:

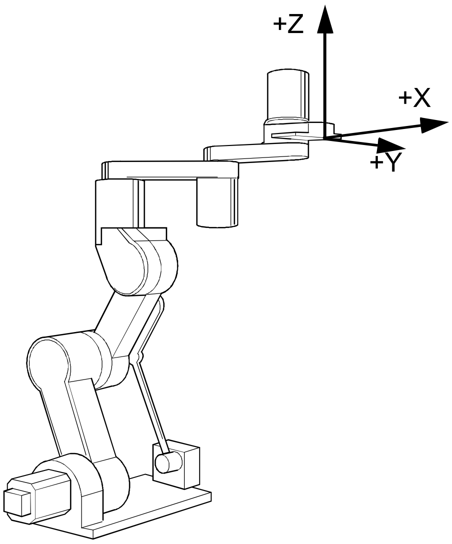
Z axis: The upward direction is positive.

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.

*RH-1000G***



Definitions of Coordinate Axes

In the coordinate system with the center of the J5 axis as the origin for the 5-axis type and with the center of the J4 axis as the origin for the 4-axis type:

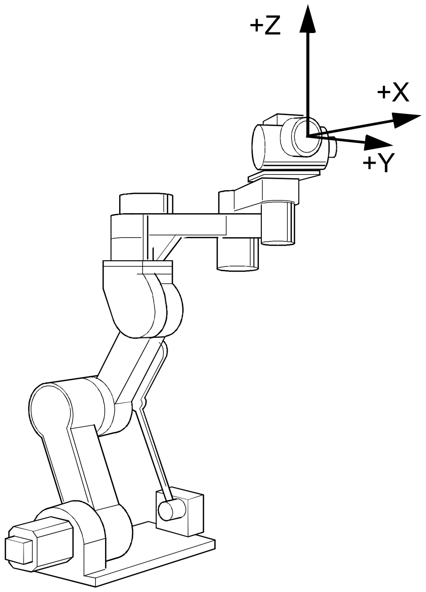
Z axis: The upward direction is positive.

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.

*RH-1500G***



Definitions of Coordinate Axes

In the coordinate system with the center of the J6 axis as the origin for the 6-axis type and with the center of the J5 axis as the origin for the 5-axis type:

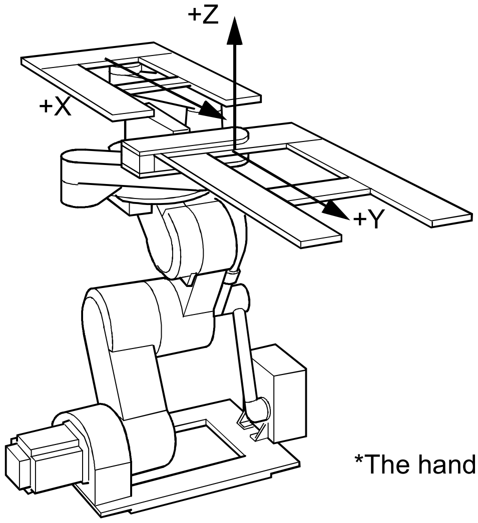
Z axis: The upward direction is positive.

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.

*RC-1000GHWDC-SA/1000GHWLC-SA



Definitions of Coordinate Axes

In the coordinate system with the center of the flange as the origin:

Z axis: The upward direction is positive.

X axis: The front of the left hand is the plus direction

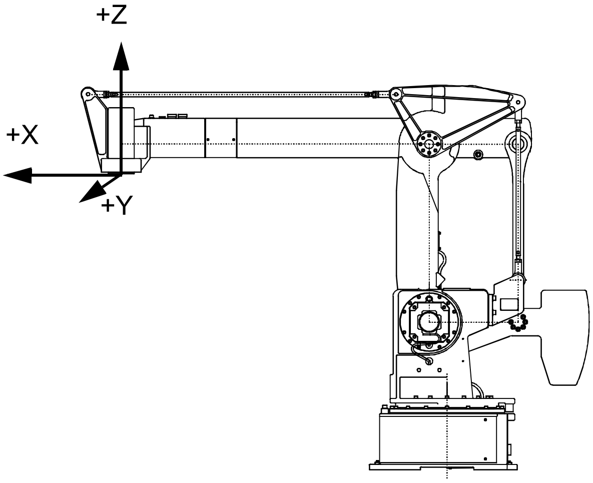
Y axis: The front of the right hand is the plus direction

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.

*The hand of the figure is an example.

*RV-100TH/100THL/150TH/150THL/60TH



Definitions of Coordinate Axes

In the coordinate system with the center 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 (In the figure, the front direction)

Axes that must be set:

Only the X and Y elements 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

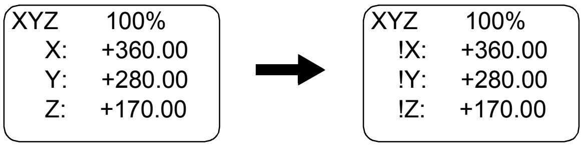
When a robot having a singular point is being operated using a T/B, a singular point adjacent alarm is generated to warn the operators of the robot if the control point of the robot approaches a singular point.
Even if an alarm is generated, the robot continues to operate as long as it can perform operation 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.
Note that this function is supported by the controller with the software version of G9 or later.

(1) Operations that generate an alarm

An alarm is generated if the control point of a robot approaches a singular point while a robot is performing any of the following operations using the T/B.

1) Jog operation (other than in joint jog mode)
2) Step feed and step return operations
3) MS position moving operation
4) Direct execution operation

If the robot approaches a singular point by any of the operations listed above, the buzzer of the controller keeps buzzing (continuous sound). However, the STATUS. NUMBER display on the operation panel does not change.
Also, in the case of "[1] Jog operation (other than joint jog mode)" above, a warning is displayed on the T/B screen together with the sound of the buzzer.



```
XYZ      100%           XYZ      100%
   X:   +360.00            !X:   +360.00
   Y:   +280.00            !Y:   +280.00
   Z:   +170.00            !Z:   +170.00
```

Fig.5-1:Warning Display During Jog Operation

(2) Operations that do not generate an alarm

No alarm is generated when a robot is performing any of the operations listed below even if the control point of the robot approaches a singular point.

1) Additional axis jog operation initiated in joint jog mode using the T/B
2) When the joint interpolation instruction is executed even by an operation from the T/B
   (Execution of the MOV instruction, MO position moving operation)
3) When the program is running automatically
4) Jog operation using dedicated input signals (such as JOGENA and JOGM)
5) When the robot is being operated using external force by releasing the brake
6) When the robot is stationary

5.18 About ROM operation/high-speed RAM operation function

Because the ROM operation /high-speed RAM function has some restrictions on program operation and data retention, please use it after thoroughly understanding the specifications.

(1) Overview

Initially, the robot programs are saved in the RAM (SRAM) that is backed up in the battery.

By saving the robot programs in Flash ROM (FLROM), a loss of files due to the depletion of the backup battery, damage to the programs due to unexpected power shutoff (including momentary power failure) during a file access operation, or changes or deletion of the programs and position data due to an erroneous operation.

By changing the parameter values, the access target of the programs can be switched between ROM and RAM. Once the access target of the programs is switched to ROM, it is referred to as in or during the ROM operation.

This function can be used with controller software version H7 or later.

Additionally, the high-speed RAM operation (using DRAM memory) function can be used in the controller's software version J1 or later.

Table 5-10:ROM operation/high-speed RAM parameter list

Parameter	Description and value
ROMDRV	Switches the access target of the programs between ROM and RAM. 0 = RAM mode (initial value) 1 = ROM mode 2 = High-speed RAM mode (high-speed RAM operation : DRAM memory is used. Can be used in software version J1 or later)
BACKUP	Copies programs, parameters, common variables and error logs from the RAM area into the ROM area. SRAM -> FLROM (fixed) * If this processing is canceled while being executed, "CANCEL" is displayed in the value field.
RESTORE	Rewrites programs, parameters, common variables and error logs in the ROM area into the RAM area. FLROM -> SRAM (fixed) * If this processing is canceled while being executed, "CANCEL" is displayed in the value field.

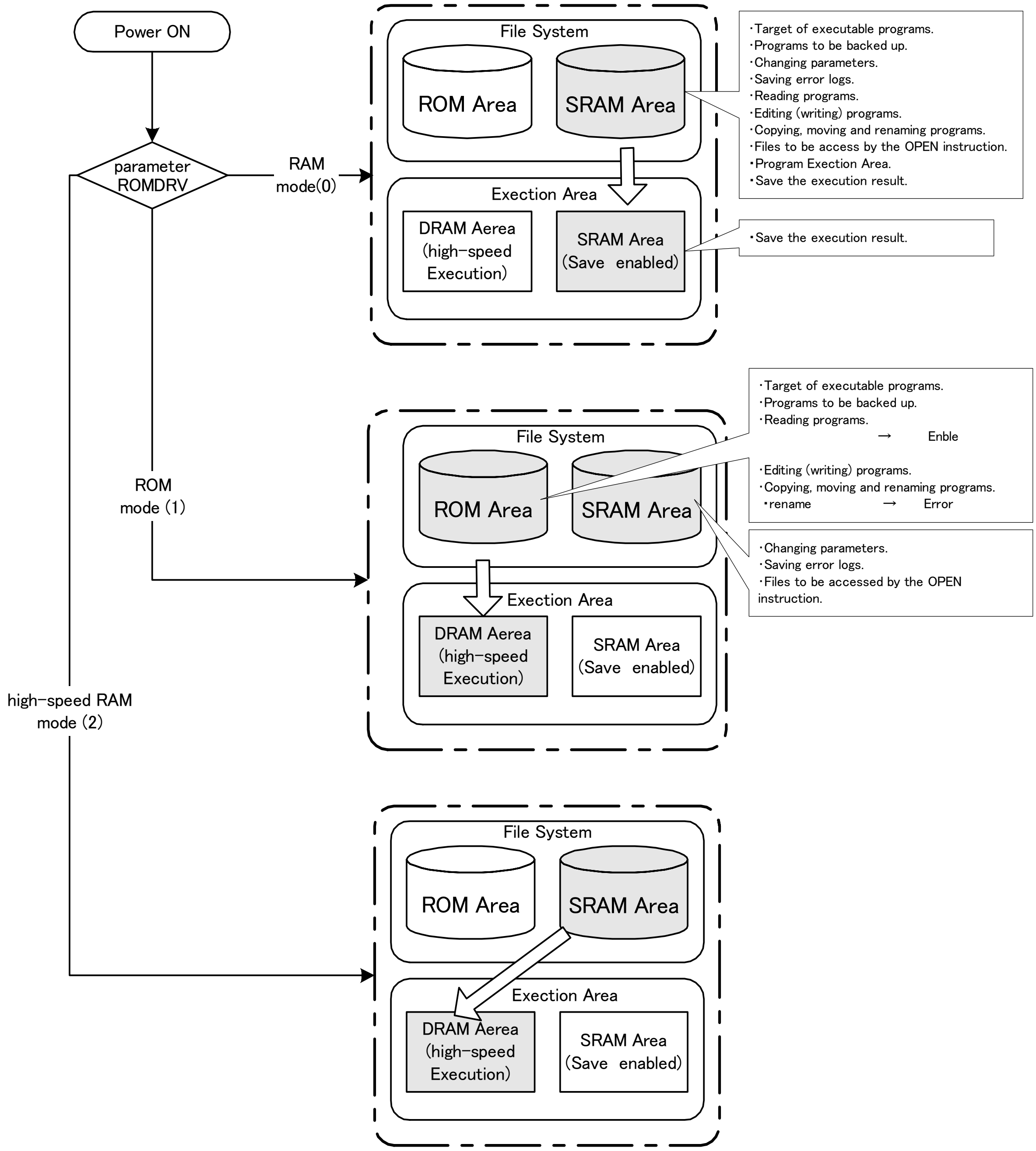
Table 5-11:Relationship between the role of each memory and the ROMDRV parameter

Memory type	Feature	ROMDRVparameter		
		0(RAM mode)	1(ROM mode)	2(High-speed RAM mode)
DRAM	High-speed execution possible Execution of programs that are erased when the power is OFF		Execution of programs (Discard the execution result)	Execution of programs (Discard the execution result)
SRAM	Not erased by power OFF Erased when a battery is consumed. Read/write enabled	Execution of programs (Save the execution result) Program management operation Read/write system data Read/write common variables Read/write programs	Read/write system data	Program management operation Read/write system data Read/write common variables Read/write programs
ROM	Not erased by power OFF Not Erased when a battery is consumed. Read only enabled		(Program management operation disabled) Read/write common variables Read/write programs	

Caution 1)"Save Parameters" and "Save Error Log" in order to save system data. The data files to be read or written by the programs (OPEN/PRINT/INPUT) are included.

Caution 2)Program management operation refers to the operations, such as copying, deleting and renaming the programs in the controller, by using the T/B and personal computer support software.

Table 5-12:ROM operation/high-speed RAM operation function image


Power ON
parameter
ROMDRV
RAM
mode(0)
ROM
mode (1)
high-speed RAM
mode (2)
File System
ROM Area
SRAM Area
Exection Area
DRAM Aerea
(high-speed
Execution)
SRAM Area
(Save enabled)
・Target of executable programs.
・Programs to be backed up.
・Changing parameters.
・Saving error logs.
・Reading programs.
・Editing (writing) programs.
・Copying, moving and renaming programs.
・Files to be access by the OPEN instruction.
・Program Exection Area.
・Save the execution result.
・Save the execution result.
・Target of executable programs.
・Programs to be backed up.
・Reading programs.
→ Enble
・Editing (writing) programs.
・Copying, moving and renaming programs.
・rename → Error
・Changing parameters.
・Saving error logs.
・Files to be accessed by the OPEN instruction.

DRAM is used as execution memory during ROM operation/high-speed RAM operation; it can perform language processing at a maximum speed of about 1.2 times faster than that of SRAM memory used for normal RAM operation. (The speed varies depending on the contents of each program.)

Note that the operations of the robot, such as program execution and step operation, can be performed similar to RAM operations (when starting in the RAM mode); however, there are restrictions on some operations. Please refer to the following precautions.

Precautions

* About variables

Variables may be changed by executing programs during the ROM operation/high-speed RAM operation; however, the changed values will be discarded when the controller power is turned off. The following lists the handling of variables during the ROM operation.

Variable Note1)	In ROM operation	In high-speed RAM operation	In RAM operation
Local variable	The values of local variables are retained during program operation; however, they will be discarded when switching programs by the OP or external I/O signal, as well as when the power is turned off. The values of variables in the program called by the CALLP instruction will be discarded when they return to the called program. The values of variables in a program called by a CALLP instruction are discarded upon returning to the calling program.	The values of variables used in a program being executed when the power was shut down are discarded. They are saved when a program is selected or a CALLP instruction finishes.	The values of variables are retained even after the power is turned off. Note2)
Program external variable	The values of variables are retained until the power is turned off. (They will not be discarded by switching programs. The contents of changes will be discarded when the power is turned off.)	The values of variables are retained as they are even after the power is shut down.	The values of variables are retained even after the power is turned off.
User-defined external variable		The contents of changes are discarded when the power is shut down.	

Note1)There are numeric value variables, character string variables, position variables and joint variables.

Note2)However, if a program is rewritten by using PC support software, the values of local variables used by programs will be discarded.

* Changing variables during program execution

If the execution of a program is aborted during the ROM operation and "Variable Monitor" (refer to Page 50, "3.12 Operating the monitor screen".) of the T/B is used, the program cannot be resumed. Although the stop lamp stays lit, if the program is executed, the program will be executed from the first line. Be careful because peripheral devices may interfere with the robot.

The values of variables cannot be changed by using "Variable Monitor" (refer to Page 50, "3.12 Operating the monitor screen".) of the T/B during the ROM operation. Program Monitor (watch function) of PC support software can be used to change the values of variables in task slots other than the editing slot.

* About programs

The target of program editing also becomes the ROM area. The programs in the controller is placed in the protected state (protect ON), and they cannot be canceled during the ROM operation. Once the ROM operation is switched to the RAM operation, the protect information reverts to the state set during the RAM operation.

Programs may be read during the ROM operation, but they cannot be written. Similarly, programs can neither be copied nor renamed.

* About parameters

Parameters and error log files are always saved in the RAM area regardless of switching between the ROM operation and the RAM operation. However, the RLNG parameter (for switching the robot language, refer to Page 321, " RLING".) cannot be changed during the ROM operation.

Only a limited number of robot models can use the RLNG parameter.

* About backup

During the ROM operation, programs are backed up from the ROM area, and parameters and error log files are backed up from the RAM area.

* About direct executio

While in the ROM operation, local variables cannot be rewritten by direct execution.

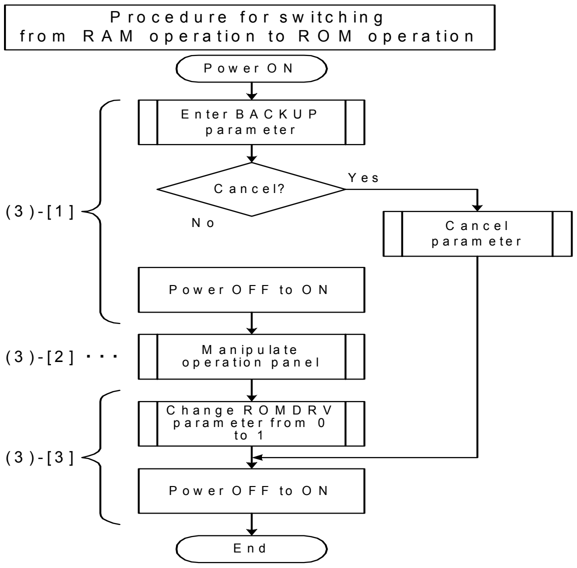
* About the continue function
While in the ROM operation, the continue function is disabled even if it is set.
The continue function saves the execution status at the time of power OFF, and starts operating from the saved status the next time the power is turned on.

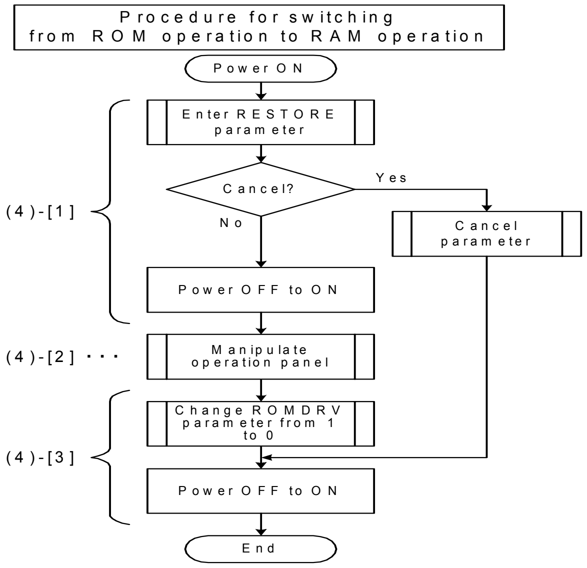
*About extension memory
When extension memory is installed or removed during the ROM operation, an error will occur. Install or remove extension memory only after switching to the RAM operation.

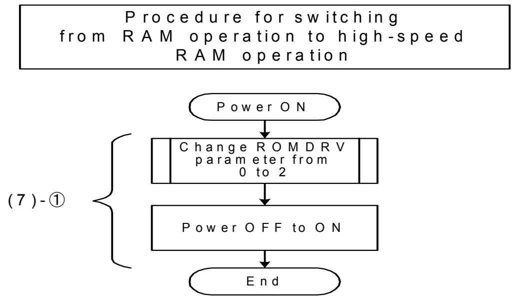
* About operating times
The operating times (power ON time and remaining battery time) are updated regardless of switching between the ROM and RAM operations.

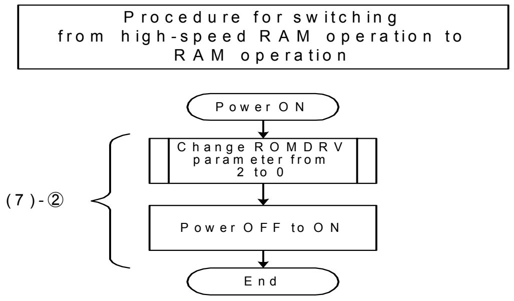
* About production information
The production information monitor (program operation count, cycle time, etc.) of Personal Computer support software is not added or updated during the ROM operation.

(2) Procedures for switching between ROM and RAM
RAM operation,The following shows the procedures for switching ROM operation, RAM operation and high-speed RAM operation:






Procedure for switching
from RAM operation to high-speed RAM operation
Power ON
Change ROMDRV parameter from 0 to 2
(7)-①
Power OFF to ON
End


Procedure for switching
from high-speed RAM operation to RAM operation
Power ON
Change ROMDRV parameter from 2 to 0
(7)-②
Power OFF to ON
End

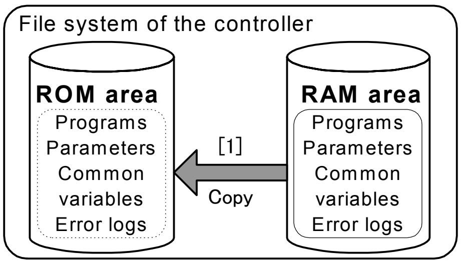
For more information about the operating procedure of each of the above, see the following pages.

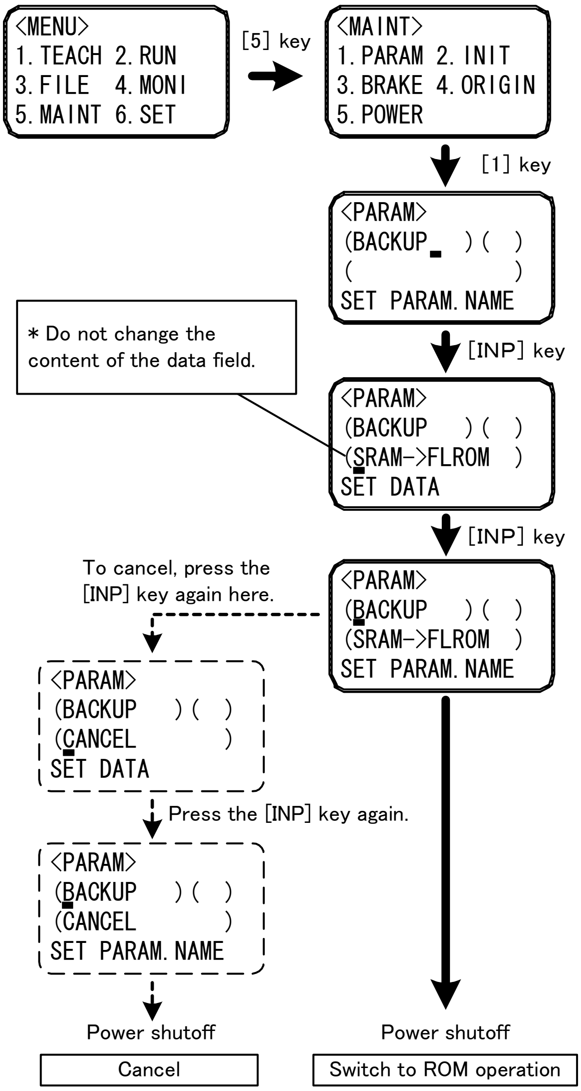
(3) Switching to the ROM operation

Use the following procedure (steps [1] to [3]) to switch to the ROM operation.

[1] Prepare to copy the information in the RAM area into the ROM area.

The programs created before the RAM operation was switched to the ROM operation are saved in the RAM area of the file system of the controller. First, copy these programs into the ROM area using the following procedure.
After the programs in the ROM area are cleared once, they are copied from the RAM area.




```
<MENU>
1.TEACH 2.RUN
3.FILE  4.MONI
5.MAINT 6.SET
```
```
<MAINT>
1.PARAM 2.INIT
3.BRAKE 4.ORIGIN
5.POWER
```
```
<PARAM>
(BACKUP_  )(  )
(            )
SET PARAM.NAME
```
```
<PARAM>
(BACKUP   )(  )
(SRAM->FLROM  )
SET DATA
```
```
<PARAM>
(BACKUP   )(  )
(SRAM->FLROM  )
SET PARAM.NAME
```
```
<PARAM>
(BACKUP   )(  )
(CANCEL       )
SET DATA
```
```
<PARAM>
(BACKUP   )(  )
(CANCEL       )
SET PARAM.NAME
```
[5] key
[1] key
[INP] key
[INP] key
* Do not change the content of the data field.
To cancel, press the [INP] key again here.
Press the [INP] key again.
Power shutoff
Cancel
Power shutoff
Switch to ROM operation

1) Display the parameter setting screen from the maintenance screen.

2) Enter "BACKUP" in the parameter name field, and press the [INP] key.

3) When "SRAM->FLROM" is displayed in the data field, press the [INP] key again as is.
* Do not change the content of the data field.

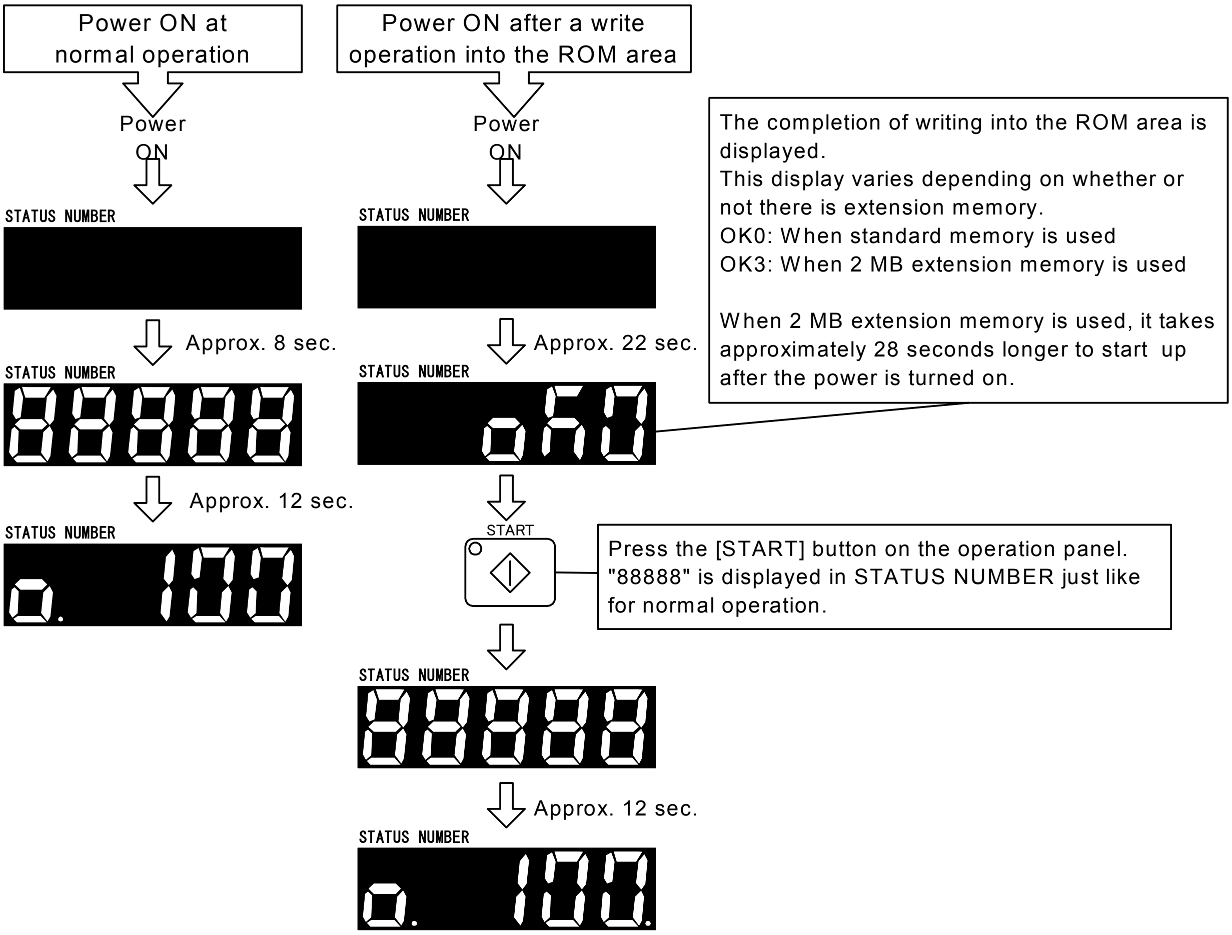
A self-check is performed to check whether or not the programs are normal prior to writing them into the ROM area. The ALWAYS program is automatically stopped during the self-check. When the program check is complete, the cursor moves to the parameter name field. If any abnormality is found during the program check, an error is output and the date of ROM write operation is registered in an error log.

[Cancel Operation]
To cancel switching to the ROM operation, press the [INP] key again here. When "CANCEL" is displayed in the data field, press the [INP] key again.

4) Be sure to shut off the power here. Also, be sure to shut off the power during [Cancel Operation] in step 3) above. Copy to the ROM area is performed the next time the power is turned on. If the power is not shut off after the operation in step 3), data will not be properly copied into the ROM area.

5) Turn on the power to the controller.
After the power is turned on, "OK" is displayed in "STATUS NUMBER" on the operation panel. After verifying that "OK" is displayed, press the [START] button on the operation panel. (The following page describes the detail of this operation.)

[2] Execute a copy operation by manipulating the operation panel.



The completion of writing into the ROM area is displayed.
This display varies depending on whether or not there is extension memory.
OK0: When standard memory is used
OK3: When 2 MB extension memory is used

When 2 MB extension memory is used, it takes approximately 28 seconds longer to start up after the power is turned on.

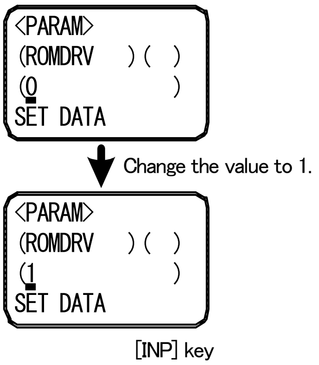
Press the [START] button on the operation panel. "88888" is displayed in STATUS NUMBER just like for normal operation.

The time required to start up after the power is turned on is based on controller software version H7.
It varies depending on the version in use and how memory is used.

⚠Caution

If switching from the RAM operation to the ROM operation is performed without copying any program into the ROM area, there would be no program to execute, or a program in which the corrected content has not been reflected would be executed. Therefore, be sure to perform the program copy operation described above.

[3] Change the parameter value and switch to the ROM operation.



```
<PARAM>
(ROMDRV   )(  )
(0           )
SET DATA
```

Change the value to 1.

```
<PARAM>
(ROMDRV   )(  )
(1           )
SET DATA
```

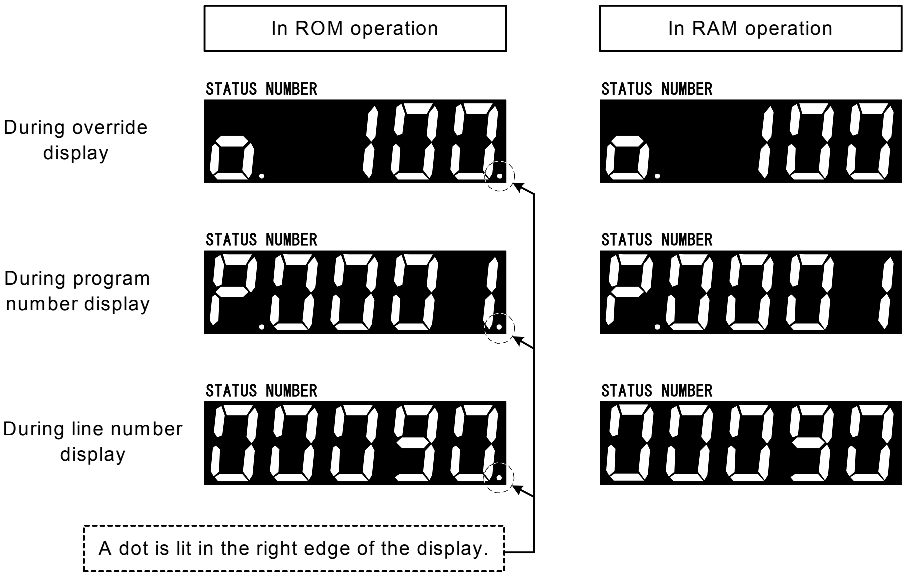
[INP] key

Change the value of the ROMDRV parameter from 0 to 1.

After changing it, be sure to shut off the power, and turn it on again.

(4) Display during the ROM operation

A dot is lit in the right edge of "STATUS NUMBER" on the operation panel during the ROM operation.



(5) Program editing during the ROM operation

It is possible to read the programs in the controller during the ROM operation; however, if a rewrite operation is performed, an error will occur. To edit a program, cancel the ROM operation (change the value of the ROMDRV parameter from 1 to 0, and reset the power to the controller) first, and then edit it. To switch back to the ROM operation after the completion of program editing, be sure to perform the operation "Copy Programs to ROM Area."

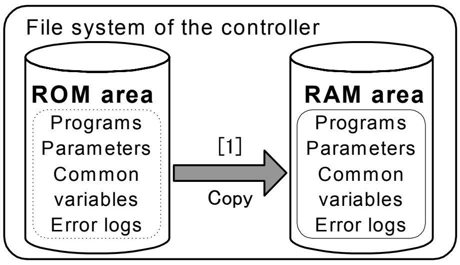
(6) Switching to the RAM operation

Use the following procedure (steps [1] to [3]) to switch to the RAM operation.

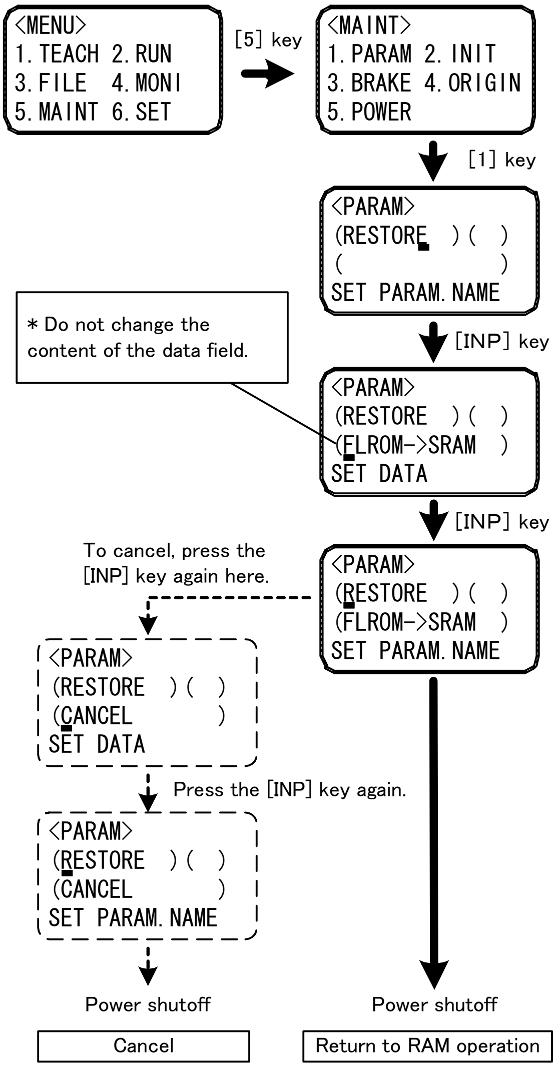
[1] Prepare to write the information in the ROM area back to the RAM area.

Write the programs and parameters written into the ROM area when the RAM operation was switched to the ROM operation back into the RAM area.

At this point, after the information (programs, parameters, values of common variables, and error logs) in the RAM area is cleared once, restore processing is performed from the ROM area.



> ⚠CAUTION If the information in the RAM area is restored into the ROM area, all the contents of the parameters changed during the ROM operation, the values of common variables, and logs of errors occurred are discarded. If any parameter was changed during the ROM operation, change the parameter again after switching to the RAM operation is complete.


```
<MENU>
1.TEACH 2.RUN
3.FILE  4.MONI
5.MAINT 6.SET
```

```
<MAINT>
1.PARAM 2.INIT
3.BRAKE 4.ORIGIN
5.POWER
```

```
<PARAM>
(RESTORE  )(  )
(            )
SET PARAM.NAME
```

```
<PARAM>
(RESTORE  )(  )
(FLROM->SRAM  )
SET DATA
```
* Do not change the content of the data field.
[INP] key
```
<PARAM>
(RESTORE  )(  )
(FLROM->SRAM  )
SET PARAM.NAME
```
To cancel, press the [INP] key again here.
```
<PARAM>
(RESTORE  )(  )
(CANCEL       )
SET DATA
```
Press the [INP] key again.
```
<PARAM>
(RESTORE  )(  )
(CANCEL       )
SET PARAM.NAME
```
Power shutoff
Cancel
Power shutoff
Return to RAM operation

1) Display the parameter setting screen from the maintenance screen.

2) Enter "RESTORE" in the parameter name field, and press the [INP] key.

3) When "FLROM->SRAM" is displayed in the data field, press the [INP] key again as is.
* Do not change the content of the data field.

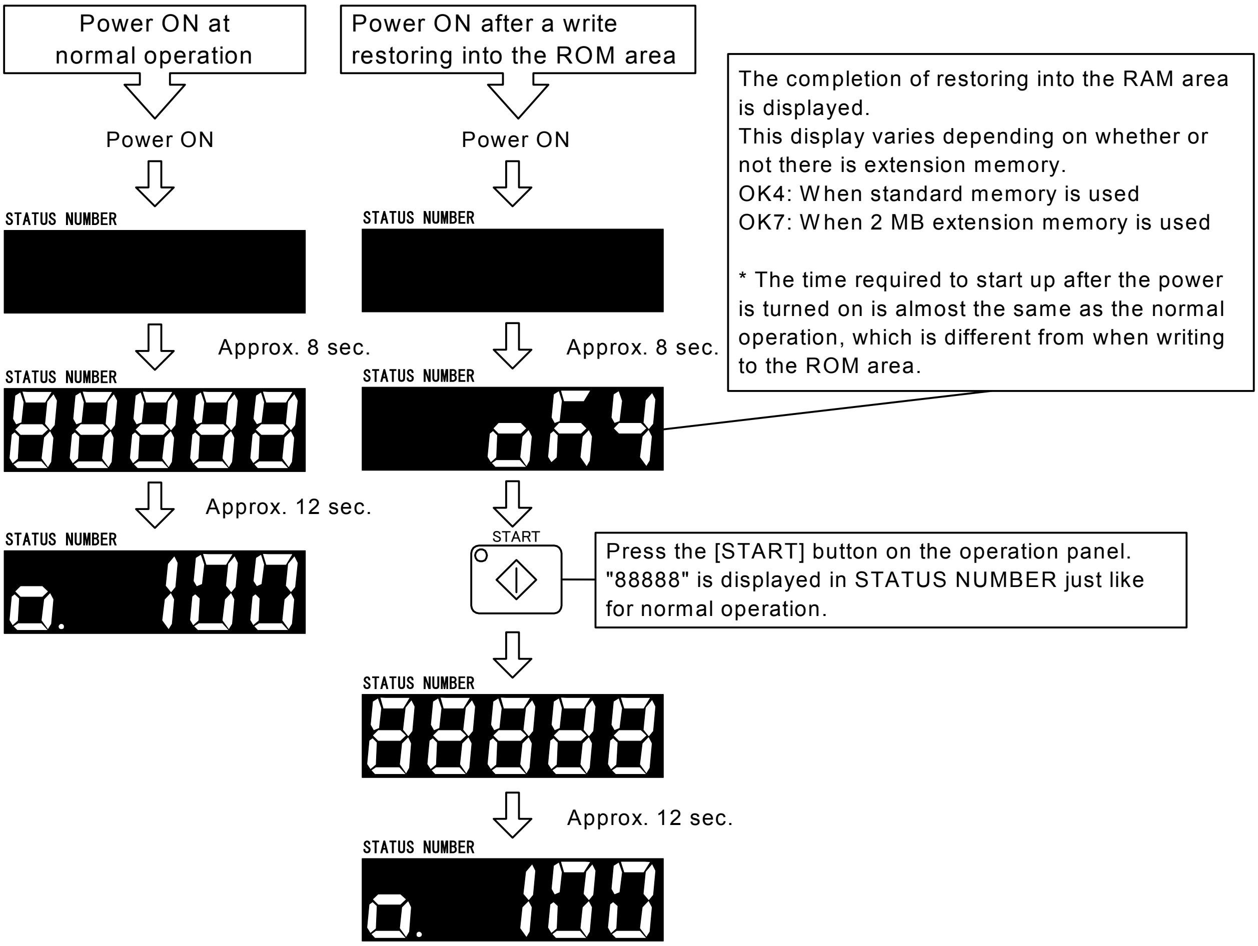
Prepare to restore the information back into the RAM area. At this point, the ALWAYS program does not stop. When the preparation is complete, the cursor moves to the parameter name field.

[Cancel Operation]
To cancel switching to the RAM operation, press the [INP] key again here. When "CANCEL" is displayed in the data field, press the [INP] key again. The cursor moves to the parameter name field.

4) Be sure to shut off the power here. Also, be sure to shut off the power during [Cancel Operation] in step 3) above. Copy to the RAM area is performed the next time the power is turned on. If the power is not shut off after the operation in step 3), data will not be properly copied into the RAM area.

5) Turn on the power to the controller.
After the power is turned on, "OK" is displayed in "STATUS NUMBER" on the operation panel. After verifying that "OK" is displayed, press the [START] button on the operation panel. (The following page describes the detail of this operation.)

[2] Execute a restore operation by manipulating the operation panel

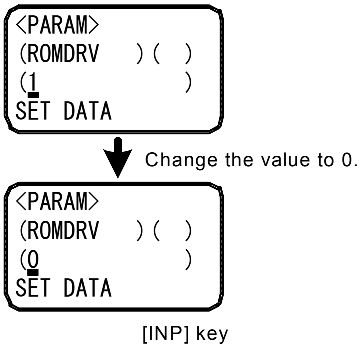


The completion of restoring into the RAM area is displayed.
This display varies depending on whether or not there is extension memory.
OK4: When standard memory is used
OK7: When 2 MB extension memory is used

* The time required to start up after the power is turned on is almost the same as the normal operation, which is different from when writing to the ROM area.

Press the [START] button on the operation panel. "88888" is displayed in STATUS NUMBER just like for normal operation.

[3] Change the parameter value and switch to the RAM operation.



```
<PARAM>
(ROMDRV   )(  )
(1           )
SET DATA
```

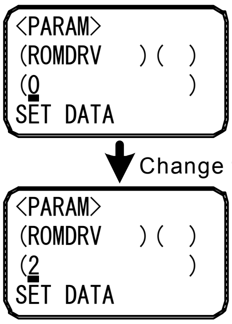
```
<PARAM>
(ROMDRV   )(  )
(0           )
SET DATA
```

Change the value of the ROMDRV parameter from 1 to 0.

After changing it, be sure to shut off the power, and turn it on again.

(7) Switching to the high-speed RAM operation(DRAM operation)
Use the following procedure to switch to the high-speed RAM operation.

[1]Change the applicable parameter and switch to high-speed RAM operation (DRAM operation).



```
<PARAM>
(ROMDRV   )(  )
(0           )
SET DATA
```

Change the value to 2.

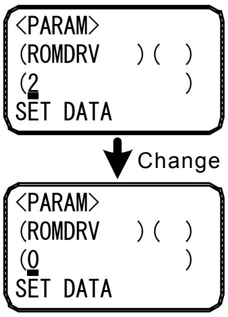
```
<PARAM>
(ROMDRV   )(  )
(2           )
SET DATA
```

[INP] key

Change the value of the ROMDRV parameter from 0 to 2.

After changing it, be sure to shut off the power, and turn it on again.

[2]Change back the parameter and return to RAM operation.



```
<PARAM>
(ROMDRV   )(  )
(2           )
SET DATA
```

Change the value to 0.

```
<PARAM>
(ROMDRV   )(  )
(0           )
SET DATA
```

[INP] key

Change the value of the ROMDRV parameter from 2 to 0.

After changing it, be sure to shut off the power, and turn it on again.

5.19 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.

This function can be used in software version J8 or later of the controller.

*To Use the Warm-Up Operation Mode

To use the warm-up operation mode, specify 1 (enable) in the WUPENA parameter and power on the controller again.

Note: To use the warm-up operation mode on the robot other than the RV-S series, it is necessary to specify a joint axis to be the target of the warm-up operation mode in the WUPAXIS parameter, other than the WUPENA parameter. For more information, see "(2)Function Details"

*When the Warm-Up Operation Mode Is Enabled

When the warm-up operation mode is enabled, powering on the controller enters the warm-up operation status (the speed is automatically reduced). In the warm-up operation status, the robot operates at a speed lower than the specified operation speed, then gradually returns to the specified speed as the operation time of a target axis elapses. The ratio of reducing the speed is referred to as the warm-up operation override. When this value is 100%, the robot operates at the specified speed. In parameter setting at shipment from the factory, the value of a warm-up operation override changes as shown in the Fig. 5-2 below according to the operation time of a target axis.

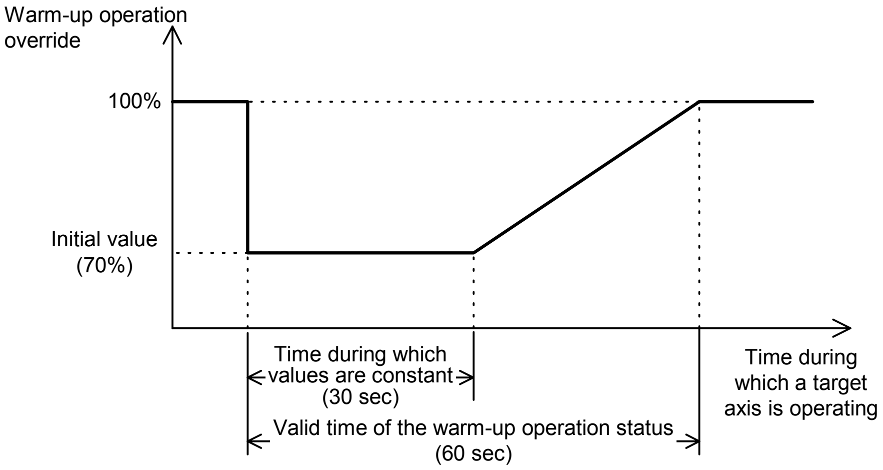


Fig.5-2:Changes in Warm-Up Operation Override

⚠CAUTION Even in the warm-up operation status, the robot does not decrease its speed if the MODE switch on the controller's front panel is set to "TEACH," for a jog operation or for an operation by real-time external control (MXT instruction), and operates at the originally specified speed.

⚠CAUTION In the warm-up operation status, because the robot operates at a speed lower than the originally specified speed, be sure to apply an interlock with peripheral units.

⚠CAUTION If the operating duty of the target axis is low, a servo error such as an excessive difference error may occur even when the warm-up operation mode is enabled.
In such a case, change the program, and lower the speed as well as the acceleration/ deceleration speed.

Also, when STATUS NUMBER on the controller's front panel is set to override display in the warm-up operation status, an underscore (_) is displayed in the second digit from the left so that you can confirm the warm-up operation status.

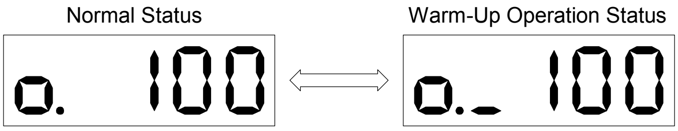


Fig.5-3:Override Display in the Warm-Up Operation Status

When a target axis operates and the warm-up operation status is canceled, the robot operates at the specified speed. Note that the joint section cools down at a low temperature if the robot continues to stop after the warm-up operation status is canceled. Therefore, if a target axis continues to stop for a prolonged period (the setting value at shipment from the factory is 60 min), the warm-up operation status is set again and the robot operates at a reduced speed.

Note 1: When powering off the controller and then powering on again, if the power-off period is short, the temperature of the robot's joint section does not decrease too much. Therefore, when powering off the controller and then powering on again after the warm-up operation status is canceled, if the power-off period is short, the robot starts in the normal status instead of the warm-up operation status.

Note 2: A target axis refers to the joint axis that is the target of control in the warm-up operation mode. It is the joint axis specified in the WUPAXIS parameter.

(2) Function Details

1)Parameters, Dedicated I/O Signals and Status Variables of the Warm-Up Operation Mode

The following parameters, dedicated I/O signals and status variables have been added in the warm-up operation mode. Refer to Page 306, "5.1 Movement parameter", Page 371, "6.3 Dedicated input/output" and Page 60, "4 MELFA-BASIC IV" for details.

Table 5-13: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, Åc 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 5-14: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 5-15: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.

Note: For robots other than the RV-S series, the setting value of the WUPAXIS parameter at shipment from the factory has been set to 0. When using the warm-up operation mode, specify a target axis in the warm-up operation mode (the joint axis to be the target of control in the warm-up operation mode; for example, a joint axis that generates an excessive difference error when operating in a low temperature environment).

*Switching with a Dedicated Input Signal

By assigning the MnWUPENA (n = 1 to 3: mechanism number) dedicated input signal, the warm-up operation mode can be enabled or disabled without powering on the controller again. Also, the current enable/disable status can be checked with the MnWUPENA (n = 1 to 3: mechanism number) dedicated output signal.

Note 1:In order for the dedicated input signal above to function, it is necessary to enable the warm-up operation mode in advance by setting the parameters described previously.

Note 2:This dedicated input signal requires the operation right of external I/O. Also, no input is accepted during operation or jog operation.

Note 3:The enable/disable status specified by this dedicated input signal is held even after the control right of external I/O is lost.

3) When the Warm-Up Operation Mode Is Enabled

When the warm-up operation mode is enabled, powering on the controller enters the warm-up operation status.

In the warm-up operation status, the robot operates at a speed lower than the actual operation speed by applying a warm-up operation override to the specified speed. The operation speed is gradually returned to the specified speed as the operation time of a target axis elapses. When the warm-up operation status is canceled, the robot will start operating at the specified speed.

*Initial Status Immediately After Power On

When the warm-up operation mode is enabled, powering on the controller enters the warm-up operation status.

However, when powering off the controller and then powering on again after the warm-up operation status is canceled, if the power-off period is short, the robot starts in the normal status instead of the warm-up operation status as the temperature of the robot's joint section has not been lowered much from power-off. To be specific, the robot starts in the normal status if the following condition is satisfied:

Condition: The robot starts in the normal status if the time during which a target axis continues to stop from the cancellation of the warm-up operation status to powering on is shorter than the time specified in the second element of the WUPTIME parameter (the resume time of the warm-up operation status).

Note that if the warm-up operation mode is switched to be enabled with the MnWUPENA (n = 1 to 3: mechanism number) dedicated input signal, the warm-up operation status is always set.

*Methods to Check the Warm-Up Operation Status
Whether the current status is the warm-up operation status or normal status can be checked in the following three methods:

- Checking with STATUS NUMBER on the controller's front panel
  The current status can be checked by setting STATUS NUMBER to override display. In the warm-up operation status, an underscore (_) is displayed in the second digit from the left.

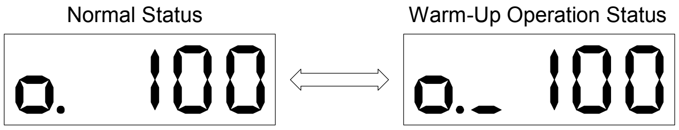


Fig.5-4:Override Display in the Warm-Up Operation Status

• Checking with a status variable
The current status can be checked by monitoring the value of the M_WUPOV status variable (the value of a warm-up operation override). In the normal status, the value of M_WUPOV is set to 100%; in the warm-up operation status, it is below 100%.

• Checking with a dedicated output signal
In the warm-up operation status, the MnWUPMD (n = 1 to 3: mechanism number) dedicated output signal is output.

*Switching Between the Normal Status and the Warm-Up Operation Status
When in the warm-up operation status, if a target axis in the warm-up operation mode continues operating and its operation time exceeds the valid time of the warm-up operation status, the warm-up operation status is canceled and the normal status is set. Thereafter, if the robot continues to stop, the joint section is cooled down in a low temperature environment. When a target axis continues to stop over an extended period of time and the resume time of the warm-up operation status is exceeded, the normal status switches to the warm-up operation status again.

• Canceling the warm-up operation status
If the accumulated time a target axis has operated exceeds the valid time of the warm-up operation status, the warm-up operation status is canceled and the normal status is set. Specify the valid time of the warm-up operation status in the first element of the WUPTIME parameter. (The setting value at shipment from the factory is 1 min.) If a multiple number of target axes exist, the warm-up operation status is canceled when all target axes exceed the valid time. Note that, with the M_WUPRT status variable, you can check when the warm-up operation status will be canceled after how much more time a target axis operates.

• Switching from the normal status to the warning-up operation status
If the time during which a target axis continues to stop exceeds the resume time of the warm-up operation status, the normal status switches to the warm-up operation status. Specify the resume time of the warm-up operation status in the second element of the WUPTIME parameter. (The setting value at shipment from the factory is 60 min.)
If a multiple number of target axes exist, the warm-up operation status is set when at least one of the axes exceeds the resume time of the warm-up operation status.
Note that, with the M_WUPST status variable, you can check when the status is switched to the warm-up operation status after how much more time a target axis continues to stop.
Note: If a target axis is not operating even when the robot is operating, it is determined that the target axis is stopping.

The following Fig. 5-5 shows an example of a timing chart for switching from the normal status to the warm-up operation status.

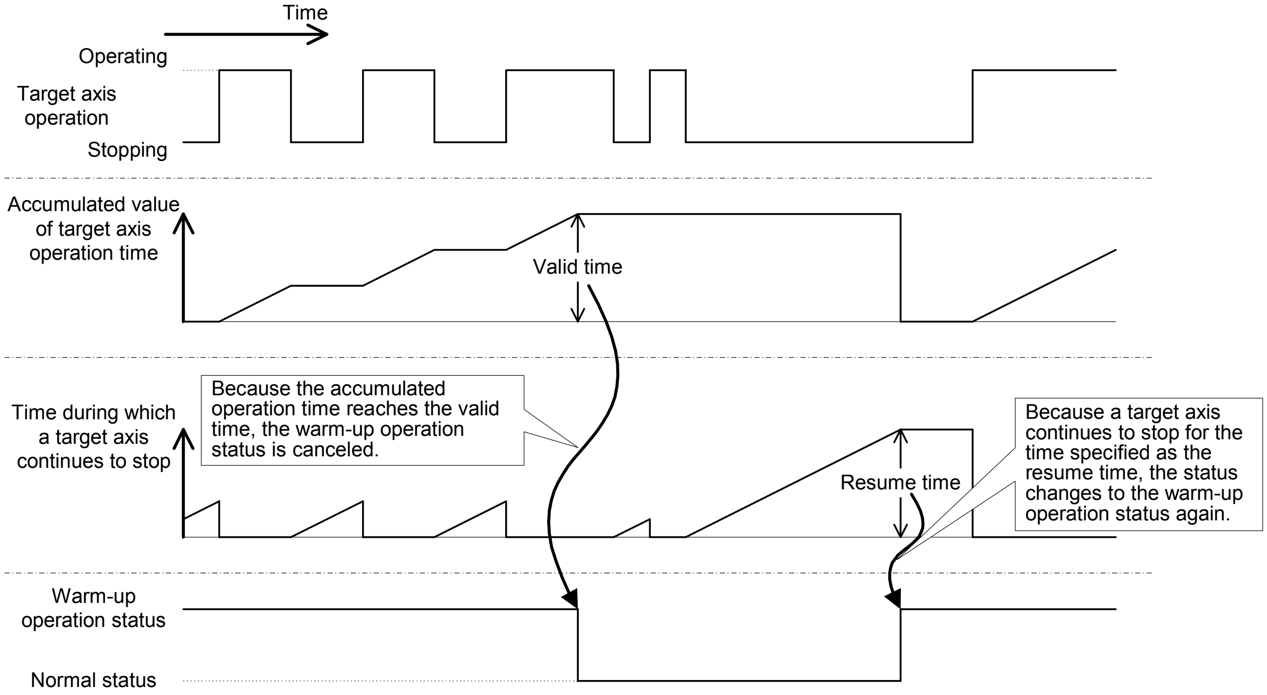

Time
Operating
Target axis operation
Stopping
Accumulated value of target axis operation time
Valid time
Time during which a target axis continues to stop
Because the accumulated operation time reaches the valid time, the warm-up operation status is canceled.
Resume time
Because a target axis continues to stop for the time specified as the resume time, the status changes to the warm-up operation status again.
Warm-up operation status
Normal 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.

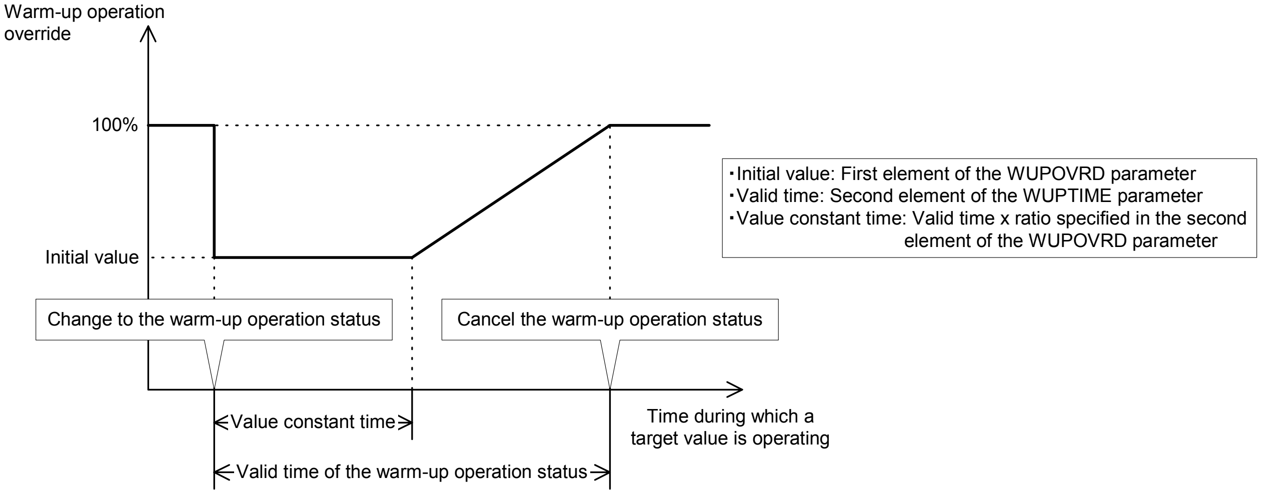


Fig.5-6:Changes in Warm-Up Operation Override

Note that the actual override in the warm-up operation status is as follows:

- During joint interpolation operation = (operation panel (T/B) override setting value) x (program override (OVRD instruction)) x (joint override (JOVRD instruction)) x warm-up operation override
- During linear interpolation operation = (operation panel (T/B) override setting value) x (program override (OVRD instruction)) x (linear specification speed (SPD instruction)) x warm-up operation override

Note 1:If the MODE switch on the controller's front panel is set to "TEACH," or for a jog operation or an operation by real-time external control (MXT instruction), the warm-up operation override is not reflected and the robot operates at the originally specified speed.

Note 2:In the warm-up operation status, because the robot operates at a speed lower than the originally specified speed, be sure to apply an interlock with peripheral units.

Note 3:If a multiple number of target axes exist, the warm-up operation override is calculated using the minimum operation time among the target axes. If a certain target axis does not operate and the value of the M_WUPRT status variable does not change, the value of the warm-up operation override does not change regardless how much other target axes operate.

Also, the value may return to the initial value before reaching 100% depending on whether each target axis is operating or stopping.

For example, when the value of a warm-up operation override is larger than the initial value, if a certain target axis switches from the normal status to the warm-up operation status, the operation time of that axis becomes the smallest (the operation time is 0 sec) and the warm-up operation override returns to the initial value.

(3) If alarms are generated

1) An excessive difference error occurs even if operating in the warm-up operation status.

- If an error occurs when the warm-up operation override is set to the initial value, decrease the value of the initial value (the first element of the WUPOVRD parameter).
- If an error occurs while the warm-up operation override is increasing to 100%, the valid time of the warm-up operation status or the value constant time may be too short. Increase the value of the first element of the WUPTIME parameter (valid time) or the second element of the WUPOVRD parameter (value constant time ratio).
- If an error cannot be resolved after taking the above actions, change the operation program, and lower the speed and/or the acceleration/deceleration speed.

2) An excessive difference error occurs if the warm-up operation status is canceled.

- Increase the value of the first element of the WUPTIME parameter, and extend the valid time of the warm-up operation status.
- Check to see if the robot's load and the surrounding temperature are within the specification range.
- Check whether the target axis continues to stop for an extended period of time after the warm-up operation status has been canceled. In such a case, decrease the value of the second element of the WUPTIME parameter, and shorten the time until the warm-up operation status is set again.
- If an error cannot be resolved after taking the above actions, change the operation program, and reduce the speed and/or the acceleration/deceleration speed.

3) The warm-up operation status is not canceled at all.

- Check the setting value of the WUPAXIS parameter to see if a joint section that does not operate at all is set as a target axis in the warm-up operation mode.
- Check to see if a target axis has been stopping longer than the resume time (the second element of the WUPTIME parameter) of the warm-up operation status.
- Check to see if an operation is continuing at an extremely low specified speed (about 3 to 5% in override during joint interpolation). If the specified speed is low, there is no need to use the warm-up operation mode. Thus, disable the warm-up operation mode.

5.20 About singular point passage function

(1) Overview of the function

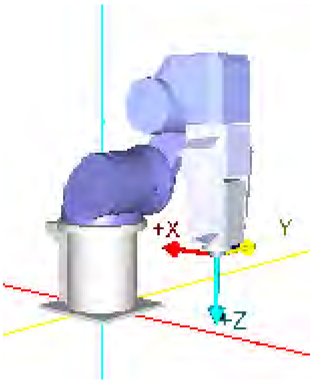
Mitsubishi's robots calculate linear interpolation movement and store teaching positions as position data in the XYZ coordinates system. In the case of a vertical 6-axis robot, for example, the position data is expressed using coordinate values of the robot's X, Y, Z, A, B and C axes, but the robot can 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.

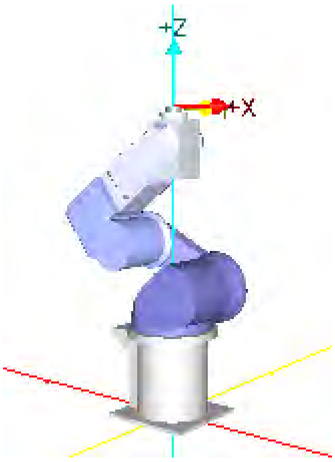
*Positions of singular points that can be passed

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

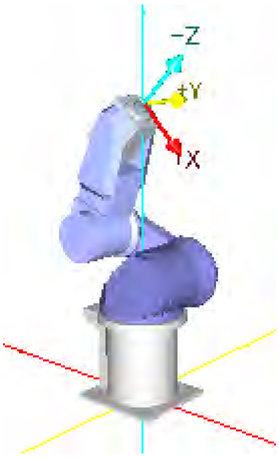


<1> Positions where axis J5 = 0 degree
In these positions, the structure flag switches between NonFlip and Flip.



<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



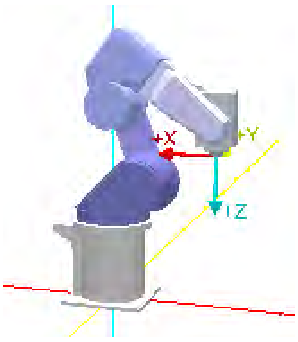
<1> Positions where the center of mechanical interface coincides with the rotation axis of axis J1
In these positions, the structure flag switches between Right and Left.

*Operation when the singular point passage function is valid

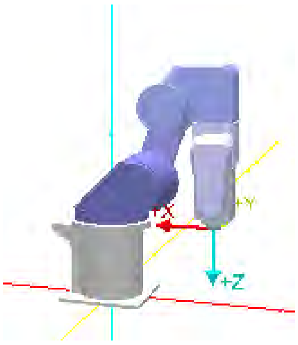
When the singular point passage function is made valid, the robot can move from position A to position C via position B (the position of a singular point) and vice versa through XYZ jog and linear interpolation operation. In this case, the value of the structure flag switches before and after passing 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.

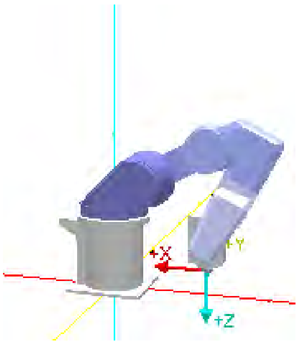
Position A



Position B



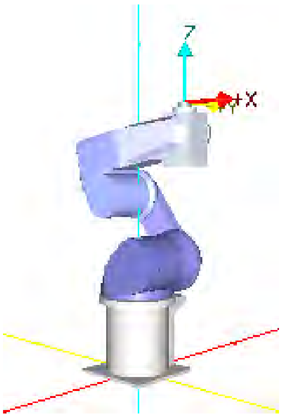
Position C



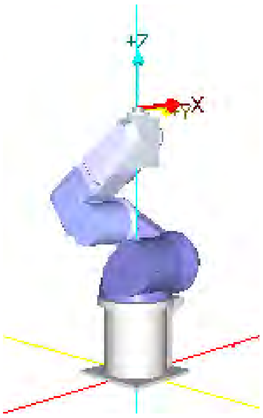
The robot can pass a singular point when the robot's motion path passes through the singular point. If the motion path does not go through the singular point (passes near the singular point), the robot continues operation without switching the value of the structure flag.

•Positions D -> E -> F: The robot's motion path passes through a singular point (the structure flag switches before and after position E).

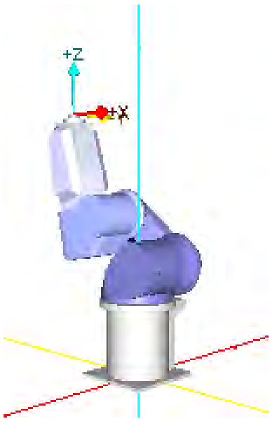
Position D



Position E

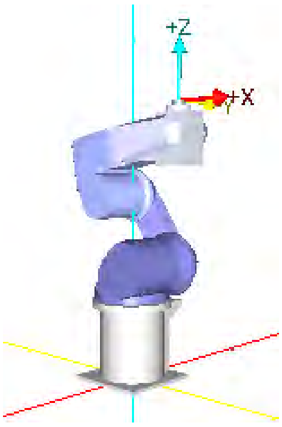


Position F

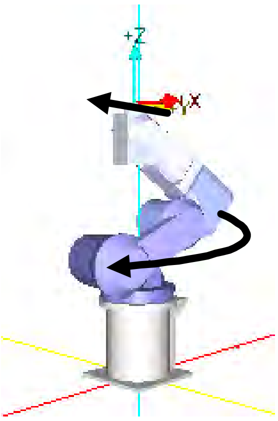


•Positions G -> H -> I: The robot's motion path passes near a singular point (the structure flag is not switched).

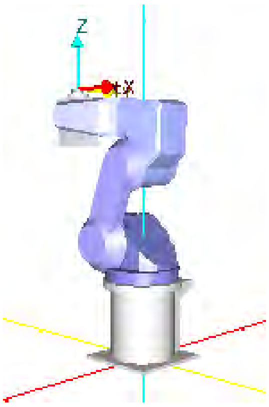
Position G



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.

*Applicable models
The models supporting the singular point passage function are the RV-3S/3SJ/3SB/3SJB series and the function can be used in controller software of version K1 or later. If the singular point passage function is made valid for non-applicable models, conventional motion is performed in the case of jog operation and an error occurs in the case of 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).
(6) MELFA-BASIC IV has instructions corresponding to the singular point passage function, but there are no corresponding commands for the MOVEMASTER COMMAND. Use MELFA-BASIC IV to use the singular point passage function.

(2) Singular point passage function in jog operation
In case of jog operation, the singular point passage function is specified to be valid (1) or invalid (0) by parameter FSPJOGMD.

FSPJOGMD	XYZ jog	TOOL jog	3-axis XYZ jog	CYLINDER jog	JOINT jog
0 (Factory setting)	Same as in the past	Same as in the past	Same as in the past	Same as in the past	Same as in the past
1	Singular point passage XYZ jog	Singular point passage TOOL jog	Same as in the past	Same as in the past	Same as in the past

1) For robots that cannot use the singular point passage function, changing the setting value of parameter FSPJOGMD has no effect; the same operation as in the past is performed (the models supporting the singular point passage function are the RV-3S/3SJ/3SB/3SJB series).
2) It is not possible to specify multiple axes to perform jog operation at the same time when passing a singular point. If it is attempted to operate an axis while another axis is operating, the operation is ignored.
3) A singular point adjacent alarm is generated if the robot comes near a singular point when performing jog operation using a T/B. See Page 344, "5.17 About the singular point adjacent alarm".
4) The specification of parameter FSPJOGMD is reflected in jog operation via dedicated input signals as well.

(3) Singular point passage function in position data confirmation (position jump)
The specification of parameter FSPJOGMD is also reflected in position data confirmation (position jump).

FSPJOGMD	MO position movement	MS position movement
0 (Factory setting)	Same as in the past	Same as in the past
1	Same as in the past	Singular point passage position movement

⚠CAUTION If an interpolation instruction (e.g., MVS P1) is executed directly from T/B when parameter FSPJOGMD is set to 1 (valid), the robot operates with the singular point passage function enabled even if the function is not made valid by the TYPE specification.

(4) Singular point passage function in automatic operation

In order to use the singular point passage function in automatic operation, make the function valid in the TYPE specification for each target interpolation instruction.

TYPE (Type)

[Function]

Specify the singular point passage function in the TYPE specification of an interpolation instruction. The interpolation instructions that support this function are linear interpolation (MVS), circular interpolation (MVR, MVR2 and MVR3) and circular interpolation (MVC). These instructions can be used in controller software of version K1 or later.

[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]

10 MVS P1 TYPE 0,2	' Perform linear interpolation from the current position to P1 with the singular point passage function enabled.
20 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.

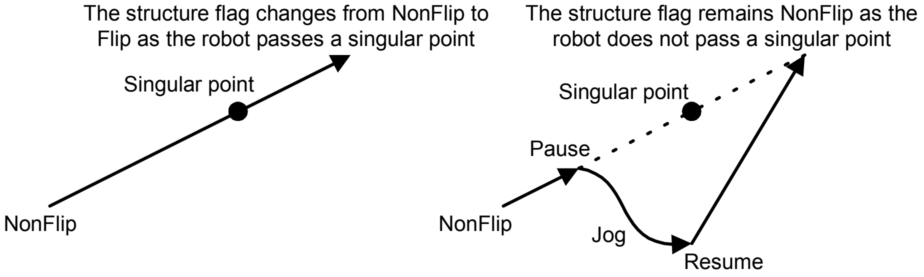
(3) If a speed is specified with the SPD instruction, the specified speed is set as the upper limit and the robot automatically lowers the speed down to the level where a speed error does not occur near a singular point.

(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.


The structure flag changes from NonFlip to Flip as the robot passes a singular point
Singular point
NonFlip
The structure flag remains NonFlip as the robot does not pass a singular point
Singular point
Pause
NonFlip
Jog
Resume

(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.

6 External input/output functions

6.1 Types

(1)Dedicated input/output................ These I/O signals are used to indicate various statuses, such as the statuses of remote operations including the execution and stopping of robot programs, status information during execution, and status of the servo power supply.
A desired function is assigned to each I/O signal. Functions are assigned in two ways: one is to set the applicable signal number in each dedicated parameter (Refer to "6.3 Dedicated input/output" on page 371), while the other is to use an emergency stop input (Refer to "6.6 Emergency stop input" on page 389). Frequently used functions have been assigned to signals in advance. The user may add new functions or modify the existing functions.

(2)General-purpose input/output..... These I/O signals are used to communicate with the PLC, etc., via robot programs. They are used to acquire positioning signals from peripheral devices or check the robot position. Usage is not limited.

(3)Hand input/output .... These I/O signals are used in the control of robot hands, for example, to instruct the hand to open or close or to acquire information from the sensors installed in the hand. They can be controlled with user programs. Wiring is performed until near the tip of the robot hand. (Hand outputs are optional.)

Table 6-1:Overall I/O Signal Map

	I/O Signal number	How to use
Standard remote I/Os	0 to 31 (15)Note1)	Refers/assigns with the M_IN,M_INB,M_INW,M_OUT,M_OUTB or M_OUTW variable Example) IF M_IN(0)=1 THEN M_OUT(0)=1
Expansion remote I/Os	32 to 255 (240) Note1)	Same as the above.
Hand input/output	900 to 907	Same as the above. May be substituted by the HOPEN and HCLOSE instructions. Example) IF M_IN(900) THEN M_OUT(900)=1 HOPEN 1 , HCLOSE 1
CC-Link Bit Note2)	When one station is occupied:6000 to 6031 When four station is occupied:6000 to 6127 (This is the signal number for station number 1. The last 2 bits cannot be used.)	Refers/assigns with the M_IN, M_INB, M_INW, M_OUT, M_OUTB or M_OUTW variable Example) IF M_IN(6000)=1 THEN M_OUT(6000)=1
CC-Link Register Note2)	When one station is occupied:6000 to 6003 When four station is occupied:6000 to 6015 (This is the signal number for station number 1.)	Referenced or assigned by the M_DIN and M_DOUT variables. Example) IF M_DIN(6000)=1000 THEN M_OUT(6000)=200

Note1)The descriptions in parentheses apply to the CR1 controller.
Note2)For details on CC-Link, refer to "CC-Link Interface Instruction Manual."

6.2 Connection method

The robot and external input/output device are connected by connecting the optional external input/output cable to the parallel input/output unit connector in the controller and the external input/output device. One parallel input/output unit is mounted in the controller as a standard. However, up to eight units can be mounted using options.

The power supply (24VDC) for the remote input/output unit installed outside of the controller, and the power supply (12 to 24VDC) for the input/output circuit must be prepared by the user.

The standard input/output unit pin Nos. and signal assignment are shown in Table 6-2 and Table 6-3. The pin layout is shown in Fig. 6-1. Refer to the Table 6-4, for the CR1 controller source type.

Refer to the separate "Standard Specifications" for details on the electrical specifications of the input/output circuit.

<Differences between Controller Models>

The number of standard I/O points provided by the CR1 controller is 16 for input points and 16 for output points.

Controllers other than CR1 (CR2, CR3, CR4, CR7, CR8 and CR9) provide 32 input points and 32 output points as the standard.

Table 6-2:Table of standard parallel input/output unit CN100 pin Nos. and signal assignments

Pin No.	2A-CBL wire color Note1)	Function name		Pin No.	2A-CBL wire color Note1)	Function name	
		General-purpose	Dedicated/power supply, common			General-purpose	Dedicated/power supply, common
1	Orange/red A		FG	26	Orange/blue A		FG
2	Gray/red A		0V: for pins 4-7	27	Gray/blue A		0V: for pins 29-32
3	White/red A		12/24V: for pins 4-7	28	White/blue A		12/24V: for pins 29-32
4	Orange/red A	General-purpose output 0	Running Note2)	29	Yellow/blue A	General-purpose output 4	
5	Pink/red A	General-purpose output 1	During servo ON Note2)	30	Pink/blue A	General-purpose output 5	
6	Orange/red B	General-purpose output 2	During error occurrence Note2)	31	Orange/blue B	General-purpose output 6	
7	Gray/red B	General-purpose output 3	Operation rights Note2)	32	Gray/blue B	General-purpose output 7	
8	White/red B		0V: for pins 10-13	33	White/blue B		0V: for pins 35-38
9	Orange/red B		12/24V: for pins 10-13	34	Yellow/blue B		12/24V: for pins 35-38
10	Pink/red B	General-purpose output 8		35	Pink/blue B	General-purpose output 12	
11	Orange/red C	General-purpose output 9		36	Orange/blue C	General-purpose output 13	
12	Gray/red C	General-purpose output 10		37	Gray/blue C	General-purpose output 14	
13	White/red C	General-purpose output 11		38	White/blue C	General-purpose output 15	
14	Orange/red C		COM0(12V/24V(COM)) : for pins 15-22	39	Yellow/blue C		COM1(12V/24V(COM)) : for pins 40-47
15	Pink/red C	General-purpose input 0	Stop (all slots stop) Note3)	40	Pink/blue C	General-purpose input 8	
16	Orange/red D	General-purpose input 1	Servo OFF Note2)	41	Orange/blue D	General-purpose input 9	
17	Gray/red D	General-purpose input 2	Error reset Note2)	42	Gray/blue D	General-purpose input 10	
18	White/red D	General-purpose input 3	Servo ON Note2)	43	White/blue D	General-purpose input 11	
19	Orange/red D	General-purpose input 4	Operation rights Note2)	44	Yellow/blue D	General-purpose input 12	
20	Pink/red D	General-purpose input 5		45	Pink/blue D	General-purpose input 13	
21	Orange/red E	General-purpose input 6		46	Orange/blue E	General-purpose input 14	
22	Gray/red E	General-purpose input 7		47	Gray/blue E	General-purpose input 15	
23	White/red E			48	White/blue E		
24	Orange/red E			49	Yellow/blue E		
25	Pink/red E			50	Pink/blue E		

Note1)The wire colors indicate the identification of the optional external input/output cables.

Note2)These are assigned as the default. The assignment can be changed with the dedicated input/output parameter. Refer to "6.3 Dedicated input/output" on page 371.

Note3)The stop input assignment is fixed to the input signal 0.

Table 6-3:Table of standard parallel input/output unit CN300 pin Nos. and signal assignments

Pin No.	2A-CBL wire color Note1)	Function name		Pin No.	2A-CBL wire color	Function name	
		General-purpose	Dedicated/power supply, common			General-purpose	Dedicated/power supply, common
1	Orange/red A		FG	26	Orange/blue A		FG
2	Gray/red A		0V: for pins 4-7	27	Gray/blue A		0V: for pins 29-32
3	White/red A		12/24V: for pins 4-7	28	White/blue A		12/24V: for pins 29-32
4	Orange/red A	General-purpose output 16		29	Yellow/blue A	General-purpose output 20	
5	Pink/red A	General-purpose output 17		30	Pink/blue A	General-purpose output 21	
6	Orange/red B	General-purpose output 18		31	Orange/blue B	General-purpose output 22	
7	Gray/red B	General-purpose output 19		32	Gray/blue B	General-purpose output 23	
8	White/red B		0V: for pins 10-13	33	White/blue B		0V: for pins 35-38
9	Orange/red B		12/24V: for pins 10-13	34	Yellow/blue B		12/24V: for pins 35-38
10	Pink/red B	General-purpose output 24		35	Pink/blue B	General-purpose output 28	
11	Orange/red C	General-purpose output 25		36	Orange/blue C	General-purpose output 29	
12	Gray/red C	General-purpose output 26		37	Gray/blue C	General-purpose output 30	
13	White/red C	General-purpose output 27		38	White/blue C	General-purpose output 31	
14	Orange/red C		COM0(12V/24V(COM)) : for pins 15-22	39	Yellow/blue C		COM1(12V/ 24V(COM)) : for pins 40-47
15	Pink/red C	General-purpose input 16		40	Pink/blue C	General-purpose input 24	
16	Orange/red D	General-purpose input 17		41	Orange/blue D	General-purpose input 25	
17	Gray/red D	General-purpose input 18		42	Gray/blue D	General-purpose input 26	
18	White/red D	General-purpose input 19		43	White/blue D	General-purpose input 27	
19	Orange/red D	General-purpose input 20		44	Yellow/blue D	General-purpose input 28	
20	Pink/red D	General-purpose input 21		45	Pink/blue D	General-purpose input 29	
21	Orange/red E	General-purpose input 22		46	Orange/blue E	General-purpose input 30	
22	Gray/red E	General-purpose input 23		47	Gray/blue E	General-purpose input 31	
23	White/red E			48	White/blue E		
24	Orange/red E			49	Yellow/blue E		
25	Pink/red E			50	Pink/blue E		

Note1) The wire colors indicate the identification of the optional external input/output cables.

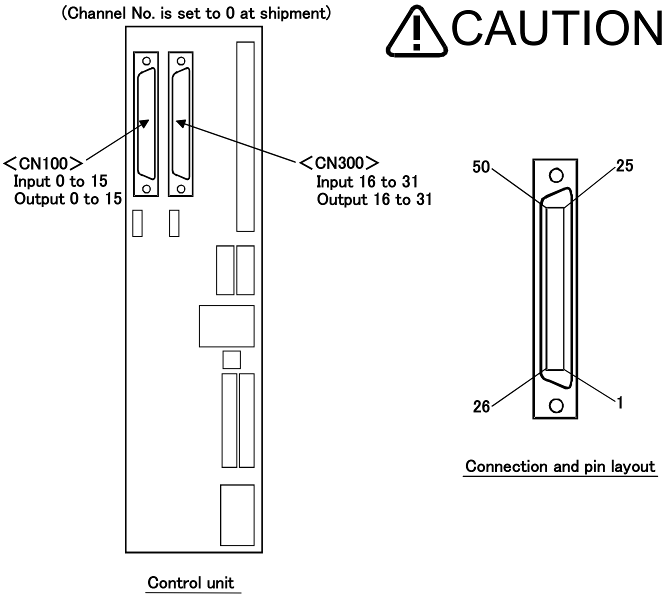
Table 6-4:Standard parallel I/O interface CN100pin No. and signal assignment list<Source type of CR1 controller>

Pin No.	2A-CBL wire color Note1)	Function name		Pin No.	2A-CBL wire color	Function name	
		General-purpose	Dedicated/power supply, common			General-purpose	Dedicated/power supply, common
1	Orange/Red A		FG	26	Orange/Blue A		FG
2	Gray/Red A		0V:For pins 4-7, 10-13	27	Gray/Blue A		0V:For pins 29-32, 35-38
3	White/Red A		12V/24V:For pins 4-7, 10-13	28	White/Blue A		12V/24V:For pins 29-32, 35-38
4	Yellow/Red A	General-purpose output 0	Running	29	Yellow/Blue A	General-purpose output 4	
5	Pink/Red A	General-purpose output 1	Servo on	30	Pink/Blue A	General-purpose output 5	
6	Orange/Red B	General-purpose output 2	Error	31	Orange/Blue B	General-purpose output 6	
7	Gray/Red B	General-purpose output 3	Operation rights	32	Gray/Blue B	General-purpose output 7	
8	White/Red B		Reserved	33	White/Blue B		Reserved
9	Yellow/Red B		Reserved	34	Yellow/Blue B		Reserved
10	Pink/Red B	General-purpose output 8		35	Pink/Blue B	General-purpose output 12	
11	Orange/Red C	General-purpose output 9		36	Orange/Blue C	General-purpose output 13	
12	Gray/Red C	General-purpose output 10		37	Gray/Blue C	General-purpose output 14	
13	White/Red C	General-purpose output 11		38	White/Blue C	General-purpose output 15	
14	Yellow/Red C		COM0(12V/24V(COM)) :For pins 15-22	39	Yellow/Blue C		COM1(12V/24V(COM)) :For pins 40-47
15	Pink/Red C	General-purpose input 0	Stop(All slot) Note2)	40	Pink/Blue C	General-purpose input 8	
16	Orange/Red D	General-purpose input 1	Servo off	41	Orange/Blue D	General-purpose input 9	
17	Gray/Red D	General-purpose input 2	Error reset	42	Gray/Blue D	General-purpose input 10	
18	White/Red D	General-purpose input 3	Start	43	White/Blue D	General-purpose input 11	
19	Yellow/Red D	General-purpose input 4	Servo on	44	Yellow/Blue D	General-purpose input 12	
20	Pink/Red D	General-purpose input 5	Operation rights	45	Pink/Blue D	General-purpose input 13	
21	Orange/Red E	General-purpose input 6		46	Orange/Blue E	General-purpose input 14	
22	Gray/Red E	General-purpose input 7		47	Gray/Blue E	General-purpose input 15	
23	White/Red E		Reserved	48	White/Blue E		Reserved
24	Yellow/Red E		Reserved	49	Yellow/Blue E		Reserved
25	Pink/Red E		Reserved	50	Pink/Blue E		Reserved

Note1) The wire colors indicate the identification of the optional external input/output cables.
Note2) The assignment of the dedicated input signal "STOP" is fixed.

⚠CAUTION The signals assigned as a dedicated input can be used as general-purpose inputs during program execution. However, for safety purposes, these must not be shared with the general-purpose inputs except for inputting values. The signals assigned as dedicated outputs cannot be used in the program. If used, an error will occur during operation

.



⚠CAUTION [*1]The channel number is set to "0".
The channel No. of 8 to F is used for the maker test. If any value of 8 to F is set, it may be dangerous since the robot unexpectedly moves. Don't set any value of 8 to F.
There is no station number setting switch in the I/O card equipped in the standard of CR1 controller.

* 2A-RZ361 is an input/output 32/32 point unit (Occupies one station)

Fig.6-1:Parallel input/output unit connection and pin layout

6.3 Dedicated input/output

The functions shown in Table 6-5 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 "(1)Setting the parameters" in "3.13 Operation of maintenance screen" on page 54 for details on setting the parameters. If a "-1" is designated for the assigned signal No., that signal will be invalidated.

The I/O parameters can be set on the T/B parameter screen or by using the maintenance tool of the PC support software (optional).

To use the dedicated I/O signals, set the key switch on the operation panel to AUTO (Ext.) beforehand.

Table 6-5:Table of dedicated input/output

Parameter name	Class	Name	Function	Signal level Note5)	Factory shipment signal number. Input, output
RCREADY	Input	-	-		-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.		-1
ATEXTMD	Input	-	-		-1(No meaning),
	Output	Remote mode output	This output indicates that the key switch on the operation panel is set to AUTO (Ext.), which is a remote operation mode. This signal must be turned ON before any control tasks using I/O signals can be performed.		-1
TEACHMD	Input	-	-		-1(No meaning),
	Output	Teaching mode output	This output indicates that the key switch on the operation panel is set to Teaching mode.		-1
ATTOPMD	Input	-	-		-1(No meaning),
	Output	Automatic mode output	This output indicates that the key switch on the operation panel is set to AUTO (OP),		-1
IOENA	Input	Operation rights input signal	Sets the validity of the operation rights for the external signal control.	Level	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 mode switch is set to AUTO (Ext.), and there is no other device that currently has the operation right.		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	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.		0
STOP	Input	Stop input	This input stops the program being executed. (This does not apply to slots whose starting condition is set to ALWAYS or ERROR.) The stop input signal No. is fixed to 0, and cannot be changed. All task slots are stopped during multitask operation. However, slots whose starting condition is set to ALWAYS or ERROR via a parameter "SLT**" will not be executed. Contacts A and B may be changed using the parameter INB.	Level	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.		-1

Parameter name	Class	Name	Function	Signal level Note5)	Factory shipment signal number. Input, output
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
	Output	Pausing output	This output indicates that the program is paused. (The specification is the same as for the STOP parameter.)		-1
STOPSTS	Input	-	-		-1(No meaning),
	Output	Stop signal input	Outputs that the stop is being input. (Logical ADD of all devices.)		-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	-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.		-1
ERRRESET	Input	Error reset input signal	Releases the error state.	Edge	2,
	Output	Error occurring output signal	Outputs that an error has occurred.		2
SRVON (Operation right required)	Input	Servo ON input signal	This input turns ON the servo power supply for the robot. With a multi-mechanism configuration, the servo power supplies for all mechanisms will be turned ON.	Edge	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.		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	1,
	Output	Servo ON disable output signal	This output indicates a status where the servo power supply cannot be turned ON. (Echo back)		-1
AUTOENA	Input	Automatic operation enabled input	Disables automatic operation when inactive. If this signal is inactive, and the AUTO mode is entered, E5010 will occur. This input is used to interlock the operations via the operation panel with the I/O signals. Use of this input is not a requirement.	Level	-1,
	Output	Automatic operation enabled output	Outputs the automatic operation enabled state.		-1
CYCLE	Input	Cycle stop input signal	Starts the cycle stop.	Edge	-1,
	Output	In cycle stop operation output signal	Outputs that the cycle stop is operating. Turns OFF when the cycle stop is completed.		-1
MELOCK (Operation right required)	Input	Machine lock input signal	Sets/releases the machine lock state for all mechanisms. This can be set or released when all slots are in the program selection state. Signal level will be set to Level when program selection is enabled.	Level	-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
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	-1,
	Output	In safe point return output signal	Outputs that the safe point return is taking place.		-1
BATERR	Input	-	-		-1,
	Output	Battery voltage drop	Outputs that the battery voltage has dropped.		-1
OUTRESET (Operation right required)	Input	General-purpose output signal reset	Resets the general-purpose output signal. The operation at the input is set with parameters ORST0 to ORST224.	Edge	-1,
	Output	-	-		-1(No meaning)

Parameter name	Class	Name	Function	Signal level Note5)	Factory shipment signal number. Input, output
HLVLERR	Input	-	-		-1(No meaning),
	Output	High level error output signal	Outputs that a high level error is occurring.		-1
LLVLERR	Input	-	-		-1(No meaning),
	Output	Low level error output signal	Outputs that a low level error is occurring.		-1
CLVLERR	Input	-	-		-1(No meaning),
	Output	Warning level error output signal	Outputs that a warning level error is occurring.		-1
EMGERR	Input	-	-		-1(No meaning),
	Output	Emergency stop output signal	Outputs that an emergency stop is occurring.		-1
SnSTART (n=1 to 32) (Operation right required)	Input	Slot n start input	Starts each slot. n=1 to 32	Edge	-1,
	Output	Slot n in operation output	Outputs the operating state for each slot. n=1 to 32		-1
SnSTOP (n=1 to 32)	Input	Slot n stop input	Outputs the operating state for each slot. n=1 to 32	Level	-1,
	Output	Slot n in pausing output	Outputs that each slot and program is temporarily stopped. n=1 to 32		-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,
	Output	Mechanism n servo ON disabled output signal	Outputs the servo ON disabled state. (Echo back)		-1
MnSRVON (n=1 to 3) (Operation right required)	Input	Mechanism n servo ON input signal	Turns the servo for each mechanism ON. n=1 to 3	Edge	-1,
	Output	Mechanism n in servo ON output signal.	Turns the servo for each mechanism ON. n=1 to 3		-1
MnMELOCK (n=1 to 3) (Operation right required)	Input	Mechanism n machine lock input signal	Sets/releases the machine lock state for each mechanism. n=1 to 3	Level	-1, -1
	Output	Mechanism n in machine lock output signal	Outputs that the machine lock state is entered. n=1 to 3		
PRGSEL (Operation right required)	Input	Program selection input signal	Designates the setting value for the program No. with numeric value input signals. The program for slot 1 is selected. Output this signal when at least 30 ms has elapsed following the start of output to the numerical input (IODATA). This signal should also be output to the robot for at least 30 ms.	Edge	-1, Note5)
	Output	-	-		-1(No meaning)
OVRDSEL (Operation right required)	Input	Override selection input signal	Designates the setting value for the override with the numeric value input signals. Output this signal when at least 30 ms has elapsed following the start of output to the numerical input (IODATA). This signal should also be output to the robot for at least 30 ms.	Edge	-1, Note5)
	Output	-	-		-1(No meaning)
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 OVRDSEL) The bit width can be set arbitrarily. However, the accuracy of output values cannot be guaranteed when they exceed the set bit width. Output this input to the robot for at least 30 ms before inputting the PRGSEL or other setting signals.	Level	-1(Start bit), -1(End bit), Note1) Note5)
	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 30 ms has elapsed following the start of input of a program number (PRGOUT) or other signal to the robot.		-1(Start bit), -1(End bit)

Parameter name	Class	Name	Function	Signal level Note5)	Factory shipment signal number. Input, output
DIODATA	Input	Numeric value input (Register number)	The specified numeric values are loaded. For numeric values, it is possible to enter real numbers consisting of 16 bits. Similar to the IODATA parameter, "program number," "override value" and other values are input to the specified registers. When inputting to the robot, input PRGSEL (program number selection) and OVRDSEL (override selection) after outputting at least for 30 ms. Note) This parameter is exclusively used for CC-Link. Also, if the IODATA parameter is set, the numeric values set in the IODATA parameter take precedence.	Level	-1
	Output	Numeric value output (Register number)	The numeric values of the specified items are output. For numeric values, it is possible to output real numbers consisting of 16 bits. Similar to the IODATA parameter, "program number," "override value," "line number," "error number" and other values are output to the specified registers. When LINEOUT (line number output request) and ERROUT (error number output request) are input, these values are output to the specified registers. Be sure to wait for at least 30 ms after that, and then load these values. Note) This parameter is exclusively used for CC-Link. Also, if the IODATA parameter is set, the numeric values set in the IODATA parameter and this parameter are output.		-1
PRGOUT	Input	Program No. output request	The program number for task slot 1 is output to the numerical output (IODATA). After the start of inputting this signal to the robot, wait at least 30 ms before reading the numerical output (IODATA) signal.	Edge	-1, Note5)
	Output	Program No. output signal	The "program number output in progress" status is output to the numerical output.		-1
LINEOUT	Input	Line No. output request	The line number for task slot 1 is output to the numerical output (IODATA). After the start of inputting this signal to the robot, wait at least 30 ms before reading the numerical output (IODATA) signal.	Edge	-1, Note5)
	Output	Line No. output request	The "line number output in progress" status is output to the numerical output.		-1
OVRDOUT	Input	Override value request	The OP override is output to the numerical output (IODATA). After the start of inputting this signal to the robot, wait at least 30 ms before reading the numerical output (IODATA) signal.	Edge	-1, Note5)
	Output	Override value output signal	The "override output in progress" status is output to the numerical output.		-1
ERROUT	Input	Error No. output request	The error number is output to the numerical output (IODATA). After the start of inputting this signal to the robot, wait at least 30 ms before reading the numerical output (IODATA) signal.	Edge	-1, Note5)
	Output	Error No. output signal	The "error number output in progress" status is output to the numerical output.		-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,
	Output	Jog valid output signal	Outputs that the jog operation is entered.		-1
JOGM	Input	Jog mode input (start No., end No.)	Designates the jog mode. 0/1/2/3/4 = Joint, XYZ, cylindrical, 3-axis XYZ, tool	Level	-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)
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, Éý, Z, A, B, C, L1 and L2 axes from the start number. 3-axis XYZ jog mode: X, Y, Z, J4, J5 and J6 axes from the start number. TOOL jog mode: X, Y, Z, A, B and C axes from the start number.	jiku	-1, Note3) -1
	Output	-	-		

Parameter name	Class	Name	Function	Signal level Note5)	Factory shipment signal number. Input, 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, Éý, Z, A, B, C, L1 and L2 axes from the start number. 3-axis XYZ jog mode: X, Y, Z, J4, J5 and J6 axes from the start number. TOOL jog mode: X, Y, Z, A, B and C axes from the start number.	Level	-1, Note3) -1
	Output	-	-		
JOGNER (Operation right required)	Input	Errors during jog operation Temporarily ignoring input signal	Temporarily ignores errors that cannot be reset during jog operation. * This signal is applicable to only machine 1. The controller software version J2 or later.	Level	-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. The controller software version J2 or later.		-1
HNDCNTLn (n=1 to 3)	Input	-	-		
	Output	Mechanism n hand output signal state (start No., end No.)	Outputs the hand output(n=1) 900 to 907 state. Outputs the hand output(n=2) 910 to 917 state. Outputs the hand output(n=3) 920 to 927 state. Example) To output the four points from 900 through 903 to general-purpose output signals 3, 4, 5 and 6, set the HNDCNTL1 to (3, 6).		-1(Start bit), -1(End bit)
HNDSTSn (n=1 to 3)	Input	-	-		
	Output	Mechanism n hand input signal state (start No., end No.)	Outputs the hand input(n=1) 900 to 907 state. Outputs the hand input(n=2) 910 to 917 state. Outputs the hand input(n=3) 920 to 927 state. Example) To output the four points from 900 through 903 to general-purpose output signals 3, 4, 5 and 6, set the HNDCNTL1 to (3, 6).		-1(Start bit), -1(End bit)
HNDERRn (n=1 to 3)	Input	Mechanism n hand error input signal	Requests the hand error occurrence. A LOW level error (error number 30) will be generated.	Level	-1,
	Output	Mechanism n hand error output signal	Outputs that a hand error is occurring.		-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.		
USRAREA Refer to "5.8 About user-defined area" on page 328	Input	-	-		
	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)		-1(Start bit), -1(End bit) Note4)
MnPTEXC (n=1 to 3)	Input	-	-		-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

Parameter name	Class	Name	Function	Signal level Note5)	Factory shipment signal number. Input, output
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,
	Output	Mechanism n warm-up operation mode output signal	Outputs that the warm-up operation mode is currently enabled. (n=1 to 3)		-1
MnWUPMD (n=1 to 3)	Input	-	-		-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

Note 1) 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 16, and the operating output signal in general-purpose output 25.
Parameter START ={16, 25}
Example)When setting 4 bits of numerical input to general-purpose inputs 6 to 9, and 5 bits of numerical output to general-purpose outputs 6 to 10.
Parameter IODATA = {6, 9, 6, 10}

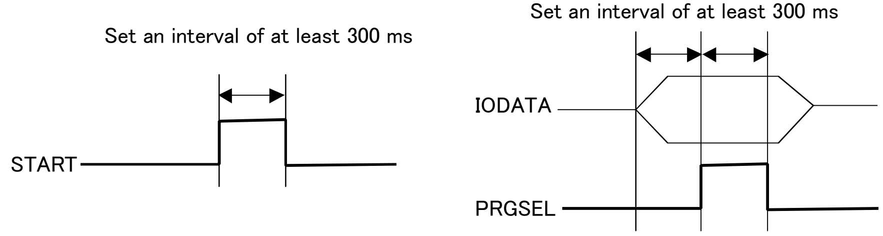
Note 2) Set in the order of input start No., input end No., output start No. and output end No.
When using as the actual jog mode, use from the start No. to the end No., and express as a binary. The start No. indicates the low-order bit, and the end No. indicates the high-order bit. Set only the numbers required to express the value.
For example, when using only the joint mode and XYZ mode, the expression can be created by setting 2 bits. Up to 3 bits can be set.

Note 3) They are in the order of an input starting number and then an input end number. Specify the J1/X axis for the input starting number and the J8/L2 axis for the input end number at its maximum.
For example, when using a 6-axis robot, only 6 bits need to be set.
Even if using a 4-axis robot, when using the XYZ mode, the C axis is required, so 6 bits must be set.
Up to 8 bits can be set.

Note 4) Set in the order of output start No. and output end No. The start number specifies area 1, while the end number specifies area 8 in the largest configuration.
For example, setting 2 bits will suffice if only two areas are used. A maximum of 8 bits can be set.

Note 5) The meanings of the signal level are explained below.
Level: The designated function is validated when the signal is ON, and the function is invalidated when the signal is OFF. Make sure the signal is turned ON for at least 30 ms.
Edge: The designated function is validated when the signal changes from the OFF to ON state, and the function maintains the original state even when the signal returns to the OFF state. .

Example)


Set an interval of at least 300 ms
START
Set an interval of at least 300 ms
IODATA
PRGSEL

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.

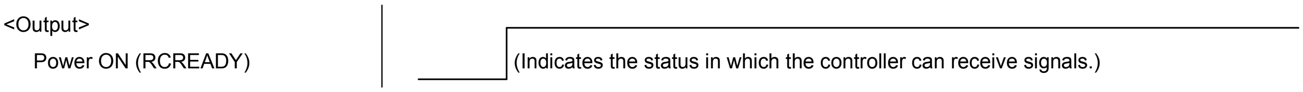
Table 6-6:Validity state of dedicated input signals

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	
PRGSEL	Program selection input	
MnWUPENA	Mechanism n warm-up operation mode enable input	
START SnSTART (n=1 to 32)	Start input	These function only when the external input/output has the operation rights (when IOENA output is ON).
SLOTINIT	Program reset	
SRVON MnSRVON (n=1 to 3)	Servo ON input	
MELOCK MnMELOCK (n=1 to 3)	Machine lock input	
SAFEPOS	Safe point return input	
PRGSEL	Program selection input	
OVRDSEL	Override selection input	
JOGENA	Jog enable input	
MnWUPENA	Mechanism n warm-up operation mode enable input	
START	Start input	These do not function in the stop input state (when STOPSTS is ON).
SLOTINIT	Program reset	
SAFEPOS	Safe point return input	
JOGENA	Jog enable input	
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).

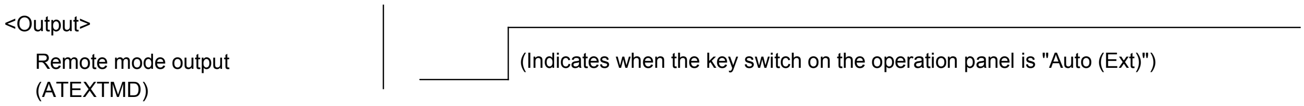
6.5 External signal timing chart

6.5.1 Individual timing chart of each signal

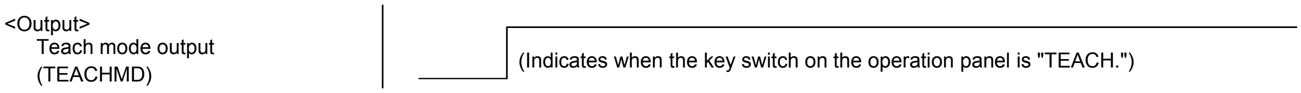
(1) RCREADY (Controller's power ON completion output)


<Output>
Power ON (RCREADY)
(Indicates the status in which the controller can receive signals.)

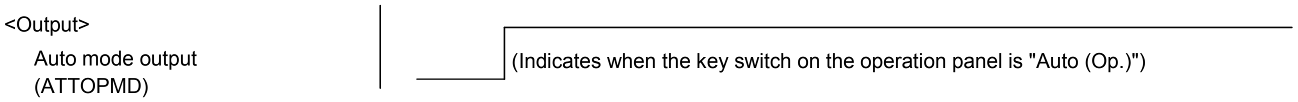
(2) ATEXTMD (Remote mode output)


<Output>
Remote mode output (ATEXTMD)
(Indicates when the key switch on the operation panel is "Auto (Ext)")

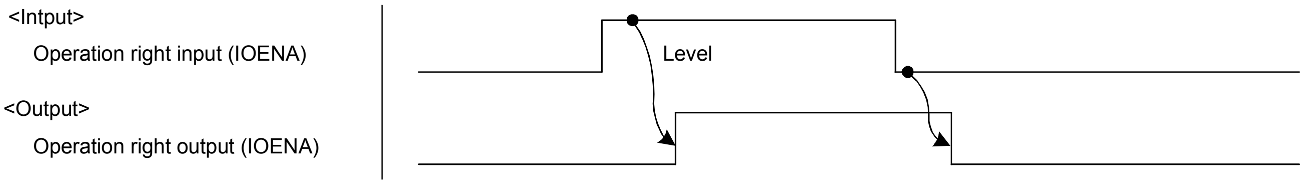
(3) TEACHMD (Teach mode output)


<Output>
Teach mode output (TEACHMD)
(Indicates when the key switch on the operation panel is "TEACH.")

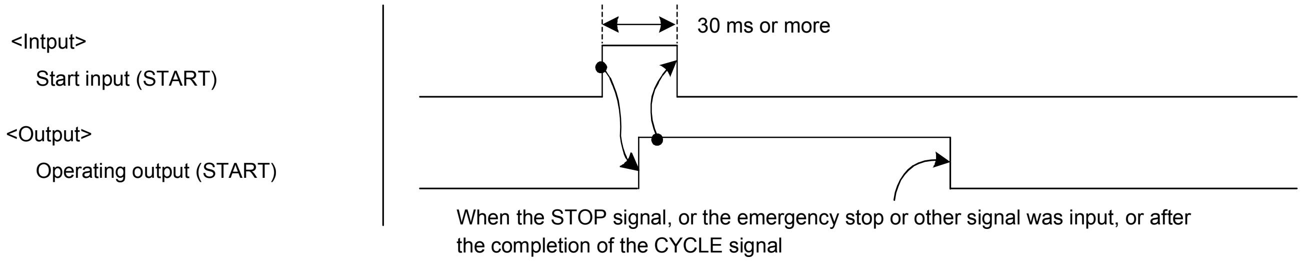
(4) ATTOPMD (Auto mode output)


<Output>
Auto mode output (ATTOPMD)
(Indicates when the key switch on the operation panel is "Auto (Op.)")

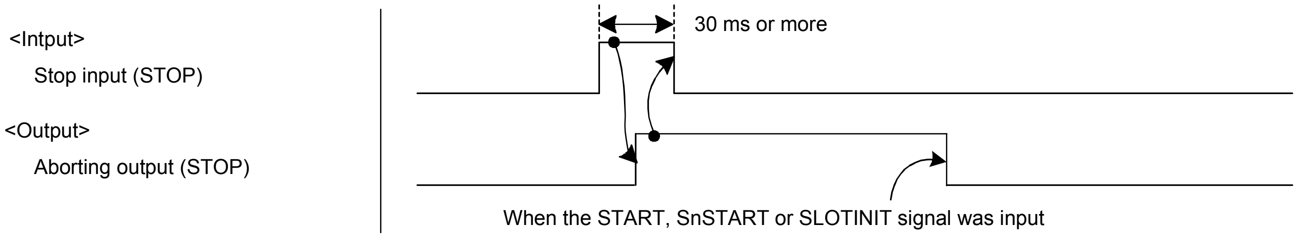
(5) IOENA (Operation right input signal/operation right output signal)



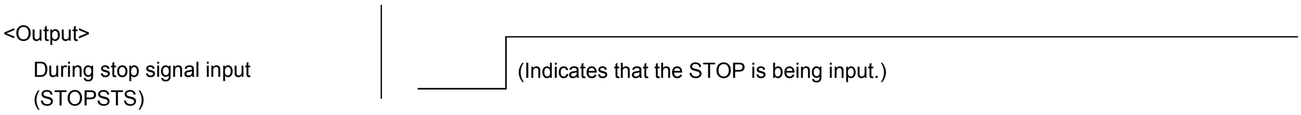
(6) START (Start input/operating output)


30 ms or more
<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

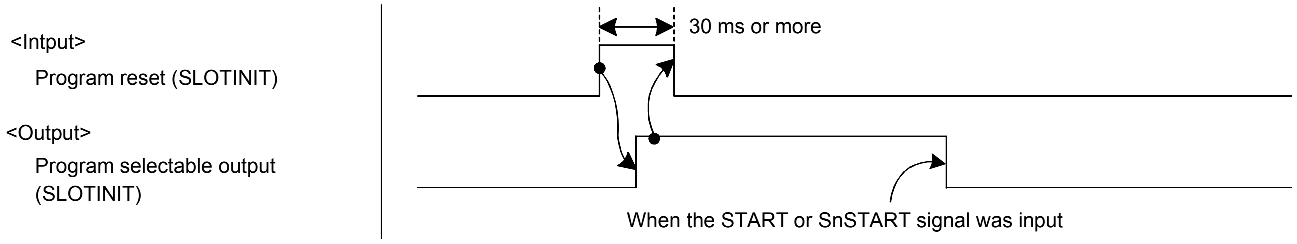
(7) STOP (Stop input/aborting output)



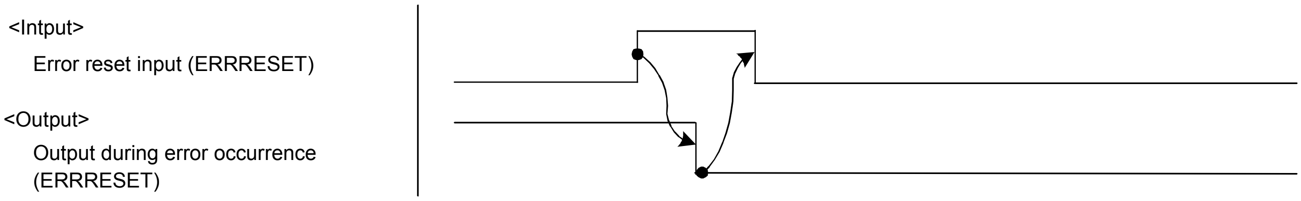
(8) STOPSTS (Output during stop signal input)


<Output>
During stop signal input (STOPSTS)
(Indicates that the STOP is being input.)

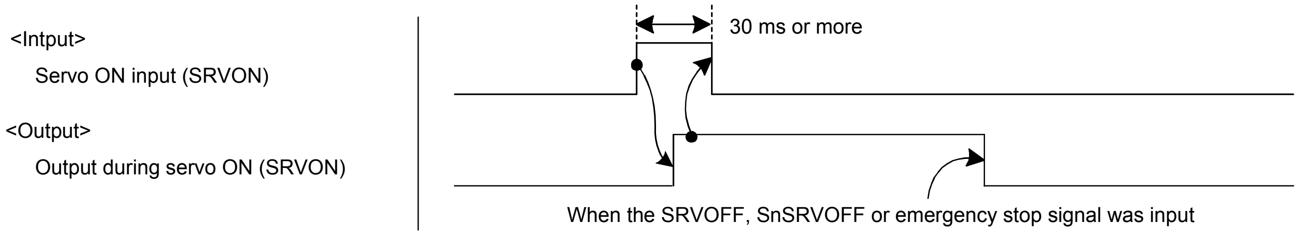
(9) SLOTINIT (Program reset input/program selectable output)


<Intput>
Program reset (SLOTINIT)
<Output>
Program selectable output (SLOTINIT)
30 ms or more
When the START or SnSTART signal was input

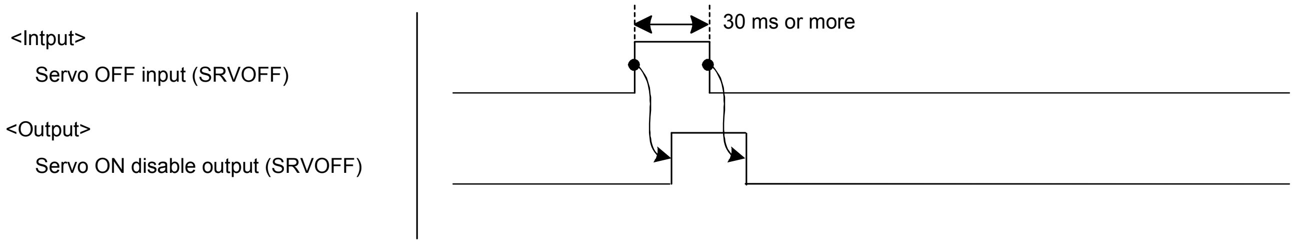
(10) ERRRESET (Error reset input/output during error occurrence)


<Intput>
Error reset input (ERRRESET)
<Output>
Output during error occurrence (ERRRESET)

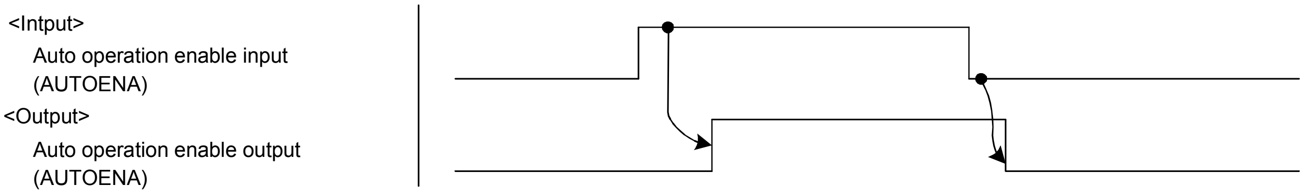
(11) SRVON (Servo ON input/output during servo ON))


<Intput>
Servo ON input (SRVON)
<Output>
Output during servo ON (SRVON)
30 ms or more
When the SRVOFF, SnSRVOFF or emergency stop signal was input

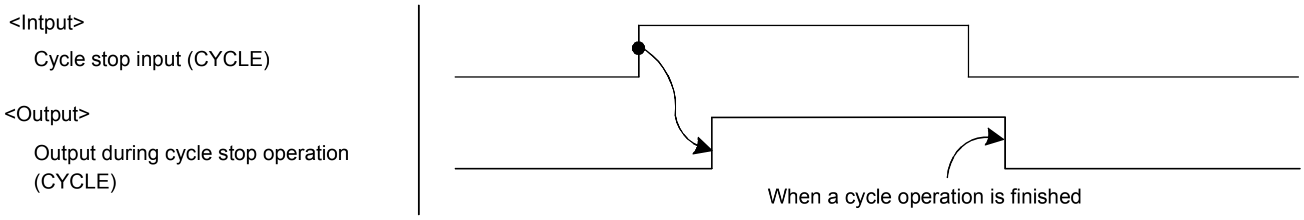
(12) SRVOFF (Servo OFF input/servo ON disable output)



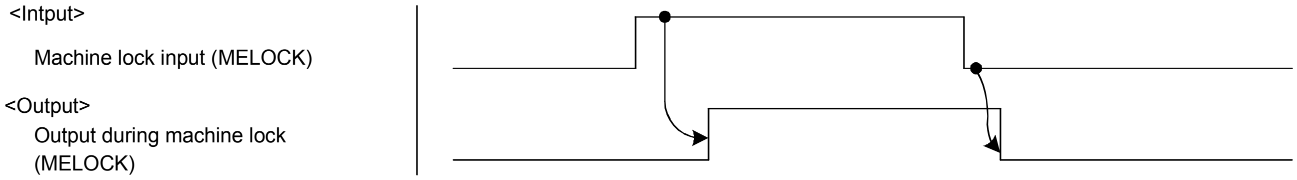
(13) AUTOENA (Auto operation input/auto operation enable output)


<Intput>
Auto operation enable input (AUTOENA)
<Output>
Auto operation enable output (AUTOENA)

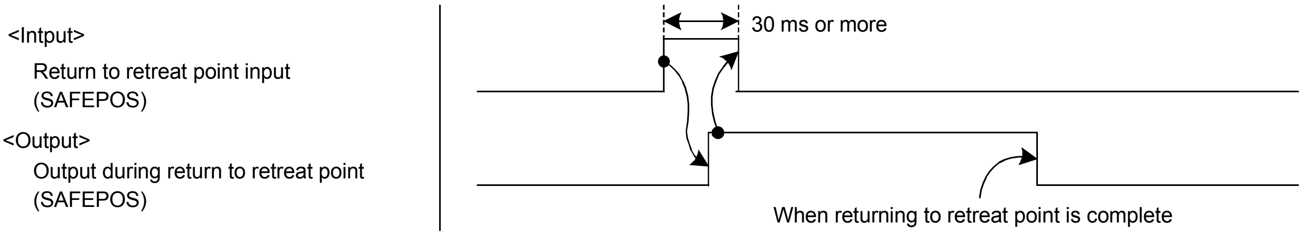
(14) CYCLE (Cycle stop input/output during cycle stop operation)


<Intput>
Cycle stop input (CYCLE)
<Output>
Output during cycle stop operation (CYCLE)
When a cycle operation is finished

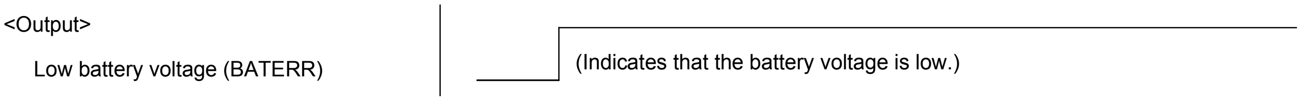
(15) MELOCK (Machine lock input/output during machine lock)



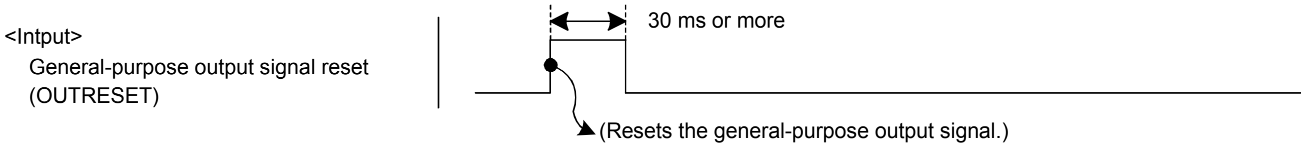
(16) SAFEPOS (Return to retreat point input/output during return to retreat point)


30 ms or more
<Intput>
Return to retreat point input
(SAFEPOS)
<Output>
Output during return to retreat point
(SAFEPOS)
When returning to retreat point is complete

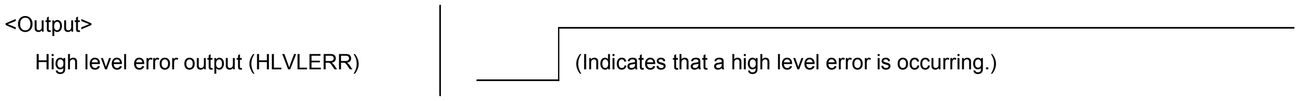
(17) BATERR (Low battery voltage output)


<Output>
Low battery voltage (BATERR)
(Indicates that the battery voltage is low.)

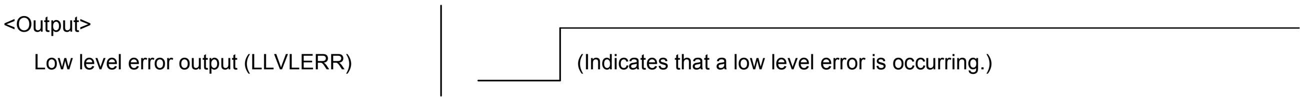
(18) OUTRESET (General-purpose output signal reset request input)


30 ms or more
<Intput>
General-purpose output signal reset
(OUTRESET)
(Resets the general-purpose output signal.)

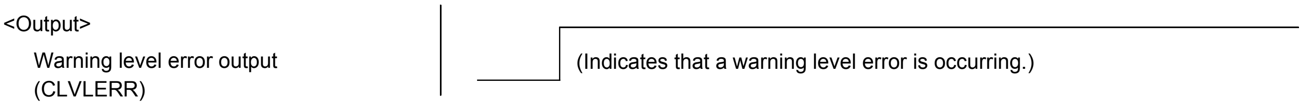
(19) HLVLERR (Output during high level error occurrence)


<Output>
High level error output (HLVLERR)
(Indicates that a high level error is occurring.)

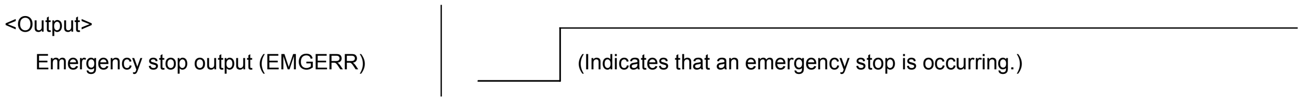
(20) LLVLERR (Output during low level error occurrence)


<Output>
Low level error output (LLVLERR)
(Indicates that a low level error is occurring.)

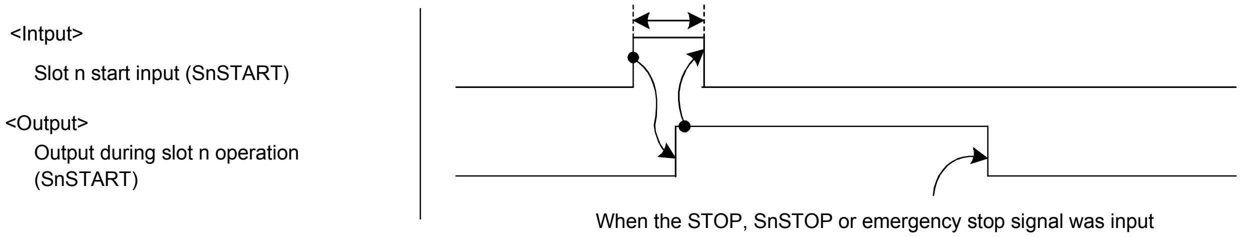
(21) CLVLERR (Output during warning level error occurrence)


<Output>
Warning level error output
(CLVLERR)
(Indicates that a warning level error is occurring.)

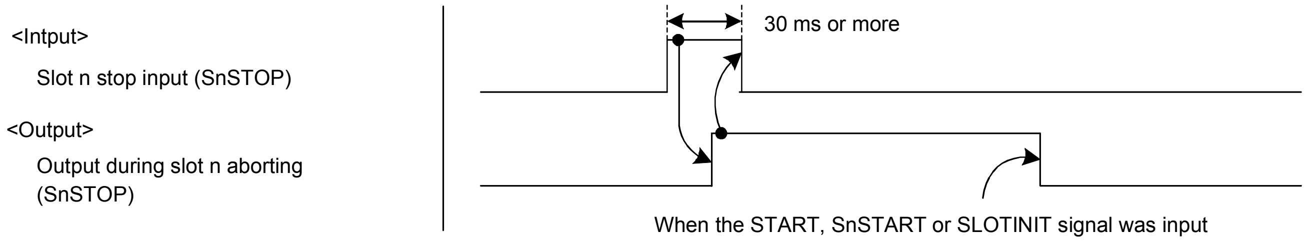
(22) EMGERR (Output during emergency stop)


<Output>
Emergency stop output (EMGERR)
(Indicates that an emergency stop is occurring.)

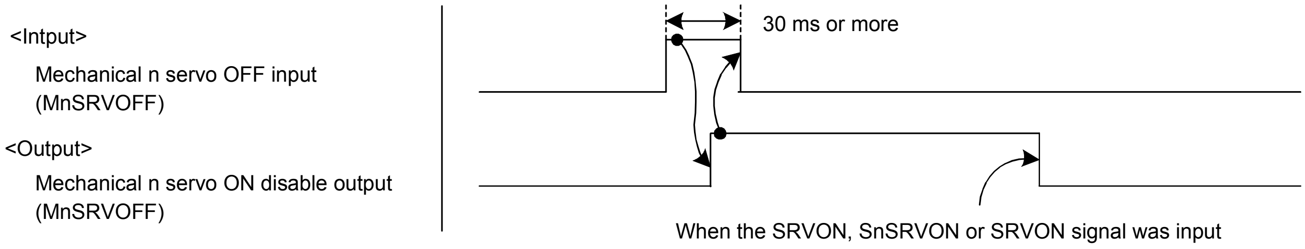
(23) SnSTART (Slot n start input/output during slot n operation)


<Intput>
Slot n start input (SnSTART)
<Output>
Output during slot n operation
(SnSTART)
When the STOP, SnSTOP or emergency stop signal was input

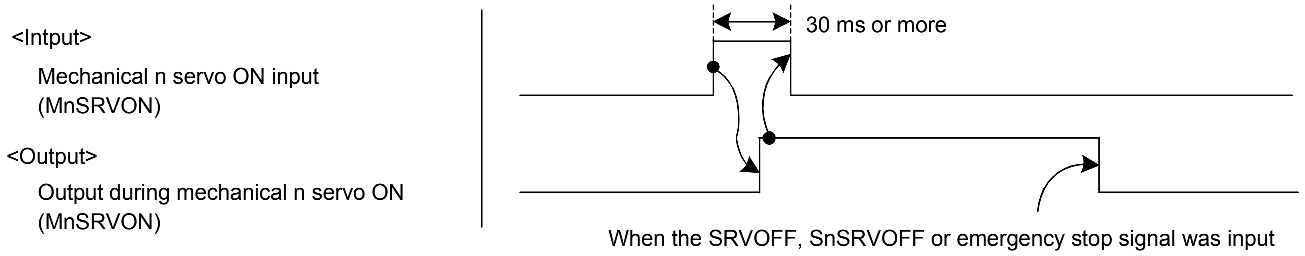
(24) SnSTOP (Slot n stop input/output during slot n aborting)


<Intput>
Slot n stop input (SnSTOP)
<Output>
Output during slot n aborting (SnSTOP)
30 ms or more
When the START, SnSTART or SLOTINIT signal was input

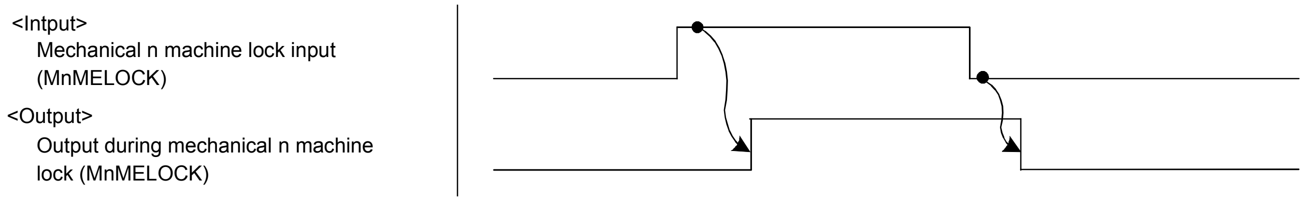
(25) MnSRVOFF (Mechanical n servo OFF input/mechanical n servo ON disable output)


<Intput>
Mechanical n servo OFF input (MnSRVOFF)
<Output>
Mechanical n servo ON disable output (MnSRVOFF)
30 ms or more
When the SRVON, SnSRVON or SRVON signal was input

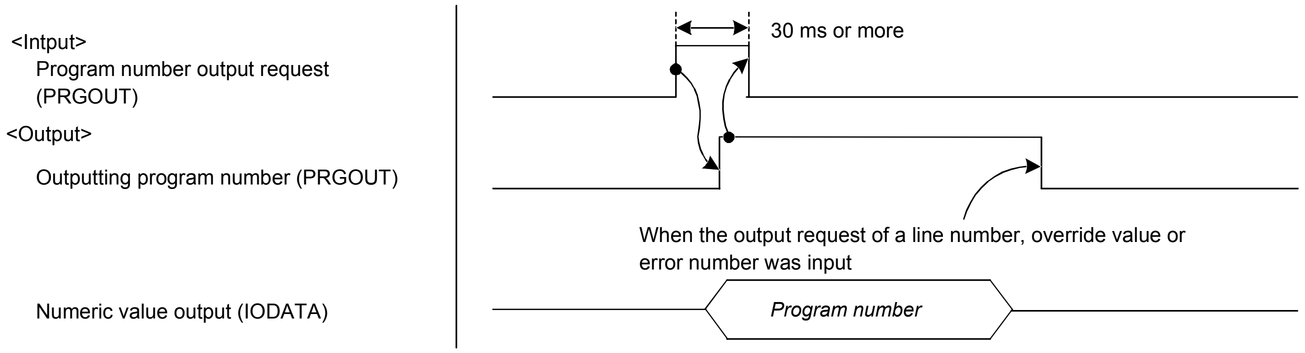
(26) MnSRVON (Mechanical n servo ON input/output during mechanical n servo ON)


<Intput>
Mechanical n servo ON input (MnSRVON)
<Output>
Output during mechanical n servo ON (MnSRVON)
30 ms or more
When the SRVOFF, SnSRVOFF or emergency stop signal was input

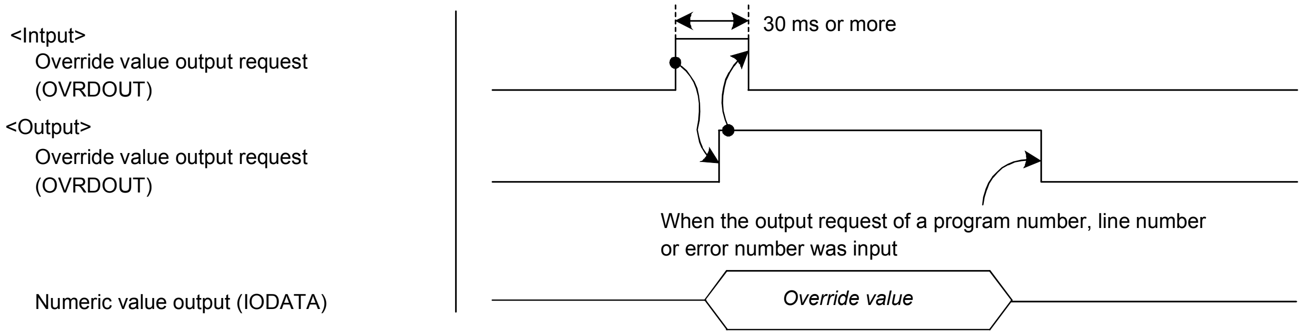
(27) MnMELOCK (Mechanical n machine lock input/output during mechanical n machine lock)


<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).

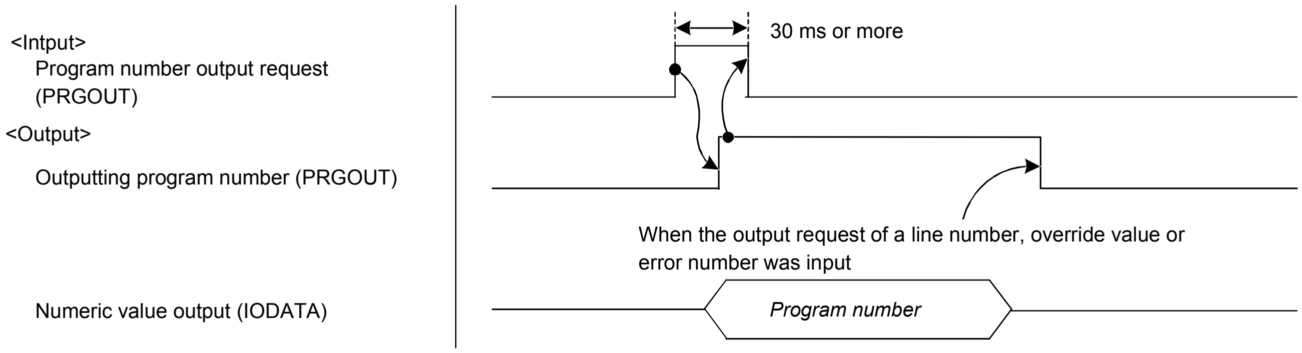

<Intput>
Program number output request (PRGOUT)
<Output>
Outputting program number (PRGOUT)
Numeric value output (IODATA)
30 ms or more
When the output request of a line number, override value or error number was input
Program number

(29) OVRDSEL (Override selection input)
* This is used together with the numeric value input (IODATA).

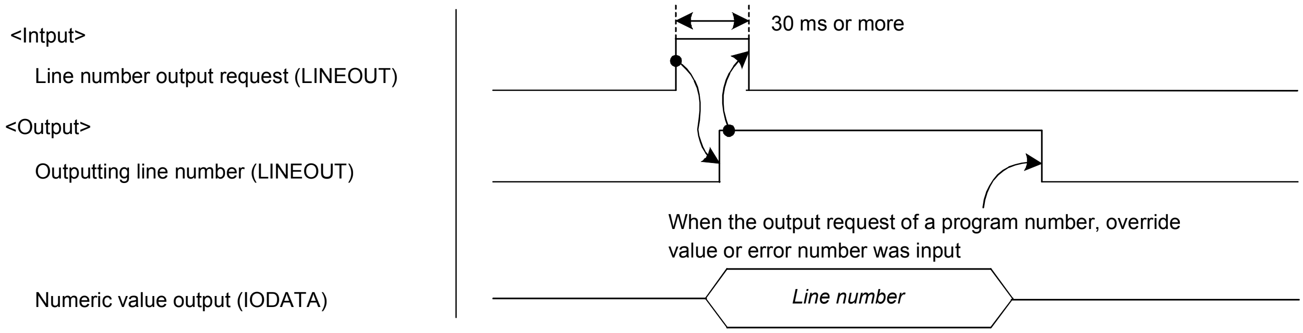

<Intput>
Override value output request (OVRDOUT)
<Output>
Override value output request (OVRDOUT)
Numeric value output (IODATA)
30 ms or more
When the output request of a program number, line number or error number was input
Override value

(30) IODATA (Numeric value input/numeric value output)
* This is used together with PRGSEL, OVRDSEL, PRGOUT, LINEOUT, OVRDOUT or ERROUT.

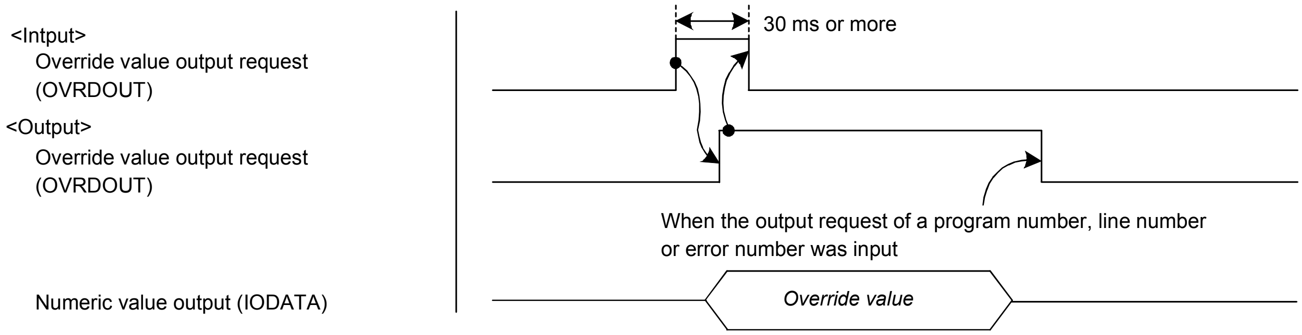
(31) PRGOUT (Program number output request input/outputting program number)
* This is used together with the numeric value output (IODATA).


<Intput>
Program number output request (PRGOUT)
<Output>
Outputting program number (PRGOUT)
Numeric value output (IODATA)
30 ms or more
When the output request of a line number, override value or error number was input
Program number

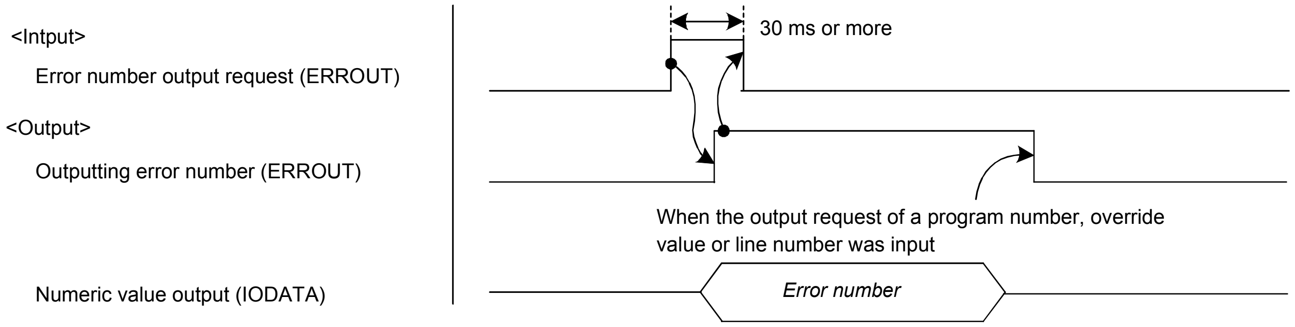
(32) LINEOUT (Line number output request input/outputting line number)
* This is used together with the numeric value output (IODATA).


<Intput>
Line number output request (LINEOUT)
<Output>
Outputting line number (LINEOUT)
Numeric value output (IODATA)
30 ms or more
When the output request of a program number, override value or error number was input
Line number

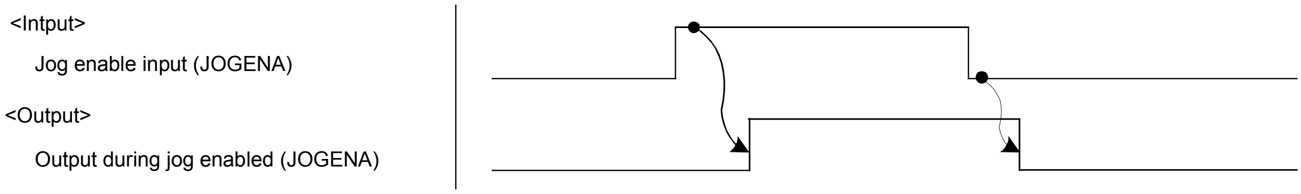
(33) OVRDOUT (Override value output request/outputting override value)
* This is used together with the numeric value output (IODATA).


<Intput>
Override value output request (OVRDOUT)
<Output>
Override value output request (OVRDOUT)
Numeric value output (IODATA)
30 ms or more
When the output request of a program number, line number or error number was input
Override value

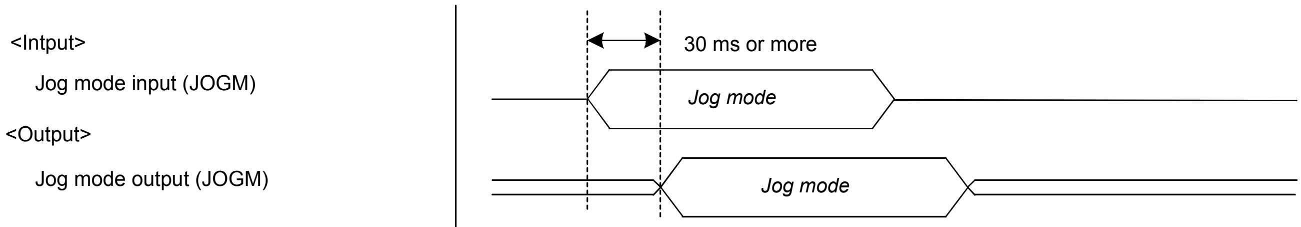
(34) ERROUT (Error number output request/outputting error number)
* This is used together with the numeric value input (IODATA).


<Intput>
Error number output request (ERROUT)
<Output>
Outputting error number (ERROUT)
Numeric value output (IODATA)
30 ms or more
When the output request of a program number, override value or line number was input
Error number

(35) JOGENA (Jog enable input/output during jog enabled)

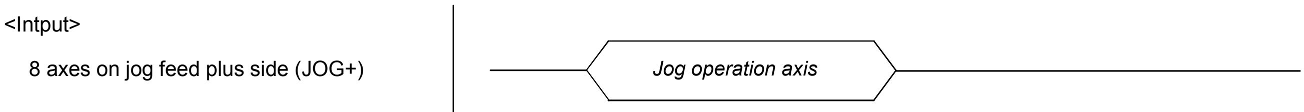


(36) JOGM (Jog mode input/jog mode output)



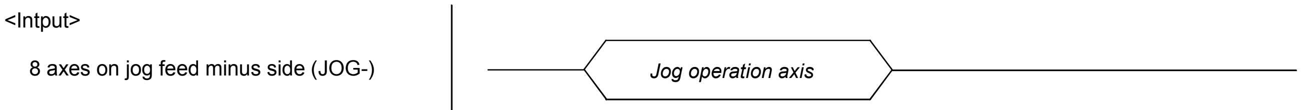
(Replies the setting value of the jog mode input signal with jog mode output.)

(37) JOG+ (Input for 8 axes on jog feed plus side)



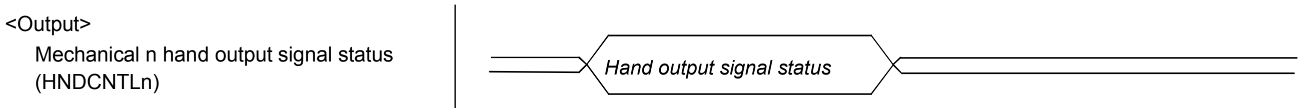
(Specify the axis that will perform jog operation in the plus direction.)

(38) JOG- (Input for 8 axes on jog feed minus side)



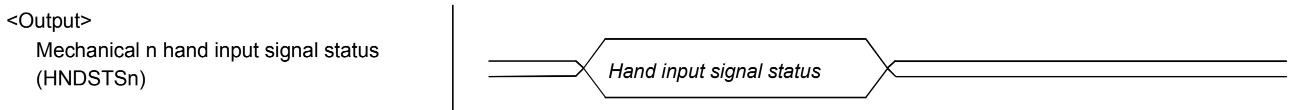
(Specify the axis that will perform jog operation in the minus direction.)

(39) HNDCNTLn (Mechanical n hand output signal status)


<Output>
Mechanical n hand output signal status (HNDCNTLn)
Hand output signal status

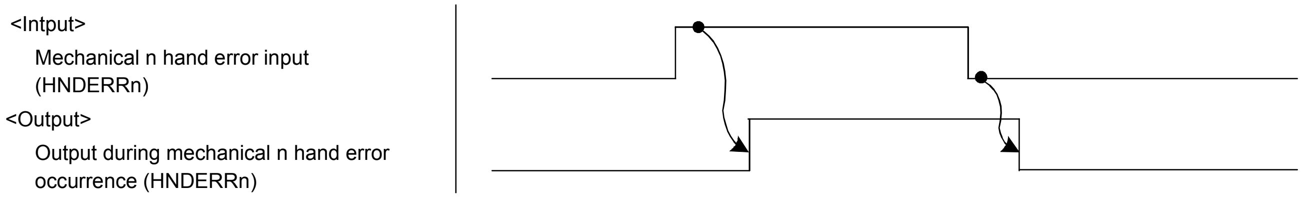
(Indicates the output signal status of the hand.)

(40) HNDSTSn (Mechanical n hand input signal status)

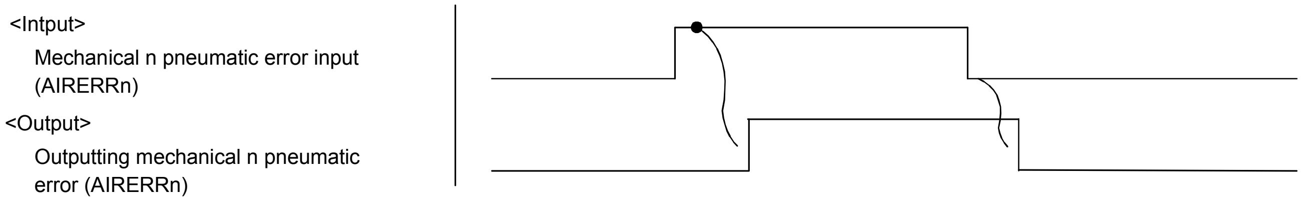


(Indicates the input signal status of the hand.)

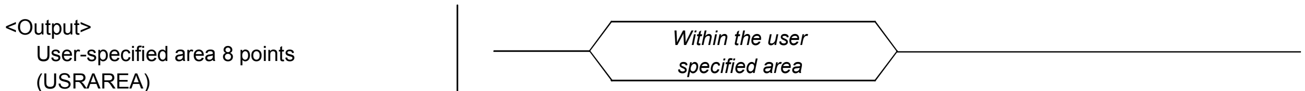
(41) HNDERRn (Mechanical n hand error input signal/output during mechanical n hand error occurrence)


<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)

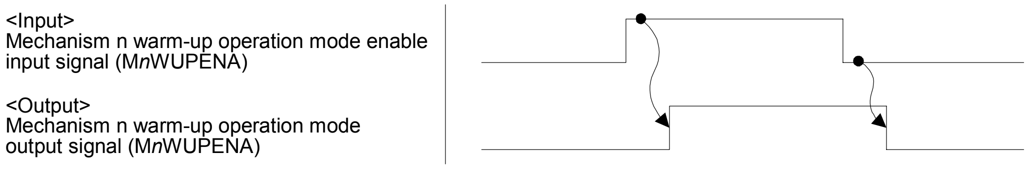

<Intput>
Mechanical n pneumatic error input
(AIRERRn)
<Output>
Outputting mechanical n pneumatic
error (AIRERRn)

(43) USRAREA (User-specified area 8 points output)

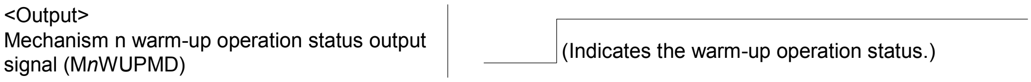


(Indicates that it is within the area specified by areas 1 though 8.)

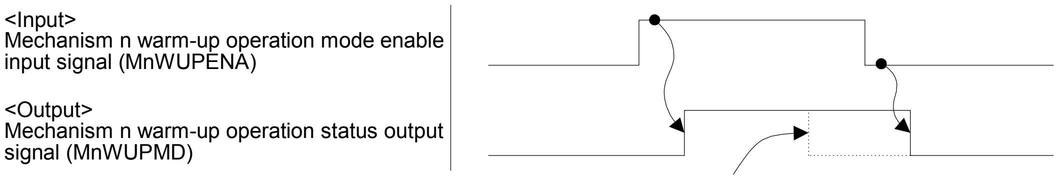
(44) MnWUPENA (Mechanism n warm-up operation mode enable input signal/ Mechanism n warm-up operation mode output signal)


<Input>
Mechanism n warm-up operation mode enable
input signal (MnWUPENA)
<Output>
Mechanism n warm-up operation mode
output signal (MnWUPENA)

(45) MnWUPMD (Mechanism n warm-up operation status output signal)


<Output>
Mechanism n warm-up operation status output
signal (MnWUPMD)
(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

6.5.2 Timing chart example

(1) External signal operation timing chart (Part 1)

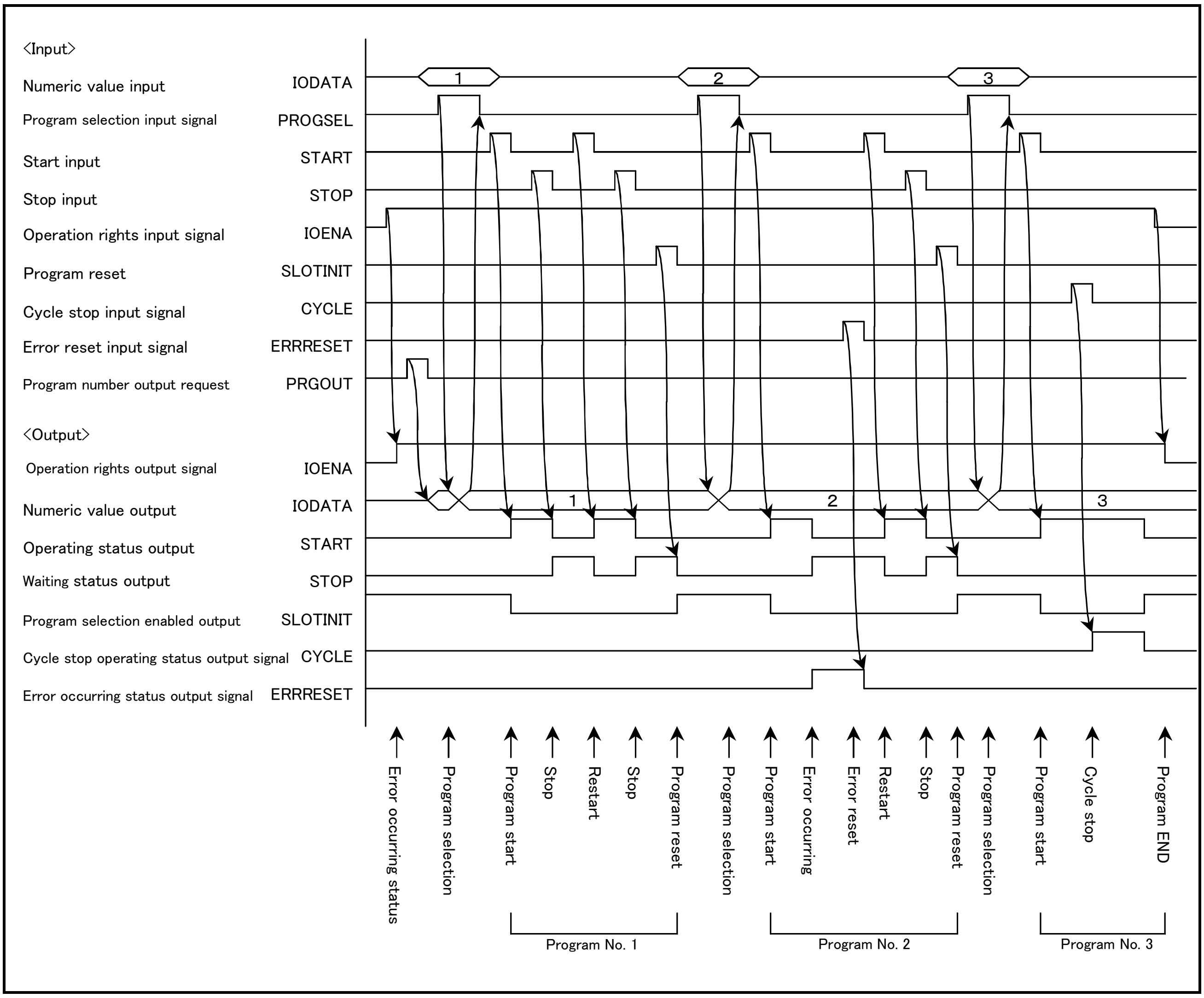

<Input>
Numeric value input IODATA
Program selection input signal PROGSEL
Start input START
Stop input STOP
Operation rights input signal IOENA
Program reset SLOTINIT
Cycle stop input signal CYCLE
Error reset input signal ERRRESET
Program number output request PRGOUT
<Output>
Operation rights output signal IOENA
Numeric value output IODATA
Operating status output START
Waiting status output STOP
Program selection enabled output SLOTINIT
Cycle stop operating status output signal CYCLE
Error occurring status output signal ERRRESET
1
2
3
Error occurring status
Program selection
Program start
Stop
Restart
Stop
Program reset
Program selection
Program start
Error occurring
Error reset
Restart
Stop
Program reset
Program selection
Program start
Cycle stop
Program END
Program No. 1
Program No. 2
Program No. 3

Fig.6-2:Example of external operation timing chart (Part 1)

(2) External signal operation timing chart (Part 2)

An example of timing chart the servo ON/OFF, selecting the program, selecting the override, starting and outputting the line No., etc., with external signals is shown in Fig. 6-3.

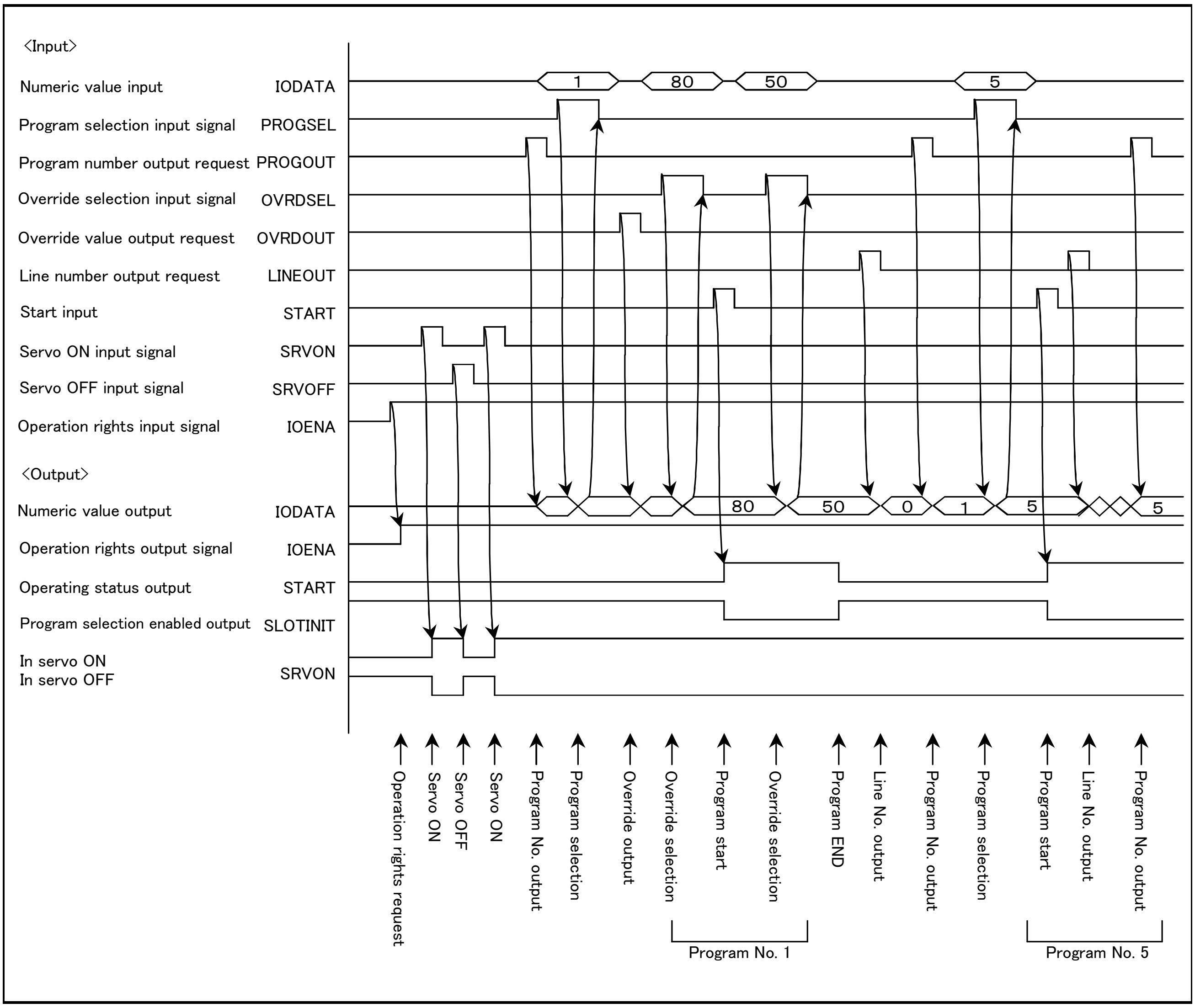

<Input>
Numeric value input IODATA
Program selection input signal PROGSEL
Program number output request PROGOUT
Override selection input signal OVRDSEL
Override value output request OVRDOUT
Line number output request LINEOUT
Start input START
Servo ON input signal SRVON
Servo OFF input signal SRVOFF
Operation rights input signal IOENA
<Output>
Numeric value output IODATA
Operation rights output signal IOENA
Operating status output START
Program selection enabled output SLOTINIT
In servo ON
In servo OFF
SRVON
1
80
50
5
80
50
0
1
5
5
Operation rights request
Servo ON
Servo OFF
Servo ON
Program No. output
Program selection
Override output
Override selection
Program start
Override selection
Program END
Line No. output
Program No. output
Program selection
Program start
Line No. output
Program No. output
Program No. 1
Program No. 5

Fig.6-3:Example of external operation timing chart (Part 2)

(3) Example of external operation timing chart (Part 3)

An example of the timing chart for error reset, general-purpose output reset and program reset, etc., with external signals is shown output in Fig. 6-4.

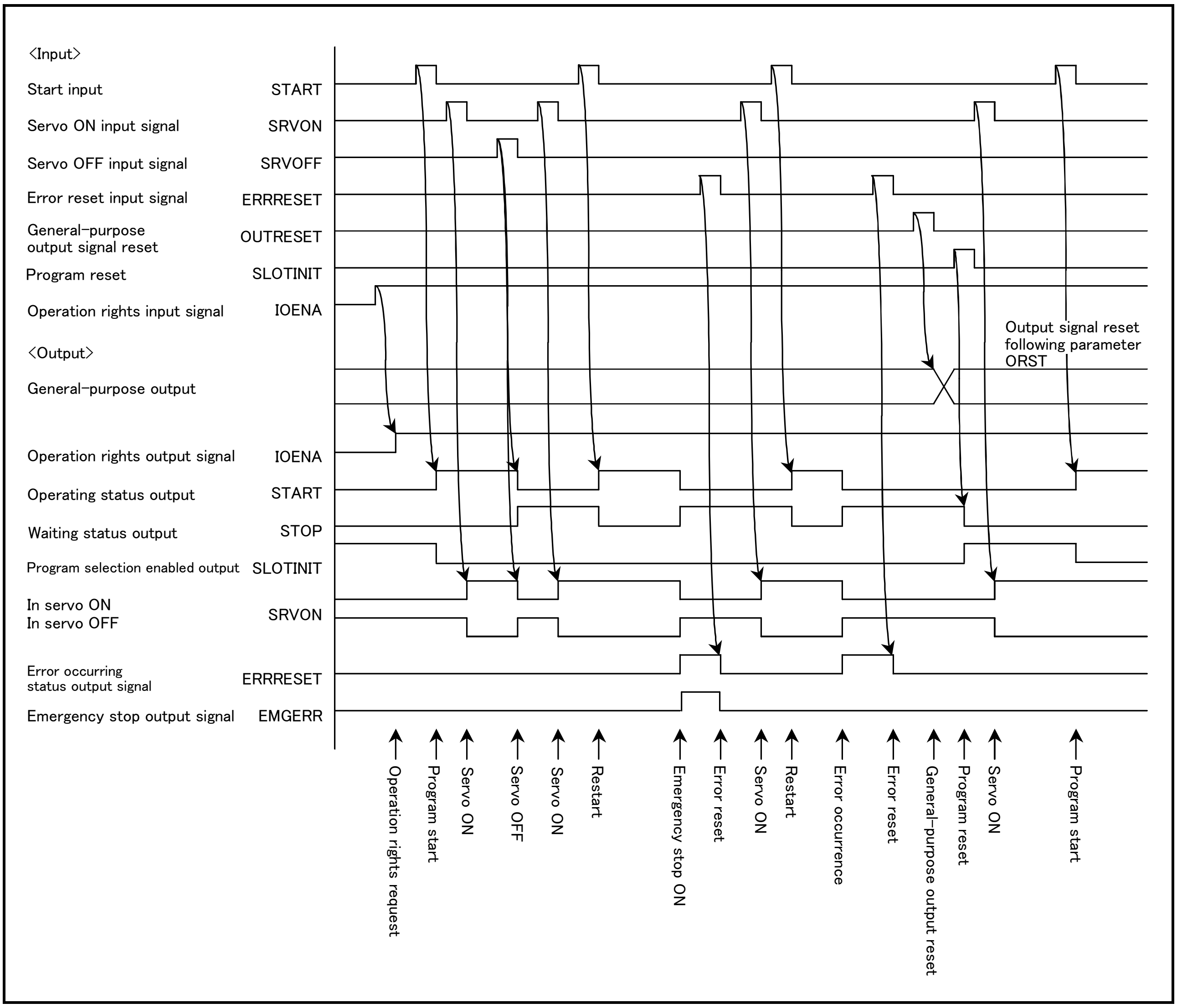

<Input>
Start input START
Servo ON input signal SRVON
Servo OFF input signal SRVOFF
Error reset input signal ERRRESET
General-purpose output signal reset OUTRESET
Program reset SLOTINIT
Operation rights input signal IOENA
<Output>
General-purpose output
Output signal reset following parameter ORST
Operation rights output signal IOENA
Operating status output START
Waiting status output STOP
Program selection enabled output SLOTINIT
In servo ON
In servo OFF
SRVON
Error occurring status output signal ERRRESET
Emergency stop output signal EMGERR
Operation rights request
Program start
Servo ON
Servo OFF
Servo ON
Restart
Emergency stop ON
Error reset
Servo ON
Restart
Error occurrence
Error reset
General-purpose output reset
Program reset
Servo ON
Program start

Fig.6-4:Example of external operation timing chart (Part 3)

(4) Example of external operation timing chart (Part 4)

An example of the timing chart for jog operation, safe point return and program reset, etc., with external signals is shown in Fig. 6-5.

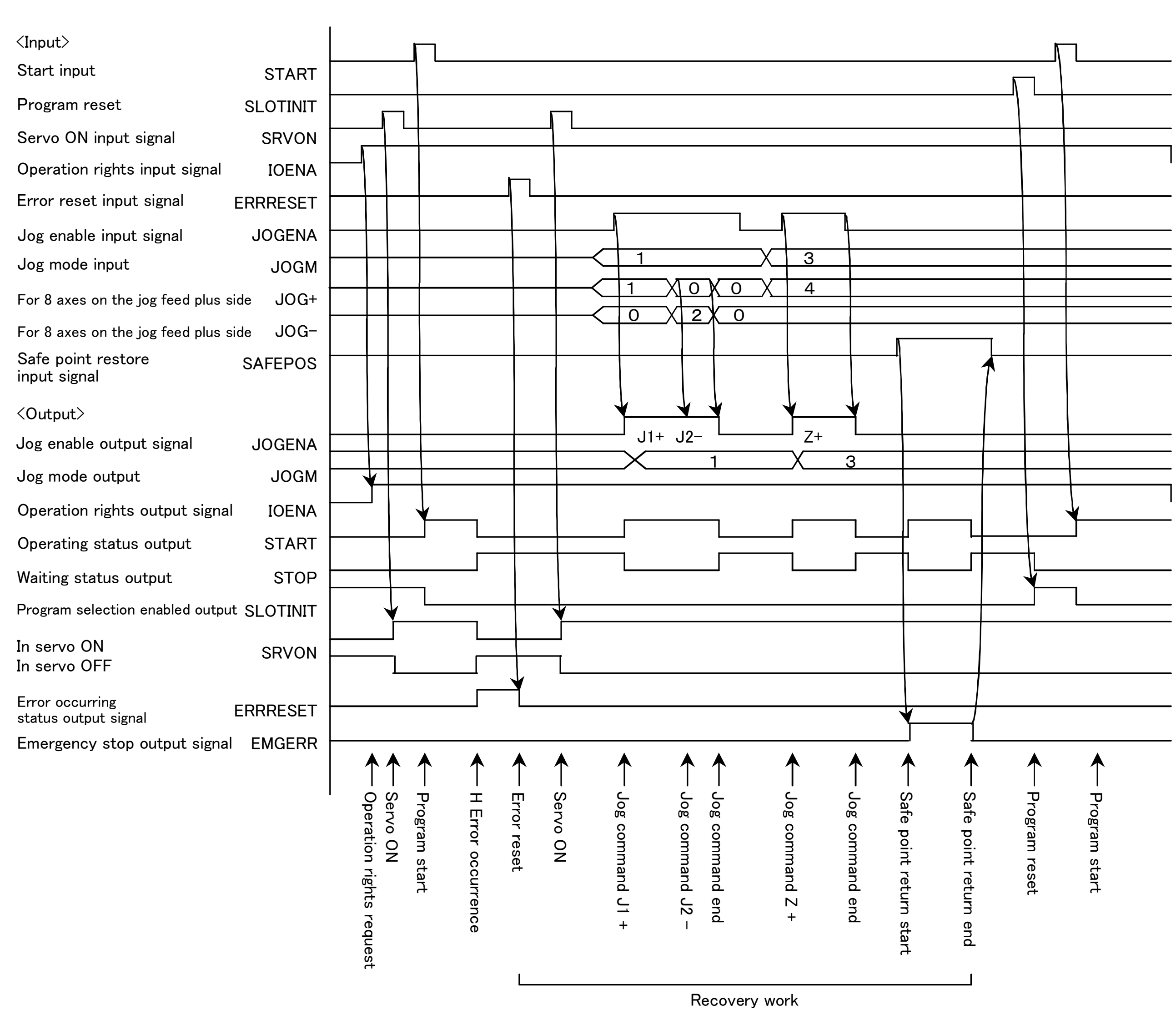

<Input>
Start input START
Program reset SLOTINIT
Servo ON input signal SRVON
Operation rights input signal IOENA
Error reset input signal ERRRESET
Jog enable input signal JOGENA
Jog mode input JOGM
For 8 axes on the jog feed plus side JOG+
For 8 axes on the jog feed plus side JOG−
Safe point restore input signal SAFEPOS
<Output>
Jog enable output signal JOGENA
Jog mode output JOGM
Operation rights output signal IOENA
Operating status output START
Waiting status output STOP
Program selection enabled output SLOTINIT
In servo ON
In servo OFF SRVON
Error occurring status output signal ERRRESET
Emergency stop output signal EMGERR
1
3
1
0
0
4
0
2
0
J1+ J2−
Z+
1
3
Operation rights request
Servo ON
Program start
H Error occurrence
Error reset
Servo ON
Jog command J1 +
Jog command J2 −
Jog command end
Jog command Z +
Jog command end
Safe point return start
Safe point return end
Program reset
Program start
Recovery work

Fig.6-5:Example of external operation timing chart (Part 4)

6.6 Emergency stop input

For wiring and other aspects of the emergency stop input, refer to the separate document entitled "Controller setup, basic operation, and maintenance."

6.6.1 Robot Behavior upon Emergency Stop Input

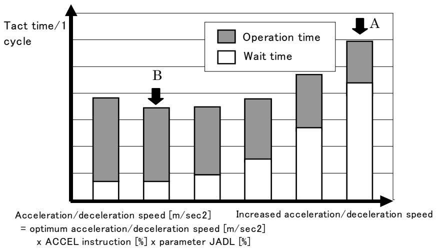
When an emergency stop signal is input while the robot is operating, the servo power supply is cut off by means of hardware control. The robot's tip path and stopping position after the input of an emergency stop signal cannot be specified. An overrun may occur depending on the robot speed or load condition of the tool.

7 Q & A

The following lists Q & A.

7.1 Movement

Q	A	Reference page
What are the features of this robot?	The features include the optimum acceleration/deceleration control, optimum speed control, compliance function, and multitask function.	Page 11, "2.3 Functions Related to Movement and Control"
I want to operate the robot tip in linear motion.	It is possible using the MVS instruction.	Page 62, "(2) Linear interpolation movement" Page 190, " MVS (Move S)"
I want to operate the robot tip in circular motion.	It is possible using the MVR, MVR2, MVR3 and MVC instruction.	Page 184, " MVR (Move R)" Page 186, " MVR2 (Move R2)" Page 188, " MVR3 (Move R 3)" Page 183, " MVC (Move C)"
I want to operate the robot tip in arch motion with ease.	It is possible using the DEF ARCH and MVA instruction.	Page 149, " DEF ARCH (Define arch)" Page 181, " MVA (Move Arch)"
I want to operate the robot in accordance with the direction of the hand. I want to move the robot tip based on relative value.	It is possible by setting a proximity/departure distance using the MVS instruction. It can be done via multiplication or addition of position variables.	Page 190, " MVS (Move S)" Page 84, "(2) Relative calculation of position data (multiplication)" Page 84, "(3) Relative calculation of position data (Addition)"
I want to skip the robot's singular points via linear interpolation.	It can be done via an 3-axes XYZ specification using the TYPE option of the MVS instruction. Although the 3-axes XYZ specification gives a linear path, the tip posture becomes unstable.	Page 190, " MVS (Move S)"
I want to control the robot in a pliable manner.	It can be done using the CMP TOOL instruction, etc. Pliableness can be specified by the hand orientation, in applicable robot axis units, etc.	Page 134, " CMP TOOL (Composition Tool)" Page 130, " CMP JNT (Comp Joint)" Page 132, " CMP POS (Composition Posture)" Page 137, " CMPG (Composition Gain)" Page 136, " CMP OFF (Composition OFF)"
I want to turn the robot's tip axis multiple times.	It is possible using the JRC instruction. It is possible. However, the tip axis cannot be turned continuously. The axis stops after each turn.	Page 177, " JRC (Joint Roll Change)"
I want to set a control point at the hand tip.	It is possible to set using the TOOL instruction or the parameter "MEXTL." If only one hand is used, setting via the parameter is recommended. Perform the necessary setting before performing teaching.	Page 217, " TOOL (Tool)" Page 324, "5.6 Standard Tool Coordinates"
I want to stop the robot for a specified period.	It is possible using the DLY instruction.	Page 160, " DLY (Delay)"
I want to verify the positioning of the robot.	The stopping pulse range can be specified using the FINE instruction. Positioning can also be performed easily with the DLY instruction.	Page 164, " FINE (Fine)" Page 160, " DLY (Delay)"

Q	A	Reference page
I want to increase the robot's tact time.	The execution time can be improved by using the following methods. 1) Perform continuous path operation using the CNT instruction. 2) Perform optimum acceleration/deceleration control using the OADL instruction. 3) Perform optimum speed control using the SPD instruction. <In the case of the RV-6S/12S series> In addition to items 1) through 3) above, the operation time can be shortened by setting a larger value in the optimum acceleration/deceleration adjustment rate parameter (JADL). In the RV-6S/12S series, the acceleration/deceleration speed is initialized to allow continuous operation Note1) with a short wait time (setting of B in the figure at right) Note2). This setting is suited for continuous operations that have a short tact time, such as palletizing work. Conversely, if quick operations (short operation time) are required, such as L/UL work on machined parts, the acceleration/deceleration speed can be increased by initial setting (setting of A in the figure at right). However, please note that some setting values of acceleration/deceleration speed tend to cause overload and overheat errors. In such a case, extend the wait time, reduce the acceleration/deceleration speed, or decrease the operating speed.	Page 138, " CNT (Continuous)" Page 193, " OADL (Optimal Acceleration)" Page 213, " SPD (Speed)" Refer to "JADL" in Page 306, "5 Functions set with parameters" Page 257, "M_SETADL"  Fig. Relationship between Acceleration/deceleration Speed and Tact Time (Conceptual Drawing)
An excessive speed error occurs even in the optimum speed control mode.	An excessive speed error may occur depending on the posture and position of the robot. In such a case, lower the speed temporarily only in that segment using an OVRD instruction.	Page 184, "SPD" reference program Page 199, "OVRD (Override)"
I want to operate the robot without stopping at each point.	It is possible using the CNT instruction.	Page 138, " CNT (Continuous)"
I want to change the acceleration time and deceleration time.	Using the ACCEL instruction is possible for set.	Page 119, " ACCEL (Accelerate)"
I want to execute an automatic return to the safe point.	A simple safe-point return can be executed using the parameter "JSAFE" and the SAFEPS input signal. The robot may be moved along a specific safe path while controlling the current position, using the user-defined area function and ZONE instruction.	Refer to "JSAFE" in Page 306, "5 Functions set with parameters" Page 328, "5.8 About user-defined area" Page 304, " ZONE" Page 305, " ZONE 2"
I want to improve the path accuracy.	It may be improved by using a PREC instruction.	Page 201, " PREC (Precision)"
I want to reduce the robot vibration.	Vibration can be suppressed by setting a smaller acceleration/deceleration rate with the ACCEL instruction. Vibrations may also be reduced by using a PREC instruction.	Page 119, " ACCEL (Accelerate)" Page 201, " PREC (Precision)"

Q	A	Reference page
An overload error occurs.	This error occurs when the motor operates under severe conditions for more than the sustainable period of time. Lower the operating speed, acceleration/deceleration time and so forth. It is more effective to reduce the acceleration/deceleration time than stopping the robot using the delay timer.	Page 199, "OVRD (Override)" Page 213, "SPD (Speed)" Page 119, "ACCEL (Accelerate)" Page 250, "M_LDFACT" Page 257, "M_SETADL"
I want to limit the movement range of the robot.	It can be done using the joint movement range parameter "MEJAR," XYZ movement range parameter "MEPAR," free plane limit parameter, etc.	Refer to "MEJAR" and "MEPAR", etc. in Page 306, "5 Functions set with parameters"
How to open and close the hand?	The hand can be controlled using the HOPEN and HCLOSE instructions or M_IN (90n) and M_OUT (90n) signals. (n=0 to 7)	Page 171, " HOPEN / HCLOSE (Hand Open/ Hand Close)" Page 248, " M_IN/M_INB/M_INW" Page 254, " M_OUT/M_OUTB/M_OUTW"
I want to limit the movement range on an arbitrary plane.	It can be done using the free plane limit function.	Page 329, "5.9 Free plane limit"
Is the impact detection function installed?	Yes. It is installed in the RV-S/RH-S series.	Page 21, "3.2.9 Impact Detection during Jog Operation" Page 141, " COLCHK (Col Check)" Refer to "COL" in Page 306, "5 Functions set with parameters"
An excessive difference error occurs when the ambient temperature of the robot is low or when starting the robot after it has been stopped over an extended period of time.	When starting the robot at a low temperature or after it has been stopped over an extended period of time, an excessive difference error may occur due to a change in the viscosity of grease. In this case, operate the robot in actual production after performing a running-in operation at low speed.For this purpose, the warm-up operation mode is provided in the controller's software version J8 or later. By using this function, errors may be resolved also.	Refer to Page 355, "5.19 Warm-Up Operation Mode".

Note1) Continuous operation: A state in which the operation smoothly continues without overload and/or overheat error(s).

Note2) The optimum acceleration/deceleration adjustment rate (the rate which is applied to the acceleration/deceleration speed calculated by optimum acceleration/deceleration control: parameter JADL) is set below 100%. The setting value varies with models. Please refer to "JADL" in Page 306, "Table 5-1: List Movement parameter".

7.2 Program

Q	A	Reference page
I want to speed up the execution processing time of the program.	I want to speed up the execution processing time of the program.	Page 345, "5.18 About ROM operation/high-speed RAM operation function"
I want to execute two or more robot programs simultaneously.	It can be done using the multitask function.	Page 86, "4.2 Multitask function"
What are the precautions when using the multitask function?	See the reference pages.	Page 89, "4.2.4 Precautions for creating multitask program" Page 90, "4.2.5 Precautions for using a multitask program"
I want to call a program from within another program.	It is possible using the CALLP instruction.	Page 125, " CALLP (Call P)"
I want to reference common positions and variables from separate programs.	External system variables can be used. Alternatively, a user base program can be created to use user-defined external variables. Program external variables can be added in the controller's software version J1 or later. See the description of the PRGGBL parameter.	Page 104, "4.3.23 Program external variables" Page 105, "4.3.24 User-defined external variables"
I want to calculate the palette position.	It is possible using the DEF PLT and PLT instruction.	Page 157, " DEF PLT (Define pallet)" Page 200, " PLT (Pallet)"
I want to output and monitor signals, etc., while the robot is operating.	It is possible to use the DEF ACT or ACT instruction when multiple paths are monitored, or using the WTH or WTHIF instruction when a single path is monitored. The WTHIF allows for conditional judgment.	Page 146, " DEF ACT (Define act)" Page 121, " ACT (Act)" Page 221, " WTH (With)" Page 222, " WTHIF (With If)"
I want to monitor the robot to see if it is at a specified position and generate an error accordingly.	It can be done using the user-defined area function or ZONE instruction. A user error can be generated using the ERROR instruction.	Page 262, " M_UAR" Page 304, " ZONE" Page 305, " ZONE 2" Page 162, " ERROR (error)"
I want to perform RS-232C communication with an external PC, etc.	It is possible.	Page 337, "5.15 About the communication setting" Page 198, " OPEN (Open)" Page 128, " CLOSE (Close)" Page 175, " INPUT (Input)" Page 202, " PRINT (Print)" Page 195, " ON COM GOSUB (ON Communication Go Subroutine)"
I want to acquire the current position of the robot.	The data can be referenced using the P_CURR or J_CURR variable.	Page 268, " P_CURR" Page 234, " J_CURR"
I want to measure the execution time.	It can be measured in milliseconds using the M_TIMER variable. 10 M_TIMER (1) = 0 20 MOV P1 30 MVS P2 40 M1 = M_TIMER (1) 50 HLT To check the time in milliseconds, monitor the M1 variable after the program is stopped.	Page 260, " M_TIMER"
I want the program to generate an error and stop.	A user error can be generated using the ERROR instruction.	Page 162, " ERROR (error)"
I want to reset an error generated by another task in a multitask program.	It is possible to set the start condition for the task slot to ALWAYS and issue a RESET ERR instruction.	Page 87, "4.2.3 Operation state of each slot" Page 206, " RESET ERR (Reset Error)"

Q	A	Reference page
I want to use multi-task instructions, such as XRUN, in programs that are constantly executed.	You will be able to use them by changing the setting of the ALWENA parameter.	Refer to "ALWENA" in Page 318, "Table 5-4: List Program Execution Related Parameter"
I cannot directly execute the XRUN instruction from the T/B, etc.	You will be able to use them by changing the setting of the ALWENA parameter. Controller's software version J1 or later.	Refer to "ALWENA" in Page 318, "Table 5-4: List Program Execution Related Parameter"

7.3 Operation

Q	A	Reference page
I want to edit programs that are constantly executed.	To edit programs whose execution attribute is set to ALWAYS via the SLTn parameter, do so after canceling the ALWAYS attribute. ALWAYS programs cannot be edited since they are constantly executed. Change ALWAYS to START in the SLTn parameter, then turn off the controller's power and turn it back on, to stop the program from being constantly executed.	Page 24, "3.5.1 Creating a program" Refer to "SLTn" in Page 306, "5 Functions set with parameters"
I want to numerically correct the position learned via teaching.	It can be corrected on the T/B position screen.	Page 32, "(8) Correcting the MDI (Manual Data Input)"
I want to cancel the paused status of the program.	It can be done via a program-reset operation. This operation can be executed on the T/B operation panel or using an I/O signal.	Page 41, "(6) Resetting the program"
I want to check the program line by line.	It is possible to execute a step feed from the T/B.	Page 35, "(1) Step feed"
I want to start with automatic operation and switch to step feed in the middle to check the operation.	Use the HLT instruction to perform automatic operation, then switch to step feed via a T/B operation.	Page 38, "3.7 Automatic operation" Page 170, " HLT (Halt)" Page 35, "(1) Step feed"
I want to move the robot to the position learned via teaching.	Position jump can be executed from the T/B.	Page 30, "(6) Confirming the position data (Position jump)"
I want to copy, delete or rename the program.	They can be done on the T/B management screen.	Page 47, "(3) Copying programs" Page 49, "(5) Deleting a program" Page 48, "(4) Changing the program name (Renaming)"
I want to write-protect the program.	They can be done on the T/B management screen. They can be done on the T/B management screen or Personal Computer Support Software.	Page 46, "(2) Program protection function"
I want to write-protect only the position data.	They can be done on the T/B management screen. They can be done on the T/B management screen or Personal Computer Support Software.	Page 46, "(2) Program protection function"
I want to check the available memory space.	It can be confirmed on the T/B directory screen.	Refer to T/B directory screen in Page 45, "3.11 Operating the program control screen"
I want to monitor the I/O statuses.	It can be done on the T/B monitor screen. They can be done on the T/B management screen or Personal Computer Support Software.	Page 50, "(1) Input signal monitor" Page 51, "(2) Output signal monitor"
I want to monitor the contents of variables.	It can be done on the T/B monitor screen. They can be done on the T/B management screen or Personal Computer Support Software.	Page 52, "(3) Variable monitor"
I want to view the error history.	It can be done on the T/B monitor screen. They can be done on the T/B management screen or Personal Computer Support Software.	Page 53, "(4) Error history"
I want to release the brake of the robot.	It can be done on the T/B brake screen.	Page 57, "(4) Releasing the brakes"
I want to view the current position of the robot.	It can be done on the T/B current position screen.	Page 29, "(5) Registering the current position data"

Q	A	Reference page
I want to change the amount of fixed-dimension feed in jog operation.	It can be set using the parameters "JOGJSP," "JOGPSP" and "JOGSPMX."	Refer to "JOGJSP", "JOGPSP" and "JOGSPMX" in Page 306, "5 Functions set with parameters"
I want to the program to operate automatically upon turning on the power.	It can be done using the multitask function.	Page 332, "5.11 Automatic execution of program at power up"
I want to turn ON the servo forcibly when the robot is outside the movement range.	It can be done by canceling the LS.	Page 44, "3.9 Error reset operation"
I want to retain the 5-digit LED display on the operation panel even after changing over the key switch.	The OPDISP parameter can be used to switch the display in the controller's software version J1 or later.	Refer to "OPDISP" in Page 315, "Table 5-3: List Operation parameter"

7.4 External input/output signal

Q	A	Reference page
I want to check if the robot controller has started	It can be checked using the dedicated signal "RCREADY."	Refer to "RCREADY" in Page 371, "6.3 Dedicated input/output"
Is the operation right also needed to operate external signals?	The operation right is also needed to control the robot using external I/O signals. It is also necessary when operating the command signals, such as those used to "turn the servo ON" and "start." Safety-related signals, such as those used for stopping the operation and turning the servo OFF can be executed without operation right.	
I want to monitor the robot via external signals to see if it is at a specified position and generate an error accordingly.	It can be done using the user-defined area function.	Page 328, "5.8 About user-defined area" Refer to "USRAREA" in Page 371, "6.3 Dedicated input/output"
I want to select programs using external signals.	It can be done using the dedicated signals "PRGSEL" and "IODATA." Set the program number as a binary value in the IODATA signal (multiple bits), and read it via the PRGSEL signal.	Refer to "PRGSEL" and "IODATA" in Page 371, "6.3 Dedicated input/output"
I want to monitor a low battery.	It can be done using the dedicated signals "BATERR".Replace the battery without delay when this signal is turned ON.	Refer to "BATERR" in Page 371, "6.3 Dedicated input/output"
I want to execute a jog feed using external signals.	It can be done using the dedicated signals "JOGENA", "JOGM", "JOG+" and "JOG-".	Refer to "JOGENA","JOGM", "JOG+" and "JOG-" in Page 371, "6.3 Dedicated input/output"
I want to check the operation of a program or signal without operating the robot.	It can be done using the dedicated signals "MLOCK". When the operation is started after this signal turned ON, the robot will not move physically. Only the program will run.	Refer to "MLOCK" in Page 371, "6.3 Dedicated input/output"
I want to cancel the paused status of the program.	It can be done using the dedicated signals "SLOTINIT".	Refer to "SLOTINIT" in Page 371, "6.3 Dedicated input/output"
I want to use robot programs to implement the PLC functions.	It can be done using the multitask function. Use one program to perform operation-related programs. Use another program to operate signal-processing programs. Use common system variables or user-defined external variables to exchange information between the programs.	Page 91, "4.2.6 Example of using multitask" Page 104, "4.3.23 Program external variables" Page 105, "4.3.24 User-defined external variables"
I want to reset the L7730 error temporarily. (Abnormalities in the CC-Link data link)	It is possible with the parameter E7730.	Refer to "E7730" in Page 314, "Table 5-2: List Signal parameter".

7.5 Parameter

Q	A	Reference page
The parameter(s) I changed have not been taken effect.	After you change parameters, power OFF and then ON again.	
I want to operate the robot continuously from the power-OFF state.	It can be done using the Continuity function.	Refer to "CTN" in Page 306, "5 Functions set with parameters"
I want to set the hand mode that becomes effective when the power is turned ON.	It can be done using the parameters "HANDINIT" and "HANDTYPE."	Refer to "HANDINIT" and "HANDTYPE" in Page 306, "5 Functions set with parameters" Page 333, "5.12 About the hand type" Page 334, "5.13 About default hand status"
I want to share the RS-232C port located on the front side of the controller between an external device such as a PC or sensor on one hand and PC support software on the other.	Some protocol-related limitations apply. See the reference page.	Page 337, "5.15 About the communication setting"
I want to cancel the triggered buzzer.	It can be done using the parameters "BZR".	Refer to "BZR" in Page 306, "5 Functions set with parameters"

8 Collection of Techniques

This chapter is intended to provide various sample programs for entry-level to advanced robot programming. This chapter mainly consists of three sections: Entry-Level Edition, Intermediate Edition, and Advance Edition. Each of these three sections describes the following items:

8.1 Entry-Level Edition

8.1.1 Describing comprehensive programs

Since MELFA-BASIC IV inherits the specifications of the BASIC language, it is generally easy to understand. On the other hand, if large programs are written in MELFA-BASIC IV, they tend to be difficult to understand because all variables can be accessed from anywhere, and functions containing arguments and/or return values cannot be described. Isn't there any technique for writing comprehensive programs?

[Technique]

To describe comprehensive programs using a nonstructural language such as BASIC, follow rules (1) through (5) listed below. Writing programs according to these rules makes the programs easy to understand. However, these rules are just examples, so please modify them or add new rules according to your environment.

(1) Clarify the structure of a program.
(2) Set up a notation that is easy to read.
(3) Define variables that are effective in a program.
(4) Define arguments and return values in subroutines.
(5) Define variables and labels that are only effective in subroutines.

Each of the above rules is described below in detail.

(1) Clarify the structure of a program

As the first step to write a program, divide the program into multiple blocks as shown in the next page. By structuring the program like this, it not only makes the program comprehensive, but also makes it possible to immediately identify the location of a problem in the event of problem occurrence since the program is clearly segmented. Additionally, because a series of operations are segmented into blocks, they can be easily used in other programs.

1000 ' Work loader (main program) 1010 ' DATE:2002.04.01 VER 1.1 1020 ' (1)Change *** -> @@@ 1030 ' (2)Add $$$ 1040 '[Revision history] 1050 ' 2002.03.01 VER 1.0 1060 ' 2002.02.01 VER 0.7	<Program version> This block describes the information necessary for version management. It is recommended to use single-byte and alphanumeric characters so that the program version can also be checked by the Teaching pendant.
1070 ’=== Variable definition === 1080 DIM ML_DATA(3) 1090 DEF IO X1_REQ=BYTE,8,&H0F 1100 DEF IO Y1_ERR=BYTE,8,&H0F 1110 DEF POS PL_PFRAM 1120 DEF INTE ML_REQ	<Variable definition> This block describes variables such as array variables (DIM), signal variables (DEF IO) and special-use variables (DEF CHAR/INTE/POS/etc.).
1130 '=== Initialization === 1140 PL_PFRAM=FRAM(PT001,PT002,PT003) 1150 DEF ACT 1,X1_REQ<>0,GOSUB *S50ACT1 1160 ACT 1=M_ON 1170 OPEN "COM1:" AS #1 1180 ON COM(1) GOSUB *S60RECV 1190 COM(1) ON	<Initialization> This block describes the initialization that is performed only once when the program is started, which includes setting initial values in variables, interrupts, and communication ports.

1200 '=== Main routine ===
1210 *MAIN
1220   IF ML_REQ=1 THEN
1230   P00TMP=P_ZERO
1240   P00TMP.X=ML_DATA(1)
1250   P00TMP.Y=ML_DATA(2)
1260   P00TMP.Z=ML_DATA(3)
1270   MOV PL_FRAM * P00TMP
1280   ENDIF
1290   IF ML_REQ=&H0F THEN
1300   MX99ERNO=9100
1310   GOSUB *S99ALARM
1320   YL_ERR=MY99RET
1330   IF M_CYS=M_ON THEN END
1340 GOTO *MAIN
	<Main routine>
	The main routine consists of a section that is executed first as shown below when the program is started. It corresponds to cycle stop on the "IF" line.
	  *MAIN
	      :
	    IF M_CYS=M_ON THEN END
	  GOTO *MAIN
	In general, the program will be easier to read by calling subroutines created in function units in order to make the program as short as possible.

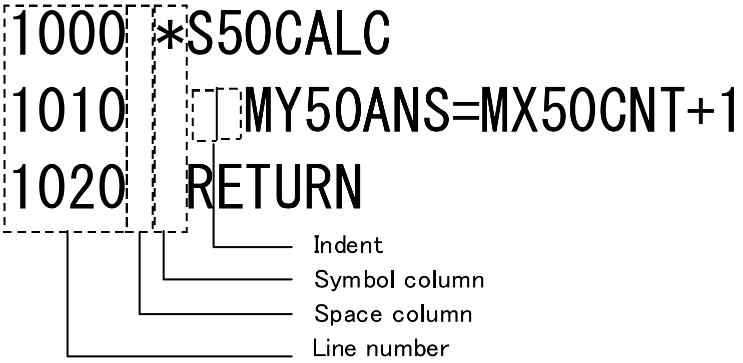
1350 '=== Subroutine50 === 1360 *S50ACT1 1370 ML_REQ=X1_REQ 1380 RETURN 0 1390 '=== Subroutine60 === 1400 *S60RECV 1410 COM(1) STOP 1420 INPUT #1,M60X,M60Y,M60Z 1430 ML_DATA(1)=M60X 1440 ML_DATA(2)=M60Y 1450 ML_DATA(3)=M60Z 1460 COM(1) ON 1470 *L60END 1480 RETURN 0 1490 '=== Subroutine99 === 1500 *S99ALARM 1510 ERROR MX99ERNO 1520 MY99RET=MX99ERNO 1530 *L99END 1540 RETURN	<Subroutines> A subroutine is composed of a section beginning with a label indicating a subroutine name and ending with a RETURN instruction; it is a program created in function units that unloads a work or analyzes received communication statements, for example.If a subroutine is called by a GOSUB instruction from the main routine or another subroutine, it returns to the line following the called line upon finishing processing. If a subroutine is called by an interrupt, it returns to the line called by RETURN 0/1 or to the line following the called line.

(2) Set up a notation that is easy to read

By giving some thought to a notation, easy-to-read programs can be created.

Indent	Indent in a subroutine, FOR to NEXT loop, IF to ENDIF block. Etc.

Allocate a blank column and a symbol column after a line number, and then describe text following these columns. The symbol column denotes a column in which an * (asterisk) that indicates a label or an ' (apostrophe) that indicates a comment is described. Also, indent two characters for IF, FOR, etc. Begin a line number with line 1000 for easier viewing without any character shift.



```
1000 *S50CALC
1010     MY50ANS=MX50CNT+1
1020   RETURN
```

Label jump	Jump by specifying a line number directly using GOTO or GOSUB

Line number jump is prohibited as it will interfere with the visibility of programs. Be sure to jump by providing labels.

(Example)

```
1150  DEF ACT 1,X1_REQ<>0,GOSUB *S50ACT1
1180  ON COM(1) GOSUB *S60RECV
1340  GOTO *MAIN
```

Comment	An explanatory statement that is described in a program using an REM instruction or an ' (apostrophe)

Please enter a comment in a section that seems to be difficult to understand by just looking at the program, such as the meaning of a program delimiter, the explanation about the input and output of a subroutine, and the meaning of a complex calculating expression.

(Example)

```
1330    IF M_CYS=M_ON THEN END    'Ends if the cycle stop request is ON

1500 '=== Subroutine 30 ===
1510 ' Move to the specified position.
1520 ' Input : MX30POS (Moving destination position number)
1530 ' Output : MY30RET (Normal end: 1, Abnormal end: 0)
1540 *S30MVPOS
```

(3) Define variables that are effective in a program

Although the values of the local variables in MELFA-BASIC IV can be referenced or rewritten from anywhere within a program, only the variables that conform to the following rules are defined as local variables in using this convention.

<table>
<tr><td>Local variables effective in a program</td><td>These variables can be used commonly between the main routine and subroutines. They are used for the result of a frame instruction, the output result of an interrupt function and so forth that can be referenced by anyone.</td></tr>
<tr><td colspan="2">1st character: Indicates the variable type
(M: numeric value, P: orthogonal position, J: joint position, C: character string, etc.).
2nd character: Indicates that it is a local variable (L).
3rd character: "_" (underscore)
4th to 8th characters: Any character string (5 characters)
(Example)
1110 DEF POS PL_PFRAM
1140 PL_PFRAM=FRAM(PT001,PT002,PT003)
1270   MOV PL_FRAM * P00TMP</td></tr>
</table>

<table>
<tr><td>Variables for teaching</td><td>These variables are subjects of teaching from the Teaching Box, or used as data such as offset values. The values of these variables can be referenced from anywhere in a program, but cannot be rewritten--i.e., reference only.</td></tr>
<tr><td colspan="2">1st character: Indicates the variable type (P, J).
2nd character: Indicates that it is a teaching/data variable (T or D).
3rd to 8th characters: Any character string (up to 6 characters)
(Example)
PT001 -> For storing the teaching result
PD001 -> For storing the offset amount in relation to PT001</td></tr>
</table>

<table>
<tr><td>Variables for signal I/O</td><td>These variables are used for reading and writing the signals defined by a DEF IO instruction. There are variables for inputs and outputs; input signals are reference only, and output signals are write only.According to the MELFA-BASIC IV specification, referencing output signal variables and writing to input signal variables are not allowed.</td></tr>
<tr><td colspan="2">1st character: Indicates the variable type (X: input signal, Y: output signal).
2nd character: Indicates that it is a local variable (L).
3rd character: "_" (underscore)4th to 8th characters: Any character string (up to 5 characters)
(Example)
1090 DEF IO X1_REQ=BYTE,8,&H0F
1100 DEF IO Y1_ERR=BYTE,8,&H0F</td></tr>
</table>

(4) Define arguments and return values in subroutines.

If GOSUB is used in MELFA-BASIC IV, it jumps to the specified label or line number, and then returns to the jump source location with a RETURN. In this convention explicit input and output (argument and return value) are defined in a subroutine for use.

However, recursive calls are not supported due to the language specification.

Subroutine names	A subroutine has a unique number, which is also reflected in the subroutine name. In addition, this number is added to variable names and label names that are effective in a subroutine.

1st character: Indicates a subroutine (S).
2nd and 3rd characters: Subroutine number (01 to 99)
4th to 8th characters: Any character string (up to 5 characters)
* "* (asterisk)" must be attached to the beginning of a label.
* The use of subroutine number 00 is not allowed since it is used by the main routine.
* Be sure to describe the explanation on the function and I/O variables of each subroutine.

(Example)

```
1360 *S50ACT1
1400 *S60RECV
1490 *S99ALARM
```

Arguments and return value of a subroutine	These variables are used for reading and writing the signals defined by a DEF IO instruction. There are variables for inputs and outputs; input signals are reference only, and output signals are write only.According to the MELFA-BASIC IV specification, referencing output signal variables and writing to input signal variables are not allowed.

1st character: Indicates the variable type (M, P, J, C, etc.).
2nd character: Indicates either an input or output (Input: X, output: Y).
3rd and 4th characters: Indicates the subroutine number.
5th to 8th characters: Any character string (up to 4 characters)

(Example)

```
1500 '=== Subroutine 30 ===
1510 ' Explanation of subroutine
1520 ' Input: MX30POS (explanation of input variable)
1530 ' Output: MY30RET (explanation of output variable)
1540 *S30MVPOS
1550    MOV PTPOS(MX30POS)
1560    MY30RET=MX30POS
1570  RETURN
```

(5) Define variables and labels that are only effective in subroutines

In MELFA-BASIC IV it is generally possible to access all variables and labels from anywhere within a program. However, since they can be written from anywhere, on the other hand, it becomes unclear when and where a certain variable was set when it was referenced. Therefore, we have fostered an idea of variables and labels that can only be used within a certain subroutine.

However, since variables and labels can be accessed from anywhere in terms of the language specification, please consider this as a rule in creating programs.

<table>
<tr><td>Effective labels in a subroutine</td><td>Label names used within a subroutine; no duplicate label names are used since each of them has a subroutine number.</td></tr>
<tr><td colspan="2">1st character: Indicates a label (L).
2nd and 3rd characters: Indicates a subroutine number.
4th to 8th characters: Any character string (up to 5 characters)
* "* (asterisk)" must be attached to the beginning of a label.
(Example)
1470 *L60END
1530 *L99END</td></tr>
</table>

<table>
<tr><td>Effective variables in a subroutine</td><td>Variable names used within a subroutine; no duplicate variable names are used since each of them has a subroutine number.</td></tr>
<tr><td colspan="2">1st character: Indicates the variable type (M, P, J, C, etc.).
2nd and 3rd characters: Indicates a subroutine number.
4th to 8th characters: Any character string (up to 5 characters)
(Example)
1230   P00TMP=P_ZERO
1420   INPUT #1,M60X,M60Y,M60Z</td></tr>
</table>

8.1.2 Managing program versions

As we create programs, it's getting harder to distinguish between old and new programs as the number of revised programs increases. Isn't there any good technique to track program versions?

[Technique]

In creating programs already created programs are often edited again in order to add specifications, modify functions, fix bugs, etc. However, when the programs are edited several times, it is sometimes hard to track which one is the latest program. Version management is thus necessary. There are several ways to manage versions. We will introduce comparatively general methods, and convenient methods when using the CRn-500 series.

(1) Writing the version in a comment

Writing the program editing history in the corresponding program is the easiest way. In addition, it is ideal to describe the version at the beginning of the program so that it can also be checked from the Teaching Box. The items that should be described include the program name, editing date, version and editing history outline. The following shows an example of description:

```
1000 'Work loader (main program)
1010 '  DATE:2002.04.01  VER 1.1
1020 '  (1)Change *** -> @@@
1030 '  (2)Add    $$$
1040 '[Revision history]
1050 '  2002.03.01  VER 1.0
1060 '  2002.02.01  VER 0.7
```

(2) Writing the version in the USERMSG parameter

Some robot systems may use multiple programs. In such a case, the entire robot system version is useful in addition to the program version. The robot system version can be checked by using the USERMSG parameter as long as there is a Teaching Box. This can be achieved by describing up to 64 single-byte characters of information including the system name, version, and date of creation in the USERMSG parameter in the specified format from the Maintenance Tool of the Personal Computer Support Software or from the Teaching Box.

8.1.3 Changing the operating speed in a program

How can we change the operating speed in a program when a work can be held more accurately if the speed is decreased while holding and releasing the work, when a work may be damaged unless the speed is decreased because the work may slightly interfere while unloading and inserting, or when it is necessary to perform a fine speed adjustment in order to shorten the cycle time?

[Technique]

Use the OVRD, JOVRD or SPD instruction.

OVRD: Effective regardless of the movement type. The speed display on the operation panel does not change.

JOVRD:Effective during joint movement.

SPD: Effective only during linear (circular, circular arc) movement.

8.1.4 Detecting fallen works while transporting

There are problems of fallen works in a robot system, which are caused by the decreased holding power due to air leak, incomplete holding due to an input of a damaged work, a drop of a work due to an interference with a peripheral device and so forth. How can we quickly detect any abnormality when such a problem occurs while transporting a work and stop the robot?

[Technique]

A conventional method uses interrupts to detect such events whose occurrences cannot be predicted. An interrupt is a method that monitors the status of signal or variables, and make a specific subroutine execute when a change occurs. This technique is very useful as it can be used in various applications, not limited to the detection of fallen works.

The method introduced here monitors signals that change when an abnormality occurs while transporting a work, and then sets up interrupts. Next, it turns on an interrupt at a timing when an abnormality should be detected, and turns off an interrupt at a timing when detection should be stopped.

To use interrupts, specify the "interrupt condition" and the "call destination when the condition is met" by using the DEF ACT instruction. It is advisable to declare interrupts at the beginning of a program unless a large number of interrupts will be used.

[Implementation example]

The following shows an example in which the robot is stopped immediately when a fallen work is detected while transporting works using a suction hand, and notifies an error according to each operation.

Point list		I/O signal list	
PT01	:Holding point	M_IN(8)	:Unloading allowed
PT02	:Front of the holding position	M_IN(9)	:Mounting allowed
PT03	:Front of the mounting position	M_IN(901)	:Work detection sensor (holding when ON)
PT04	:Mounting position		
PT99	:Retreat point		

Program	Comment
1000 DEF ACT 1,M_IN(901)=M_OFF GOTO *S50FALL,S 'Setting an interrupt for fallen works. 1010 '	Attach "S" to the argument, so that the robot decelerates and stops when it drops a work.
1020 PL_ZON1=P99-(10,10,10,1,1,1,10,10)(0,0) 1030 PL_ZON2=P99+(10,10,10,1,1,1,10,10)(0,0) 1040 ML_ZCHK=ZONE(P_CURR,PL_ZON1,PL_ZON2) 'Check on the retreat point. 1050 IF ML_ZCHK=0 THEN MOV PT99 'If not there, move to the retreat point. 1060 ' 1070 *MAIN	Calculate in advance because computation cannot be performed within the functions.
1080 OVRD 100 'Move to the Front of the holding position 1090 MOV PT02 'by increasing the speed. 1100 *L00WAITG 1110 IF M_IN(8)=M_OFF THEN *L00WAITG 'Wait until unloading is allowed. 1120 OVRD 30 'Move to the holding position 1130 MVS PT01 'by reducing the speed. 1140 HCLOSE 1 'Hold a work. 1150 DLY 0.1 'Wait for a while.	In principle, set the operating speed to 100% in programs that place importance in speed, and lower the speed only when necessary.
1160 ACT 1=M_ON 'Set the fallen work interrupt to ON.	The interrupt becomes valid from here.
1170 MVS PT02 'Set the fallen work interrupt to ON. 1180 OVRD 100 'Move to the front of the mounting position 1190 MOV PT03 'by increasing the speed. 1200 *L00WAITP 1210 IF M_IN(9)=M_OFF THEN *L00WAITP 'Wait until mounting is allowed. 1220 OVRD 30 'Move to the mounting position 1230 MVS PT04 'by reducing the speed.	
1240 ACT 1=M_OFF 'Set the fallen work interrupt to OFF. 1250 HOPEN 1 'Release the work. 1260 DLY 0.1 'Wait for a while. 1270 OVRD 100 'Move to the front of the mounting position 1280 MVS PT03 'by increasing the speed. 1290 GOTO *MAIN 'Go to the next work. 1300 ' 1310 *S50FALL 1320 ERROR 9102 'Error output 1330 END	Disable the interrupt before releasing the work.

8.1.5 Positioning works accurately

One of the main purposes of using a robot is to accurately place works at the target position. It is enough to just place works at approximate positions if a system is equipped with a positioning device. However, considering the cost, installation space and cycle time, in many cases, you just want to use a robot to accurately position works. How can we just use the robot to accurately position works?

[Technique]

For accurate positioning using just a robot, it is necessary for the robot to hold and release works at exact positions. To achieve this, use the FINE instruction. If there is a margin in the cycle time, it is simple and effective to insert the DLY instruction after the move instruction.

8.1.6 Awaiting signal ON/OFF during the specified number of seconds

It takes some time until the holding confirmation signal to turn ON depending on the characteristics of the hand after the hold command is issued, for example. How can we check signal ON/OFF during only the specified period of time?

[Technique]

Since no special instruction is provided for this purpose, we will introduce a subroutine that can achieve this function. By creating several subroutines that can be used for general purposes, the efficiency of programming can be improved, and easy-to-understand programs can be created.

[Implementation example]

This example loads/unloads works, assuming the case in which the hold confirmation signal cannot be obtained immediately although the hand issues the hold command and/or the release command to the suction sensor.

Point list	I/O signal list

PT01 :Holding point
PT02 :Front of the holding position
PT03 :Front of the mounting position
PT04 :Mounting position
	M_IN(901) :Vacuum pressure sensor (holding when ON)
	M_OUT(901) :Suction instruction (holding when ON)

Program	Comment
1000 *MAIN	
1010 OVRD 100 'Move to the front unloading position	
1020 MOV PT02 'by increasing the speed.	
1030 MVS PT01 'Move to the unloading position .	
1040 M_OUT(901)=M_ON 'Output the hold command.	
1050 MX50SIG=901	Set arguments for the subroutine.
1060 MX50SEC=3.0	
1070 GOSUB *S50WON 'Wait for the holding confirmation signal to turn ON.	
1080 IF MY50SKP=1 THEN GOTO *L40ALARM 'An error for detecting timeout	
1090 OVRD 10 'Move to the front unloading position	
1100 MVS PT02 'by reducing the speed.	
1110 '	
1120 OVRD 100 'Move to the front of the mounting position	
1130 MOV PT03 'by increasing the speed.	
1140 OVRD 10 'Move to the front of the mounting position	
1150 MVS PT04 'by reducing the speed.	
1160 M_OUT(901)=M_OFF 'Release the work.	
1170 MX51SIG=901	Set arguments for the subroutine.
1180 MX51SEC=3.0	
1190 GOSUB *S51WOFF 'Wait for the holding confirmation signal to turn OFF.	
1200 IF MY51SKP=1 THEN GOTO *L40ALARM 'An error for detecting timeout	
1210 OVRD 100 'Move to the front of the mounting position	
1220 MVS PT03 'by increasing the speed.	
1230 GOTO *MAIN	
1240 '	
1250 *L40ALARM 'Error handling	
1260 ERROR 9100	
1270 END	
1280 '	

```
1290 '---------- Wait for signal ON for specified number of seconds ----------
1300 ' IN:MX50SIG  Monitor signal
1310 '    MX50SEC  No. of seconds to monitor (s)
1320 'OUT:MY50SKP  0: normal end, 1: TIMEOUT
1330 *S50WON
1340    M_TIMER(1)=0                    'Timer reset
1350    MY50SKP=1                      'Output value reset
1360    M50SEC=MX50SEC * 1000          'Seconds Milliseconds
1370 *L50WON1
1380    IF M_TIMER(1)>M50SEC THEN *L50WON2 'End if timeout occurs.
1390    IF M_IN(MX50SIG)=1 THEN MY50SKP=0    'Set an output value if the signal turns
                                             ON.
1400    IF MY50SKP=0 THEN *L50WON2            'End as signal ON was confirmed.
1410    GOTO *L50WON1                        '
1420 *L50WON2
1430 RETURN
1440 '
```

It is useful to write a title and I/O specification in subroutines that are used for general purposes.

```
1450 '---------- Wait for signal OFF for specified number of seconds ----------
1460 ' IN:MX51SIG  Monitor signal No.
1470 '    MX51SEC  No. of seconds to monitor (s)
1480 'OUT:MY51SKP  0: normal end, 1: TIMEOUT
1490 *S51WOFF
1500    M_TIMER(1)=0                    'Timer reset
1510    MY51SKP=1                      'Output value reset
1520    M51SEC=MX51SEC * 1000          'Seconds Milliseconds
1530 *L51WOF1
1540    IF M_TIMER(1)>M51SEC THEN *L51WOF2 'End if timeout occurs.
1550    IF M_IN(MX51SIG)=0 THEN MY51SKP=0    'Set an output value if the signal turns
OFF.
1560    IF MY51SKP=0 THEN *L51WOF2           'End as signal OFF was confirmed.
1570    GOTO *L51WOF1
1580 *L51WOF2
1590 RETURN
1590 RETURN
```

8.1.7 Interlocking by using external input signals

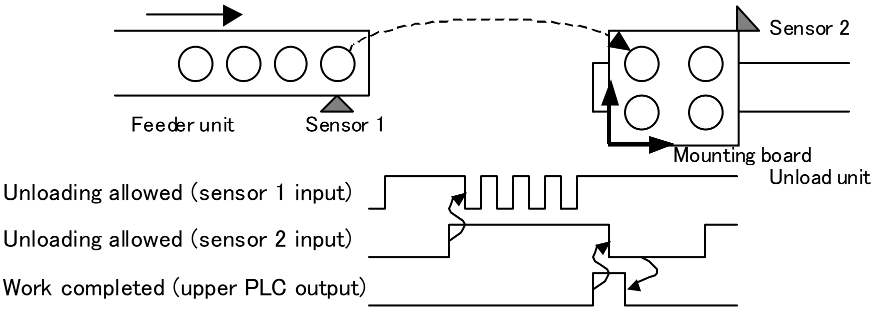
After a robot is incorporated into a system, the robot will be seldom operated by itself. How can we interlock the robot with peripheral units?

[Technique]

To achieve this, it is necessary to use the input and output of external signals. To simply turn ON/OFF I/O signals, use M_IN and M_OUT. Also, it is useful to use the DEF IO instruction as it allows to handle multiple signals together and name signals.

[Implementation example]

A system as shown below is assumed in this example. Also refer to the timing chart. In this system, the robot unloads works supplied by the feeder unit when there is a mounting board in the unload unit, and mounts four works on each mounting board. Once the mounting of the four works is complete, the robot outputs the work completion signal, and then the unload unit unloads the completed mounting board.


Sensor 2
Feeder unit
Sensor 1
Mounting board
Unload unit
Unloading allowed (sensor 1 input)
Unloading allowed (sensor 2 input)
Work completed (upper PLC output)

Point list		I/O signal list	
PTGET	:Holding position	M_IN(8)	:Sensor 1 (unloading from the feeder unit allowed)
PTPUT	:Calculated mounting position	M_IN(9)	:Sensor 2 (Mounting onto the board allowed)
PT99	:Retreat point		
PL_FRM	:Board origin position	M_OUT(8)	:Work completed
PL_POS(4)	:Mounting position on the board (relative coordinates)		

Program	Comment
1000 DIM PL_POS(4)	
1010 DEF IO XL_GET=BIT,8 'Allow unloading signal	Make the program easy-to-understand by assigning the signals to variables.
1020 DEF IO XL_PUT=BIT,9 'Allow mounting signal	
1030 DEF IO YL_OUT=BIT,8 'Work completion signal	
1040 PL_FRM=FRAM(PTO,PTX,PTY)	
1050 MOV PT99 'Return to the retreat point.	
1060 HOPEN 1 'Place the hand in the release state.	
1070 '	
1080 *MAIN	Wait until the allow mounting signal to turn OFF.
1090 IF XL_PUT=M_OFF THEN *MAIN 'Wait for the completion of preparing the unload unit.	
1100 M00NUM=0 'Reset the number of parts mounted on the board.	
1110 WHILE M00NUM<4 'Loop until four works are mounted.	Wait until the allow mounting signal to turn OFF.
1120 MOV PTGET,50 'Move to the front of the holding position.	
1130 *L00WAIT1	
1140 IF XL_GET=M_OFF THEN *L00WAIT1 'Wait if a part has not been supplied.	
1150 MOV PTGET 'Move to the holding position.	
1160 HCLOSE 1 'Hold a work.	
1170 MOV PTGET,50 'Move to the front of the holding position.	
1180 PTPUT=PL_FRM*PL_POS(M00NUM+1) 'Calculate the mounting position.	
1190 MOV PTPUT,50 'Move to the front of the mounting position.	
1200 MOV PTPUT 'Move to the mounting position.	
1210 HOPEN 1 'Release the work.	
1220 MOV PTPUT,50 'Move to the front of the mounting position.	Turn ON the work completion signal
1230 M00NUM=M00NUM+1 'No. of mounted parts + 1	
1240 WEND 'Move to the next part.	
1250 YL_OUT=M_ON 'Turn ON the work completion signal once 4 parts have been mounted.	Turn OFF the work completion signal.
1260 *L00WAIT2	
1270 IF XL_PUT=M_ON THEN *L00WAIT2 'Wait until the completed board is unloaded.	
1280 YL_OUT=M_OFF 'Turn OFF the work completion signal.	
1290 GOTO *MAIN 'Move to the next board.	

8.1.8 Sharing data among programs

It is necessary to reference the same data when we want to execute two or more programs together in order to obtain the results of calculations done by another program, or have another program retain position data. How can we accomplish this?

[Technique]

Wen sharing data among programs, it is not possible to directly reference the data of another program. Therefore, data must be transferred via program external variables (see Section 4.3.23 on page 101) or user-defined external variables (see Section 4.3.24 on page 102).

[Implementation example]

This example transfers data PT001() retained by subprograms to the main program by using program external variables or user-defined external variables.

Program	Comment
<Example of using system external variables> Main program 1010 CALLP "SUB1" 'Execute subprogram (SUB1.PRG). 1020 MOV P_100(1) 1030 MOV P_100(2) : 1010 CALLP "SUB2" 'Execute subprogram (SUB2.PRG). 1020 MOV P_100(1) 1030 MOV P_100(2) : Subprogram (SUB*.PRG) 1000 DIM PT001(5) 1010 FOR M01=1 TO 5 1020 M_100(M01)=PT001(M01) 'Copy own data into an external variable. 1030 NEXT 1040 END	The contents of position data can be completely switched by just calling a subroutine. Prepare the same program with different position data.
<Example of using user external variables> Base program (BASE.PRG) 1000 DIM POS P_POS(5) Main program 1000 DIM POS P_POS(5) 1010 CALLP "SUB1" 'Execute subprogram (SUB1.PRG). 1020 MOV P_POS(1) 1030 MOV P_POS(2) : 1010 CALLP "SUB2" 'Execute subprogram (SUB2.PRG). 1020 MOV P_POS(1) 1030 MOV P_POS(2) : Subprogram (SUB*.PRG) 1000 DIM POS P_POS(5) 1010 DIM PT001(5) 1020 FOR M01=1 TO 5 1030 M_POS(M01)=PT001(M01) 'Copy own data into an external variable. 1040 NEXT 1050 END	After installation, set "BASE.PRG" in parameter PRGUSR, and reset the power.

8.1.9 Checking whether the current position and the commanded position are the same

There are instances in which we want to determine whether the robot is located at the desired position, for example, when the robot should be located at the initial position at startup, when we want to check whether the robot has reached the target position, or when we want to change the path to return to the retreat point according to the position at startup. How can we determine this?

[Technique]

Since there are several ways to check positions, select the most suitable one according to use.

(1) Use a user area

It is the best way to use a user area to check the robot posture at startup, for instance, which is not changed frequently, of which the check area is small, or which is likely to be changed later. Up to eight user areas can be registered. This method have two advantages: it is constantly checked regardless of whether the program is started, and it can be changed easily by just changing parameters as it does not depend on the program. For more information, see Section 5.3, "About user-defined area" on page 280.

(2) Use the ZONE function and ZONE 2 function

Use these functions to check whether the robot is in the designated area in a program. ZONE checks whether the robot is in a cube, and ZONE 2 checks whether the robot is in a spherical or cylindrical body.

(3) Use the DIST function

If performing a simple check is sufficient--for example, checking whether the robot has reached the target position, it can be checked by calculating the distance between the target position and the current position (P_CURR) using the DIST function. However, be aware that angles are not considered.

Also, to check the deviation between the target position and the current position when using the compliance function, it is convenient to reference M_CMPDST.

(4) Compare each component of position variables

To check whether each position component is within the specified range, compare each component as in the program shown below. For example, the following program checks that the value of PX52A is near PX52B. The X component is +/-0.01mm, the Y component is +/-10.0mm, and the Z component is not checked. When the A, B and C components are within the range of +/-1.0 degree, "1" is set in output value MY52RET.

Caution) The A, B and C components are output in radians when elements are taken out. Using the RAD function, 1.0 degree is converted into radians and then used for checking.

Program

```
1000 'PX52A                    'Coordinate 1 to be checked
1010 'PX52B                    'Coordinate 2 to be checked
1020 'MY52RET                  '1 when PX52A is near PX52B, 0 for others
1030 *S52PSCHK                 'A subroutine to check position change
1040    MY52RET=0              'Initialize the return value.
1050    IF ABS(PX52B.X-PX52A.X)>0.01 THEN GOTO *L52END
1060    IF ABS(PX52B.Y-PX52A.Y)>10.0 THEN GOTO *L52END
1070    'No check is performed for Z components.
1080    IF ABS(PX52B.A-PX52A.A)>RAD(1.0) THEN GOTO *L52END
1090    IF ABS(PX52B.B-PX52A.B)>RAD(1.0) THEN GOTO *L52END
1100    IF ABS(PX52B.C-PX52A.C)>RAD(1.0) THEN GOTO *L52END
1110    MY52RET=1                'Set the return value again.
1120 *L52END
1130  RETURN
```

8.1.10 Shortening the cycle time (entry-level edition)

You wan to make the robot produce more. In many cases, it means to shorten the cycle time. What kinds of methods are available to easily reduce the cycle time?

[Technique]

Here, we will introduce basic check items to reduce the cycle time.

(1) Check whether speed control is appropriate.

Do you always revert the speed after decreasing it using the OVRD instruction and the SPD instruction? In writing programs, use the maximum operating speed in general, and decrease the speed only when necessary. If the speed is decreased, be sure to execute OVRD 100 and SPD M_NSPD (optimum speed control mode) and revert the speed to the maximum speed.

(2) Set up or reduce relay points.

In most cases when moving from one work point (for example, a holding position) to another work point (for example, a mounting position), it will pass through relay points. Have these relay points been set up at appropriate locations? Relay points should be reduced and optimized so that the target position can be reached with as few operations as possible. There is also another method that sets up a relay point just before the target position, even if it is possible to move to the target position by a single instruction, so that the operation can be performed at the maximum position to the very limit.

(3) Review the operation path in order to perform smooth path connection as much as possible.

Review the operation path currently in use or on the drawings. Use the CNT instruction if there is no interference with peripheral units when passing through relay points. This will prevent the speed at the relay points from decreasing; thus, improving the speed.

[Implementation example]

Three programs that perform the identical operation are shown successively on the next page. All of these programs move the robot from the starting position (TP03) to the target position (PT01) by passing through the front position (PT02).

The second and third blocks employ a technique to reduce the cycle time.

Block 1: Moves the robot from the starting position (TP03) to the target position (PT01) by passing through the front position (PT02).

Block 2: Moves the robot at the maximum speed to the very limit by providing a relay point in front of the target position (PT01).

Block 3: Moves the robot further by passing through inside relay points.

Each of the above three blocks measures the cycle time using the M_TIMER function, and stores the result in the corresponding M01, M02 and M03 variables. Check the differences of each cycle time.

Point list	I/O signal list
PT01 : Target position PT01 : Front position PT01 :Starting position	None

Program	Comment
1000 MOV PT03 'Move to the starting position.	
1010 '=== Block 1 ===	Regular description method
1020 M_TIMER(1)=0 'Timer reset	
1030 OVRD 100 'Move to the front position by	
1040 MOV PT02 'increasing to the maximum speed.	
1050 OVRD 10 'Move to the target position by	
1060 MVS PT01 'reducing the speed.	
1070 DLY 0.1 '	
1080 OVRD 100 'Move to the front position by	
1090 MVS PT02 'increasing to the maximum speed.	
1100 MOV PT03 'Move to the starting position.	
1110 DLY 0.1 'Measure the time after stopping	
1120 M01=M_TIMER(1) 'completely.	
1130 '=== Block 2 ===	Method that added the preceding position
1140 M_TIMER(1)=0 'Timer reset	
1150 OVRD 100 'Move to the front position by	
1160 MOV PT02 'increasing to the maximum speed.	
1170 MVS P01,-10 'Move to the front of the front position.	
1180 OVRD 10 'Move to the target position by	
1190 MVS PT01 'reducing the speed.	
1200 DLY 0.1 '	
1210 OVRD 100 'Move to the front position by	
1220 MVS PT02 'increasing to the maximum speed.	
1230 MOV PT03 'Move to the starting position.	
1240 DLY 0.1 'Measure the time after stopping	
1250 M02=M_TIMER(1) 'completely.	
1260 '=== Block 3 ===	Method that added CNT
1270 M_TIMER(1)=0 'Timer reset	
1280 OVRD 100 'Pass through inside by increasing to	
1290 CNT 1 'the maximum speed (enabled).	
1300 MOV PT02 'Move to the front position.	
1310 CNT 0 'Pass through inside (disabled).	
1320 MVS PT01,-10 'Move to the front of the front position.	
1330 OVRD 10 'Move to the target position by	
1340 MVS PT01 'reducing the speed.	
1350 DLY 0.1 '	
1360 OVRD 100 'Move to the front position by	
1370 MVS PT02 'increasing to the maximum speed.	
1380 MOV PT03 'Move to the starting position.	
1390 DLY 0.1 'Measure the time after stopping	
1400 M03=M_TIMER(1) 'Completely.	

8.2 Intermediate Edition

8.2.1 How to quickly support for the addition of types

There are not too many types and layers are not changed frequently, so it is not necessary to install a host computer. But, it is troublesome to do teaching every time. Is there any good way to change and use position data?

[Technique]

The program should be divided into the main program and data programs, and one data program should be created for each type. This method allows to store data for each type or group in a separate program. Use external variables to transfer data between data programs and the main program. Each data program contains position data and a program that writes that position data into an external variable. Thus, by starting a specific data program from the main program, the desired data can be written into an external variable.

[Implementation example]

In this example, the data programs ("WK1.PRG" - "WK63.PRG) corresponding to the type numbers obtained from the external input signals are started, and position data is transferred from each of the data programs to the main program via system external variable (P_100()).

Program	Comment

```
<Main program>
1000      DEF IO XL_SETNO=BIT,8              'Complete type number setting.
1010      DEF IO XL_PRGNO=BYTE,10,&H3F   'Receive type numbers 1 to 63.
1020     '
1030     *MAIN
1040        IF XL_SETNO=M_OFF THEN *MAIN   'Loop until the type numbers are received.
1050        C00PRG$="WK"+STR$(XL_PRGNO)   'Generate a program name.
1060        CALLP C00PRG$                         'Write data into an external variable.
1070        '
1080        FOR M00WK=1 TO 10                  'No, of work count
1090          P00TMP=P_100(M00WK)           'Generate position data.
1100          MOV P00TMP,50                     'Move to the front of the target position.
1110          MOV P00TMP                         'Move to the target position.
1120          DLY 1
1130          MOV P00TMP,50                     'Move to the front of the target position.
1140        NEXT M00WK                            'Move to the next work.
1150      GOTO *MAIN                              'Go to next layer data.
1160      END
```

```
<Data programs: "WK1.PRG" - "WK63.PRG">
1000      DIM PTDATA(10)                       'Data storage area
1010      FOR M01=1 TO 10                      'No. of work count
1020        P_100(M01)=PTDATA(M01)        'Copy data into an external variable.
1030      NEXT M01                  '
1040      END
```

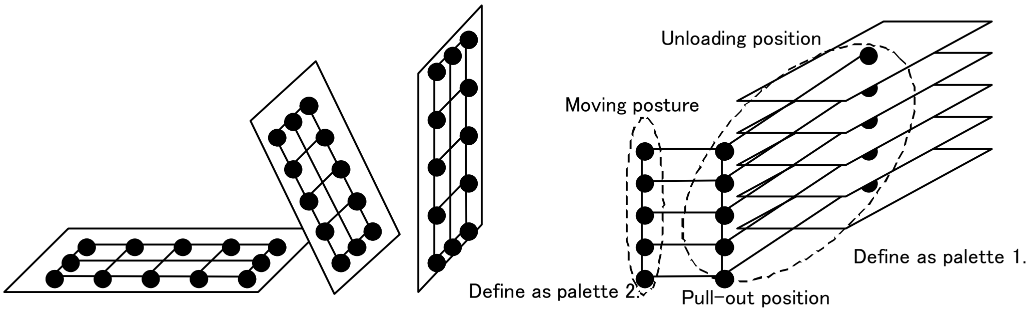
8.2.2 Convenient ways to use the pallet instruction

Generally, the pallet instruction is associated with taking out parts from lattice-like parts boxes or stacking boxes on a distribution pallet, but it can be used in various ways. We will explain convenient ways to use the pallet instruction here.

[Technique]

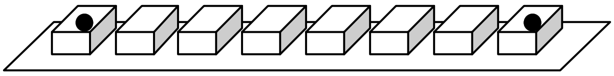
(1) The pallet instruction can be used on a free plane.

The pallet instruction is associated with a lattice on a plane placed on ground from its name, but it can actually be used on any flat planes.



The pallet instruction can define lattices as shown in the illustration above. This makes it possible, for example, to easily create coordinates such as the holding position and front position of works arranged in layers. Of course, it is possible to calculate these positions without using the pallet instruction, but using the pallet instruction can reduce the cycle time since calculations can be performed with a single instruction.

(2) The pallet instruction can be used in a row.



As shown in the illustration above, it is not necessary to teach all works when holding works arranged in a row. By using the pallet instruction, the rest of works can be interpolated as long as teaching is performed on the first and last works.

*For regular lattice-like pallets, see Page 71, "4.1.2 Pallet operation".

[Implementation example]

In the case of "works arranged in layers," the program can be streamlined by defining the work unloading position and pull-out position as pallet 1, and the pull-out position and moving posture as pallet 2. Also, in the case of "works arranged in a row," this can be accomplished by giving the same position variable twice to the argument specified by the DEF PLT instruction.

Program	Comment
<In the case of works arranged in layers> PT1:Bottom layer PT2:Top layer PT3:Front of the bottom layer PT4:Front of the top layer PT5:Moving posture of the bottom layer PT6:Moving posture of the top layer (for 5 layers)	
1000 DEF PLT 1,PT1,PT2,PT3,PT4,5,2,2 1010 DEF PLT 2,PT5,PT6,PT5,PT6,5,1,1 1020 PTMP1 = PLT 1,1 'Get the bottom layer position. 1030 PTMP2 = PLT 1,5 'Get the top layer position. 1040 PTMP3 = PLT 1,5+1 'Get the position in front of the bottom layer 1050 PTMP4 = PLT 1,5+5 'Get the position in front of the top layer. 1060 PTMP5 = PLT 2,1 'Get the moving posture of the bottom layer. 1070 PTMP6 = PLT 2,5 'Get the moving posture of the top layer.	
<In the case of works arranged in a row> P1: 1st position, P2: Nth position (when 10 works are lined up)	
1000 DEF PLT 1,PT1,PT2,PT1,PT2,10,1,1 1010 PTMP1 = PLT 1, 1 'Get the 1st position. 1020 PTMP2 = PLT 1,10 'Get the 10th position.	

8.2.3 How to write communication programs

How should we program to connect a robot with a personal computer and PLCs with a communication cable in order to issue work requests?

[Technique]

There are several ways to program, and various specifications can be supported ranging from simply using the INPUT instruction to wait until some sort of command is received to reading by multi-tasking in advance. When classified broadly, the following three types of methods can be used. The features of these methods are listed below.

Degree of programming difficulty	Method	Explanation
Low	INPUT instruction only	Programs can be written in an extremely simple manner. However, because the INPUT instruction does not end until something is received, the robot cannot do anything during that period. For more information about this method, see Page 337, "5.15 About the communication setting".
Medium	Communication interrupt	Reception processing is performed only when some sort of work instruction is received. Thus, other work can be performed as long as there is no work instruction. However, the robot operation will stop if a work instruction is received while in operation.
High	Multi-tasking	Because communication can be carried out concurrently with robot operation, it is possible to perform transport work while reading instructions in advance and analyzing complicated text received. Thus, consecutively requested operations can be executed one by one without stopping the robot.

[Implementation example]

Here, the following two types of methods are introduced among the three methods mentioned in Technique.

Communication interrupt method

Multi-tasking method

(1) Communication interrupt method

The following shows an example of a program that uses an communication interrupt. After receiving a command ("GET" or "PUT" in this example) from communication line 1, this program performs corresponding processing, and sends a completion command ("FIN" in this example) upon completion of processing.

```
                     Communication interrupt method                            Comment
1000   OPEN "COM1:" AS #1       'Open communication line 1 with file No. 1 (#1).
1010   ON COM(1) GOSUB *S98COMRC   'Setting for communication interrupt
1020   '
1030   ML_REQ=0                 'Clear the work request flag
1040   COM(1) ON                'Set to wait for communication.
1050  *MAIN                     'Beginning of the main loop
1060     GOSUB *S50CHECK        'Regular operation until reception
1070     SELECT ML_REQ          'If a work request arrives
1080     CASE 1
1090       GOSUB *S51GET        'Call a work routine
1100       GOSUB *S99FIN        'Work completed
1110       BREAK
1120     CASE 2
1130       GOSUB *S52PUT        'Call a work routine
1140       GOSUB *S99FIN        'Work completed
1150       BREAK
1160     END SELECT
1170     GOTO *MAIN
1180  '
1190  *S50CHECK                 'Regular operation                         Enable a communication inter-
1200     COM(1) ON              'Set to wait for communication.            rupt while in regular operation
1210  'Describe regular work.
1220     COM(1) STOP            'Stop waiting for communication.
1230   RETURN
1240  '
1250  *S51GET                   'When "GET" is received
1260   ' Describe the work when "GET" is received.
1270   RETURN
1280  '
1290  *S52PUT                   'When "PUT" is received
1300   ' Describe the work when "PUT" is received
1310   RETURN
1320  '
1330  *S98COMRC                 'Communication interrupt callback
1340     INPUT #1,C98CMD$       'Load the character string received.
1350     ML_REQ=0               'Clear the request reception flag.
1360     IF C98CMD$="GET" THEN ML_REQ=1   'If "GET"
1370     IF C98CMD$="PUT" THEN ML_REQ=2   'If "PUT"
1380   RETURN 1
1390  '
1400  *S99FIN
1410     PRINT #1,"FIN"         'Send a completion report.
1420     ML_REQ=0               'Clear the work request flag.
1430   RETURN
```

(2) Multi-tasking method

The following programs are examples of communication using multi-tasking. After receiving a command ("GET" or "PUT" in this example) from communication line 1, these programs perform corresponding processing, and send a completion command ("FIN" in this example) upon completion of processing.

First, the main program is started in task slot 1, and the communication program in task slot 2. The communication program in task slot 2 is started with the startup condition = ALWAYS.

Checking as to whether or not there is a reception by the communication program and transmission requests from the main program to the communication program are done by using external variables. External variable M_100(10) is used to check whether or not there is a reception by the communication program, and external variable M_05 is used for transmission requests from the main program to the communication program. Considering cases when the communication program would receive a command again before the main program completes post-reception processing, the communication program uses a maximum of 10 reception buffers (external variable M_100(10)).

Multi-tasking method	Comment
<Main program>	
1000 *MAIN	
1010 IF M_01 = M_02 THEN GOTO *MAIN 'If a queue is empty, nothing is performed.	
1020 M_01 = (M_01 MOD 10) + 1 'Read index + 1	Read the read index after adding 1 to it.
1030 SELECT M_100(M_01)	
1040 CASE 1	
1050 GOSUB *S51GET 'Call a work routine.	
1060 M_05=1 'Request to send the end of work.	
1070 BREAK	
1080 CASE 2	
1090 GOSUB *S52PUT 'Call a work routine.	
1100 M_05=1 'Request to send the end of work.	
1110 BREAK	
1120 END SELECT	
1130 GOTO *MAIN	
1140 '	
1150 *S51GET 'Perform work based on the command received.	
1160 ' Describe processing when "GET" is received.	
1170 RETURN	
1180 '	
1190 *S52PUT 'Perform work based on the command received.	
1200 ' Describe processing when "PUT" is received.	
1210 RETURN	
<Communication program>	
1000 CLOSE #1 'Close communication line 1.	
1010 OPEN "COM1:" AS #1 'Open communication line 1 with #1.	
1020 ON COM(1) GOSUB *S98COMRC 'Set a communication interrupt.	
1030 '	
1040 M_01=1 'Clear the read index	An index for using M_100() as a reception queue.
1050 M_02=1 'Clear the write index	
1060 COM(1) ON 'Set to wait for communication.	
1070 *MAIN 'Beginning of the main loop	Check the send request (M_05) from the main program.
1080 IF M_05=1 THEN GOSUB *S99FIN 'Wait for a send request	
1090 GOTO *MAIN	
1100 '	
1110 *S98COMRC '	Subroutine for communication interrupts
1120 INPUT #1,C98CMD$ 'Input the character string received.	
1130 IF C98CMD$="GET" THEN M_100(M_02)=1 ' IF "GET"	
1140 IF C98CMD$="PUT" THEN M_100(M_02)=2 ' IF "PUT"	
1150 M_02=(M_02 MOD 10) + 1 'Write index + 1	Add 1 to the write index after writing.
1160 RETURN 1	
1170 '	
1180 *S99FIN	
1190 PRINT #1,"FIN" 'Send a completion report.	
1200 M_05=0 'Clear the work request flag.	
1210 RETURN	

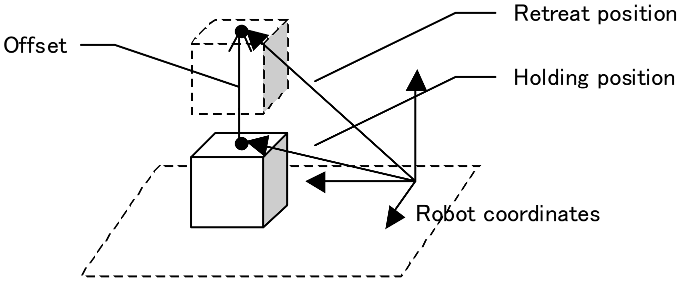
8.2.4 How to reduce teaching points

I wrote a program for a teaching operation, but the operation cannot be carried out smoothly because there are too many teaching points. Is there any way to reduce the number of teaching points?

[Technique]

For the robot to handle works and access peripheral units or move to remote points in an appropriate manner, there must be positions for the robot to pass through. However, it is not always necessary to perform teaching. In many cases, the number of teaching points can be reduced by setting relative positions* from certain positions and utilizing these positions.

*In contrast, teaching points (positions from the OP of the robot) are called absolute positions.



Please look at the illustration above. Two methods are available for the robot to hold and lift a work:

(1) Teach two points of the holding position and the retreat position.

(2) Teach the holding point, and use the position multiplied by an offset as the retreat position.

Method (2) requires a calculation, but it only needs one teaching point. Furthermore, this method only requires to teach one point of the holding position again, whereas method (1) requires two points to be taught when teaching again.

[Implementation example]

In the following example, position calculations are performed using the "+" operator in order to lift in the upper air in the robot coordinate system. To obtain the retreat position in the tool coordinate system when a 6-axial robot is used, for example, perform relative position calculations using the "*" operator, or perform calculations using the second argument (proximity distance) of the MOV or MVS instruction.

Program	Comment
<Teaching method> 1000 MOV PGET 'Move to the holding position (teaching). 1010 MOV PAPR 'Move to the retreat position (teaching). 1020 HLT	
<Offset method (robot coordinate system)> 1000 PAPR=PGET+POFS 'Calculate the retreat position. 1010 MOV PGET 'Move to the holding position (teaching). 1020 MOV PAPR 'Move to the retreat position (specifying a calculated value). 1030 HLT	
<Offset method (tool coordinate system)> 1000 PAPR=PGET*POFS 'Calculate the retreat position. 1010 MOV PGET 'Move to the holding position (teaching). 1020 MOV PAPR 'Move to the retreat position (specifying a calculated value). 1030 ' 1040 MOV PGET 'Move to the holding position (teaching). 1050 MOV PGET,-50 'Move to the retreat position (specifying a proximity distance). 1060 HLT	Note1)

Note1)Be careful as the directions of the tool coordinates are reversed between horizontal multi-joint robots and vertical multi-joint robots.

[Others]

This is a special example, but when using an RC-1000GHW*C or similar double-hand robot, the amount of teaching can be reduced by half if teaching is done for one hand instead of for both hands, and then both hands are supported by correcting the Z component.

Also, if a work position is located at constant intervals, the intermediate positions can be interpolated by using the pallet instruction.

8.2.5 Using a P variable in a counter, etc.

I want to initialize a program when it is installed on a controller. How can I initialize a counter that is not initialized when the power is turned off or the program is restarted in daily operations?

[Technique]

To achieve this, you can use an external variable, but an operation such as initializing the variable after installation is required. Here, we will introduce a technique to achieve this by using a position variable. Position variables are normally used to store coordinate values, but information other than coordinate values can also be stored in each component of a position variable. Unlike numeric variables and string variables, position variables can be saved as a part of a program just like instruction lines. By utilizing this characteristic, it is possible to initialize programs when they are installed, and use them as counters that will not be reset thereafter.

[Implementation example]

This example shows a program that counts how many times the power is turned on and off after installation.

(1) Add the following position data to the program.
PDATA = (0.0 , 0.0 , 0.0 , 0.0 , 0.0 , 0.0)(0.0 , 0.0)
X component: Initialization flag, Y component: Counter value

(2) Describe the following codes at the beginning or the program or in the initialization routine.

Program	Comment
<Main program> 1000 IF PDATA.X=1 THEN 1010 PDATA.Y=0 1020 PDATA.X=0 1030 ENDIF 1040 PDATA.Y=PDATA.Y + 1 1050 ' 1060 ' 1070 *LOOP 1080 'Main routine 1090 GOTO *LOOP	

(3) Write "1" in PDATA.X, and save or back up the program in a batch.
This clears the counter (PDATA.Y) to 0 once each time this program is loaded or restored, and the counter is counted up each time the power is turned on and off thereafter. However, the counter is counted up every time the beginning of the program is executed, so be sure not to return to the beginning of the program.

8.2.6 Getting position information when the sensor is on

Is there any way to read the coordinates of a robot when the sensor is turned on and off without stopping the robot so that the coordinates can be used for compensation operations? Also, how can we increase accuracy?

[Technique]

If you can stop the robot when sensing, it is desirable to use an interrupt; if you don't want to stop the robot, it is desirable to use a multi-task. If you will be using an interrupt, refer to the description of the DEF IO instruction for more information. Here, we will explain the method that uses a multi-task.

Two key points for reading the coordinates of the robot at the timing when the sensor is turned on and off are as follows:

(1) Match the sensor ON timing and the timing to read position information as much as possible.

Even when the program detects the sensor being turned on, it is meaningless if the robot advances before actually reading coordinates. The detection of sensor ON and the reading of coordinates should be carried out on the same line as much as possible. Because of this reason, reading the coordinates inside a loop is much more effective than using an interrupt.

(2) Check the sensor at high speed as much as possible.

It is obvious that more accurate data can be obtained by measuring 20 times than 10 times in a second. It is recommended to increase the priority of the sensing program as much as possible using the PRIORITY instruction at least during sensing.

[Implementation example]

In this example, the main program is started in task slot 1, and the sensing program is set as starting condition = ALWAYS and started in task slot 2. For the instruction from the main program to the sensing program, an external variable (M_01) is used.

A robot whose hand is equipped with two sensors (input signals 901 and 902) scans from PT01 to PT02 by a linear interpolation operation at 50 mm/sec, and reads the position coordinates where each of these two sensors are turned on. The current positions read are passed to the main program using external variables (P_01 and P_02). An error is issued if the robot finishes scanning with either of the sensors not turning on.

Program	Comment
<Main program>	
1000 M_01=0	
1010 MOV PT01 'Move to the front of the sensing position	
1020 SPD 50 'Reduce the speed to the sensing speed.	
1030 M_01=1 'Request to start sensing.	
1040 PRIORITY 1,1 'Status in which the sensing program is given priority	
1050 PRIORITY 10,2	
1060 MVS PT02 'Move to the target position.	
1070 PRIORITY 1,1 'Status in which the sensing program is given priority	
1080 PRIORITY 1,2	
1090 IF M_01=1 THEN 'End sensing	
1100 M_01=0 'if sensing has not been completed.	
1110 ERROR 9100 'Generate an error.	
1120 ELSE	
1130 'Processing using P_01 and P_02	
1140 PTMP = (P_01 + P_02) / 2	
1150 MVS PTMP 'Move to a mid-point where the sensor was turned on.	
1160 HLT	
1170 ENDIF	
<Sensing program>	
1000 WAIT M_01=1 'Wait for a sensing request.	
1010 MS1=0	
1020 MS2=0	
1030 *LOOP	
1040 IF MS1=0 AND M_IN(901) THEN P_01=P_CURR(1) 'Read the current position.	Execute sensor checking and coordinate reading on the same line.
1050 IF MS1=0 AND M_IN(901) THEN MS1=1 'Sensor 1 input completed	
1060 IF MS2=0 AND M_IN(902) THEN P_02=P_CURR(1) 'Read the current position.	
1070 IF MS2=0 AND M_IN(902) THEN MS2=1 'Sensor 1 input completed	
1080 IF (MS1=0 OR MS2=0) AND M_01=1 THEN *LOOP	
1090 M_01=0	
1100 END	

8.3 Advance Edition

8.3.1 Using the robot as a simplified PLC (sequencer)

Can the multi-task function of a robot be used to perform signal processing without using a PLC when configuring a robot system?

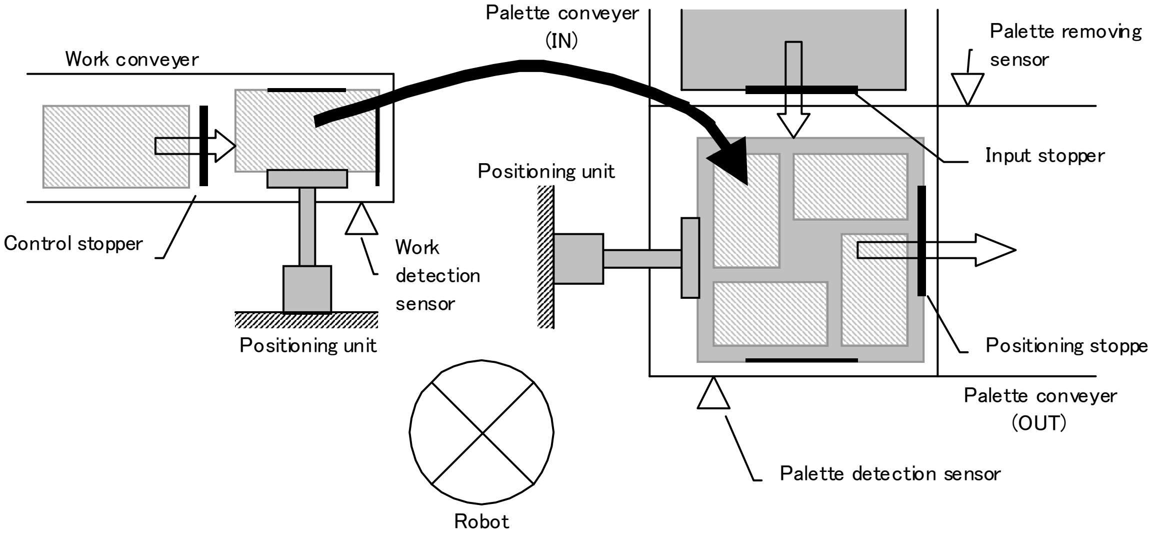
[Technique]

Although it cannot be said that the multi-task function is totally equivalent to the PLC, it can be used for simple processing such as controlling a conveyer or controlling lamp ON/OFF.
If the number of standard external I/O points is insufficient, extend the number of signal points by using an optional parallel I/O module.

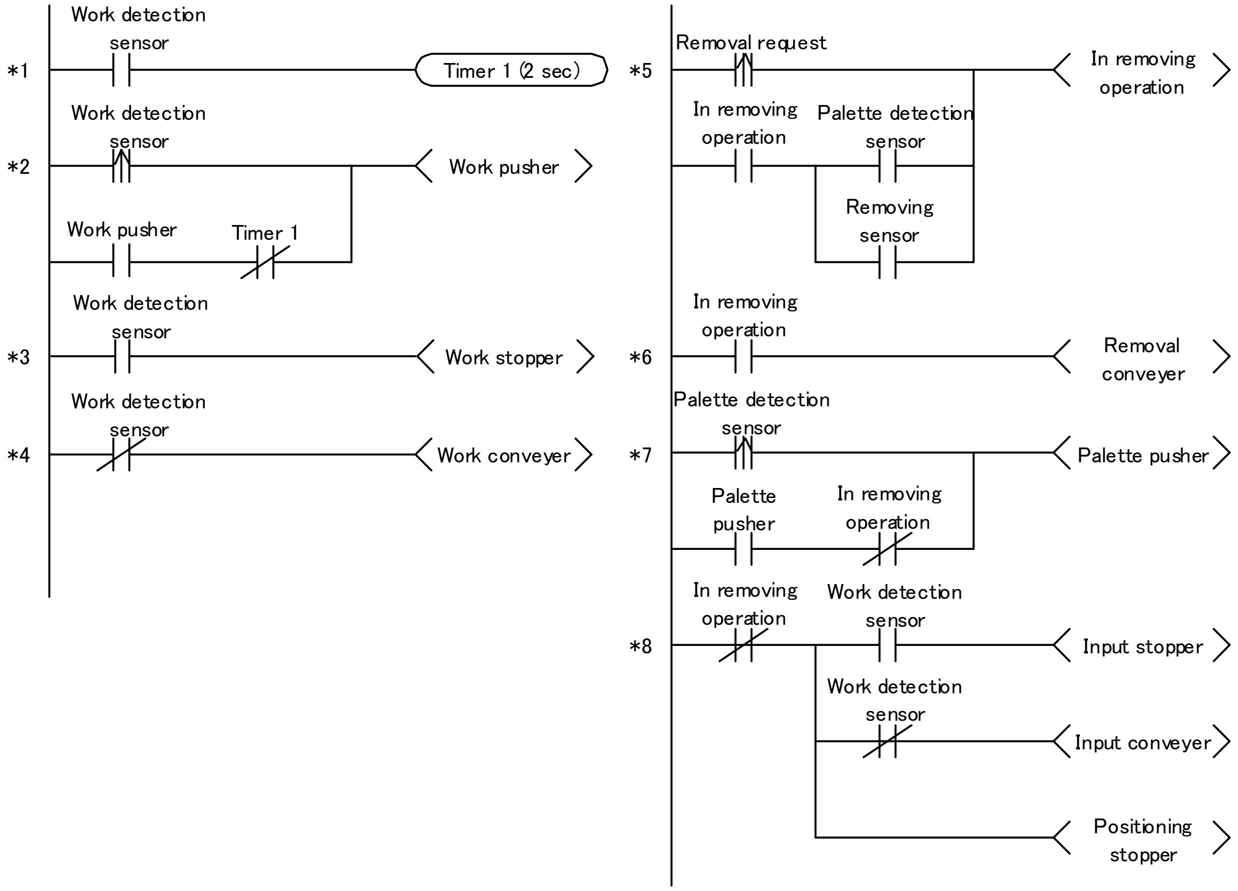
[Implementation example]

Create a program for processing signals, and start it with the ALWAYS attribute. The program will start at the same time when the robot's power is turned on, which will not stop even if an error occurs externally. Do not make any description to stop processing using the ERROR , HLT and/or WAIT instructions in this program.

As an example, a palletized system as shown in the illustration below is assumed, where a work conveyer and a palette conveyer are controlled by the robot. After thoroughly designing a signal processing program, program it in MELFA-BASIC IV.



Input signal list		Output signal list	
M_IN(8)	:Work detection sensor	M_OUT(8)	:Work conveyer ON/OFF
M_IN(9)	:Palette detection sensor	M_OUT(9)	:Work stopper ON/OFF
M_IN(10)	:Palette removing sensor	M_OUT(10)	:Work positioning unit ON/OFF
M_IN(11)	:Palette removal request signal	M_OUT(11)	:Palette input conveyer ON/OFF
		M_OUT(12)	:Palette removal conveyer ON/OFF
		M_OUT(13)	:Palette input stopper ON/OFF
		M_OUT(14)	:Palette positioning stopper ON/OFF
		M_OUT(15)	:Palette positioning unit ON/OFF


*1
Work detection sensor
Timer 1 (2 sec)
*2
Work detection sensor
Work pusher
Work pusher
Timer 1
*3
Work detection sensor
Work stopper
*4
Work detection sensor
Work conveyer
*5
Removal request
In removing operation
In removing operation
Palette detection sensor
Removing sensor
*6
In removing operation
Removal conveyer
*7
Palette detection sensor
Palette pusher
Palette pusher
In removing operation
*8
In removing operation
Work detection sensor
Input stopper
Work detection sensor
Input conveyer
Positioning stopper

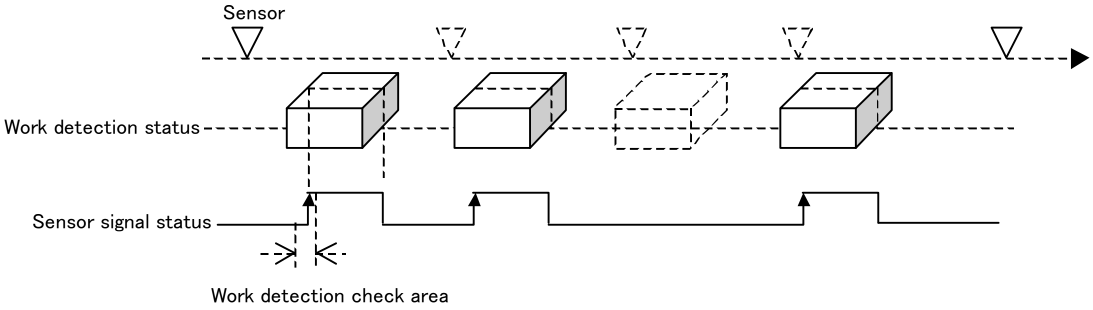
Program	Comment
1000 MTMLOOP=100*1000 'No. of seconds per loop (100 seconds)	Unit: msecBe aware of duplications with timers used by other programs.
1010 M_TIMER(1)=0 'Timer initialization	
1020 MTM01=0 'Timer variable initialization	
1030 MIN08=0 'A variable to retain the status of the work detection sensor	
1040 MIN09=0 'A variable to retain the status of the palette detection sensor	
1050 MIN11=0 'A variable to retain the palette removal request signal	
1060 MPLTOUT=0 'A variable for in palette removing operation	
1070 '	
1080 *LOOP	
1090 'N-second timer creation	Set a 100-second loop for overflow control of the timer. Clear the timer when an overflow is detected, and deduct the reference time from the timer variable.
1100 IF M_TIMER(1)>MTMLOOP THEN 'Clear the timer to 0 if the designated	
1110 M_TIMER(1)=0 'number of seconds has been exceeded.	
1120 IF MTM01+MTMLOOP>0 THEN MTM01=MTM01-MTMLOOP 'A timer variable	
1130 ENDIF	
1140 '	
1150 '##### Work conveyer #####	
1160 MUPPLS08=(M_IN(8) AND MIN08=0) 'Generate a rise of the detection sensor.	A rising edge is detected.
1170 MIN08=M_IN(8)	
1180 '=== Ladder *1 ===	Because the timer cannot be described in one line, two lines are used.
1190 IF MUPPLS08 THEN MTM01=M_TIMER(1) 'Reset the timer when a rising pulse is detected.	
1200 MTMOUT01=M_IN(8) AND (M_TIMER(1)-MTM01>2000) 'Set to ON after 2 seconds.	
1210 '=== Ladder *2 ===	
1220 IF MUPPLS08 OR (M_OUT(10) AND MTMOUT01=0) THEN M_OUT(10)=1 ELSE M_OUT(10)=0	
1230 '=== Ladder *3, *4 ===	
1240 IF M_IN(8) THEN 'When the work detection sensor is ON	
1250 M_OUT(9)=1 'Stopper ON	Be sure to set either ON or OFF.
1260 M_OUT(8)=0 'Conveyer OFF	
1270 ELSE 'When the work detection sensor is ON	
1280 M_OUT(9)=0 ' Stopper OFF	
1290 M_OUT(8)=1 ' Conveyer ON	
1300 ENDIF	
1310 '	
1320 '##### Palette conveyer #####	
1330 MUPPLS09=(M_IN(9) AND MIN09=0) 'Generate a rise of the detection sensor.	
1340 MIN09=M_IN(9)	
1350 MUPPLS11=(M_IN(11) AND MIN11=0) 'Generates a rise of a removal request.	
1360 MIN11=M_IN(11)	
1370 '=== Ladder *5 ===	It is designed as a retaining circuit.
1380 IF MUPPLS11 OR (MPLTOUT AND (M_IN(9) OR M_IN(10))) THEN MPLTOUT=1 ELSE MPLTOUT=0	
1390 '=== Ladder *6 ===	
1400 IF MPLTOUT THEN M_OUT(12)=1 ELSE M_OUT(12)=0	
1410 '=== Ladder *7 ===	
1420 IF MUPPLS09 OR (M_OUT(15) AND MPLTOUT=0) THEN M_OUT(15)=1 ELSE M_OUT(15)=0	
1430 '=== Ladder *8 ===	
1440 IF MPLTOUT=0 THEN 'When not in removing operation	
1450 IF M_IN(9) THEN 'When the palette detection sensor is ON	
1460 M_OUT(13)=1 'Input stopper ON	
1470 M_OUT(11)=0 'Input conveyer OFF	
1480 ELSE	
1490 M_OUT(13)=0 'Input stopper OFF	
1500 M_OUT(11)=1 'Input conveyer ON	
1510 ENDIF	
1520 M_OUT(14)=1 'Positioning stopper ON	
1530 ELSE	
1540 M_OUT(14)=0 'Positioning stopper OFF	
1550 ENDIF	
1560 GOTO *LOOP	

8.3.2 Implementing a mapping function

The robot moves to the designated location to fetch works, but sometimes operators process them or damaged works are removed in advance, so works are not always present. Although the robot can move to the next work if no work is present when the robot tries to fetch one, this method will extend the cycle time. How can we detect the presence of works in advance?

[Technique]

One available method is to detect the present of works in advance by installing an optical sensor at the tip of the robot's hand and scanning over works (this method is called mapping). However, in this case, because the processing speed of the robot controller will be limited, it may not be able to fetch works at the maximum speed.



[Implementation example]

It is desirable to use a multi-task and have the robot's operation performed by the main task and the reading from the sensor by the subtask. Try to write a program that can run at high speed by describing the sensor reading routine short and simple. Also, manipulate priority dynamically, and the priority of the subtask should be changed between normal operation and mapping execution.

In this example, the robot scans works lined up in a row using a non-contact sensor (using external input signal 901), checks whether each work is located where it should be (reference position), and only grabs a work if present (work holding processing not installed). For checking, whether or not works are present are determined by executing a calibration program once when all works are present at the time of system startup in order to obtain the reference position, and by comparing the reference position with the scan data obtained by a mapping operation performed before the robot grabs a work. When the difference between the reference data and the scan data is +/-10 mm or less, it is determined that a work is present.
Please execute this program according to the following procedure:

(1) Start the sensing program in slot 2 with the ALWAYS attribute.
(2) Start the calibration program when all works are present, and get the reference position.
(3) Start the main program.

Specification	I/O signal list
M_01 :Sensing request M_100() :Work detection result storage destination P_100() :Sensing result storage destination P_101() :Reference value storage destination	M_IN(8) :Work request signal M_OUT(8) :Work completion signal M_IN(901) :Sensor input

Program	Comment
<Main program (start in slot 1)> 1000 WAIT M_IN(8)=M_ON 'Wait for the work request signal to turn on. 1010 MOV PT1 'Move to the sensing starting point. 1020 M_01=M_ON 'Start sensing. 1030 SPD 30 'Reduce the speed to the sensing speed. 1040 MVS PT2 'Move to the sensing end position. 1050 DLY 0.1 'Wait until it stops completely. 1060 M_01=M_OFF 'End sensing.	The same codes as those of the calibration program should be used.
1070 SPD M_NSPD 'Revert the speed. 1080 MVS PT1 'Return to the sensing starting point. 1090 ' 1100 M01RANGE=10.0 'Work judgment range is 10 mm. 1110 FOR M01=1 TO 10 'Compare with 10 reference data. 1120 M_100(M01)=0 'Clear the detection flag. 1130 FOR M00=1 TO 10 'Compare with 10 scan data. 1140 M01DIST=DIST(P_101(M01),P_100(M00)) 'Determine that a work is present 1150 IF M01DIST<M01RANGE THEN 'if within the distance range between the scan value and reference value. 1160 M_100(M01)=1 'A work is determined to be present. 1170 M00=10 'Exit from the FOR loop. 1180 ENDIF 1190 NEXT M00 1200 NEXT M01 1210 ' 1220 FOR M02=1 TO 10 'There should be 10 works. 1230 IF M_100(M02)=0 THEN *L01SKIP 'Skip if there is no work. 1240 ' GOSUB *S50WKGET 'Work holding processing (not installed) 1250 *L01SKIP 1260 NEXT 1270 M_OUT(8)=M_ON 'Set the work completion signal to ON. 1280 WAIT M_IN(8)=M_OFF 'Wait until the work request signal turns OFF. 1290 M_OUT(8)=M_OFF 'Set the work completion signal to OFF. 1300 END	
<Sensing program (start in slot 2 with ALWAYS attribute)> 1000 *S01SENS 1010 PRIORITY 10,1 'Standard status 1020 PRIORITY 1,2 1030 M_01=M_OFF 'Set the sensing request signal to OFF. 1040 M01POS=1 'Clear the storage destination counter. 1050 WAIT M_01=M_ON 'Wait until the sensing request signal turns ON. 1060 PRIORITY 1,1 'Status in which the sensing program is given priority 1070 PRIORITY 10,2 1080 *L01LOOP 1090 IF M_01=M_OFF THEN *S01SENS 'End if the sensing request signal is OFF. 1100 IF M_IN(901)=M_ON THEN 'Save the current position 1110 P_100(M01POS)=P_CURR 'when the sensor turns ON. 1120 M01POS=M01POS+1 'Storage destination counter + 1 1130 WAIT M_IN(901)=M_OFF 'Wait until the sensor turns OFF. 1140 ENDIF 1150 GOTO *L01LOOP	The current position is stored in a system external variable. Use a user external variable if more than 10.
<Calibration program (start in slot 1)> 1000 MOV PT1 'Move to the sensing starting point. 1010 M_01=M_ON 'Start sensing. 1020 SPD 30 'Reduce the speed to the sensing speed. 1030 MVS PT2 'Move to the sensing end position. 1040 DLY 0.1 'Wait until it stops completely. 1050 M_01=M_OFF 'End sensing. 1060 SPD M_NSPD 'Revert the speed. 1070 MVS PT1 'Return to the sensing starting point. 1080 ' 1090 FOR M01=1 TO 10 'Copy the sensing result to 1100 P_101(M01)=P_100(M01) 'calibration data. 1110 NEXT M01 ' 1120 HLT	In some cases, it is desirable to scan several times and obtain the average.

8.3.3 Finding out executed lines

I am debugging the programs I created, and want to know which lines were executed when an error occurred? Is there any good method to find that out?

[Technique]

This controller employs the round-robin method by which all programs are executed one line at a time if the priority of the programs has not been changed. So, by using this method, all of the currently executed lines can be obtained by a multi-task. Look at lines 1050 and 1060 in the following program.

```
1050  PRIORITY 1, 1  ' Execute 6 lines of this program while
1060  PRIORITY 6, 2  ' executing one line of the main program.
```

This means that six lines of this program is executed in task slot 2 while one line of the main program is executed in task slot 1. These six lines correspond to lines 1070 through 1120.

[Implementation example]

To execute this example program, start this program in task slot 2 with the ALWAYS attribute, and start the main program in task slot 1. If you are starting this program in other than task slot 2, also change the priority setting on line 1060. The logs of executed line in task slot 1 are stored in variable MLN (100), and are overwritten from the beginning once the storage area becomes full.

Program	Comment
1000 DIM MLN(100) 'Storage area for 100 lines	
1010 FOR M01=1 TO 100 'Clear the storage area.	
1020 MLN(M01)=-1	
1030 NEXT	
1040 MIDX=1	
1050 PRIORITY 1,1 'Execute 6 lines of this program while	
1060 PRIORITY 6,2 'executing 1 line of the main program.	
1070 *LOOP	
1080 MBF=MIDX 'Save the previous ID.	
1090 MIDX=(MIDX MOD 100)+1 'Ring buffer ID	
1100 MLN(MIDX)=M_LINE(1) 'Save the current line No.	
1110 IF MLN(MBF)=MLN(MIDX) THEN MIDX=MBF 'Return if no line has been changed.	
1120 GOTO *LOOP	

8.3.4 Saving the status when an error has occurred

Is there any method to save the status when an error has occurred?

[Technique]

Robot programs have startup attributes (START, ALWAYS, ERROR), and the startup condition can be changed by using these attributes. In your case, you can create a program having the ERROR attribute that is started when an error has occurred, and save the status at the time of error occurrence.

[Implementation example]

Set the task slot that uses this program as follows:
Startup condition = ERROR (executed when an error occurs)
Operating mode = CYC (ends after executing one cycle.)
(See the Page 318, " SLT*".)
In this example, the information of past five errors is saved in variable PERINF (,). The following content are saved:

1) Number of the error occurred
2) Number of the line where an error occurred
3) Remaining distance to the target position
4) Position (orthogonal coordinates) of the robot when an error occurred

It is also possible to change the information to be saved depending on the error content (line 1110 and succeeding lines).
When using this program for the first time, verify that M_01 is "0." This clears the counter at the storage destination. Subsequently, every time an error occurs, this program is called and information is written into variable PERINF (,). The latest information is an element indicated by M_01, and is a ring buffer.

Program	Comment
<Get error information program> 1000 DIM PERINF(5,5) 'Allocate an area for saving past 5 errors. 1010 IF M_01=0 THEN 'Clear only once. 1020 M_01=1 'No. of error occurrences 1030 M_02=1 'Ring buffer ID 1040 ENDIF 1050 'Those independent of the error type 1060 PERINF(M_02,1).X=M_01 'No. of error occurrences 1070 PERINF(M_02,1).Y=M_ERRNO 'No. of the error occurred 1080 PERINF(M_02,1).Z=M_LINE(1) 'No. of the line where an error occurred 1090 PERINF(M_02,2).X=M_RDST 'Remaining distance to the target position 1100 PERINF(M_02,3)=P_CURR 'Current position	
1110 'Change information to be saved depending on the error content. 1120 SELECT M_ERRNO 1130 CASE 2802 1140 'Write information to be acquired when error 2802 occurred into PERINF. 1150 'PERINF(M_02,4).X=M_??? 1160 'PERINF(M_02,5)=P_??? 1170 BREAK 1180 CASE 9100 1190 'Write information to be acquired when error 9100 occurred into PERINF. 1200 BREAK 1210 DEFAULT 1220 'Write information to be acquired when an unexpected error occurred into PERINF. 1230 BREAK 1240 END SELECT 1250 '	It is possible to change information to be saved according to the error content.
1260 M_01=M_01+1 1270 M_02=(M_02 MOD 5)+1 'Generate an ID as a ring buffer. 1280 END	Increase the count of the buffer index.

9 Appendix

9.1 Reference Material

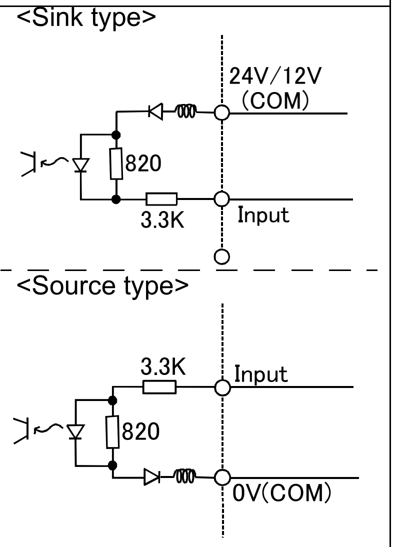
9.1.1 About sink/source type of the standard external input and output

There are two types of external input/output circuit specifications: sink type and source type. The circuit specification type specified by the customers at ordering is built into the product (sink type for general Japanese/international products, and source type for CE mark products for Europe). The external input/output circuit built into the controller as standard is explained below, together with the two types of circuit specifications mentioned above.

(1) Electrical specifications of input/output circuit

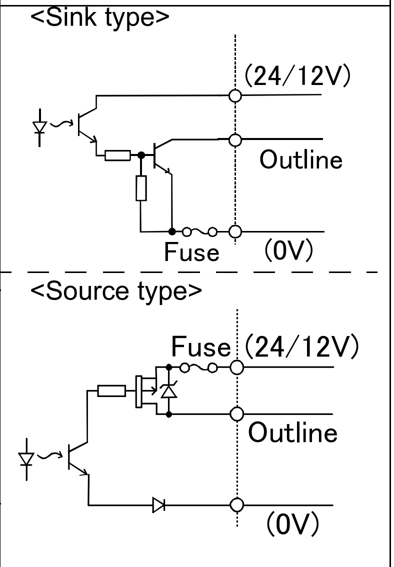
The external input circuit specification is shown in Table 9-1. The external output circuit specification is shown in Table 9-2.

Table 9-1:Electrical specifications of input circuit

Item		Specifications	Internal circuit
Type		DC input	
No. of input points		32 or 16 Note1)	
Insulation method		Photo-coupler insulation	
Rated input voltage		DC12V/DC24V	
Rated input current		Approx. 3mA/approx. 7mA	
Working voltage range		1 0.2VDC to 26.4VDC(ripple rate within 5%)	
ON voltage/ON current		8VDC or more/2mA or more	
OFF voltage/OFF current		4VDC or less/1 mA or less	
Input resistance		Approx. 3.3k É¹	
Response time	OFF-ON	1 0ms or less(DC24V)	
	ON-OFF	1 0ms or less(DC24V)	
Common method		8 points per common	
External wire connection method		Connector	

Note1)The number of input points differ with robot type. Refer to the separate manual "Standard specifications".

Table 9-2:Electrical specifications of output circuit

Item		Specifications	Internal circuit
Type		Transistor output	
No. of output points		32 or 16 Note1)	
Insulation method		Photo-coupler insulation	
Rated load voltage		DC12V/DC24V	
Rated load voltage range		DC 1 0.2 Å`30V(peak voltage 30VDC)	
Max. load current		0.1 A/point (1 00 Åì)	
Leakage current at OFF		0.1 mA or less	
Max. voltage drop at ON		DC0.9V(TYP.)	
Response time	OFF-ON	2ms or less (hardware response time)	
	ON-OFF	2ms or less (Resistance load) (hardware response time)	
Fuse rating		Fuse 3.2A (one per common) Replacement not possible	
Common method		4 points per common (common terminal: 4 points)	
External wire connection method		Connector	
External power supply	Voltage	DC12/24V(DC10.2Å`30V)	
	Current	60mA (TYP. 24VDC per common) (base drive current)	

Note1)The number of output points differ with robot type. Refer to the separate manual "Standard specifications".

(2) Connection example

Shows the connection example with a Mitsubishi PLC. The Fig. 9-1 is the sink type example, and the Fig. 9-2 is the source type example.

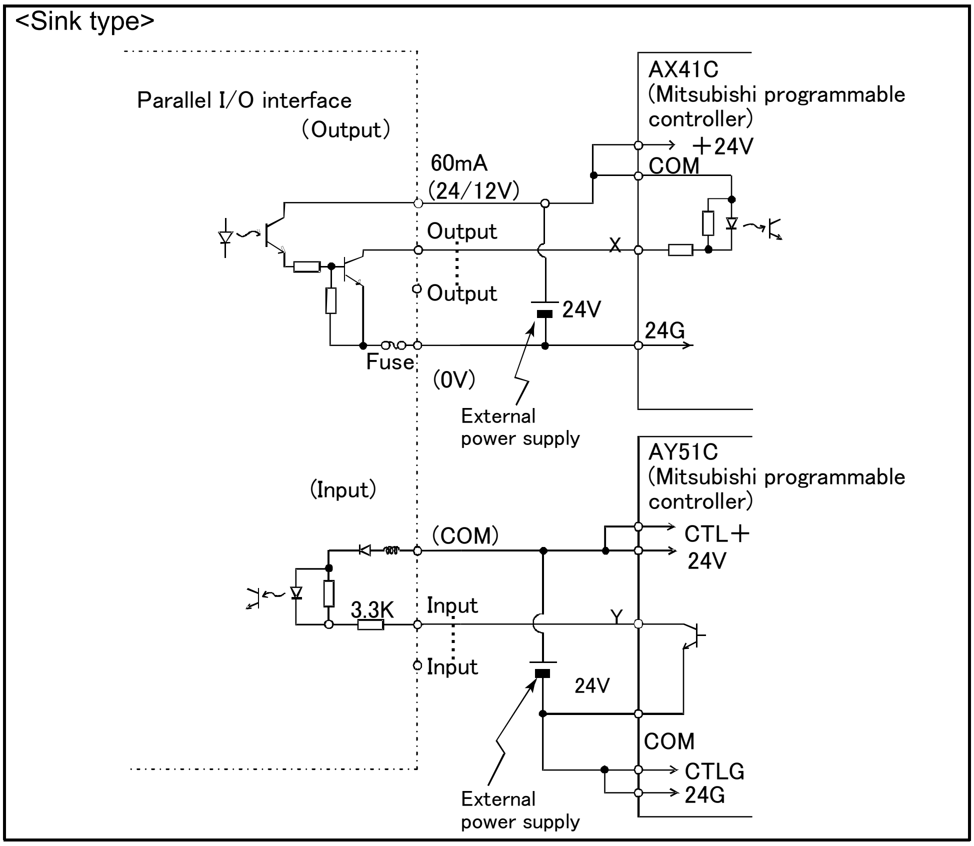


Fig.9-1:Connection with a Mitsubishi PLC (Example of sink type)
*The input/output circuit external power supply (24 VDC) must be prepared by the customer.

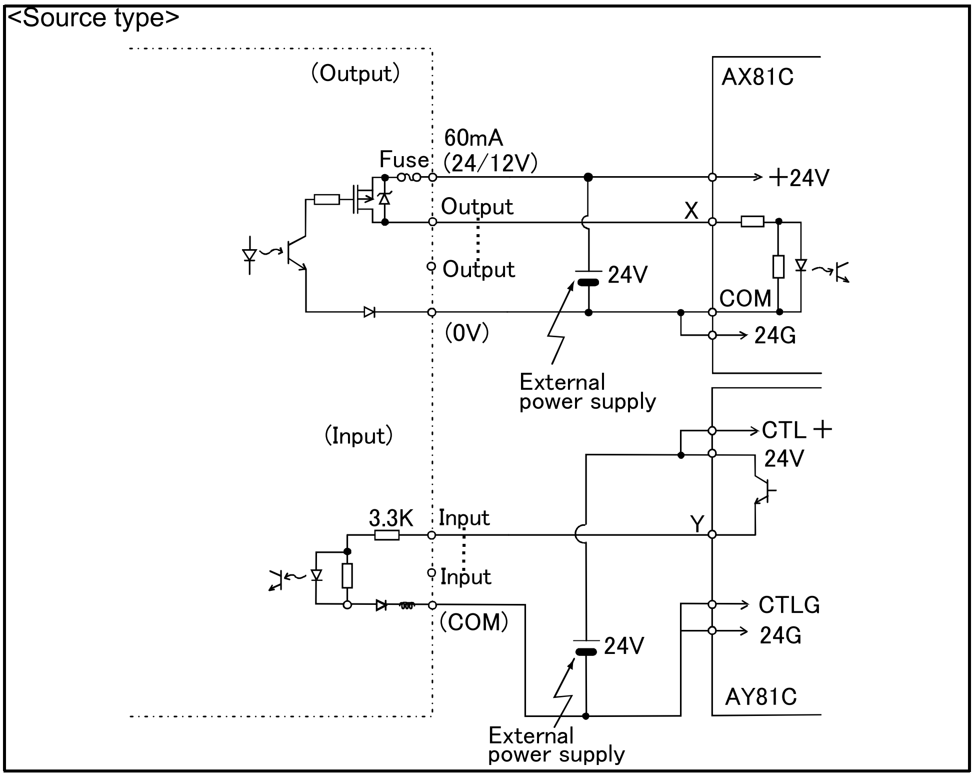


Fig.9-2:Connection with a Mitsubishi PLC (Example of source type)
*The input/output circuit external power supply (24 VDC) must be prepared by the customer.

(3) Connector pin assignment

A list of connector pin numbers and signal assignments of the external input/output card is shown below.

In case of the CR1-571 controller, the assignments of pin numbers differ between the sink type and source type. Table 9-3 and Table 9-4 show the connector pin assignments of the sink type and source type, respectively.

For other controllers, the pin assignments are the same for the sink type and source type. Table 9-5 and Table 9-6 show the connector pin assignments of CN100 and CN300, respectively.

*For CR1 controller

Table 9-3:Connector pin No. and signal assignment list (CR1 controller: sink type)

Pin No.	Line color Note1)	Function name		Pin No.	Line color Note1)	Function name	
		General-purpose	Dedicated/power supply, common			General-purpose	Dedicated/power supply, common
1	Orange/Red A		FG	26	Orange/Blue A		FG
2	Gray/Red A		0V:For pins 4-7	27	Gray/Blue A		0V:For pins 29-32
3	White/Red A		12V/24V:For pins 4-7	28	White/Blue A		12V/24V:For pins 29-32
4	Yellow/Red A	General-purpose output 0	Running	29	Yellow/Blue A	General-purpose output 4	
5	Pink/Red A	General-purpose output 1	Servo on	30	Pink/Blue A	General-purpose output 5	
6	Orange/Red B	General-purpose output 2	Error	31	Orange/Blue B	General-purpose output 6	
7	Gray/Red B	General-purpose output 3	Operation rights	32	Gray/Blue B	General-purpose output 7	
8	White/Red B		0V:For pins 10-13	33	White/Blue B		0V:For pins 35-38
9	Yellow/Red B		12V/24V:For pins 10-13	34	Yellow/Blue B		12V/24V:For pins 35-38
10	Pink/Red B	General-purpose output 8		35	Pink/Blue B	General-purpose output 12	
11	Orange/Red C	General-purpose output 9		36	Orange/Blue C	General-purpose output 13	
12	Gray/Red C	General-purpose output 10		37	Gray/Blue C	General-purpose output 14	
13	White/Red C	General-purpose output 11		38	White/Blue C	General-purpose output 15	
14	Yellow/Red C		COM0:For pins 15-22 Note2)	39	Yellow/Blue C		COM1:For pins 40-47 Note1)
15	Pink/Red C	General-purpose input 0	Stop(All slot) Note3)	40	Pink/Blue C	General-purpose input 8	
16	Orange/Red D	General-purpose input 1	Servo off	41	Orange/Blue D	General-purpose input 9	
17	Gray/Red D	General-purpose input 2	Error reset	42	Gray/Blue D	General-purpose input 10	
18	White/Red D	General-purpose input 3	Start	43	White/Blue D	General-purpose input 11	
19	Yellow/Red D	General-purpose input 4	Servo on	44	Yellow/Blue D	General-purpose input 12	
20	Pink/Red D	General-purpose input 5	Operation rights	45	Pink/Blue D	General-purpose input 13	
21	Orange/Red E	General-purpose input 6		46	Orange/Blue E	General-purpose input 14	
22	Gray/Red E	General-purpose input 7		47	Gray/Blue E	General-purpose input 15	
23	White/Red E		Reserved	48	White/Blue E		Reserved
24	Yellow/Red E		Reserved	49	Yellow/Blue E		Reserved
25	Pink/Red E		Reserved	50	Pink/Blue E		Reserved

Note1) "Line color" shows the line color of the external input/output cable 2A-CBL **.

Note2) Sink type:24V/12V(COM), Source type:0V(COM)

Note3) The assignment of the dedicated input signal "STOP" is fixed.

Table 9-4:Connector pin No. and signal assignment list (CR1 controller: source type type)

Pin No.	Line color Note1)	Function name		Pin No.	Line color Note1)	Function name	
		General-purpose	Dedicated/power supply, common			General-purpose	Dedicated/power supply, common
1	Orange/Red A		FG	26	Orange/Blue A		FG
2	Gray/Red A		0V:For pins 4-7, 10-13	27	Gray/Blue A		0V:For pins 29-32, 35-38
3	White/Red A		12V/24V:For pins 4-7, 10-13	28	White/Blue A		12V/24V:For pins 29-32, 35-38
4	Yellow/Red A	General-purpose output 0	Running	29	Yellow/Blue A	General-purpose output 4	
5	Pink/Red A	General-purpose output 1	Servo on	30	Pink/Blue A	General-purpose output 5	
6	Orange/Red B	General-purpose output 2	Error	31	Orange/Blue B	General-purpose output 6	
7	Gray/Red B	General-purpose output 3	Operation rights	32	Gray/Blue B	General-purpose output 7	
8	White/Red B		Reserved	33	White/Blue B		Reserved
9	Yellow/Red B		Reserved	34	Yellow/Blue B		Reserved
10	Pink/Red B	General-purpose output 8		35	Pink/Blue B	General-purpose output 12	
11	Orange/Red C	General-purpose output 9		36	Orange/Blue C	General-purpose output 13	
12	Gray/Red C	General-purpose output 10		37	Gray/Blue C	General-purpose output 14	
13	White/Red C	General-purpose output 11		38	White/Blue C	General-purpose output 15	
14	Yellow/Red C		COM0:For pins 15-22 Note2)	39	Yellow/Blue C		COM1:For pins 40-47 Note1)
15	Pink/Red C	General-purpose input 0	Stop(All slot) Note3)	40	Pink/Blue C	General-purpose input 8	
16	Orange/Red D	General-purpose input 1	Servo off	41	Orange/Blue D	General-purpose input 9	
17	Gray/Red D	General-purpose input 2	Error reset	42	Gray/Blue D	General-purpose input 10	
18	White/Red D	General-purpose input 3	Start	43	White/Blue D	General-purpose input 11	
19	Yellow/Red D	General-purpose input 4	Servo on	44	Yellow/Blue D	General-purpose input 12	
20	Pink/Red D	General-purpose input 5	Operation rights	45	Pink/Blue D	General-purpose input 13	
21	Orange/Red E	General-purpose input 6		46	Orange/Blue E	General-purpose input 14	
22	Gray/Red E	General-purpose input 7		47	Gray/Blue E	General-purpose input 15	
23	White/Red E		Reserved	48	White/Blue E		Reserved
24	Yellow/Red E		Reserved	49	Yellow/Blue E		Reserved
25	Pink/Red E		Reserved	50	Pink/Blue E		Reserved

Note1) "Line color" shows the line color of the external input/output cable 2A-CBL **.

Note2) Sink type:24V/12V(COM), Source type:0V(COM)

Note3) The assignment of the dedicated input signal "STOP" is fixed.

*For CR2/CR3/CR4/CR7/CR8/CR9 controller

Table 9-5:Connector CN100pin No. and signal assignment list (CR2/CR3/CR4/CR7/CR8/CR9 controller: sink and source common)

Pin No.	Line color Note1)	Function name		Pin No.	Line color Note1)	Function name	
		General-purpose	Dedicated/power supply, common			General-purpose	Dedicated/power supply, common
1	Orange/Red A		FG	26	Orange/Blue A		FG
2	Gray/Red A		0V:For pins 4-7	27	Gray/Blue A		0V:For pins 29-32
3	White/Red A		12V/24V:For pins 4-7	28	White/Blue A		12V/24V:For pins 29-32
4	Yellow/Red A	General-purpose output 0	Running	29	Yellow/Blue A	General-purpose output 4	
5	Pink/Red A	General-purpose output 1	Servo on	30	Pink/Blue A	General-purpose output 5	
6	Orange/Red B	General-purpose output 2	Error	31	Orange/Blue B	General-purpose output 6	
7	Gray/Red B	General-purpose output 3	Operation rights	32	Gray/Blue B	General-purpose output 7	
8	White/Red B		0V:For pins 10-13	33	White/Blue B		0V:For pins 35-38
9	Yellow/Red B		12V/24V:For pins 10-13	34	Yellow/Blue B		12V/24V:For pins 35-38
10	Pink/Red B	General-purpose output 8		35	Pink/Blue B	General-purpose output 12	
11	Orange/Red C	General-purpose output 9		36	Orange/Blue C	General-purpose output 13	
12	Gray/Red C	General-purpose output 10		37	Gray/Blue C	General-purpose output 14	
13	White/Red C	General-purpose output 11		38	White/Blue C	General-purpose output 15	
14	Yellow/Red C		COM0:For pins 15-22 Note2)	39	Yellow/Blue C		COM1:For pins 40-47 Note1)
15	Pink/Red C	General-purpose input 0	Stop(All slot) Note3)	40	Pink/Blue C	General-purpose input 8	
16	Orange/Red D	General-purpose input 1	Servo off	41	Orange/Blue D	General-purpose input 9	
17	Gray/Red D	General-purpose input 2	Error reset	42	Gray/Blue D	General-purpose input 10	
18	White/Red D	General-purpose input 3	Start	43	White/Blue D	General-purpose input 11	
19	Yellow/Red D	General-purpose input 4	Servo on	44	Yellow/Blue D	General-purpose input 12	
20	Pink/Red D	General-purpose input 5	Operation rights	45	Pink/Blue D	General-purpose input 13	
21	Orange/Red E	General-purpose input 6		46	Orange/Blue E	General-purpose input 14	
22	Gray/Red E	General-purpose input 7		47	Gray/Blue E	General-purpose input 15	
23	White/Red E		Reserved	48	White/Blue E		Reserved
24	Yellow/Red E		Reserved	49	Yellow/Blue E		Reserved
25	Pink/Red E		Reserved	50	Pink/Blue E		Reserved

Note1) "Line color" shows the line color of the external input/output cable 2A-CBL **.

Note2) Sink type:24V/12V(COM), Source type:0V(COM)

Note3) The assignment of the dedicated input signal "STOP" is fixed.

Table 9-6:Connector CN300pin No. and signal assignment list (CR2/CR3/CR4/CR7/CR8/CR9 controller: sink and source common)

Pin No.	Line color Note1)	Function name		Pin No.	Line color Note1)	Function name	
		General-purpose	Dedicated/power supply, common			General-purpose	Dedicated/power supply, common
1	Orange/Red A		FG	26	Orange/Blue A		FG
2	Gray/Red A		0V:For pins 4-7	27	Gray/Blue A		0V:For pins 29-32
3	White/Red A		12V/24V:For pins 4-7	28	White/Blue A		12V/24V:For pins 29-32
4	Yellow/Red A	General-purpose output 16		29	Yellow/Blue A	General-purpose output 20	
5	Pink/Red A	General-purpose output 17		30	Pink/Blue A	General-purpose output 21	
6	Orange/Red B	General-purpose output 18		31	Orange/Blue B	General-purpose output 22	
7	Gray/Red B	General-purpose output 19		32	Gray/Blue B	General-purpose output 23	
8	White/Red B		0V:For pins 10-13	33	White/Blue B		0V:For pins 35-38
9	Yellow/Red B		12V/24V:For pins 10-13	34	Yellow/Blue B		12V/24V:For pins 35-38
10	Pink/Red B	General-purpose output 24		35	Pink/Blue B	General-purpose output 28	
11	Orange/Red C	General-purpose output 25		36	Orange/Blue C	General-purpose output 29	
12	Gray/Red C	General-purpose output 26		37	Gray/Blue C	General-purpose output 30	
13	White/Red C	General-purpose output 27		38	White/Blue C	General-purpose output 31	
14	Yellow/Red C		COM0:For pins 15-22 Note2)	39	Yellow/Blue C		COM1:For pins 40-47 Note1)
15	Pink/Red C	General-purpose input 16		40	Pink/Blue C	General-purpose input 24	
16	Orange/Red D	General-purpose input 17		41	Orange/Blue D	General-purpose input 25	
17	Gray/Red D	General-purpose input 18		42	Gray/Blue D	General-purpose input 26	
18	White/Red D	General-purpose input 19		43	White/Blue D	General-purpose input 27	
19	Yellow/Red D	General-purpose input 20		44	Yellow/Blue D	General-purpose input 28	
20	Pink/Red D	General-purpose input 21		45	Pink/Blue D	General-purpose input 29	
21	Orange/Red E	General-purpose input 22		46	Orange/Blue E	General-purpose input 30	
22	Gray/Red E	General-purpose input 23		47	Gray/Blue E	General-purpose input 31	
23	White/Red E		Reserved	48	White/Blue E		Reserved
24	Yellow/Red E		Reserved	49	Yellow/Blue E		Reserved
25	Pink/Red E		Reserved	50	Pink/Blue E		Reserved

Note1) "Line color" shows the line color of the external input/output cable 2A-CBL **.

Note2) Sink type:24V/12V(COM), Source type:0V(COM)

MITSUBISHI ELECTRIC

HEADQUARTERS

MITSUBISHI ELECTRIC **EUROPE**
EUROPE B.V.
German Branch
Gothaer Straße 8
D-40880 Ratingen
Phone: +49 (0)2102 486-0
Fax: +49 (0)2102 486-1120
e mail: megfamail@meg.mee.com

MITSUBISHI ELECTRIC **FRANCE**
EUROPE B.V.
French Branch
25, Boulevard des Bouvets
F-92741 Nanterre Cedex
Phone: +33 1 55 68 55 68
Fax: +33 1 55 68 56 85
e mail: factory.automation@fra.mee.com

MITSUBISHI ELECTRIC **IRELAND**
EUROPE B.V.
Irish Branch
Westgate Business Park, Ballymount
IRL-Dublin 24
Phone: +353 (0) 1 / 419 88 00
Fax: +353 (0) 1 / 419 88 90
e mail: sales.info@meir.mee.com

MITSUBISHI ELECTRIC . **ITALY**
EUROPE B.V
Italian Branch
Via Paracelso 12
I-20041 Agrate Brianza (MI)
Phone: +39 (0)39 / 60 53 1
Fax: +39 (0)39 / 60 53 312
e mail: factory.automation@it.mee.com

MITSUBISHI ELECTRIC **SPAIN**
EUROPE B.V.
Spanish Branch
Carretera de Rubí 76-80
E-08190 Sant Cugat del Vallés
(Barcelona)
Phone: +34 9 3 / 565 3160
Fax: +34 9 3 / 589 1579
e mail: industrial@sp.mee.com

MITSUBISHI ELECTRIC **UK**
EUROPE B.V.
UK Branch
Travellers Lane
GB-Hatfield Herts. AL10 8 XB
Phone: +44 (0)1707 / 27 61 00
Fax: +44 (0)1707 / 27 86 95
E-mail: automation@meuk.mee.com

MITSUBISHI ELECTRIC **JAPAN**
CORPORATION
Office Tower "Z" 14 F
8-12,1 chome, Harumi Chuo-Ku
Tokyo 104-6212
Phone: +81 3 6221 6060
Fax: +81 3 6221 6075

MITSUBISHI ELECTRIC **USA**
AUTOMATION
500 Corporate Woods Parkway
Vernon Hills, IL 60061
Phone: +1 847 / 478 21 00
Fax: +1 847 / 478 22 83

EUROPEAN REPRESENTATIVES

GEVA **AUSTRIA**
Wiener Straße 89
AT-2500 Baden
Phone: +43 (0)2252 / 85 55 20
Fax: +43 (0)2252 / 488 60
e mail: office@geva.at

Koning & Hartman b.v. **BELGIUM**
Researchpark Zellik
Pontbeeklaan 43
BE-1731 Brussels
Phone: +32 (0)2 / 467 17 51
Fax: +32 (0)2 / 467 17 45
e mail: info@koningenhartman.com

AutoCont **CZECH REPUBLIC**
Control Systems s.r.o.
Nemocnicni 12
CZ-70200 Ostrava 2
Phone: +420 59 / 6152 111
Fax: +420 59 / 6152 562
e mail: consys@autocont.cz

HERSTAD + PIPER A/S **DENMARK**
Jernholmen 48 C
DK-2650 Hvidovre
Phone: +45 (0) 36 / 77 40 00
Fax: +45 (0) 36 / 77 77 40
e mail: mail@herstad-piper.dk

Beijer Electronics OY **FINLAND**
Ansatie 6a
FIN-01740 Vantaa
Phone: +358 (0)9 / 886 77 500
Fax: +358 (0)9 / 886 77 555
e mail: info@beijer.fi

Kouvalias **GREECE**
Robot + Vision Systems
25, El. Venizelou Ave
GR-17671 Kallithea
Phone: +30 22950 / 42902/3/4
Fax: +30 22950 / 42690
e mail: info@kouvalias.com

Axicont Automatika Kft. **HUNGARY**
Reitter F. U. 132
HU-1131 Budapest
Phone: +36 (0)1 / 412-0882
Fax: +36 (0)1 / 412-0883
e mail: office@axicont.hu

EUROPEAN REPRESENTATIVES

Koning & Hartman b.v. **NETHERLANDS**
Donauweg 2 B
NL-1000 AK Amsterdam
Tel: +31 (0)20 / 587 76 00
Fax: +31 (0)20 / 587 76 05
e mail: info@koningenhartman.com

Beijer Electronics AS **NORWAY**
Teglverksveien 1
NO-3002 Drammen
Phone: +47 (0)32 / 24 30 00
Fax: +47 (0)32 / 84 85 77
e mail: info@beijer.no

MPL Technology Sp. z o.o. **POLAND**
ul. Sliczna 36
PL-31-444 Kraków
Phone: +48 (0)12 / 632 28 85
Fax: +48 (0)12 / 632 47 82
e mail: krakow@mpl.pl

INEA SR d.o.o. **SERBIA AND MONTENEGRO**
Karadjordjeva 12/260
SCG-113000 Smederevo
Phone: +381 (0)26 / 617 163
Fax: +381 (0)26 / 617 163
e mail: vladstoj@yubc.net

AutoCont Control s.r.o. **SLOVAKIA**
Radlinského 47
SK-02601 Dolný Kubín
Phone: +421 435868 210
Fax: +421 435868 210
e mail: info@autocontcontrol.sk

INEA d.o.o. **SLOVENIA**
Stegne 11
SI-1000 Ljubljana
Phone: +386 (0)1- 513 8100
Fax: +386 (0)1- 513 8170
e mail: inea@inea.si

Beijer Electronics AB **SWEDEN**
Box 426
S-20124 Malmö
Phone: +46 (0)40 / 35 86 00
Fax: +46 (0)40 / 35 86 02
e mail: info@beijer.se

ECONOTEC AG **SWITZERLAND**
Postfach 282
CH-8309 Nürensdorf
Phone: +41 (0)1 / 838 48 11
Fax: +41 (0)1 / 838 48 12
e mail: info@econotec.ch

GTS **TURKEY**
Darülaceze Cad. No. 43 Kat. 2
TR-80270 Okmeydani-Istanbul
Phone: +90 (0)212 / 320 1640
Fax: +90 (0)212 / 320 1649
e mail: gts@turk.net

EURASIAN REPRESENTATIVES

ELEKTROSTYLE **RUSSIA**
Poslannikov Per., 9, Str.1
RU-107005 Moscow
Phone: +7 095 / 542-4323
Fax: +7 095 / 956-7526
e mail: info@estl.ru

ELEKTROSTYLE **RUSSIA**
Krasnij Prospekt 220-1,Office 312
RU-630049 Novosibirsk
Phone: +7 3832 / 10 66 18
Fax: +7 3832 / 10 66 26
e mail: info@estl.ru

ICOS **RUSSIA**
Industrial Computer Systems Zao
Ryazanskij Prospekt, 8A, Office 100
RU-109428 Moscow
Phone: +7 095 232-0207
Fax: +7 095 232-0327
e mail: mail@icos.ru

MIDDLE EAST REPRESENTATIVE

ILAN & GAVISH LTD **ISRAEL**
Automation Service
24 Shenkar St., Kiryat Arie
IL-49001 Petach-Tiqva
Phone: +972 (0) 3 / 922 18 24
Fax: +972 (0) 3 / 924 07 61
e mail: iandg@internet-zahav.net

AFRICAN REPRESENTATIVE

CBI Ltd **SOUTH AFRICA**
Private Bag 2016
ZAF-1600 Isando
Phone: +27 (0) 11 / 928 2000
Fax: +27 (0) 11 / 392 2354
e mail: cbi@cbi.co.za

ROB 11/05 - Printed in Germany

MITSUBISHI ELECTRIC INDUSTRIAL AUTOMATION
Gothaer Strasse 8
D-40880 Ratingen
Phone: +49 2102 486-0
Hotline: +49 1805 000-765
Fax: +49 2102 486-7170
megfa-mail@meg.mee.com
www.mitsubishi-automation.de
www.mitsubishi-automation.com