Simple and friendly

Kawasaki

Kawasaki Robot Controller
F Series

AS Language Reference Manual

Robot

Kawasaki Heavy Industries, Ltd.

90209-1025DEB

Preface

This manual describes the AS* language used in the Kawasaki Robot Controller F series. The objective for this manual is to provide detailed information on the outline of the AS system, basic usages, data types, robot trajectory control and all the commands/instruction to allow effective usage of the AS system. The robot operation procedures are not included here, so refer to the Operation Manual for that information. This manual should be read with careful review of the related manuals listed below. Once the contents of all the manuals are thoroughly read and understood the robot can be used.

1. Safety Manual
2. Installation and Connection Manual for Arm
3. Installation and Connection Manual for Controller
4. External I/O Manual (for connecting with peripheral devices)
5. Inspection and Maintenance Manual

The contents of this manual are described on condition that installation and connection of the robot are done in accordance with the above listed manuals.

The explanations in this manual include information on optional functions, but depending on the specification of each unit, not every optional function detailed here may be included with the robot. Should any unexplained questions or problems arise during robot operation, please contact Kawasaki. Refer to the contact information listed on the rear cover of this manual for the nearest Kawasaki.

NOTE* AS is pronounced [az].

---

1. This manual does not constitute a guarantee of the systems in which the robot is utilized. Accordingly, Kawasaki is not responsible for any accidents, damages, and/or problems relating to industrial property rights as a result of using the system.
2. It is recommended that all personnel assigned for activation of operation, teaching, maintenance or inspection of the robot attend the necessary education/training course(s) prepared by Kawasaki, before assuming their responsibilities.
3. Kawasaki reserves the right to change, revise, or update this manual without prior notice.
4. This manual may not, in whole or in part, be reprinted or copied without the prior written consent of Kawasaki.
5. Store this manual with care and keep it available for use at any time. If the robot is reinstalled or moved to a different site or sold off to a different user, attach this manual to the robot without fail. In the event the manual is lost or damaged severely, contact Kawasaki.

Symbols

The items that require special attention in this manual are designated with the following symbols.

Ensure proper and safe operation of the robot and prevent physical injury or property damage by complying with the safety matters given within the boxes with these symbols.

⚠ DANGER

Failure to comply with indicated matters can result in imminent injury or death.

⚠ WARNING

Failure to comply with indicated matters may possibly lead to injury or death.

⚠ CAUTION

Failure to comply with indicated matters may lead to physical injury and/or mechanical damage.

[NOTE]

Denotes precautions regarding robot specification, handling, teaching, operation and maintenance.

⚠ WARNING

1. **The accuracy and effectiveness of the diagrams, procedures, and detail explanations given in this manual cannot be confirmed with absolute certainty. Should any unexplained questions or problems arise, please contact Kawasaki.**
2. **Safety related contents described in this manual apply to each individual work and not to all robot work. In order to perform every work in safety, read and fully understand the safety manual, all pertinent laws, regulations and related materials as well as all the safety explanation described in each chapter, and prepare safety measures suitable for actual**

Table of Contents

1 Overview of AS

The Kawasaki robots are controlled by a software-based system called AS. This chapter describes the overall view of the AS system.

1.1 Overview of the AS System

In the AS system, AS language is used for communication with robots or for programming. The AS system is written in the nonvolatile memory in the robot control unit. When the controller power is turned on, the AS system starts and waits for a command to be input.

The AS system controls the robot according to the given commands and programs. It can also execute several types of functions while a program is running. Some of the functions that can be used while a program is running are: displaying the system status, defining pose variable, saving data in external memory devices, and writing/editing programs.

1.2 Characteristics of the AS System

In the AS system, robots are controlled and operated based on a program. A program is prepared before a robot operation is conducted and describes the necessary tasks for that operation. (Teaching Playback Method)

AS language can be divided into two types: monitor commands and program instructions.

Monitor commands: Used to write, edit, and execute programs. They are entered after the prompt (>) shown on the screen, and are immediately executed. Some of the monitor commands are used within the programs to work as program instructions.

Program instructions: Used to direct the movements of the robot, to monitor or to control external signals, etc. in programs. A program is a collection of program instructions.

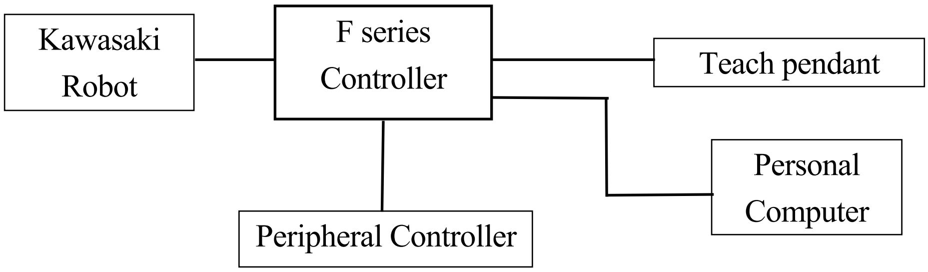
In this manual, monitor command is referred to as command, and program instruction as instruction.

AS is unique in following ways:

1. Robot can be moved along a continuous path trajectory (CP motion: Continuous Path motion).
2. Two coordinate systems are provided: base coordinates with its origin at the robot base, and tool coordinates fixed on the tool attached to the end of the arm. The robot can be moved based on either of the coordinate systems.
3. The coordinates can be shifted or rotated corresponding to the task situation.
4. When in teach or repeat mode, robot can be moved along a linear path. In teach mode, this can be done while keeping the tool orientation.
5. Programs can be named freely and saved without limits in numbers within the memory capacity.
6. Each operation unit can be defined as a program and these programs can be combined to make a complex one. (Subroutine)
7. By monitoring signals, programs can be interrupted and branched to a different program suspending current motions when an external signal is input. (Interruption)
8. A Process Control program (PC program) without a motion instruction can be executed simultaneously with a robot control program.
9. Programs and pose data can be displayed on terminals and saved in devices such as USB flash drive memory.
10. Programming can be done using a personal computer loaded with the terminal software (KRterm) provided by Kawasaki. (Off-line programming)

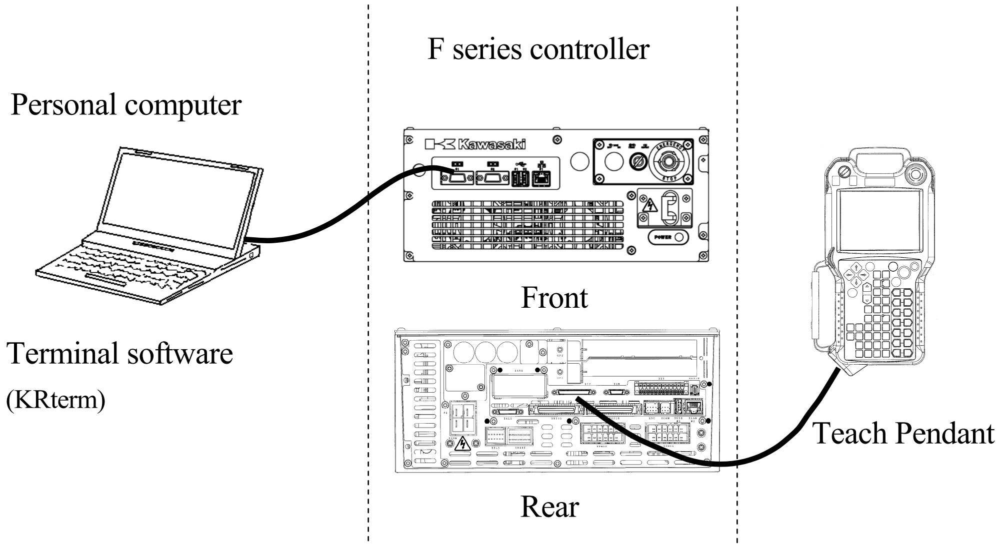
1.3　AS System Configuration

Kawasaki Robot controller F series is composed of following components:



By connecting a personal computer loaded with the terminal software (KRterm) to an F series controller, the following operations can be done:

・ Writing AS commands and instructions
・ Saving and loading to and from personal computers



Personal computer	Controller	Teach Pendant
· Enters AS monitor commands · Enters AS program instructions · Saves/loads to storage devices such as programs	Daily operations	· Selects program · Displays program names and steps · Manually controls the robot · Monitors signals · Sets repeating conditions · Teaches pose data · Teaches auxiliary data (block teaching)

[**NOTE**]

KRterm operates on a PC with Microsoft Windows 7/8/8.1/10. Please prepare an appropriate PC.

2 AS System

This chapter describes the AS system status, AS system switches and system setup.

2.1 AS System Status

The AS system consists of the following three modes:

1. Monitor Mode
   This is the basic mode in the AS system in which the execution of the AS system is controlled and monitored. The Monitor commands are executed in this mode. Access to Editor Mode (by executing EDIT command) or Playback Mode (by executing EXECUTE command) from this mode.

2. Editor Mode
   This mode enables you to create a new program or to modify an existing one. Only editor commands are executed by the system in this mode.

3. Playback Mode
   The system is in Playback Mode during program execution. Commands entered from the terminal are processed in this mode. At the same time, computations for robot motion control are performed at a certain cycle. Most monitor commands can be input in this mode. See 5.0 Monitor Commands for monitor commands that are allowed input during Playback Mode.

2.2 AS System Switches

The following system switches can be set in the AS System using the monitor command SWITCH. The status and the conditions set for each switch can be checked or changed from the terminal.

1. CHECK.HOLD
   Determines whether or not to accept input from the keyboard of EXECUTE, DO, STEP, MSTEP, and CONTINUE commands only in HOLD state.

2. CP
   Enables or disables continuous path movement. When this switch is ON, the robot makes smooth transitions between motion segments. When it is OFF, the robot decelerates and stops at the end of each motion segment.

3. CYCLE.STOP
   Determines whether to keep CYCLE START in ON state or to turn it OFF when an external hold signal is input to stop the motion of the robot.

4. MESSAGES
   Enables or disables message output to the terminal in response to the PRINT or TYPE command.

5. OX.PREOUT
   Sets the timing for OX signal output in block instructions.

6. PREFETCH.SIGINS
   Determines whether to allow or not the early processing of signal input and output via AS commands/instructions.

7. QTOOL
   Determines whether the tool data is changed only when TOOL command/instruction or block instruction is executed in repeat mode, or to allow automatic change also in teach mode according to the tool number taught in block instructions.

8. REP ONCE (Repeat Once)
   When this switch is ON, the program runs once. When it is OFF, the program runs continuously.

9. RPS (Random Program Selection)
   Enables or disables the function to allow selection of programs via external signals.

10. SCREEN
    Enables or disables the scrolling of the screen when the information is too large to fit in one screen.

11. STP_ONCE
    Sets whether the program is performed one step at a time or continuously.

Refer to 5.6 Monitor Command SWITCH, ON, OFF for further information on how to set the system switches.

2.3 AS System Setup

The following system settings can be changed depending on the need, using the monitor commands.

1. Zeroing (ZZERO command)
   ZZERO command is used to set the encoder value corresponding to the mechanical origin of each axis of a robot as zeroing data. When replacing the servo motor or performing maintenance on an encoder, the encoder value will need adjustment using this command. (This command is for maintenance purposes only.)

2. Clamp setting (HSETCLAMP command)
   This setting is made prior to shipment from the factory. The settings, single/double and output spec (ON when closed/OFF when closed), can be changed using HSETCLAMP command. However, the change will only affect the software, so be sure to check the consistency with the hardware.

3. Maximum number of input and output signals (ZSIGSPEC command)
   ZSIGSPEC command sets the maximum number of input and output signals that can be used. It is set prior to shipment from the factory. (This is a default setting that functions as a software error check, thus be sure it is consistent with the hardware.)

4. Software Dedicated Signals (DEFSIG command)
   In addition to the hardware dedicated signals, there are I/O signals in the software that can be used as dedicated signals (Software dedicated signals). The signals in the table below can be used as Software dedicated signals. Note that since the number of I/O signals in the software is the sum of Software dedicated signals and general purpose signals, the number of general purpose signal decreases as more software dedicated signals are used. Please refer to External I/O Manual for details of dedicated signals.

Software Dedicated Input Signal	Software Dedicated Output Signal
EXT. MOTOR ON	MOTOR_ON
EXT. ERROR RESET	ERROR
EXT. CYCLE START	AUTOMATIC
EXT. PROGRAM RESET	CYCLE START
Ext. prog. select (JUMP_ON, JUMP_OFF RPS_ON, RPSxx)	TEACH MODE HOME1, HOME2
EXT_IT	POWER ON
EXT. SLOW REPEAT MODE	RGSO
	Ext. prog. select enable (RPS)
	Ext. prog. select (JMP_ST, RPS_ST)

2.4 Input/Output Control

2.4.1 Terminal Control

Data and commands input at a terminal are first received by the system buffer. Then they are read by the monitor or program and echoed or displayed on the terminal screen. The maximum number of characters that can be input at a terminal is 79 (or 80 when ">" is input in prompt), and additional characters input are ignored.

Output of data to a terminal can be controlled using the PRINT and TYPE instructions. 8 bits are displayed on the terminal screen. Unless format is specified using specification code "/S" with the PRINT/TYPE instruction, data are displayed with a new line starting after each command. (See 6.8 Message Control Instructions for detailed information.)

Terminal input and output can be controlled using the commands shown below. These are called terminal control commands. Ctrl (Control Key) is pressed with each alphabetical character (the character may be either lower or upper case letters). Unlike other AS commands, there is no need to press the ENTER key after these command.

Commands	Functions
Ctrl + S	Stops the scrolling of the display terminal.
Ctrl + Q	Resumes the data output stopped by Ctrl + S.
Ctrl + C	Cancels the last input line.
Ctrl + H	Deletes the last input character. (Backspace)
Ctrl + M	Ends the input of the current line.
Ctrl + L	Displays the content of the line entered previously on the current input line. It can be used up to seven times. (Last)
Ctrl + N	Displays the content of the line input after the line displayed using Ctrl + L. This operation can be used only after Ctrl + L is used more than once. (Next)
Backspace	Deletes the last input character.

Input TAB (Ctrl + I or TAB) as space (blank).

2.4.2 External Memory Devices

The commands below are used to save programs, variables and pose information in the robot memory, USB flash drive memory, or computer hard disk.

1. Displays the contents on the USB flash drive memory. (USB_FDIR)
2. Saves the data on the robot memory to PC or USB flash drive files. (SAVE*, USB_SAVE)
3. Loads the data on PC or USB flash drive to the robot memory. (LOAD, USB_LOAD)
4. Deletes the files on PC or USB flash drive. (USB_FDEL)

Commands with USB_ refer to USB flash drive memory.

NOTE* SAVE/LOAD command may be used only when the computer is connected.

See also 5.2 Program and Data Control Commands, 5.3 Program and Data Storage Commands.

2.5 Installing Terminal Software (KRterm)

The robot can be controlled from a personal computer using the AS language. To do so, load KRterm on to a PC and connect the PC to an F series controller. The operation environment needed for KRterm is as follows:

Hardware	Microsoft Windows running PC
OS	Microsoft Windows 7/8/8.1/10
Tested models	DELL PRECISION M6600 (Windows 7) Sony VAIO Fit 13A (Windows 8) Sony VAIO Duo 13 (Windows 8.1) Sony VAIO Fit 13A (Windows 10)

NOTE* KRterm may not operate properly on untested models.

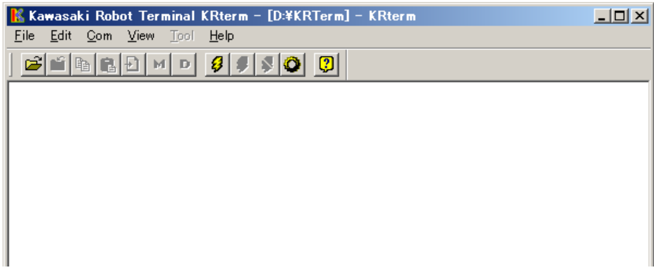
Connecting the computer and the controller using the RS-232C cable enables a single computer to control a single robot. An Ethernet connection enables multiple computers to control multiple robots.

Follow the below procedure to install KRterm on to the PC.

2.5.1 Installing KRterm

Copy the setup software for KRterm (SetupE.exe), provided by Kawasaki to a file on a Windows PC and execute it. Follow the installer direction to complete the installation.

After the installation is completed, an icon for KRterm is created, so double-click on it. KRterm starts and the window as shown below is displayed.



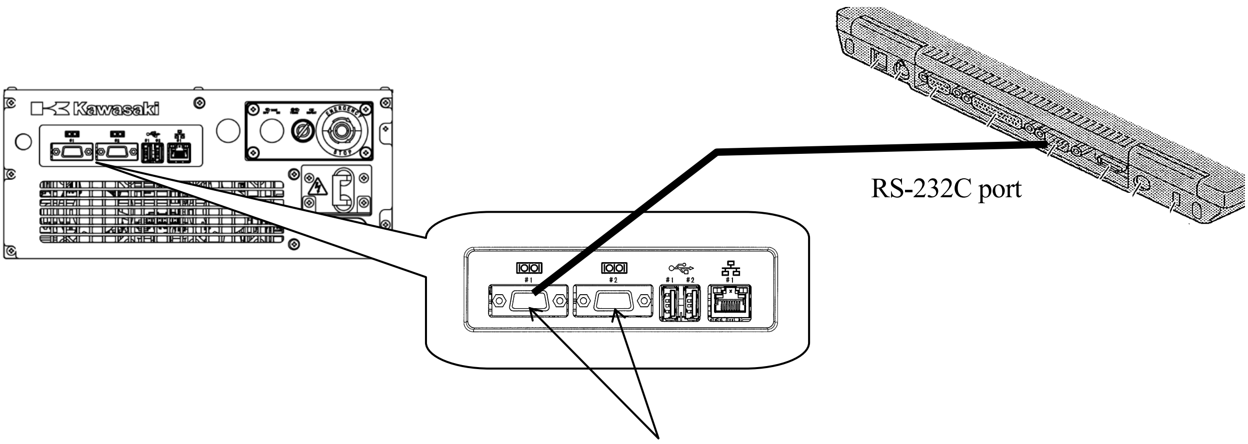
2.6 Operations from Personal Computer

2.6.1 System Setup

2.6.1.1 Connecting to RS-232C Port

The operation of the controller from a PC via RS-232C is possible by using KRterm.

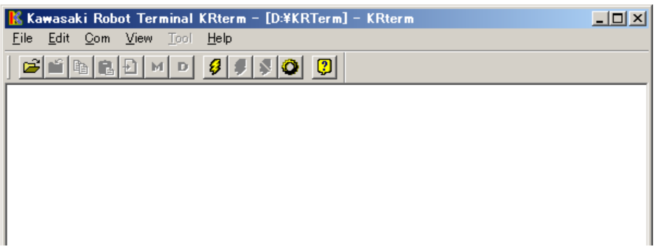
1. Connect the personal computer with the controller using the RS-232C cable. Make sure the CONTROLLER POWER on the controller and the computer power are both turn off.



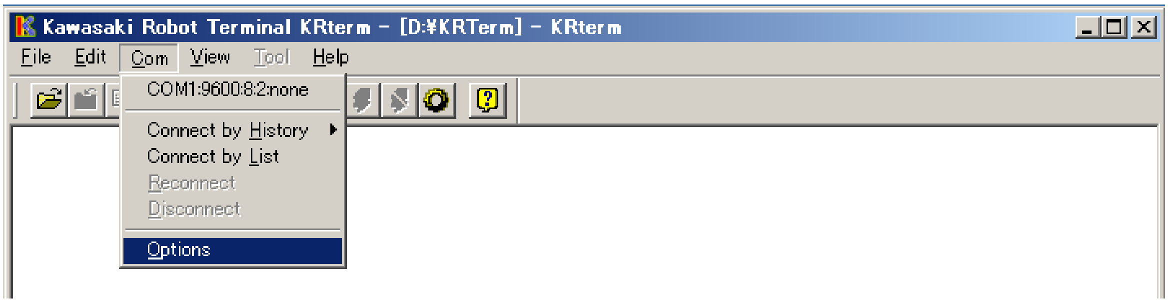
Use #1 for the RS232C port.
For PC connection cable, use straight cable with female- female Dsub9 pin connector. The connectors are allocated as below:

Pin	Name	Content
1	CD	Carrier detection
2	RD	Receive data
3	SD	Send data
4	ER	Data terminal ready
5	SG	Ground
6	DR	Data set ready
7	RS	Request send
8	CS	Send allowed
9	CI	Calling indicate

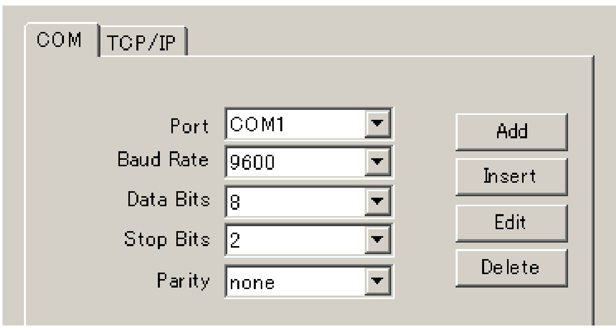
2. Turn on the computer, and start KRterm by clicking on the icon.



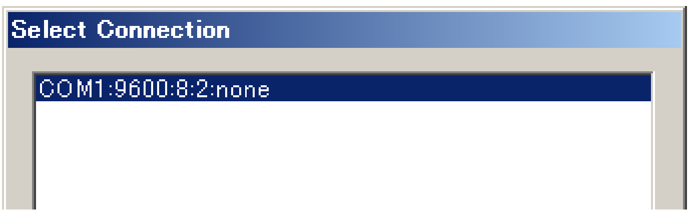
3. When the KRterm screen opens, set the connection.
   Select from the menu bar, [Com(munication)] →[Options].



4. Enter 9600 for <Baud Rate>, 8 for <Data Bits>, 2 for <Stop Bits>, “none” for <Parity>.
   Select the <COM> tab and similarly set the parameters, then click <OK>.



Next, select [Com(munication)]→[Connect by List] and in the window that appears, select the connection set above.　Click <OK>.



5. Turn ON the CONTROLLER POWER on the controller.
   (See “Operation Manual” 3.1 Power ON Procedure.)

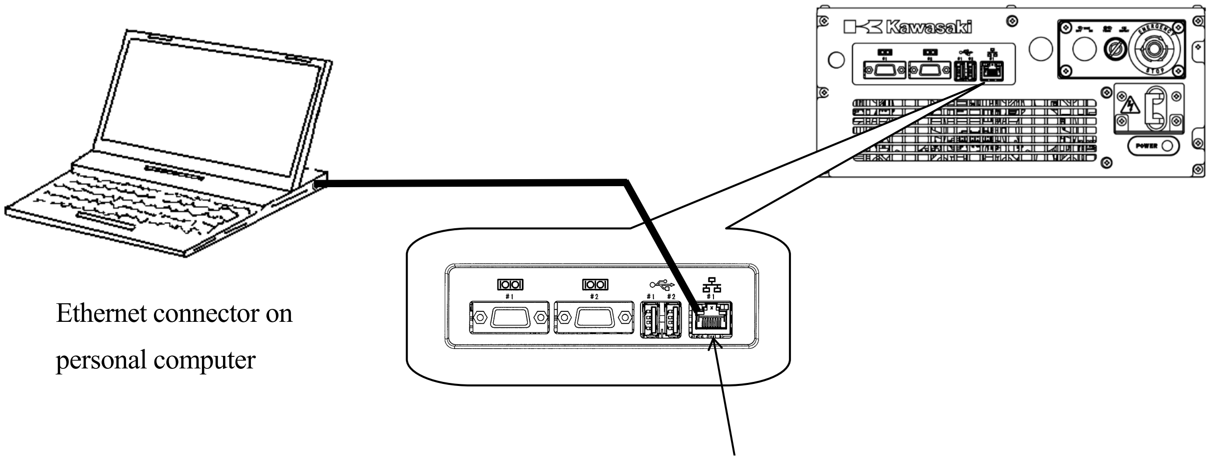
6. The initial screen for KRterm will appear on the display.

When the CONTROLLER POWER is turned ON before connecting the PC to the controller, only the prompt “>” will appear and not the initial screen. However, KRterm works the same.

2.6.1.2 Connecting Robots Using the Ethernet

The operation of the controller from a PC via Ethernet is possible by using KRterm.

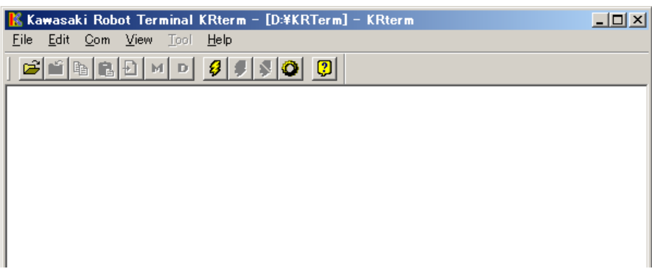
1. Connecting the cables.



For Ethernet cable, use straight cable with RJ45 connector. The connectors are allocated as below:

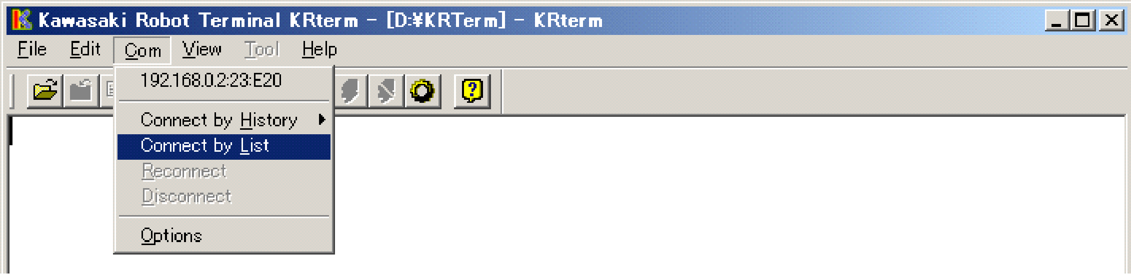
Pin	Name	Content
1	TD+	Send+
2	TD-	Send-
3	RD+	Receive+
4	Not used	
5	Not used	
6	RD-	Receive-
7	Not used	
8	Not used	

2. Turn ON the PC and double click on the icon for KRterm.

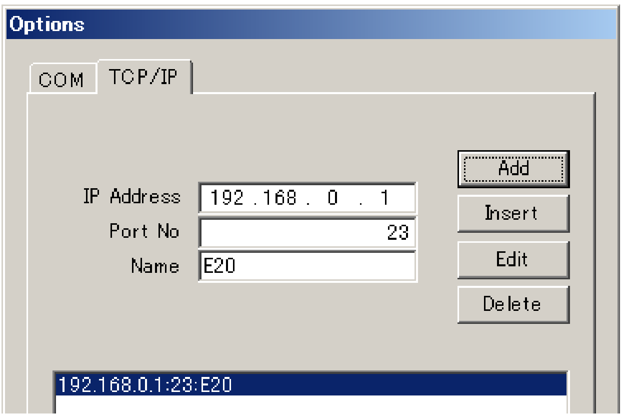


3. Next, register the IP address for the robot to connect.

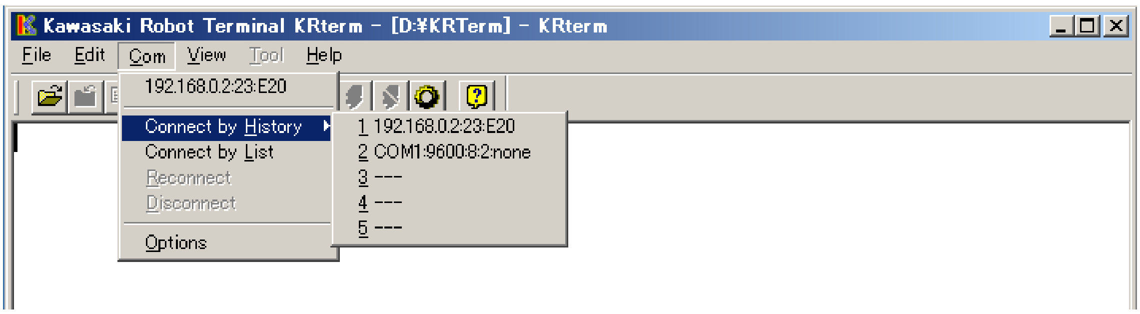
Select [Com(munication)] →[Options] from the menu bar.



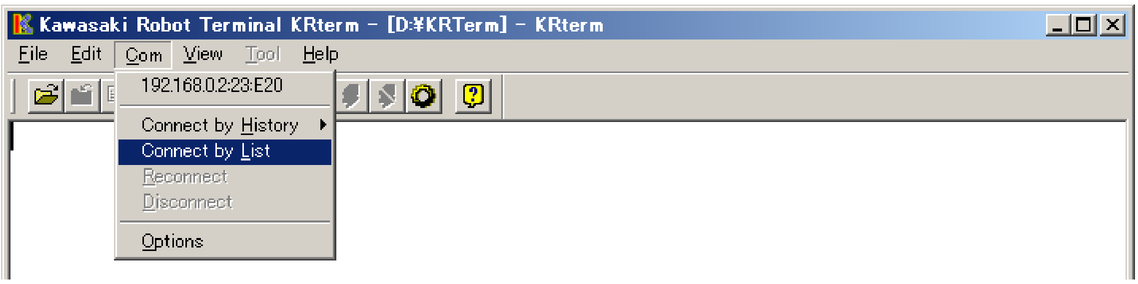
Click on [TCP/IP] tab and enter the IP address and name (optional) for the robot controller to connect on the network. Click on <Add>.



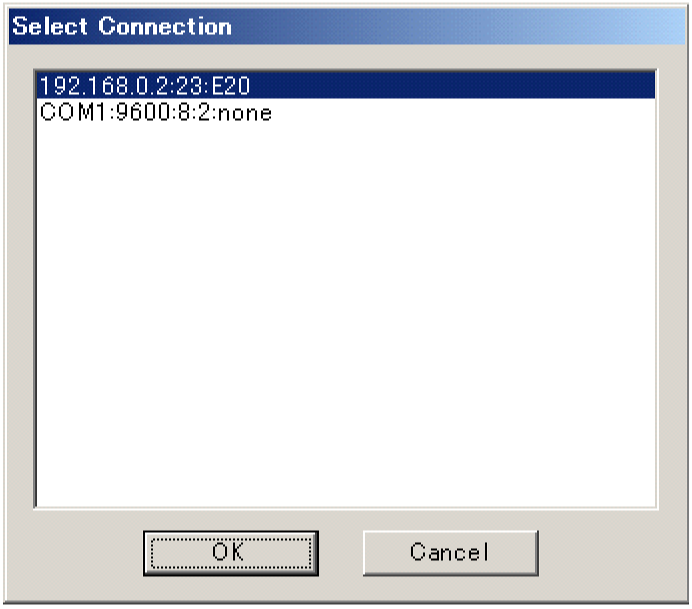
4. Connect to the registered robot on the network.
   (1) The robot last used is displayed at the top of the drop-down list that is displayed when clicking on [Com] on the menu bars. OR
   (2) Select [Com] →[Connect by History] to displayed a list of robots used in the past. Select the robot to connect from this list.



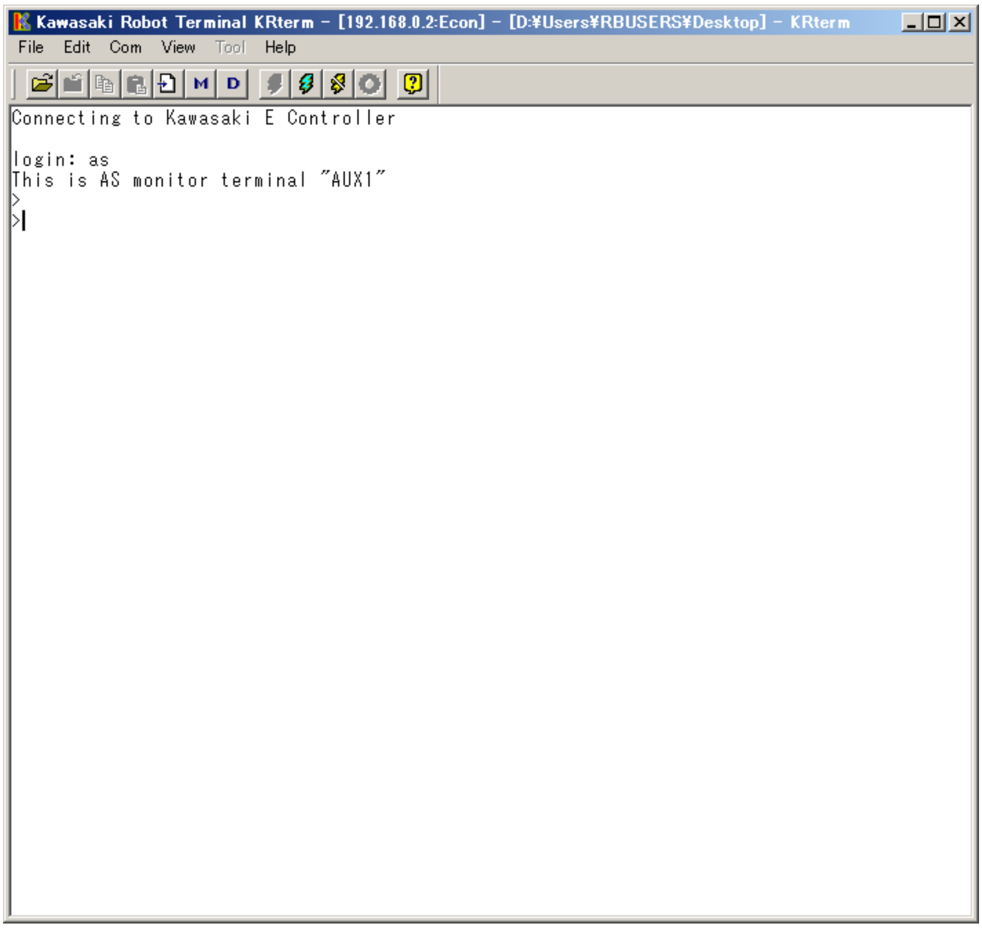
(3) To connect to robots not in the list, select [Com(C)] → [Connect by List].



Select the robot to connect and click on <OK>.



5. If the connection is established, robot information such as its name followed by the message login :. Enter “as” after this message. A prompt “>” returned from the robot is then displayed.



AS commands can be input once the prompt appears.

2.6.2 Uploading and Downloading Data

1. SAVE command

To save the data on the computer, use the SAVE command (See 5.3 SAVE command).

Example	>SAVE test.pg	This saves the data in the same directory as KRterm in the computer hard disk.
	>SAVE My Documents¥ test.pg	This saves the data in the specified file.

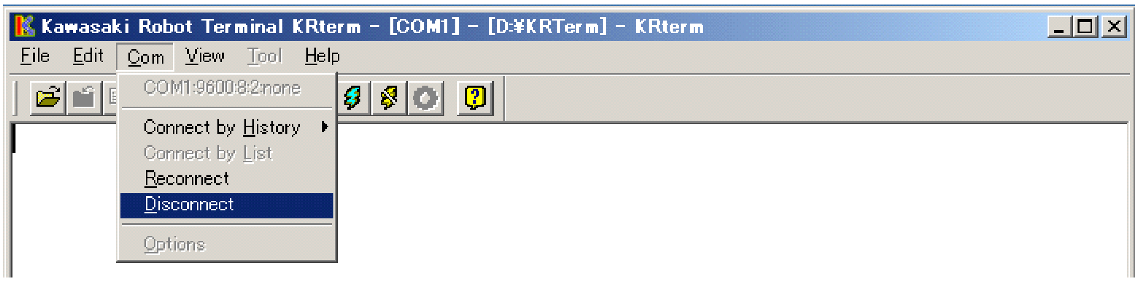
2. LOAD command

To load data from the computer to the robot memory, use the LOAD command.

Example >LOAD data01.as

2.6.3 System Shutdown

1. When the robot is connected, choose from the menu bar [Com(munication)] → [Disconnect] to disconnect the robot.



2. Turn off the robot controller. (See "Operation Manual" 3.2 POWER OFF procedure).

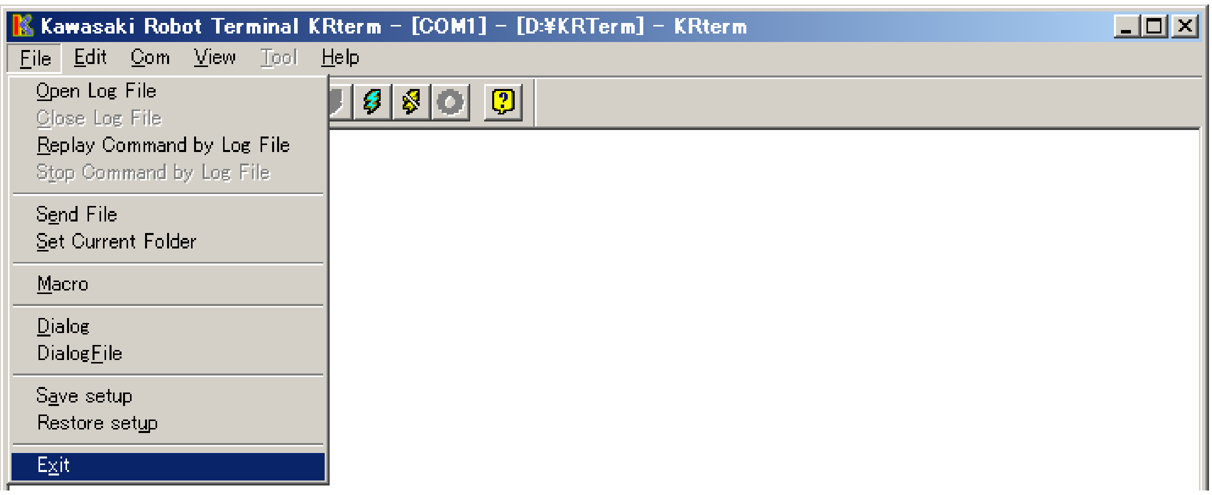
(1) Change HOLD/RUN state from RUN to HOLD.

(2) Turn OFF the motor power by pressing the EMERGENCY STOP button.

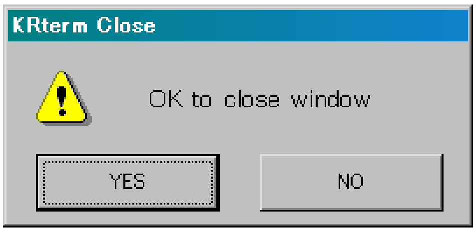
(3) Turn OFF the CONTROLLER POWER.

3. Shut down KRterm.

(1) Choose from the menu bar [File] → [Exit].



(2) Click <YES>.



4. Shut down the computer.

5. If there is no need to keep the computer connected to the controller, disconnect the cable.
Make sure the controller and the computer power are both turned off before disconnecting.

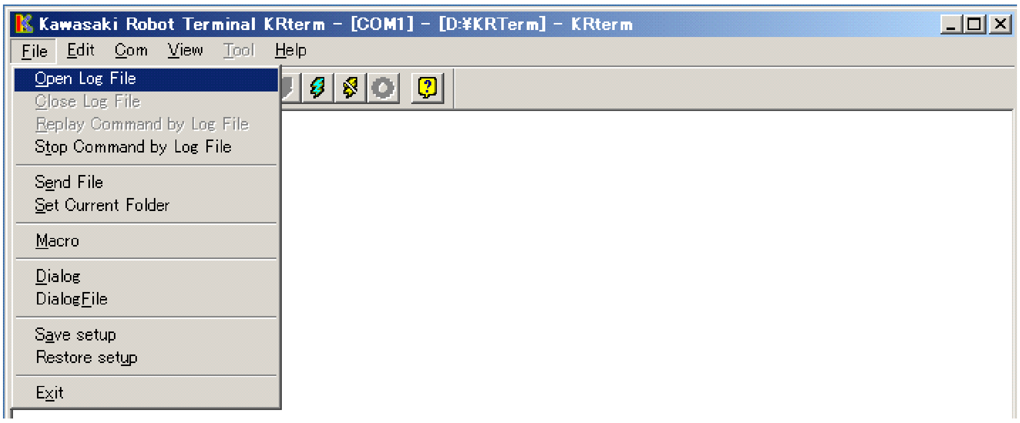
2.6.4 Useful Functions of KRterm

2.6.4.1 Creating Log Files

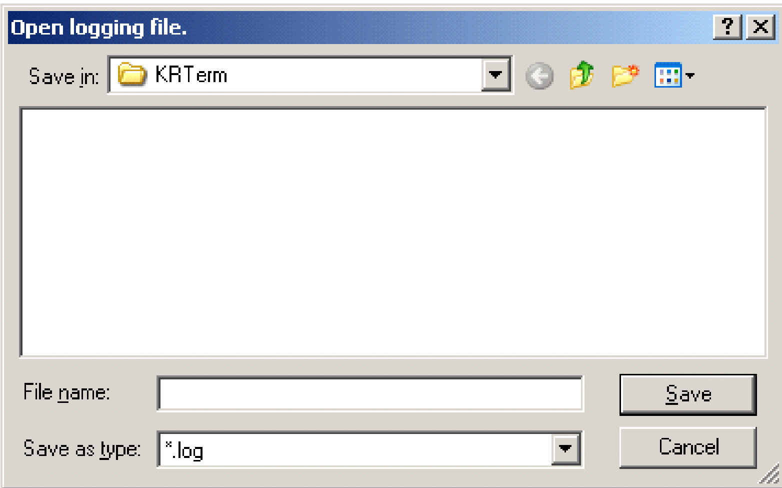
The contents displayed on the KRterm screen can be saved as a log file. This is useful when making printout of the robot operation procedures.

1. Start logging.

(1) Choose from the menu bar [File] → [Open Log File].



(2) Select the folder to save the log file, and name the file.



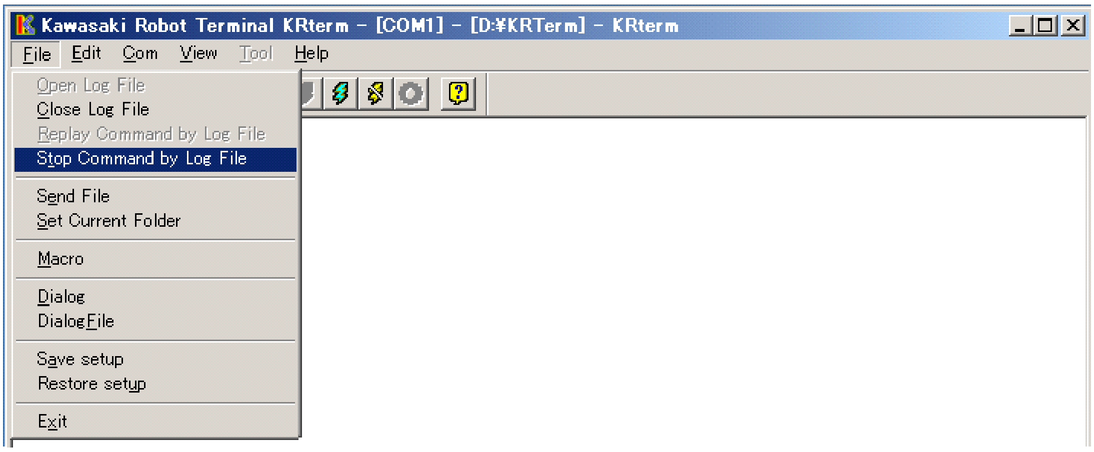
(3) The message [Logging Now] appears on the title bar. The contents on the display are recorded until the log file is closed.

The contents on the display are recorded while this message is shown.

2. End log

Once logging starts, all the contents on the KRterm display will be recorded until the log file is closed.

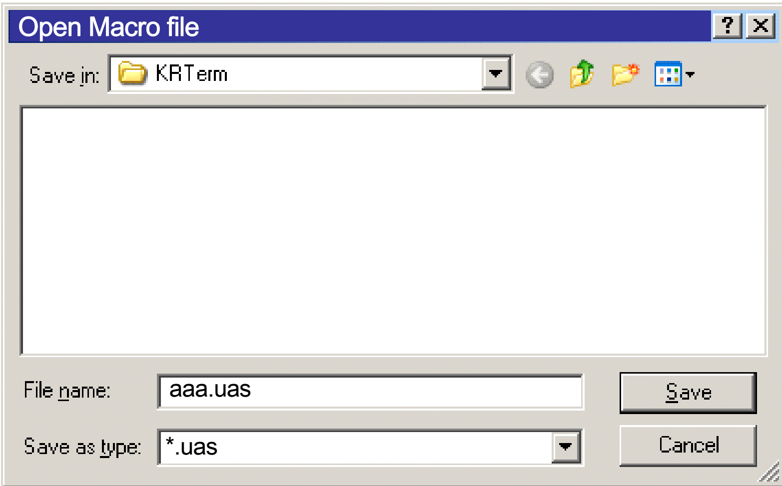
To close the log file and end log, choose from the menu bar [File] →
[Stop Command by Log File].



2.6.4.2 Macro Functions

Macro functions are provided in KRterm. If a task needs to be executed repeatedly, recording the series of instructions/commands for that task inside a macro can be very useful and will increase efficiency.

To record a macro, choose from the menu bar [FILE (F)] → [MACRO (M)] and enter the file name to save that macro. To run a macro, use the SEND command on the KRterm screen.



See Help in KRterm for more details.

3 Information Expressions in AS Language

This chapter describes the types of information and variables used in AS language.

3.1 Notation and Conventions

1. Uppercase and lowercase letters
   For easier understanding, the following rules apply to the usage of upper and the lowercase letters in this manual. All AS keywords (commands, instructions, etc) are shown in uppercase. Variables and any other items that can be specified are shown in lowercase. However, both can be used when entering at an AS terminal.

2. Keys and switches
   The keys on the teach pendant or the computer keyboard and the switches on the controller are expressed in this manual with their names surrounded by a □.
   Example Backspace, CONTROLLER POWER

3. Abbreviations
   Keywords can be abbreviated. For example, EXECUTE command can be abbreviated as EX. See Appendix 5 AS Language List.

4. Space, Tab
   At least one blank space or tab is necessary as a delimiter between the command (or instruction) and the parameter*. Also, a space or tab is necessary between those parameters not divided by commas or other delimiters. Excess spaces or tabs are ignored by the system.
   NOTE* A parameter is a data necessary for completing commands or other functions. For example, in SPEED command, parameter data is needed for specifying the robot speed. When the command or function uses several parameters, a comma or a space separates each parameter.
   Example SPEED 50

5. ENTER key
   Monitor commands and program instructions are processed by pressing the ENTER key. In this manual, the ENTER key is shown as ↵.

6. Omitted Parameters
   Many monitor commands and program instructions have parameters that can be omitted. If there is a comma after these optional parameters, the comma should be retained even if the parameter is omitted. If all successive parameters are omitted, comma may also be omitted.

7. Numeric values

Values are expressed in decimal notations, unless noted otherwise. Mathematical expressions can be used to designate these values as parameters in AS monitor commands and program instructions. However, note that acceptable values are restricted. The following rules show how the values are interpreted in various cases.

(1) Distance

Used to define the length the robot moves between two points. The unit for distance is millimeter (mm); the unit is omitted when entering. The input values can be either negative or positive.

(2) Angles

Describes the tool orientation and axis value by Euler's 3 angles and rotation angle of a robot joint, respectively. The values can be negative or positive, with the maximum angles limited to 180 degrees or 360 degrees, depending on the commands used.

(3) Scalar variables

Unless noted otherwise, these variables represent real values. The values for the variables can range from −3.4E+38 to 3.4E+38 (-3.4×10^{38} to $3.4+10^{38}$). When it exceeds ±999999, it is expressed as xE+y (x is the mantissa, y is an exponent).

(4) Joint number

Expresses the joints of the robot in integer from 1 to the number of joints available (standard type has 6 joints). The joints are numbered in order starting from the base joint. (Usually expressed JT1, JT2 ....).

(5) Signal number

Identifies binary (ON/OFF) signals. The values are given as integers and take the following ranges.

	Standard range	Maximum range
External output signal	1 – 16	1 – 960
External input signal	1001 – 1016	1001 – 1960
Internal signal	2001 – 2960	2001 – 2960

Negative signal numbers indicate OFF state.

8. Keywords

Generally, variable names can be freely assigned within the AS system. However, keywords defining commands, instructions, etc. in the AS system are reserved, and cannot be used to name pose data, variables, etc.

3.2 Pose Information, Numeric Information, Character Information

There are three types of information in the AS system: pose* information, numeric information, and character information.

NOTE* "Pose" was formerly called "location", but in accordance with the international standards (the ISO), in this manual, it is referred to as pose to express both the position and the orientation of the robot in one word.

3.2.1 Pose Information

Pose information is used to specify the position and orientation of the robot in the given working area. The robot's position and orientation refer to the position of the tool center point (TCP) and orientation of the tool (coordinates), unless otherwise specified. The position and orientation together is called the pose of a robot.

The pose is determined by where the robot is and which way it is facing, therefore, when a robot is instructed to move, these two things are done at the same time:

1. Robot's TCP moves to the specified position.
2. Robot's tool coordinates rotate to the specified orientation.

The pose data is described by a set of joint displacement values or by transformation value:

1. Joint displacement values

   This pose information is given by a set of angular or linear displacement values from each of the robot axes origins. Using encoder values, angular displacement and linear displacement are calculated and described in degrees and millimeters, respectively. Once the joint displacement values are determined, the position and orientation of the TCP is uniquely specified.

   Example The joints are expressed in order from JT1,…JT6, and the displacement value of each joint is shown beneath the joint number.

```
              JT1    JT2      JT3      JT4   JT5   JT6
   #pose = 0.00,  33.00,  -15.00,     0,   -40,   30
```

2. Transformation values (X,Y,Z,O,A,T)

Describes a pose of coordinates in relation with reference coordinates. Unless specified otherwise, it refers to the transformation values of the tool coordinates relative to the base coordinates of a robot. The position is given by the XYZ values of the TCP on the base coordinates, and the orientation is given by Euler's OAT angles* of the tool coordinates in respect with the base coordinates. Some of the commonly used transformation values are: the tool transformation values, describing the pose of the tool coordinates relative to the null tool coordinates, and workpiece transformation values, describing the pose of the tool coordinates relative to the workpiece coordinates.

NOTE* See Appendix 3 Euler's O,A,T Angles.

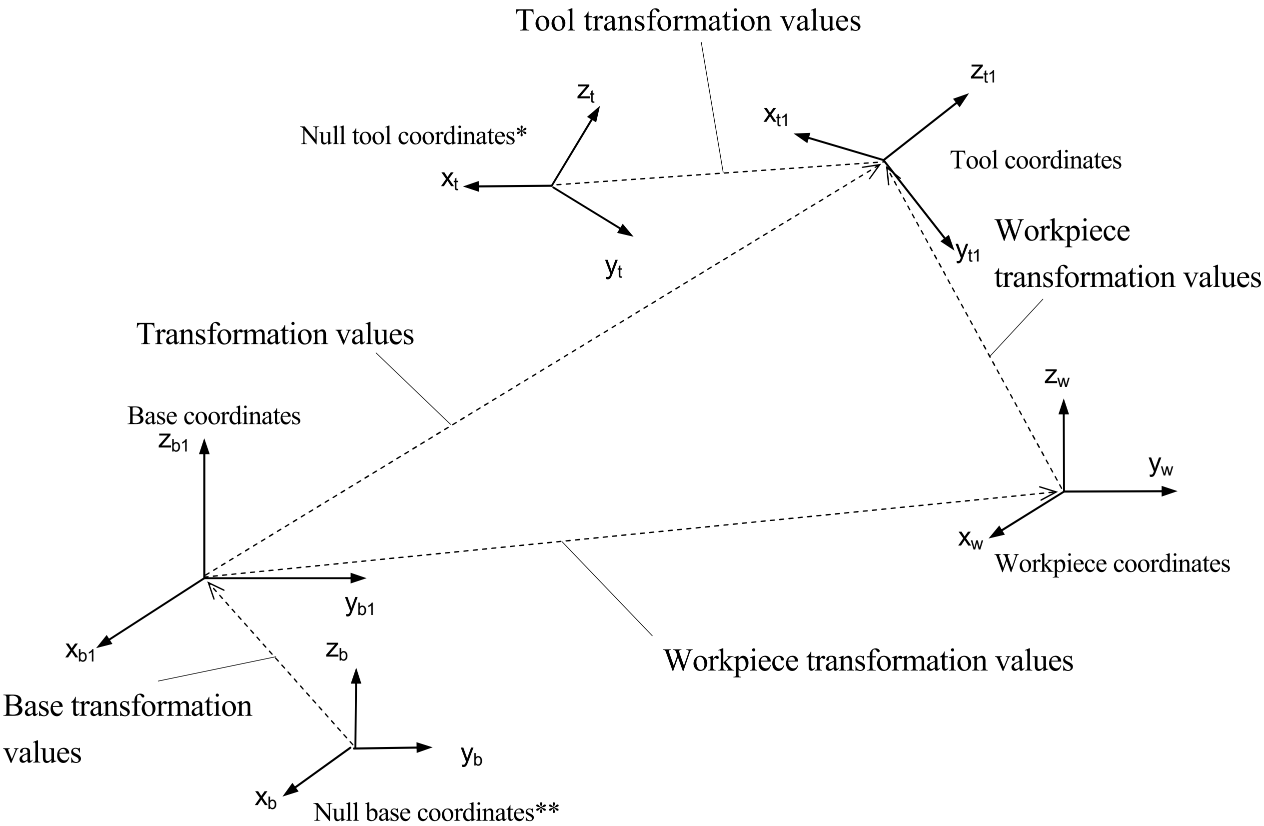
Example

	X	Y	Z	O	A	T
pose =	0,	1434,	300,	0,	0,	0

If the robot has more than six axes, the value of the extra axis is shown with the transformation values.

Example

	X	Y	Z	O	A	T	JT7
pose =	0,	1434,	300,	0,	0,	0	1000



NOTE* Null tool coordinates have their origin at the center of the robot's tool mounting flange surface, and they are described by the tool transformation values (0,0,0,0,0,0).

NOTE** Null base coordinates are set as the robot's default value, and are described by the base transformation values (0,0,0,0,0,0).

The joint displacement values and the transformation values have advantages and disadvantages. Use them to suit your need.

	Joint displacement values	Transformation values
Advantage	· Playback precision is achieved and there is no ambiguity about robot configuration at a pose	· The tool coordinates origin used in repeat mode does not change even if the tool is changed. (The null tool coordinates shift) · Can use relative coordinates. (e.g. workpiece coordinates) · Convenient for processing as the data are shown in XYZOAT values.
Disadvantage	· TCP changes when the tool is changed (null tool coordinates remain the same) · Cannot use relative coordinates (e.g. workpiece coordinates, etc.)	· Coordinates will change according to base or tool transformation values, so a full understanding is needed of the effect of any change for safe usage. · Robot configuration may change if it is not set before repeating movements.
Suggested usage	· Setting the starting pose of a program · Setting the robot configuration at or just before a pose described by transformation values · Use for other common poses	· Describing relative coordinates such as workpiece coordinates · Describing a pose that is to be changed using numeric values with functions such as SHIFT · Describing a pose that is to be changed by sensor information

[NOTE]

1. Unlike at a pose defined by joint displacement values, where the robot configuration is set uniquely, when a pose is defined by transformation values, the robot may take different configurations with respect to that pose. It is because transformation values only set the XYZOAT values of the tool coordinates of the robot and do not define the axis value of each joint. Therefore, before starting the robot in repeat mode, be sure to fix the robot's configuration using configuration commands (LEFTY, etc.) or by recording the joint displacement values.

2. Since transformation values are described by the base coordinates, if the base coordinates are shifted using the BASE command/instruction, the robot's TCP will also be shifted the same amount. This is one of the advantages of using the transformation values, but pay attention to the effect that changing the base coordinates will have on transformed points. Failure to do so may cause accidents such as interference with peripheral devices.

   Take the same caution when using the TOOL command/instruction.

3.2.2 Numeric Information

In the AS system, numeric values and expressions can be used as numeric information. A numeric expression is a value expressed by using numerals and variables combined with operators and functions. Numeric expressions are used not only for mathematical calculations, but also as parameters for monitor commands and program instructions.

For example in the DRIVE command, three parameters, joint number, motion amount, and speed, are specified. The parameters can be expressed either in numeric values or in expressions as in the following example:

DRIVE 3,45,75	Moves joint number 3 by 45° at the speed of 75%.
DRIVE joint, (start+30)/2, 75	When specified joint=2, start=30 then joint 2 moves by +30° at 75% speed.

Numeric values used in AS system are divided into three types:

1. Real numbers
   Real numbers can have both integers and fractions. It can be a positive or a negative value between −3.4 E+38 and 3.4 E+38(-3.4×10^{38} and 3.4×10^{38}) or zero. Real numbers can be represented in scientific notations. The symbol E divides between the mantissa and the exponent. The exponent may either be negative (power of 1/10) or positive (power of 10).

Example	8.5E3	8.5×10^{3}	(+ in the exponent is omitted)
	6.64	6.64×10^{0}	(E, 0 is omitted)
	–9E-5	-9.0×10^{-5}	(decimal point is omitted)
	–377	-377×10^{0}	(decimal point, E, 0 are omitted)

Note that the first seven digits are valid, but the number of valid digits might lessen through calculation procedures.

Real values without fractional parts are called integers. The range is from −16,777,216 to +16,777,215 and for those exceeding this limit, the first seven digits are valid. Integer values are usually entered in decimal numbers although there are times when it is convenient expressed in binary or hexadecimal notation. ^B states that the number entered is in binary notation. ^H states that the number entered is in hexadecimal notation.

Example	^B101	(5 in decimal)
	^HC1	(193 in decimal)
	–^B1000	(–8 in decimal)
	–^H1000	(–4096 in decimal)

2. Logical values
Logical values have only two states, ON and OFF, or TRUE and FALSE. A value of −1.0 is assigned for the TRUE or ON state, and a value of 0 (or 0.0) is assigned for FALSE or OFF state. ON, OFF, TRUE and FALSE are all reserved as AS language.

Logical true = TRUE, ON, −1.0
Logical false= FALSE, OFF, 0.0

3. ASCII values
Shows the numeric value of one ASCII character. The character is prefixed with an apostrophe (’) to differentiate from other values.

’A ’1 ’v ’%

3.2.3 Character Information

Character information referred to in the AS system is indicated as a string of ASCII characters enclosed in quotation marks (“”). Since the quotation marks indicate the beginning and the end of the string, they cannot be used as a part of the string. Also, the ASCII Control characters (CTRL, CR, LF, etc.) cannot be included in the string.

Example

```
>PRINT  “KAWASAKI”
```

command character string

3.3 Variables

In the AS system, names can be assigned to pose information, numeric information, and character information. These are called variables, and the variables can be divided into two types: global variables and local variables. Unless otherwise noted, global variables are referred to as variables.

3.3.1 Variables (Global Variables)

Variables for pose information, numeric information, and character information are called pose variable, real variable*, string variable, respectively. Several values can be grouped and be assigned to an array variable as array element values.

NOTE* Since most numeric values used in AS are real numbers, numeric variables are referred to as real number variables or real variables. However, note that integers, logical values and ASCII values are all expressed using real number values. Therefore, a real variable may refer to any of these values.

Once a variable is defined, it is saved with that value in the memory. Therefore, it can be used in any program.

3.3.2 Local Variables

In contrast with the global variables above, local variables are not saved in the memory at the time they are defined. They are saved in the memory when the step they are defined in is executed for the first time after the program is started. A variable with a "." (period) at the beginning of its name is considered a local variable.

Local variables are useful in cases when several programs use the same variable name wherein the value of the variable changes every time the program runs. Local variables can also be used as a parameter of a subroutine. (See also 4.4.2 Subroutine with Parameters.)

[**NOTE**]

1. Local variables cannot be defined using monitor commands.

2. The value of a local variable cannot be confirmed directly via monitor command. For example, inputting the monitor command as below will not display the current value or the local variable:

   ```
   >POINT. pose
   ```

   To see the current value of the local variable, set its value to a global variable in the program where the local variable is defined, and then use the POINT command.

   POINT a=.pose　　Execute the program that defines the local variable before using the POINT command.

   ```
   >POINT a
   X[mm]     Y[mm]     Z[mm]     O[deg]    A[deg]    T[deg]
   xxxxxxx   xxxxxxx   xxxxxxx   xxxxxxx   xxxxxxx   xxxxxxx
   Change?(If not hit RETURN key) ↵
   ```

3.4 Program and Variable Names

Program and variable names must start with an alphabetical character and can contain only letters, numbers, periods, and underscores. The letters can be entered either in uppercase or lowercase (it will appear in lowercase on the display screen). The length of the program and variable names are limited to 15 characters. Only the first 15 characters will be valid for names with more than 15 characters. The following are some examples of names that cannot be used:

3p••••••••••••••••••••the first letter is not an alphabet
part#2••••••••••••••••”#” is prefix for joint displacement value variable name and cannot be used in middle of a variable name
random•••••••••••••••keyword

[NOTE]

1. Variables describing joint displacement values are preceded by the symbol “#” to differentiate them from transformation value variables. Character string variables are preceded by “$” to differentiate them from real value variables. Variables with integers only are prefixed with an @ before their names.

pick	(transformation value variable)		
#pick	(joint displacement value variable)		
count	(real value variable)	$count	(string variable)
@count	(integer variable)		

2. All variables can be used as array variables. Arrays consist of several values under the same name and these values are distinguished from each other by their index value. Each value in the array is called an array element. To specify an array element, attach an element index value enclosed in brackets. For example, “part [7]” indicates the seventh element of the array “part”. For the indexes, use integers within the range 0 to 9999. For three-dimensional arrays use syntax similar to this: part [7, 1, 1]=1.

3. When a variable is defined, that variable can be used in various programs. Therefore, be careful not to make unnecessary changes to variables that are used in different programs.

3.5 Defining Pose Variables

Variables that describe pose information are called pose variables. A pose variable is defined only when it is given a name and a value is assigned to it. It remains undefined until a value is assigned, and if a program using an undefined variable is executed, an error occurs.

Pose variables are useful in the following ways:

1. The same pose data can be used repeatedly without teaching the pose every time.
2. A defined pose variable may be used in different programs.
3. A defined pose variable can be used or changed to define a different pose.
4. Numeric values can be directly input for specifying pose information instead of time consuming process of teaching poses to the robot using the teach pendant.
5. Pose variables can be named freely, so programs can be made more legible.

Pose variables are defined as follows.

3.5.1 Defining by Monitor Commands

1. HERE command stores the robot's current pose data as the value of the pose variable with the specified name.

   Example 1 Using joint displacement values
   Start the variable name with # to differentiate it from transformation values.
   Following the command, the joint displacement values of the current pose will appear:

```
              > HERE   #pose↵
 JT1      JT2      JT3      JT4      JT5      JT6
xxxxxxx  xxxxxxx  xxxxxxx  xxxxxxx  xxxxxxx  xxxxxxx
Change?   (if not, hit RETURN only)↵
>
```

   Example 2 Using transformation values
   Following the command, the transformation values of the current pose will appear:

```
>HERE pose↵
 X[mm]    Y[mm]    Z[mm]    O[deg]   A[deg]   T[deg]
xxxxxxx  xxxxxxx  xxxxxxx  xxxxxxx  xxxxxxx  xxxxxxx
Change?(if not, hit RETURN only)  ↵
>
```

2. POINT command is used to define a pose using another defined pose variable or, to define it by the numerical data entered from the terminal.

Example 1 Using joint displacement values

(1) Defining a new, undefined variable

```
>POINT   #pose ↵
    JT1       JT2      JT3       JT4       JT5       JT6
    0.000     0.000    0.000     0.000     0.000     0.000
Change? (if not, hit RETURN only) ↵
    >
```

Enter the new values by separating each value with a comma:

xxx, xxx, xxx, xxx, xxx, xxx

(2) Changing the value of a defined variable

```
>POINT   #pose ↵
    JT1       JT2       JT3       JT4       JT5       JT6
    10.000    20.000    30.000    40.000    50.000    40.000
Change? (if not, hit RETURN only) ↵
```

Enter the value to be changed:

30, , , ,20, ;changes the value of JT1 and JT 5 to 30 and 20

(3) Substitute the value of a defined variable

```
>POINT   pose_1=pose_2 ↵
    JT1       JT2       JT3       JT4       JT5       JT6
    10.000    20.000    30.000    40.000    50.000    40.000
Change? (if not, hit RETURN only) ↵
```

The value to be defined as pose_1 (the recent value of pose_2) appears. Hit ↵ to set the values as they are, or change them in the same procedure as in (2) above.

Example 2 Using transformation values

Follow the same procedures as above, only the variable name should not start with #.

[NOTE]

For joint displacement value variable, define the variable with its name starting with #.
For transformation value variable, define the variable without the #.

3.5.2 Defining by Program Instructions

1. HERE instruction stores the robot’s current pose as the values of the pose variable with the specified name.

   HERE pose

2. POINT instruction substitutes the specified pose variable values with the values from a previously defined pose.

   POINT pose_1=pose_2

   Values of “pose _1” are substituted with the values of the defined variable “pose_2”. An error will occur if “pose _2” is not defined.

[NOTE]

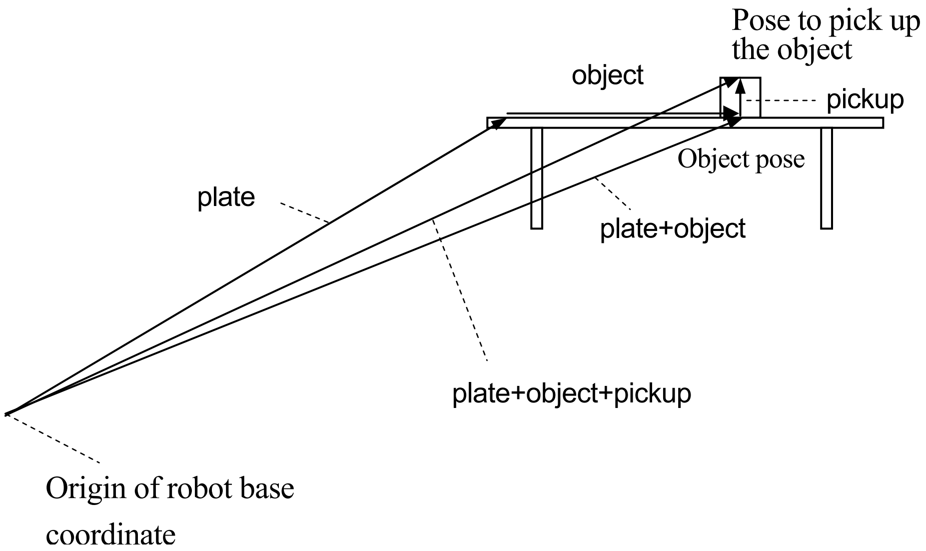
For joint displacement value variable, define the variable with its name starting with #.
For transformation value variable, define the variable without the #.

3.5.3 Using Compound Transformation Values

The transformation values between two coordinates can be expressed as a combination of transformation values between two or more transitional coordinates. This is called compound transformation values or relative transformation values.

For example, say that “plate” is the name of the variable defined by the transformation values relative to the base coordinates describing the coordinates at the table where the object is placed. Then, if the pose of an object relative to the pose “plate” is defined as “object”, the compound transformation values of the object relative to the robot base coordinates can be described as “plate+object”.

In the example below, even if the pose “plate” changes (e.g. the table moves), only the transformation values for “plate” will need revising and the rest can be used as is.



The compound transformation values can be defined in various ways. Normally, the transformation values for a coordinate in reference with the robot base coordinate is defined. Then the next transformation values for the coordinate in reference to that coordinate are defined, and so on. The transformation values can be defined using any command or instruction used to define pose variables. (It is easiest to use the HERE command/ instruction.)

First, use the teach pendant to move the robot tool to the pose that is to be named “plate”. Then, enter as below to define that pose as plate.

```
>HERE   plate ↵
```

Next, move the robot tool to the pose to be named “object” and enter:

```
>HERE   plate + object
```

The transformation value “object” now defines the current pose relative to “plate”* (If “plate” is not defined at this point, “object” will not be defined and an error will occur).

NOTE* What appears on the screen after entering the HERE command is the transformation values of the pose for the rightmost variable (i.e. “object” in this case). It is not the values for “plate + value”. To see the values for “plate +object”, use the WHERE command when the robot is at that pose.

Finally, move the robot hand to the pose where it picks up the object and enter:

```
>HERE   plate + object + pickup ↵
```

This last command defines “pickup” relative to the transformation values “object”.

As shown above, compound transformation values are defined by a combination of several transformation values separated by "+". Do not include any spaces in between the "+" and the transformation values. Using this method, you can combine as many transformation values as needed.
If the robot is to pick up the object at the pose specified as "pickup" defined relative to "object", the program will be written as follows:

```
        JMOVE   plate+object+pickup
or      LMOVE   plate+object+pickup
```

[NOTE]

1. Do not change the order in which the relative transformation is expressed. For example, if the transformation value of pose variable "b" is defined relatively to transformation value of pose variable "a", "a+b" results as expected, but "b+a" may not.
2. The pose data "object" and "pickup" from the example above are defined in relation to other pose data. Therefore, do not use commands such as "JMOVE object" or "LMOVE pickup" unless you are certain of its purpose and its effect on the program.

When using compound transformation values repeatedly, use the POINT command to lessen the time to calculate the compound transformation values. For example, to approach the pose "pickup" and then to move to that pose, you might enter:

```
JAPPRO  plate + object + pickup, 100    approach 100 mm above "pickup"
LMOVE   plate + object + pickup         move in linear motion to "pickup"
```

Instead, if you enter as below, this will save calculation time:

```
POINT   x = plate + object + pickup     calculate the target pose
JAPPRO  x, 100                          approach 100 mm above the target
LMOVE   x                               move in linear motion to the target
```

These two programs result in the same motion, but the latter calculates the compound transformation only once, so the execution time is shorter. In such simple examples, the difference will be minor, but in more complex programs, it may make a big difference and improve overall cycle time.

[NOTE]

For robots with 7 joints, note the following:

1. When using POINT command, note the value of JT7. For example, in

   ```
   POINT p=p1+p2
   ```

   The value of JT7 assigned to "p" will be the value of JT7 for "p2". The value of the rightmost variable on the right side of the expression is assigned to the variable p on the left side as JT7 value.

2. When assigning a specific value to JT7, add "/7" to the end of the POINT command. For example,

   ```
   POINT/7 p = TRANS(,,,,,,value)
   ```

   assigns "value" to the variable "p" as JT7 value.

3.6 Defining Real Variables

Real variables are defined by using the assignment instruction (=). The format for assigning a real variable is:

Real variable = numeric value

Example

```
a=10.5
count=i*2+8
Z[2]=Z[1]+5.2
```

The variable on the left side may be either a scalar variable (i.e., count) or an array element (i.e., Z[2]). A variable is defined only when a value is assigned to it. It remains undefined until a value is assigned, and if a program using an undefined variable is executed, an error occurs.

The numeric value on the right side may be a constant, a variable or a numeric expression. When the assignment instruction is processed, the value on the right side of the assignment instruction is computed first, and then the value is assigned to the variable on the left side.

If the variable on the left side of the instruction is a new one and has never been assigned a value before, the value on the right is assigned to that variable automatically. If the left side variable is already defined, the new value will replace the current value.

For example, the instruction “x=3” assigns the value 3 to the variable “x”. It is read, “assign 3 to x” and not “x is equal to 3”. The following example illustrates the processing order clearly:

```
x= x+1.
```

If this example is a math equation, it is read “x is equal to x plus 1”, which does not make sense. As an assignment instruction, it is read, “assign the value of x plus 1 to x”. In this case, the sum of the current value “x” and 1 is calculated and then the resulting value is assigned to “x” as a new value. Such an equation requires that x be defined in advance, as below:

```
x=3
x=x+1
```

In this case, the resulting value of “x” is 4.

3.7 Defining Character String Variables

Character string variables are defined by using the assignment instruction (=). The format for assigning a character variable is:

$string variable=character string value

Example $a1=$a2
$error mess[2]="time over"

The string variable on the left can be a variable (i.e., $name), or an array element (i.e., $line[2]). A variable with specified name is defined only when a value is assigned to it. It remains undefined until a value is assigned, and if a program using an undefined variable is executed, an error occurs.

The character string on the right side may be a string constant, a string variable or a string expression. When an assignment instruction is processed, the value on the right side is computed first, and then the value is assigned to the variable on the left side.

```
$name = "KAWASKI   HEAVY   INDUSTRIES   LTD."
```

In the above instruction, the string enclosed in “” will be assigned to the variable “$name”. If the variable on the left side of the instruction has never been used before, this string will be assigned automatically. If the left side variable is already defined, the new value specified on the right side will replace the current value.

3.8 Numeric Expressions

Numeric expressions may consist of numerals, variables, specific functions or other numeric expressions combined together with operators. All numeric expressions evaluated by the system result in real number values. Numeric expressions can be used anywhere in place of numeric values. They can be used as parameters in monitor commands and program instructions, or as array indexes.

The interpretation of the value depends on the context in which the expression appears. For example, an expression specified for an array index is interpreted as yielding an integer value. An expression specified for a logical value is interpreted as false when it is evaluated as 0, and true if it is other than 0.

3.8.1 Operators

For describing expressions, arithmetic, logical, and binary operators are provided. All the operators combine two values to obtain a single resulting value. Exceptions: the two operators (NOT and COM) operate on a single value and the operator (−) operates on one or two values. The operators are described below.

Arithmetic Operators	+	Addition
	−	Subtraction or negation
	*	Multiplication
	/	Division
	^	Power
	MOD	Remainder
Relational Operators	<	Less than
	<=, =<	Less than or equal to
	==	Equal
	<>	Not equal to
	>=, =>	Greater than or equal to
	>	Greater than
Logical Operators	AND	Logical AND
	NOT	Logical complement
	OR	Logical OR
	XOR	Exclusive logical OR
Binary Operators	BAND	Binary AND
	BOR	Binary OR
	BXOR	Binary XOR
	COM	Complement

[NOTE]

1. Relational operator “==”is a operator to check if the two values are equal, and different from the assignment indicator “=“.

2. Binary operator BOR performs OR operation for the respective binary bit of two numeric values. (In this example the value is expressed in binary notation, but this operation may be used with any notation.)

   ^B101000 BOR ^B100001 → ^B101001

   This result is different from what you can get in OR operation.

   ^B101000 OR ^B100001 → -1(TRUE)

   In this case, ^B101000 and ^B100001 are interpreted as logical values, and since neither is 0 (FALSE), the expression is evaluated as TRUE.

3.8.2 Order of Operations

Expressions are evaluated according to a sequence of priorities. The priority is listed below, from 1 to 14. Note that the order of operations can be controlled using parentheses to group the components of an expression. With expressions containing parentheses, the expression within the innermost pair of parentheses is evaluated first, and then the system works toward the outer most pair.

1. Evaluate functions and arrays
2. Process relational operators concerning character strings (See 3.9 String Expressions)
3. Process power operator “^”
4. Process unary operators “-“(negation), NOT, COM
5. Process multiplication “*” and division”/” from left to right
6. Calculate remainder (MOD operation) from left to right
7. Process addition”+” and subtraction”-“ from left to right
8. Process relational operators from left to right
9. Process BAND operators from left to right
10. Process BOR operators from left to right
11. Process BXOR operators from left to right
12. Process AND operators from left to right
13. Process OR operators from left to right
14. Process XOR operators from left to right

3.8.3 Logical Expressions

Logical expressions result in logical value TRUE or FALSE. A logical expression can be used in a program as a condition to determine the next operation in a program. In the following example, a simple logical expression, “x>y”, is used in a subroutine to determine which of the two variables to assign to variable “max”.

```
IF x>y GOTO 10
max=y
GOTO 20
10          max=x
20 RETURN
```

When evaluating logical expressions, the value zero is considered FALSE and all nonzero values are considered TRUE. Therefore, all real values or real value expressions can be used as a logical value.

For example, the following two statements have the same meanings.

```
IF x   GOTO   10
IF x<>0 GOTO 10
```

However, the second statement shows the logical operator clearly and is easier to understand. It is recommended to use the logical operators.

3.9 String Expressions

String expressions consist of character strings, string variables, specific functions or other string expressions combined together with operators. The following operators are used with the string expressions.

String operator	+	Combine
Relational operators	<	Less than
	<=, =<	Less than or equal to
	==	Equal to
	<>	Not equal to
	>=, =>	Greater than or equal to
	>	Greater than

The result of using the string operator will be a string, and that of using relational operators will be a real value.

When using relational operators with character strings, the strings are compared character for character from the first character in the string. If all the characters are the same, the two strings are considered equal, but if there is even one difference, the string with the character having higher character code is evaluated as the greater string. If one of the strings is shorter, the shorter one is evaluated less. In relational operations with strings, spaces and tabs are regarded as a character.

```
"AA"      <    "AB"
"BASIC" == "BASIC"
"PEN."    >    "PEN"
"DESK"    <    "DESKS"
```

[NOTE]

Uppercase and lowercase letters in string expressions are regarded as different characters.

4 AS Program

This chapter explains about AS programs. It explains about programming and execution of programs, and about the robot motions. For better understanding, actually operate the actual system or PC-ROSET* as you read this chapter.

NOTE* PC-ROSET is a personal computer robot simulator compatible with the AS system.

4.1 Types of AS Programs

A program is a series of instructions telling the robot how to move, output signals, do calculations etc. per a set process. A program name consists of no more than 15 characters starting with an alphabetical character, and can contain only letters, numbers, and periods. You can create as many programs as the memory can store. Programs are usually created using the AS system editor mode, but you may also use a separate computer loaded with KRterm or PC-ROSET and later load it to the robot memory.

4.1.1 Robot Control Program

Robot control programs are programs that control the robot movements. You may use all the program instructions including robot motion instructions to create these programs.

4.1.2 PC Program (Process Control Program)

PC or process control programs are programs executed simultaneously with the robot control programs. PC programs are commonly used to control or monitor external devices by monitoring external I/O signals. The PC program and the robot control program can communicate with each other by using common variables or internal signals.

PC programs and robot control programs use instructions in common. Therefore, in some cases, a PC program can be executed as a robot control program. However, motion instructions other than BRAKE instruction cannot be used in PC programs. BASE and TOOL functions are also not available for PC programs.

[NOTE]

The execution time of each step in the program differs according to the instruction included in the program, and the number of programs running simultaneously. If the execution time needs to be shortened, take countermeasures such as to halt the execution of other programs using wait instructions such as TWAIT, or reduce the number of programs running at the same time, etc.

4.1.3 Autostart

A PC program can be set to start automatically when the controller power is turned ON.

1. Turn ON the system switch AUTOSTART.PC (or AUTOSTART2.PC – AUTOSTART5.PC).

2. Create the program you want to start automatically and name it AUTOSTART.PC (or AUTOSTART2.PC – AUTOSTART5.PC).

Some monitor commands can be executed in programs by using program instruction MC.

This is a sample autostart program. In this example, after the controller power is turned ON, the robot monitors for motor power ON and executes program pg1 when it is turned ON. For easier understanding safety checks are ignored here, but in actual usage, be sure to include safety check procedures.

```
autostart.pc( )
1   WAIT SWITCH (POWER)   ;waits for motor power ON
2   WAIT SIG(27)          ;checks if the robot is at home pose*
3   MC EXECUTE pg1        ;Executes pg1(robot motion program)
```

NOTE* Set home pose and assign the dedicated signal HOME1 to signal 27, before executing this program.

4.2 Creating and Editing Programs

In this section, a simple program is made to instruct the robot to perform a task. A program is a list of procedures that the robot will be made to do. When executing a program through the AS system, program steps (lines) are processed in order from top to bottom and the operations defined in each step are carried out by the robot.

4.2.1 AS Program Format

Each line (step) of an AS language program is expressed in the following format.

step number label program instruction ;comment

1. Step number
   A step number is automatically assigned to each line of a program. Steps are numbered consecutively beginning with 1 and are automatically renumbered whenever lines are inserted or deleted.

2. Label
   Labels are used in a program to branch the program. A label can be a string of up to 15 alphanumeric characters, periods, or underscores, which starts with an alphabetical character or integer, followed by a colon (:). Labels are inserted at the beginning of a program line, right after the step number. Labels can be used as branch destinations from anywhere within the program.

3. Comment
   A semicolon (;) indicates that all information to the right of the semicolon is a comment. Comments are not processed as program instructions when the program is executed, and are only used for explaining the program contents. You can make a program line with only a comment and no label or instruction. Blank lines can also be made to improve program legibility. (A blank line consists of at least one space or tab after the semicolon.)

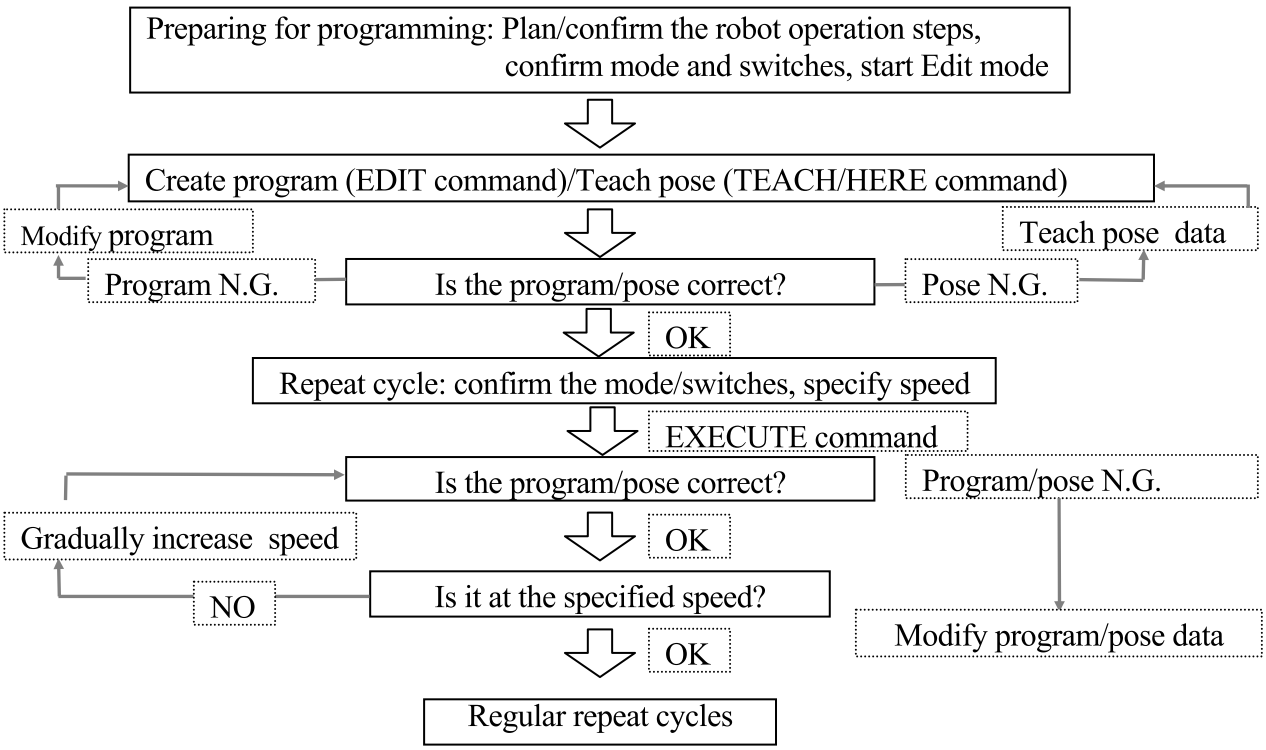
4.2.2 Editor Commands

The following editor commands are used to create and edit programs. (Highlighted parameters can be omitted.)

EDIT program name, step	Starts editor mode.
Program instructions	Replaces the current steps with a new instruction.
ENTER key (↵)	Goes to the next step without changing the current step.
D number of steps	Deletes specified number of program steps. (Delete)
E	Exits editor mode, and returns to monitor mode. (Exit)
F character string	Searches characters and displays that line. (Find)
I	Inserts a new step.
L	Displays the previous step. (Last)
M /existing characters /new _characters	Replaces the existing characters with new characters. (Modify)
O	Places the cursor on current step for editing. (One line)
P number of steps	Displays specified number of program steps. (Print)
R character string	Replaces characters within a step.
S step number	Selects program step. (Step)
XD	Cuts the selected step or steps and stores in clipboard.
XY	Copies the selected step or steps and stores in clipboard.
XP	Pastes the content of clipboard.
XQ	Pastes the content of the clipboard in the reverse order.
XS	Shows the contents of the clipboard.
T	Teaches while in editor mode (option).

4.2.3 Programming Procedures

Programming is done as shown in the following steps:



4.2.4 Creating Programs

In an AS program, two things have to be taught to the robot:

1. Working conditions for the robot
2. Path (pose) to be followed by the robot tool

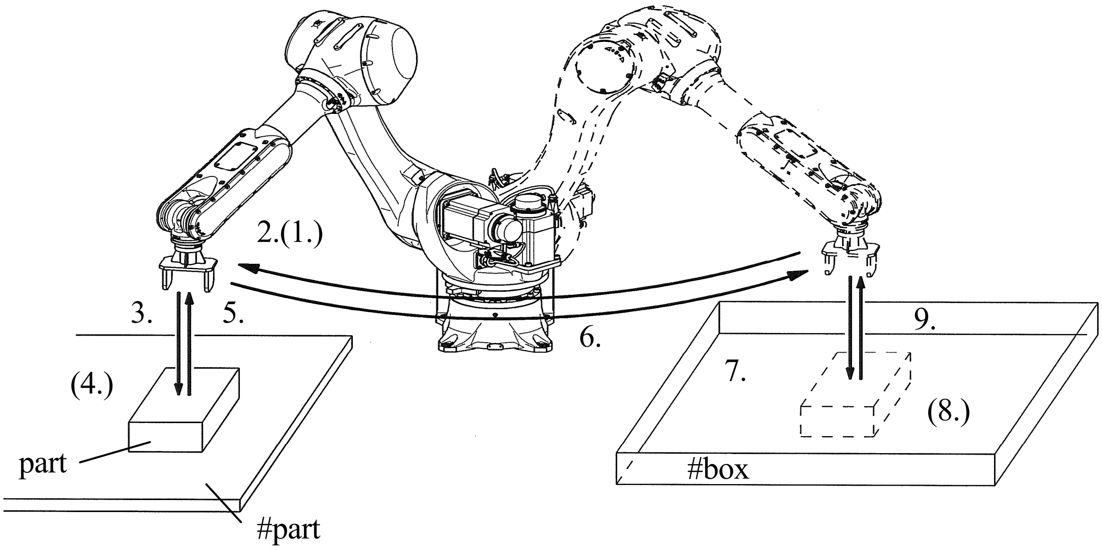
Here is a sample program.　The robot will perform the task shown on the next page: pick up a part fed in by the supply shoot (conveyor), and place it in the box.

First define all the motions required to complete the task:

1. Check if the hand is open.
2. Move to a pose 50 mm above the part (#part) on the supply shoot.
3. Move straight down to the part (#part).
4. Close the hand and grab the part.
5. Move straight up 150 mm above the supply shoot.
6. Move to a position 200 mm above the box (#box).

7. Move the part down into the box.
8. Open the hand and release the part.
9. Move back up to a position 180 mm above the box.

The variables #part and #box which express the position and the orientation of the robot are called pose data in the AS system. Define the pose variables as shown in Chapter 3 before executing the program.



Programs are created and edited via AS Editor. To create a program named “demo”, enter “EDIT demo ↵”. The screen should appear as follows:

```
> EDIT demo ↵
  .PROGRAM demo
  1 ?
```

Now, AS is waiting for the first step to be entered. Enter “OPENI ↵” after “1?”

```
> EDIT demo
 .PROGRAM demo
 1 ? OPENI ↵
 2 ?
```

Next enter “JAPPRO #part, 50 ↵”for the second step.

```
> EDIT demo
  .PROGRAM demo
  1 ? OPENI
  2 ? JAPPRO #part, 50 ↵
  3 ?
```

Enter the rest of the program in the same manner. Correct any mistakes when entering the steps by pressing Backspace before pressing ↵.

If the ↵ key is hit at the end of an erroneous step, error message appears and that step is rejected. In this case, enter the step again. When the entire program has been entered, the screen should appear as follows:

```
>EDIT demo
    .PROGRAM
   1 ? OPENI
   2 ? JAPPRO #part,50
   3 ? LMOVE #part
   4 ? CLOSEI
   5 ? LDEPART 150
   6 ? JAPPRO #box,200
   7 ? LMOVE #box
   8 ? OPENI
   9 ? LDEPART 180
  10 ? E  ↵
>
```

The last step “E ↵” is not a command for the robot but a command to exit the Editor mode (see the table in 4.2.1). The program is now complete. When the program is executed, the AS system follows the steps in order, from step 1 to step 9.

See 11.0 Sample Programs for further information on how to create programs.

4.3 Program Execution

The robot control programs and the PC programs are executed in different ways.

4.3.1 Executing Robot Control Programs

To execute a program, turn the TEACH/REPEAT switch to REPEAT position. Next, ensure the TEACH LOCK switch on the teach pendant is in the OFF position. Then, turn ON the motor power and change the HOLD/RUN state from HOLD to RUN.

1. Running program via EXECUTE command

   First, set the monitor speed. The robot will move at this speed when the program is executed. The speed should be set under 30%, with the initial setting at10%.

   > SPEED 10 ↵

   To start execution, use the EXECUTE command. Type as below:

   > EXECUTE demo ↵

   The robot should then perform the selected task. If it does not move as expected, change from RUN to HOLD. The robot will decelerate and stop. In case of emergency, press the EMERGENCY STOP button on the controller operation panel or on the teach pendant. The brakes are applied and the robot stops immediately.

   If the robot moves correctly at 10% speed, gradually raise the speed.

   > SPEED 30 ↵
   > EXECUTE demo ↵ The robot operates at 30% speed.

   > SPEED 80 ↵
   > EXECUTE demo ↵ The robot operates at 80% speed.

   After the EXECUTE command has been issued at least once, A + CYCLE START can be used to execute programs.

   To execute the program more than once, enter the number of repetitions after the program name:

   > EXECUTE demo,5 ↵ Executes 5 times.
   > EXECUTE demo,–1 ↵ Runs the program continuously.

2. Running program via PRIME command
   Set the monitor speed in the same way as with the EXECUTE command, and execute PRIME command.

   >PRIME demo ↵

   Robot is now ready to execute the program. Pressing A + CYCLE START begins execution. Execution can also be started using the CONTINUE command.

3. Running program via STEP command or CHECK GO/BACK key
   It is possible to check the motion and the contents of a program by executing the program step by step. Use either the STEP monitor command or the CHECK GO/BACK key* on the teach pendant.

   NOTE* When using the CHECK GO/BACK key, the program execution pauses at the end of each motion instruction.

During execution of the robot control program, some monitor commands are disabled. Likewise, the EXECUTE command cannot be entered twice during execution.

4.3.2 Stopping Programs

There are several ways to stop a program in progress. The following three are described in order from most to least urgent.

1. Press the EMERGENCY STOP button either on the controller panel or on the teach pendant. Breaks are applied and robot stops immediately. Unless there is an emergency, use methods 2 and 3.

2. Change from RUN to HOLD. The robot slows down and stops.

3. Entering the ABORT command stops the program execution after the robot completes the current step (motion instruction).

   > ABORT ↵

   HOLD command can also be used to stop execution.

   > HOLD ↵

4.3.3 Resuming Robot Control Programs

Depending on how the program was stopped, there are several methods to resume the program.

1. When the robot was stopped with EMERGENCY STOP button, release the lock of EMERGENCY STOP, and turn ON the motor power. Robot starts moving when you press A + CYCLE START.

2. When HOLD was used to stop the robot, press A+ RUN to change to RUN.

3. To resume after ABORT or HOLD command or when program execution was suspended by an error, use CONTINUE command. (When restarting after an error, the error should be reset before resuming the program.)

```
> CONTINUE ↵
```

4.3.4 Executing PC Programs

PC programs are executed by PCEXECUTE monitor command or by a program instruction that is executed from within a robot control program. PCABORT command can be used to stop execution of the PC program at any time. PCEND command ends the execution of the program after the current cycle is completed.

PCCONTINUE command resumes execution of a program suspended by either PCABORT or because of an error. (When restarting after an error, the error should be reset before resuming the program.)

4.4 Program Execution Flow

The program instructions are regularly executed in order from top to bottom of the program. This consecutive flow is changed when there is an instruction such as GOTO or IF....GOTO. A CALL instruction calls up and executes a different program, but this does not change the order of the flow. When a RETURN instruction is executed, the processing returns to the caller program and resumes from where it has left.

WAIT instruction stops the program from proceeding to the next step until the specified condition is met. PAUSE and HALT instructions stop the program at the step where these instructions are used.

STOP instruction may not stop the execution in some cases. If the specified execution cycles remain, execution continues with the first step in the main program. (Even if the STOP instruction is executed in a subroutine, the execution returns to the beginning of the main program.) If there are no cycles remaining, the execution stops at the step where the instruction is used.

4.4.1 Subroutine

A main program can be temporarily suspended and a different program, called the subroutine, can be called up and executed. By using the subroutine, you can make the program into a modular structure that is easier to understand.

4.4.2 Subroutine with Parameters

Parameters can be used with subroutines for more convenience. For example, when a calculation that uses different input data is done repetitively, create a subroutine to do the calculation. Use the CALL instruction to branch to the subroutine, and use the input data as parameters in the calculation. (See examples 1 and 2 below)

Up to 25 parameters can be set using real variables, pose variables or string variables. The variable type must be the same in the main program and the subroutine. When assigning transformation values to a parameter put a “&” in front of the parameter variable name in order to differentiate it from real number variables. Also, use local variables in the CALL destination (subroutine).

Example 1 The value of real number variable “c” is the sum of input data “a” and “b”.

```
main()
 1  a=1
 2  b=2
 3  CALL calc(a,b,c)
 4  TYPE c
 calc(.aa,.bb,.cc)
 1  .cc=.aa+.bb
```

Example 2 The value of transformation value variable “c” is the sum of transformation value of “a” and “b”.

```
pose()
 1  point a = trans(10)
 2  point b = trans(0,20)
 3  CALL add(&a,&b,&c)
 4  point d = c
 add(.&aa,.&bb,.&cc)
 1   point .cc=.aa+.bb
```

[NOTE]

To set parameters in the subroutine, as in example 1 above, enter “EDIT calc, 0” then the following appears in the display:

```
0.()
0?
```

Enter (.aa,.bb,.cc) after the ?.

4.4.3 Asynchronous Process (Interruption)

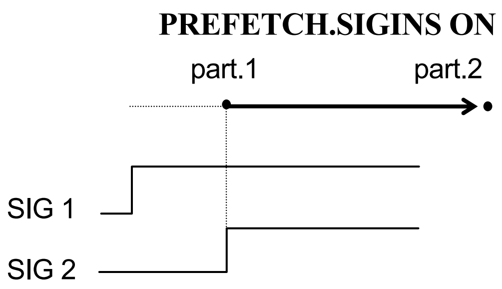
Under certain conditions, like when an error occurs or when a specific external signal is input, program execution may be interrupted and another program will be executed. This occurs independently from the flow of execution of the main program and is called asynchronous processing (interruption). As soon as the specified signal (e.g. an external signal or an error) is detected, the interruption occurs regardless of the execution of the main program. This process is activated using the ON (or ONI) ...CALL instruction.

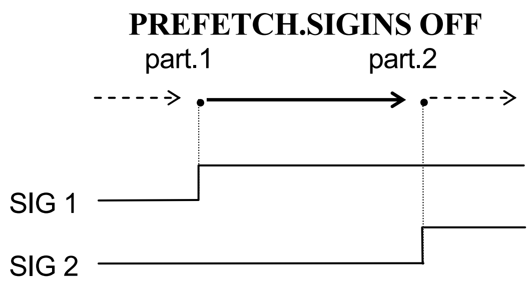
4.5 Robot Motion

4.5.1 Timing of Robot Motion and Program Step Execution

In the AS system, the timing of program execution and of the robot motion can be changed by setting the system switches. For example, the timing of step execution changes as following when PREFETCH.SIGINS switch is turned ON (allow early processing of signal I/O commands) or OFF (not allow early processing of signal I/O commands).

```
JMOVE part1
SIGNAL 1
JMOVE part2
SIGNAL 2
```



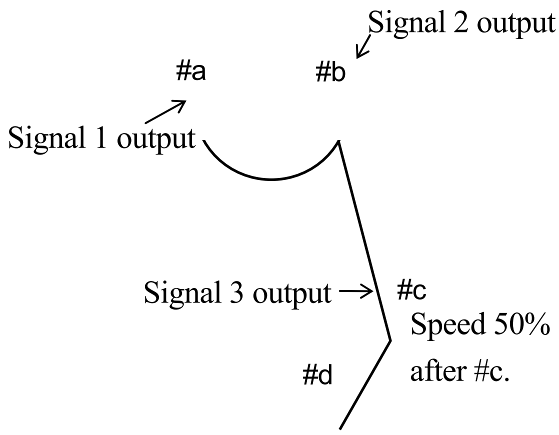


When PREFETCH.SIGINS is ON, the external signal 1 (SIGNAL 1) is output as soon as the robot starts moving toward part1. When the program reaches the second JMOVE instruction, it waits until the robot reaches part1 before performing that instruction. As soon as the robot reaches part 1, it starts for part 2, and at the same time, external signal 2 (SIGNAL 2) is output.

When PREFETCH.SIGINS is OFF, the signals are output after the robot reaches the destination of the motion instructions and the axes coincide.

The sample below demonstrates how the program steps are executed in AS system when PREFETCH.SIGINS is ON.

```
1 JMOVE    #b
2 SIGNAL    1
3 a=2
4 LMOVE    #c
5 SIGNAL    2
6 SPEED    50
7 LMOVE    #d
8 SIGNAL    3
```



The signal is processed in advance when PREFETCH.SIGINS is ON, therefore all the instructions up to the next motion instruction are executed as soon the robot starts executing the current motion instruction. If the above program is executed when the robot is at #a, the steps proceed in the following order:

1. At #a, the robot plans the motion for JMOVE #b and starts moving toward #b.
2. As soon as the motion starts, the next step, SIGNAL 1, is executed, i.e., signal 1 is turned on right after the robot departs #a.
3. The execution proceeds to step 4, plans LMOVE #c and waits for the robot to reach #b.
4. As soon as the robot reaches #b, the robot starts moving toward #c. The execution proceeds to step 7 (plans motion for LMOVE #d), and waits for the robot to reach #c.

[NOTE]

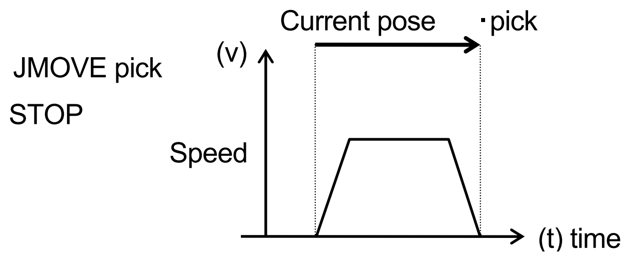
When PREFETCH.SIGINS is ON, the program processes the next step until it has to wait for the robot to reach the specified pose. However, the timing is affected by other settings and command/ instructions such as WAIT instruction or the CP switch. WAIT instruction suspends the processing of steps until the given condition is satisfied. When the CP switch is OFF, the program processes all the steps before the step that includes motion instruction, and stops there before proceeding. Keep in note the settings of the system switches and instructions when programming.

As demonstrated here, it is important to note that the timing in which the AS system processes the program and in which the robot moves are affected by the system switch settings and some certain program instructions. Pay careful attention to the output timing of signals during programming.

For details on each system switch, refer to 7.0 AS System Switch or the Operation Manual.

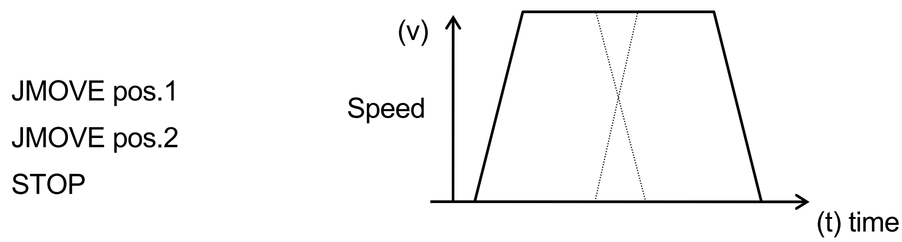
4.5.2 Continuous Path (CP) Motion

This example shows the execution of one motion instruction.



When executing a motion instruction like the one above, the robot accelerates smoothly up to the current speed setting as it moves towards the pose “pick”. As the robot approaches “pick”, it gradually decelerates until it stops at the pose. Series of motions such as this, carried out by one motion instruction, is called a “motion segment”.

In the case for the figure below, if the CP system switch is ON, the robot first accelerates to reach the specified speed, but does not decelerate when it approaches pos.1. Instead, it makes a smooth transition to the motion toward pos.2. When the robot approaches pos.2, it gradually decelerates and stops at that point. This motion consists of two motion instructions, and is thus structured by two motion segments.



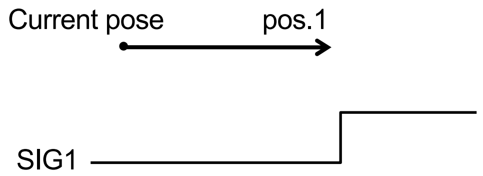
Motion like this, where the robot performs a series of motions making a smooth transition between the motion segments without stopping at each destination, is called CP (Continuous Path) motion. Turning OFF the CP system switch disables the CP function. If the CP switch is turned OFF, the robot will decelerate and stop at the end of each motion segment. (See 5.6 SWITCH and ON/OFF command, 6.9 ON/OFF instruction on how to set the CP switch).

CP motions can be used in both linear motions and joint interpolated motions or in a combination of them. For example, CP motions can be used throughout all of the following steps:
linear motion (e.g. LDEPART) → joint interpolated motion (e.g. JAPPRO)→ linear motion (e.g. LMOVE).

4.5.3 Breaks in CP Motions

Some instructions can suspend the execution of a program until the robot actually reaches the destination pose. This is called the break in CP motions. These instructions are useful when the robot should be stationary while certain operations are performed (e.g. closing the hand). See the example below.

```
JMOVE pos.1
BREAK
SIGNAL1
```



The JMOVE instruction starts moving the robot toward pos.1. Next, the BREAK instruction is executed. This instruction suspends the execution of the program until the movement towards pos.1 is completed. In this way, the external signal is not output until the robot comes to a stop.

The following instructions suspend program execution until the robot movement is completed. However, be careful not to use these instructions when the robot should be moving.

BASE	BREAK	BRAKE	CLOSEI	HALT	OPENI	PAUSE	RELAXI
TOOL	ABOVE	BELOW	DWRIST	UWRIST	LEFTY	RIGHTY	

In addition to the above, ONI instruction also interrupts the program execution, but note that the break set by ONI instruction may occur at any place of the motion segment.

[NOTE]

1. The robot decelerates and stops if an instruction is not given before the execution of the current motion is completed. Some of the reasons that cause such situation are:
   (1) The WAIT instruction is executed but the conditions to resume the program are not set before robot movement is completed.
   (2) Program steps before the next motion instruction are not completed before the current motion finishes.

2. When moving in CP motion, a certain amount of time is required to calculate the condition for smooth transition to the next motion segment. Therefore, if the distance between the two points is too short, the calculation time may become insufficient and cause the robot to stop in between the motion segments. To avoid this, it is necessary to slow down the speed. If the speed is not to be changed, do not specify the points unnecessarily close together.

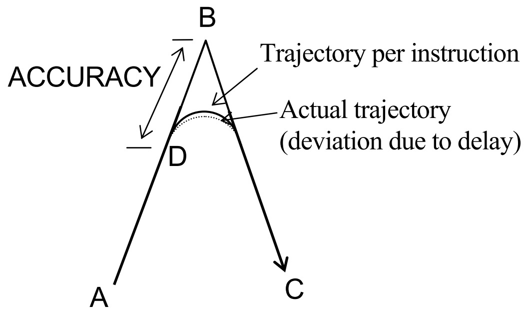
4.5.4 Relation between CP Switch and ACCURACY, ACCEL, and DECEL Instructions

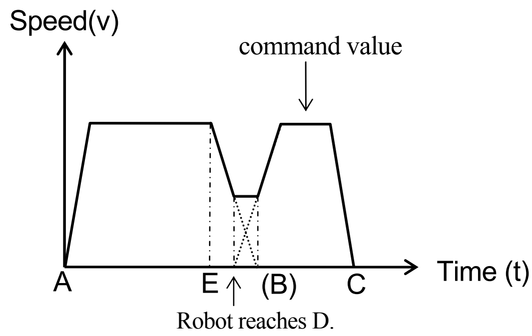
ACCURACY instruction:	Sets the robot's positioning accuracy at the end of each motion segment. (When the robot enters the range set by this instruction, it considers that it has reached the destination, and starts the movement for the next destination.)
·ACCEL instruction:	Sets acceleration of the robot at the beginning of a movement.
·DECEL instruction:	Sets deceleration of the robot at the end of a movement.
·CP Switch:	Enables or disables CP motion.

4.5.4.1 CP ON: Motion Type 1 (Standard)

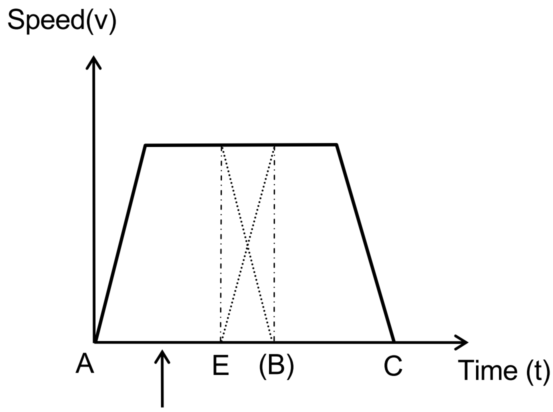
For example the robot takes the motions below with the CP switch ON: A →B→C.

As soon as the current pose values for the robot enters the accuracy range (i.e. robot reaches point D), superposing begins of the values of the current motion path with the motion command values for the next path. The robot will shift movement continuously toward the next path according to these command values.



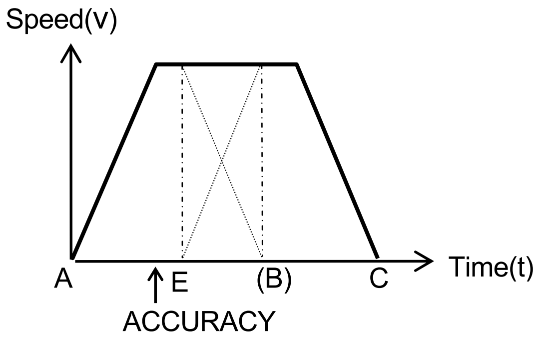


The greater the range specified by ACCURACY, the earlier the superposing will begin. However, acceleration on the next path does not begin before the point where the robot starts to decelerate (point E). Therefore, it can be said that the ACCURACY effect is saturated at a certain value, i.e. there is no effect in setting the accuracy value greater than the distance between point B and E. (See the diagram below.)

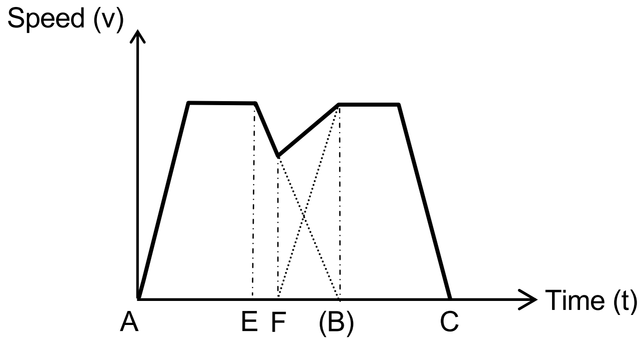


Even if command value reaches the accuracy point at this time, acceleration for next path will not start until deceleration begins at point E.

If the acceleration and the deceleration values are set smaller, the superposing begins earlier and the robot will move in a trajectory with larger radius, but the total time it takes to reach C does not differ significantly.



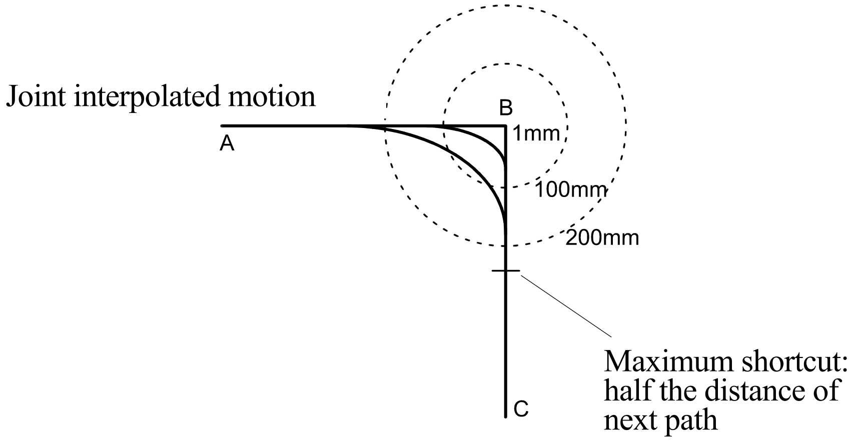
Even if the deceleration is decreased and the acceleration for the next path is increased, the compound speed will not exceed the specified maximum speed, since the superposing does not begin until the robot reaches point F (the point where acceleration starts). In other words, the time taken to complete deceleration and acceleration is the same (point B).



4.5.4.2 CP ON: Motion Type 2

In motion type 2, the concept of accuracy and velocity in linear motion and circular motion is different from that of Standard motion type described above. Standard motion type and motion type 2 can use the same programs without modifications, but the actual motion path and motion speed will change.

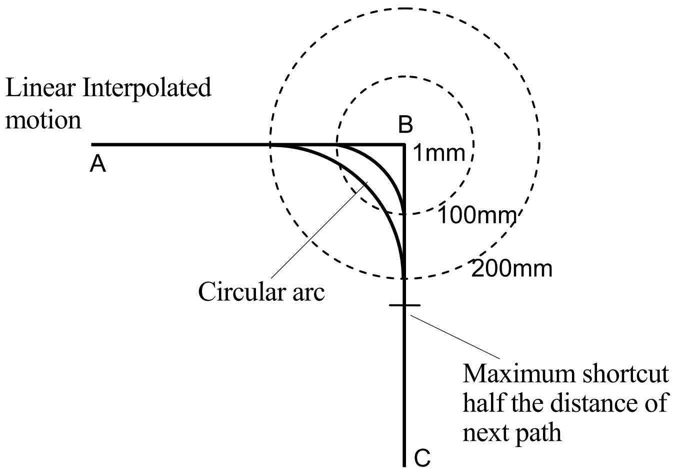
1. Accuracy setting
   (1) Accuracy in joint interpolated motion
   The motion path of the robot corresponding to the accuracy setting is shown in the figure below. In this example the accuracy values at point B are 1 mm, 100 mm, and 200 mm. In the same way as Standard motion, the robot starts to shortcut before reaching point B, but does not necessarily start turning at the point where it enters the accuracy range. How close the robot approaches point B before turning is determined by the angle of each axis calculated proportionally to the accuracy value. By setting the accuracy value larger, the robot can shortcut the shorter distance of either the remaining distance of the current path or half the distance of the next path from B to C.



(2) Accuracy in linear and circular interpolation motion

The motion trajectory of the robot corresponding to the accuracy setting is as shown in the figure below. In this example the accuracy values at point B are 1 mm, 100 mm, and 200 mm. The robot starts turning at the point where it enters the accuracy range. The robot follows a circular trajectory within the radius of accuracy range.

By setting the accuracy range larger, the robot can shortcut the shorter distance of either the remaining distance of the current path or half the distance of the next path from B to C. The accuracy value can be set up to the value equal to half the distance of the second path.



By shortcutting, the cycle time can be shortened. However, when the following conditions are set, the processing of the accuracy setting will be the same as in Standard motion:

· When a waiting instruction (TWAIT, SWAIT, etc.) is executed at point B.
· When a workpiece/tool is changed at point B.
· When the interpolation mode for the next point is changed to joint interpolation.
· When the motion mode is changed at point B. (ordinary mode ↔ motion based on the fixed tool coordinates)
· When the processing branches due to conditions set by instruction such as IF and END.

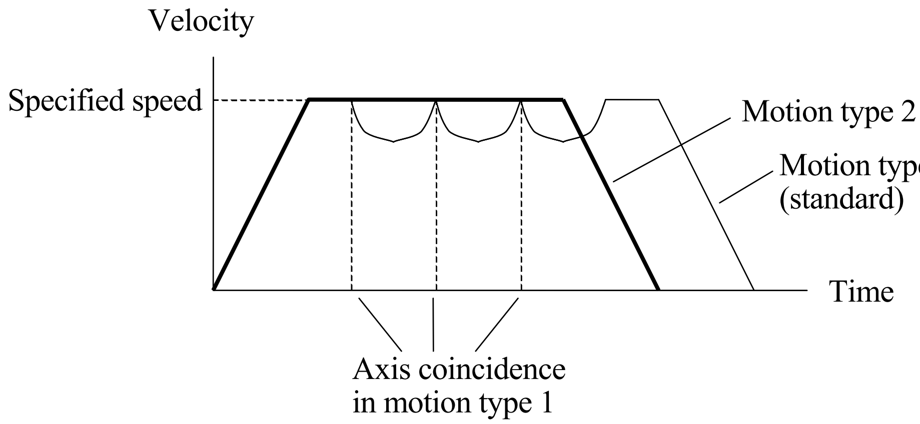
2. Speed setting

(1) Speed in joint interpolated motion

Same as in Standard motion type.

(2) Speed in linear and circular interpolated motion

In motion type 2, if the accuracy value is set larger and the configuration of the robot does not change between two defined poses, the specified speed is attained even if the distance between the two poses is small.



However, when the following conditions are set, the process will be the same as in Standard motion type:

· When a waiting instruction (TWAIT, SWAIT, etc.) is executed at point B.
· When a workpiece/tool in changed at point B.
· When the interpolation mode for the next point is changed to joint interpolation.
· When the motion mode is changed at point B. (ordinary mode ↔ motion based on the fixed tool coordinates)
· When the processing branches due to conditions set by instruction such as IF and END.

[NOTE]

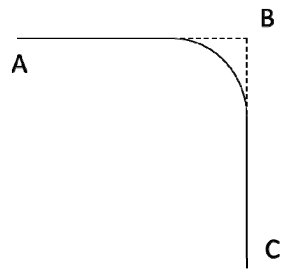
When attempting to execute a program where the robot orientation changes greatly within a short distance, the time it takes to change the orientation will exceed the time it takes to move that distance at the specified speed. In this case, the joint movements are given priority, thus the motion will not reach the specified speed.

(3) Speed in circular interpolation
In motion type 2 the maximum speed is automatically set according to the robot's capacity to carry out proper circular interpolation motion.

In motion type 2, the robot follows a circular trajectory within the accuracy range circle. The maximum speed of this trajectory is also set by the robot's capacity.

3. Precautions for programming in motion type 2
In motion type 2, the motion is planned with the next motion instruction as the target value. For example, in the figure below, the pose information for point C is used as reference in motion from point A to point B.

```
ACCURACY 100 ALWAYS
LMOVE A
LMOVE B
LMOVE C
```



When programming in motion type 2, make sure that the target value of the next motion is determined (example: LMOVE C) before executing a motion instruction step (example: LMOVE B).
For example, the following programming must not be done:

```
LMOVE B
POINT C = pos[1]
LMOVE C
```

In this example, the pose information (C) for the second motion instruction is set between motion LMOVE B and LMOVE C. This is incorrect and error (E0102) "Variable is not defined" will occur. This program must be written as below:

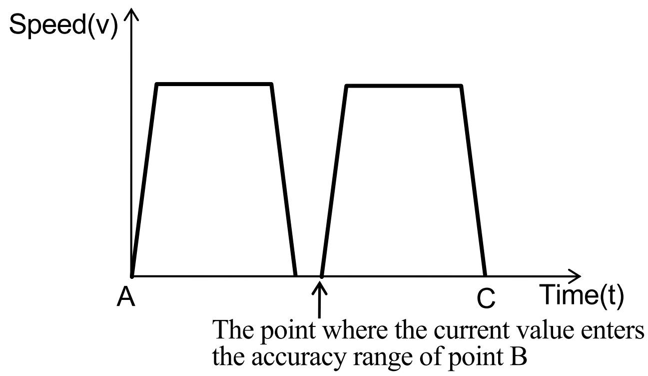
```
POINT C = pos[1]
LMOVE B
LMOVE C
```

As in the example above, the target value for the next motion instruction (example: LMOVE C) should be defined before executing the motion instruction step (example: LMOVE B).

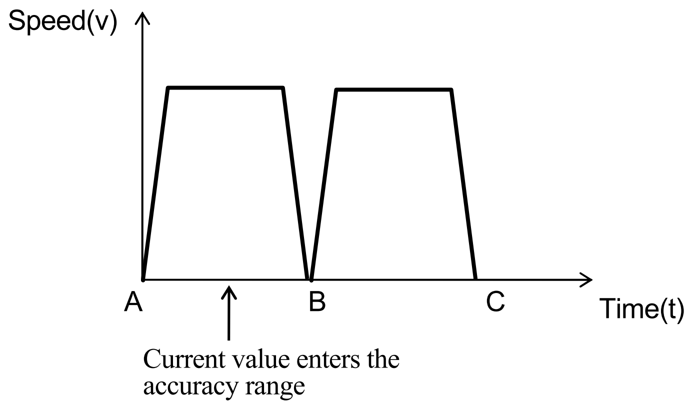
4.5.4.3 CP OFF

When the CP switch is OFF, there is no superposing of motions. The acceleration for the second path starts after the first motion segment is completed and the current value enters the ACCURACY* range.

NOTE* For example, for RS2ON, the default value is 1 mm.



When the CP switch is OFF, the motion for the second path begins only when the deceleration speed of the first motion reaches zero, even if the accuracy range is set larger than the end of the first path.



4.5.5 Motion along Specified Path

Linear interpolated and joint interpolated motions are standard functions on all the robots. However, occasionally it is necessary to move the robot along a specified or calculated path. The AS system can run calculations while the robot is moving, making it possible to realize complex motions. This feature is called “Motion along a specified path”.

The system enables the motions via a program loop that performs a series of continuous calculations of short-distance motions performed while motion instructions are executed. Such a program loop is possible because AS can perform non-motion instructions while the robot is moving. The calculated motion segments are connected smoothly using the CP function.

The following is an example of a program for motion along a specified path. The robot tool will follow the path defined by a series of pose data specified by the array variable “path”.

```
FOR  index=0 TO 10
LMOVE   path[index]
END
```

Array variables path[0] to path[10] are to be defined by manual teaching or by calculation.

In this example, END instruction exists between the first LMOVE and the next LMOVE, so the motion type for this program is standard motion type and not motion type 2. (Refer to 4.5.4.2).

4.5.6 Setting Load Data

By setting the load data for the robot’s current motion, the optimal acceleration and deceleration for the load are determined automatically. Set the correct load data according to the robot’s current motion.

> **CAUTION**
>
> **Always set the correct load mass and center of gravity location. Incorrect data may weaken or shorten the longevity of parts or cause overload/deviation errors. For detailed information see WEIGHT command/instruction.**
> **The load data can be set automatically by using the auxiliary function 0406 Auto Load Measurement. See the Operation Manual for details.**

5 Monitor Commands

This chapter groups the monitor commands in the following categories, and describes each command in detail. A monitor command consists of a keyword expressing the command and parameter(s) following that key word, as shown in the example below.

EXAMPLE

Keyword Parameter

EDIT **program name, step number**

Parameters marked with [highlighted] can be omitted.

Always enter a space between the keyword and the parameter.

↵ represents the Enter key in the examples.

5.1 Editor Commands

EDIT	Starts program editor.
C	Finishes editing current program and changes to another program (Change).
S	Selects program step to display (Step).
P	Displays specified number of program steps (Print).
L	Selects the previous step (Last).
I	Inserts a new step (Insert).
D	Deletes program steps (Delete).
F	Searches for characters (Find).
M	Replaces characters (Modify).
R	Replaces characters (Replace).
O	Places the cursor on the current step (One line).
E	Exits editor (Exit).
XD	Cuts and stores the selected step or steps in clipboard.
XY	Copies and stores the selected step or steps in clipboard.
XP	Pastes content of the clipboard.
XQ	Pastes content of the clipboard in the reverse order.
XS	Shows the contents of the clipboard.
T	Teaches motion instructions while in editor mode. (Option)

EDIT program name, step number

Function

Enters the editor mode that enables program creation and editing.

Parameter

Program name

Selects a program for editing. If a program name is not specified, then the last program edited or held (or stopped by an error) is opened for editing. If the specified program does not exist, a new program is created.

Step number

Selects the step number to start editing. If no step is specified, editing starts at the last step edited. If an error occurred during the last program executed, the step where the error occurred is selected.

[NOTE]

A program cannot be edited during execution.

A program cannot be executed or deleted while it is being edited. If a program calls a program that is being edited, an error occurs, and the execution of that program stops.

C program name, step number

Function

Changes the program currently selected in editor mode.

Parameter

Program name

Selects the program to be edited.

Step number

Selects the step number to start editing. If no step is specified, the first step of the program is selected.

S step number

Function

Selects and displays the specified step for editing. (Step)

Parameter

Step number

If no step is specified, the first step of the program is selected. If the step number is greater than the number of steps in the program, a new step following the last step in the program is selected.

P number of steps

Function

Displays the specified number of steps starting with the current step.

Parameter

Number of steps
Sets the number of steps to display. If the number of steps is not specified, only the current step is displayed.

Explanation

Displays only the specified number of steps. The last step on the list is ready for editing.

L

Function

Displays the previous (last) step for editing.
([Current step number] −1=[step number of the step to be displayed])

I

Function

Inserts lines before the current step.

Explanation

The steps after the inserted line are renumbered. To exit insert mode, press the ↵ key. All lines written before exiting the insert mode are inserted in the program.

Example

CLOSEI instruction is inserted between steps 3 and 4.

```
1?OPENI
2?JAPPRO  #PART, 500
3?LMOVE  #PART
4?LDEPART  1000
5? S   4                        ;Display step 4 to insert a line before it.
4      LDEPART  1000
4? I                            ;Type the I command.
4I  CLOSEI   ↵                  ;Type in the instructions for the inserted line.
5I ↵                            ;Press enter to finish inserting the lines.
5      LDEPART  1000            ;Step 4 is now renumbered as step 5.
5?
```

[NOTE]

To insert blank line, press Spacebar or TAB, then ↵ while in the insert mode.

D number of steps

Function

Deletes the specified number of steps including the current step.

Parameter

Number of steps

Specifies number of steps to delete beginning with the current step. If no number is specified, only the current step is deleted.

Explanations

Deletes only the specified number of steps beginning with the current step. Once deleted, all remaining steps are automatically renumbered and displayed.

[NOTE]

If the number of steps specified is greater than the number of steps in the program, all the steps after the current step are deleted.

F character string

Function

Searches (finds) the current program for the specified string from the current step to the last, and displays the first step that includes the string.

Parameter

Character string

Specifies the string of characters to be searched.

Example

Searches for character string “abc” in steps after the current step and displays the step containing that string.

```
1?F abc ↵
3          JMOVE abc
3?
```

M　/existing characters/new characters

Function

Modifies the characters in the current step.

Parameter

Existing characters

Specifies which characters are overwritten in the current step.

New characters

Specifies the characters that replace the existing characters.

Example

Modifies step 4 by replacing the pose variable abc with def.

```
4       JMOVE abc ↵
4?M/abc/def ↵
4       JMOVE def
4?
```

R character string

Function

Replaces existing characters in the current step with the specified characters.

Parameter

Character string

Specifies the new characters that replace the existing characters.

Explanation

The procedure for using the R command is as follows:

1. Using the Spacebar, move the cursor under the first character to replace.

2. Press the R key and then the Spacebar.

3. Enter the new replacement character(s). Note that the characters entered do not replace characters above the cursor but those two spaces to the left, starting above the R. (See example below.)

4. Press ↵.

Once ↵ is pressed, the AS system checks if the line is correct. If there is an error, the entry is ignored.

Example

The speed is changed from 20 to 35 using the R command.

```
1       SPEED  20 ALWAYS
1?             R 35 ↵
1       SPEED  35 ALWAYS
1?
```

O

Function

Places the cursor on the current step for editing.　(“O” for “one line”, not zero).

Example

The pose variable abc is changed to def using the O command.　The cursor is moved using ← or → key.

```
3        JMOVE abc
3?O ↵
3        JMOVE abc [BackSpace]          ; delete “abc” using [Backspace]
3        JMOVE def ↵                    ; Enter “def”
3        JMOVE def
3?
```

[NOTE]

This command cannot be used via teach pendant.

E

Function

Exits from the editor mode and returns to monitor mode.

XD number of steps

Function

Cuts the specified number of steps from a program and stores them in the paste buffer.

Parameter

Number of steps

Specifies number of steps to cut and store in the paste buffer beginning with the current step. Up to ten steps can be cut. If not specified, only the current step is cut.

Explanation

Cuts the specified number of steps and stores them in the paste buffer.

The XY command copies and does not cut the steps, but the XD command cuts the steps. The remaining steps in the program are renumbered accordingly.

XY number of steps

Function

Copies the specified number of lines and stores in the paste buffer.

Parameter

Number of steps

Specifies number of steps to copy and store in the paste buffer. Up to ten steps can be copied. If the number is not specified, only the current step is copied.

Explanations

Copies the specified number of steps including the current step and stores them in the paste buffer.

The XD command cuts the steps, but XY command copies the steps. The program remains the same and step count does not change after the XY command is used.

XP

Function

Inserts the contents of the paste buffer before the current step.

Explanation

Use the XD or XY command prior to this command to store the desired contents in the paste buffer.

XQ

Function

Inserts the contents of the paste buffer before the current step with the contents being inserted in reverse order.

Explanation

Inserts the contents of the paste buffer in reverse order as it would be inserted using XP command.

XS

Function

Displays the contents of the paste buffer.

Explanation

Displays the current contents of the paste buffer. If the paste buffer is empty, nothing will be displayed.

T pose variable

Option

Function

Enables teaching of motion instructions (JMOVE, LMOVE, etc.) using the teach pendant while in editor mode.

Parameter

Pose variable

Specifies pose variable name of destination to be taught, expressed in transformation values or joint displacement values. It is read as an array variable if specified in the form of A[ ]. In this case, variables cannot be used in the element numbers. If omitted, the current joint displacement values are taught as constants (pose constant).

Explanation

Enter this command while in editor mode. When executed, teach pendant displays a specialized teaching screen. Motions taught here are recorded as instructions in the program, and are written on the step where the T command is entered. When more than one step is taught, the variable is renamed by incrementing the last number in the variable name. See Operation Manual for more details.

Example

With pose variable

```
2    JAPPRO #a
3?    T pos
```

Teach using TP. Press R to return to AS.
(3 steps are taught here)

```
3    JMOVE pos0
4    JMOVE pos1
5    LMOVE pos2
        ⋮
```

Without pose variable

```
2    JAPPRO #a
3?   T                         Teach joint values using TP. Press R to end.
                               (2 steps are taught here)
3    JMOVE #[0,10,20,0,0,0]
4    JMOVE #[10,10,20,0,0,0]
        ⋮
```

WARNING

Teach pendant must be connected to the controller to use this command. Also, the robot has to be in Teach mode, and TEACH LOCK ON.

5.2 Program and Data Control Commands

USB_FDIR	Lists names of files on USB flash drive.
DIRECTORY	Displays program or data names in list format.
DIRECTORY/P	Displays program names in list format.
DIRECTORY/L	Displays pose variable names in list format.
DIRECTORY/R	Displays real variable names in list format.
DIRECTORY/S	Displays string variable names in list format.
DIRECTORY/INT	Displays integer variable names in list format.
LIST	Displays all program steps and variable values.
LIST/P	Displays all program steps.
LIST/L	Displays all pose variables and their values.
LIST/R	Displays all real variables and their values.
LIST/S	Displays all string variables and their data.
LIST/INT	Displays all integer variables and their data.
DELETE/D	Deletes programs and variables in robot memory.
DELETE/P/D	Deletes programs in robot memory.
DELETE/L/D	Deletes pose variables in robot memory.
DELETE/R/D	Deletes real variables in robot memory.
DELETE/S/D	Deletes string variables in robot memory.
DELETE/INT/D	Deletes integer variables in robot memory.
USB_FDEL	Deletes files on USB flash drive.
RENAME	Changes the name of a program.
USB_RENAME	Changes the name of a program in USB flash drive.
XFER	Copies steps from one program to another.
COPY	Copies programs.
USB_COPY	Copies programs in USB flash drive.
TRACE	Turns ON/OFF the TRACE Function.
SETTRACE	Reserves memory for logging.
RESTRACE	Releases memory reserved with SETTRACE.
LSTRACE	Displays the logging data.
SHUTDOWN	Executes the data backup to CFast.

USB_FDIR folder name

Function

Displays the name of files on USB flash drive.

Parameters

Folder name

Specifies the folder which contains the list of files to display. When omitted, the files in the root folder of the USB flash drive memory are displayed. Writing “USB1¥” to “USB4¥” as a folder name can specify the file on the USB memory of USB#1 to USB#4. USB#1 and USB#2 indicate the controller front USB slot, USB#3 does the controller rear USB slot and USB#4 does the internal USB slot. Also, writing “CFast¥” before the file name can specify the file on the CFast in the controller.

Explanation

By using the USB_FDIR command, all the files on USB flash drive are displayed.

Example

>USB_FDIR ↵	Displays the names all files on the USB flash drive.

When the switch SCREEN is ON, the display does not scroll and stops at the end of the screen. To continue display, press Spacebar. To end the display, press ↵.

DIRECTORY program name, ...
DIRECTORY/P program name, ...
DIRECTORY/L pose variable, ...
DIRECTORY/R real variable, ...
DIRECTORY/S string variable, ...
DIRECTORY/INT integer variable, ...

Function

Displays the specified program and data in a list format.

Program name (/P), pose variable (/L), real variable (/R), string variable (/S), integer variable (/INT)
Specifies the type of data to display. If not specified, all the data in memory is displayed.

Explanation

The DIRECTORY command displays all the program names, their subroutines and variables. On the other hand, DIRECTORY/P command displays only the contents of the specified program itself.

Example

>DIRECTORY	Displays the contents of all programs, including the variables and their values in a list.
>DIRECTORY/L	Displays all the pose variable names in a list.
>DIRECTORY/R	Displays all the real variables names in a list.
>DIRECTORY test3	Displays all the names of the variables used in program test3.
>DIRECTORY test*	Displays all the names of programs that start with test.

When the SCREEN switch is ON, the display does not scroll and stops when the screen is full. To continue the display, press Spacebar. To quit the display, press ↵.

[NOTE]

Program and variable names with * or ~ in front of the name indicate that no data are set to that program or variable.

LIST program name, ...
LIST/P program name, ...
LIST/L pose variable, ...
LIST/R real variable, ...
LIST/S string variable, ...
LIST/INT integer variable, ...

Function

Displays the specified program and data.

Parameters

Program name (/P), pose variable (/L), real variable (/R), string variable (/S), integer variable (/INT)

Specifies the type of data to display. If not specified, all the data in memory is displayed. If an array variable is selected, all the elements of that array variable are displayed on the screen.

Explanation

The LIST command displays all the program names, their subroutines and variables. On the other hand, LIST/P command displays only the contents of the specified program itself.

Example

>LIST ↵	Displays the name of all programs, including the variables and their values.
>LIST/L ↵	Displays all the pose variables and their values.
>LIST/R ↵	Displays all the real variables and their values.
>LIST test* ↵	Displays the contents of all programs that start with “test”, their subroutines and variables.

When the SCREEN switch is ON, the display does not scroll and stops when the screen is full. To continue the display, press Spacebar. To quit the display, press ↵.

DELETE/D program name, ...
DELETE/P/D program name, ...
DELETE/L/D pose variable [array elements] , ...
DELETE/R/D real variable [array elements] , ...
DELETE/S/D string variable [array elements] , ...
DELETE/INT/D string variable [array elements] , ...

Function

Deletes the specified data from the memory.

Parameters

Program name (/P), pose variable (/L), real variable (/R), string variable (/S),
integer variable (/INT), forced delete(/D)
Specifies the type of data to delete.

Explanation

DELETE command deletes the specified program completely; i.e. the main program itself and, if used in the program, the following data. (However, data used in other programs are not deleted).

· All subroutines called by the program or by subroutines within that program.
· All pose variables used in the program and in the subroutines in that program.
· All real variables used in the program and in the subroutines in that program.
· All string variables used in the program and in the subroutines in that program.

DELETE/P command, unlike the DELETE command, deletes only the program itself, and not the subroutines and variables used by that program.

If the array elements are not specified with the DELETE/L, DELETE/R, DELETE/S and DELETE/INT commands, all the elements in that array variable are deleted. If the element(s) are specified, only the specified element(s) are deleted.

When forced delete (/D) is specified, all subroutine and variable including those used in other programs are deleted. Those programs that are in execution or selected cannot be deleted. Also, robot programs and PC programs in execution cannot be deleted forcibly.

Example

>DELETE test ↵	Deletes the program "test", and all the subroutines and variables used in them.
>DELETE/P pg11, pg12 ↵	Deletes the programs "PG11" and "PG12". (The subroutines and the variables are not deleted.)
>DELETE/R a ↵	Deletes all the elements of the array variable "a".
>DELETE/R a[10] ↵	Deletes the 10th element of array variable "a".
>DELETE/D test ↵	Deletes the program "test" even if it is the subroutine called by another program.
>DELETE/R/D a[10] ↵	Deletes only the 10th element of array variable "a" even if the array variable "a[]" is used in a program.

```
.PROGRAM aa()
  CALL bb
.END
.PROGRAM bb()
  …
.END
```

When the program "aa" mentioned before is selected,

>DELETE/D aa ↵	Cannot delete "aa" because it is selected.
>DELETE/D bb ↵	Deletes "bb".

When the program "aa" mentioned before is executed and the program "bb" is in HOLD in execution,

>DELETE/D aa ↵	Cannot delete "aa" because it is in execution.
>DELETE/D bb ↵	Cannot delete "bb" because it is in execution.

USB_FDEL file name, ...

Function

Deletes the specified file from the USB flash drive memory.

Parameters

File name

Specifies the file names of a program and a variable. If no file name extension is specified, the extension ".as" is automatically added to the file name. To specify the folder including the file, add "folder name¥" before the file name. Writing "USB1¥" to "USB4¥" as a folder name can specify the file on the USB memory of USB#1 to USB#4. USB#1 and USB#2 indicate the controller front USB slot, USB#3 does the controller rear USB slot and USB#4 does the internal USB slot. Also, writing "CFast¥" before the file name can specify the file on the CFast in the controller.

Explanation

Deletes the programs in the specified file completely.

RENAME new program name＝existing program name

Function

Changes the name of a program currently held in memory.

Parameters

New program name

Sets new name for the program.

Existing program name

Specifies the current name of the program.

Explanation

If the new program name already exists, RENAME command results in an error.

Example

> RENAME test=test.tmp ↵ Changes the name of the program from "test.tmp" to "test".

USB_RENAME new file name＝existing file name

Function

Changes the name of a file currently held in USB flash drive memory.

Parameters

New file name

Sets new name for the file. To specify the folder including the file, add “folder name¥” before the file name. Writing “USB1¥” to “USB4¥” as a folder name can specify the file on the USB memory of USB#1 to USB#4. USB#1 and USB#2 indicate the controller front USB slot, USB#3 does the controller rear USB slot and USB#4 does the internal USB slot. Also, writing “CFast¥” before the file name can specify the file on the CFast in the controller.

Existing file name

Specifies the current name of the file. To specify the folder including the file, add “folder name¥” before the file name. Writing “USB1¥” to “USB4¥” as a folder name can specify the file on the USB memory of USB#1 to USB#4. USB#1 and USB#2 indicate the controller front USB slot, USB#3 does the controller rear USB slot and USB#4 does the internal USB slot. Also, writing “CFast¥” before the file name can specify the file on the CFast in the controller.

Explanation

If the new file name already exists, USB_RENAME command results in an error.

Example

>USB_ RENAME file_new =file ↵	Changes the name of the file in USB memory, from “file” to “file_new”.

**XFER destination program name, step number1 =
source program name, step number2, number of steps**

Function

Copies and transfers steps from one program to another program.

Parameters

Destination program name

Sets the program for receiving the copied data. If a program of that name does not exist, the data is transferred to a new program with that name.

Step number 1

Sets the step number before which the copied data is inserted. If no step is specified, the data is inserted at the end of the specified program.

Source program name

Sets the name of the program from where the data is copied.

Step number 2

Sets the step number in the source program where the data is copied. If no number is specified, the data is copied starting from the top of the program.

Number of steps

Sets the number of steps to copy from the source program, starting from the step number set above (parameter: step number 2). If no step count is specified, all remaining steps in the program are copied.

Explanation

Copies from the specified program a specified number of steps, and inserts the data before the specified step in the destination program.

[**NOTE**]

If the destination program is being displayed using the STATUS or PCSTATUS commands, or if it is being edited (EDIT command), XFER command cannot be used.

COPY new program name =
source program name + source program name +

Function

Copies the complete program to a new program.

Parameters

New program name

Specifies the name of the program to where the copied program is placed. This must be specified.

Source program name

Specifies the name of the program to be copied. At least one program must be specified.

Explanation

When two or more source programs are specified, the programs are combined into one program under the new program name. The name specified for the new program cannot be an existing program.

USB_COPY new file name = source file name

Function

Copies the specified files to a new file on USB flash drive memory.

Parameters

New file name

Specifies the new name for the file to be created. This must be specified. To specify the folder including the file, add “folder name¥” before the file name. Writing “USB1¥” to “USB4¥” as a folder name can specify the file on the USB memory of USB#1 to USB#4. USB#1 and USB#2 indicate the controller front USB slot, USB#3 does the controller rear USB slot and USB#4 does the internal USB slot. Also, writing “CFast¥” before the file name can specify the file on the CFast in the controller.

Source file name
Specifies the name of the file(s) to be copied. This must be specified. To specify the folder including the file, add “folder name¥” before the file name. Writing “USB1¥” to “USB4¥” as a folder name can specify the file on the USB memory of USB#1 to USB#4. USB#1 and USB#2 indicate the controller front USB slot, USB#3 does the controller rear USB slot and USB#4 does the internal USB slot. Also, writing “CFast¥” before the file name can specify the file on the CFast in the controller.

Explanation

The name specified for the new file cannot be an existing file name.

TRACE stepper number: ON/OFF

Function

Starts (ON) or ends (OFF) logging of the robot or PC program to allow program tracing.

Parameters

Stepper number
Specifies the program to trace using the following number selection:

1: Robot program
1001: PC program 1
1002: PC program 2
1003: PC program 3
1004: PC program 4
1005: PC program 5

If the program is not specified, the program currently in execution is logged.

ON/OFF
Starts/ ends logging.

Explanation

If the necessary memory is not reserved using the SETTRACE command before TRACE ON, the error (P2034) “Memory undefined” occurs. Execute SETTRACE before retrying.

SETTRACE number of steps

Function

Reserves the necessary memory to log the data for program tracing.

Parameters

Number of steps

Specifies the number of steps to log (setting range: 1 to 9999). If not specified, memory for 100 steps will be reserved.

Explanation

A portion of the user memory is set aside to accommodate the specified number of steps and the current number of existing robot and PC programs.

If TRACE ON and LSTRACE commands are executed without reserving the memory for logging, the error (P2034) "Memory undefined" occurs. If the SETTRACE command is used while logging, error (P2033) "Logging is in process" occurs, and all tracing are turned OFF (ends).

RESTRACE

Function

Releases the memory set aside by SETTRACE command.

Explanation

If the RESTRACE command is used while logging, the error (P2033) "Logging is in process" occurs.

LSTRACE stepper number: logging number

Function

Displays the logging data of the specified robot program or PC program.

Parameters

Stepper number

Specifies the program to be displayed by the following number selection:

1: Robot program	
1001: PC program 1	1004: PC program 4
1002: PC program 2	1005: PC program5
1003: PC program 3	

If no program is specified, the log for robot program is displayed.

Logging number

Specifies the line number of the logging data from which to start the display. If not specified, line 1 is selected.

Explanation

If the necessary memory is not reserved using the SETTRACE command before executing LSTRACE, error (P2034) "Memory undefined" occurs.

If the LSTRACE command is used while logging, error (P2033) "Logging is in process" will occur.

When the LSTRACE command is executed, the logging data is displayed. The prompt appears after the data and the following commands can be entered:

N ↵ displays the next 9 lines.

L ↵ displays the previous 9 lines.

S number ↵ displays the specified log line number, and the 4 lines logged before and after that line (total of 9 lines). If no line number is specified, lines 1 to 9 are displayed. If the number is greater than the existing lines, the highest line number is displayed.

F character ↵ displays the line that includes the specified character(s), and the 4 lines logged before and after that line (total of 9 lines). If no character(s) are specified, the characters entered previously with the F command are used. If the characters are not found in the data, nothing is displayed.

E ↵ ends the display and returns to AS monitor mode.

↵ entered alone displays the next 9 lines.

Example

```
91    pg1    31  JOINT SPEED9 ACCU1 TIMER0 TOOL1 WORK0   CLAMP1 (OFF,0,0,0)   2
92    pg1    32  SIGNAL   14;sig on
93    pg1    33  JOINT SPEED9 ACCU1 TIMER0 TOOL1 WORK0   CLAMP1 (OFF,0,0,0)   2
94    pg1    34  CALL "sub1"
95    sub1    1  PRINT "SUB1"
96    sub1    2  xyz:
97    sub1    3  JMOVE a
98    sub1    4  JMOVE b
99    sub1    5  JMOVE c
----N: Next page, L: Previous page, S number: Jump to number, F character: Find character,
E: End-------
```

SHUTDOWN

Function

Executes the data backup to the CFast in the controller.

Explanation

The data backup to the CFast starts when the SHUTDOWN command is executed.
If the data backup is completed normally, the message (D0906) "Data backup to CF is completed. Turn OFF the controller power." appears. If the data backup is not completed after the specified time (10[sec]), the message (D0907) "Data backup to CF is failed. Turn OFF & ON the controller power." appears.
Executing following operations becomes impossible after the SHUTDOWN command is executed, because the memory of the controller is locked. Turn OFF and ON the controller power.
The SHUTDOWN command cannot be used while a robot motion program is running.
If the SHUTDOWN command is executed while the robot motion program is running, the error (P1012) "Robot is moving now." occurs.

5.3 Program and Data Storage Commands

SAVE*	Saves the specified programs.
SAVE/P *	Saves programs.
SAVE/L *	Saves pose variables.
SAVE/R *	Saves real variables.
SAVE/S *	Saves character strings.
SAVE/A *	Saves auxiliary information.
SAVE/SYS *	Saves system data.
SAVE/ROB *	Saves robot data.
SAVE/ELOG *	Saves error log data.
SAVE/OPLOG *	Saves operation log data.
SAVE/ALLLOG *	Saves all log data.
SAVE/FULL *	Saves all data that can be saved.
SAVE/STG *	Saves logging data for data storage function. (Option)
LOAD *	Loads programs and data to robot memory.
USB_MKDIR	Creates a new folder on the USB flash drive.

NOTE* These commands save data to personal computers. To save the data to the USB flash drive, add the prefix USB_ to the command. See the explanation for each command for further information.

SAVE/SEL file name=program name, ………

USB_SAVE/SEL file name=program name, ………

Function

SAVE command stores programs and variable data on the computer hardware. (Use only when a PC is connected to the robot controller.)
USB_SAVE command stores programs and variable data on USB flash drive.

Parameters

File name
Specifies the file names of a program and a variable. If no file name extension is specified, the extension ".as" is automatically added to the file name. To specify the folder including the file, add "folder name¥" before the file name. For the USB_SAVE command, writing "USB1¥"-"USB4¥" as a folder name can specify the file on the USB memory of "USB1¥" to "USB4¥" as a folder name can specify the file on the USB memory of USB#1 to USB#4. USB#1 and USB#2 indicate the controller front USB slot, USB#3 does the controller rear USB slot and USB#4 does the internal USB slot. Also, writing "CFast¥" before the file name can specify the file on the CFast in the controller.

Program name
Select the program to save. If not specified, all the programs in the memory are saved.

Explanation

The commands SAVE/P, SAVE/L, SAVE/R, SAVE/S, SAVE/SYS store each data type (program, pose variable, real variable, string variable, and system data, respectively) in separate files. Using the SAVE command alone stores all the five data types in one file.

SAVE command (without /SEL) stores the specified program(s), including any variables and subroutines used by the program(s). SAVE/SEL command stores only the program and not the subroutines and variables used by that program.

Example

>SAVE f3=cycle,motor ↵	Stores under the file name "f3.as" the system data, the two programs "cycle" and "motor", the subroutines called from those programs, and the variables used in those programs.

[NOTE]

If the specified file name already exists in memory, then the existing file is automatically renamed with a "b" in front of the file extension. For example if "file1.as"already exists in memory and the command >SAVE file1 ↵ is executed, then that file is renamed "file1.bas". The newly created file will be named "file1.as".

SAVE/P/SEL　file name=program name,　………
SAVE/L/SEL　file name=program name,　………
SAVE/R/SEL　file name=program name,　………
SAVE/S/SEL　file name=program name,　………
SAVE/A/SEL　file name=program name,　………
SAVE/SYS　file name
SAVE/ROB　file name
SAVE/ELOG　file name
SAVE/OPLOG　file name
SAVE/ALLLOG　file name
SAVE/FULL　file name
SAVE/STG　file name

USB_SAVE/P/SEL　file name=program name,　………
USB_SAVE/L/SEL　file name=program name,　………
USB_SAVE/R/SEL　file name=program name,　………
USB_SAVE/S/SEL　file name=program name,　………
USB_SAVE/A/SEL　file name =program name,　………
USB_SAVE/SYS　file name
USB_SAVE/ROB　file name
USB_SAVE/ELOG　file name
USB_SAVE/OPLOG　file name
USB_SAVE/ALLLOG　file name
USB_SAVE/FULL　file name
USB_SAVE/STG　file name

Function

Stores in the file the program (/P), pose variable (/L), real variable (/R), string variable (/S), auxiliary information (/A), system data (/SYS), robot data (/ROB), error log (/ELOG), operation log (/OPLOG), all logs (/ALLLOG), all savable data (/FULL) and data logging function (option) logging data (STG).

As with SAVE command, USB_SAVE/ commands are used to save files to the USB flash drive memory.　Use SAVE/ command only when a PC is connected to the robot controller.　See 2.6.2 Uploading and Downloading Data.

Parameters

File name

Saves the data under this file name. If no file name extension is specified, the following extensions are automatically added to the file name according to the type of data in the file:

program	.PG	pose information	.LC	real variables	.RV
string variables	.ST	auxiliary information	.AU	system data	.SY
robot data	.RB	error log	.EL	operation log	.OL
all logs	.AS	all data	.AS	data storage	.CSV

For the USB_SAVE command, writing "USB1¥" to "USB4¥" as a folder name can specify the file on the USB memory of USB#1 to USB#4. USB#1 and USB#2 indicate the controller front USB slot, USB#3 does the controller rear USB slot and USB#4 does the internal USB slot. Also, writing "CFast¥" before the file name can specify the file on the CFast in the controller.

Program name

Selects the name of the program to save. If not specified, all the programs and data in the memory will be saved on the file.

Explanation

1. SAVE/P

   Stores in the specified file the selected program(s) and the subroutines called by those program(s) (including the subroutines called by the subroutines).

   The names of the program(s) that were saved to the file are displayed on the system terminal. Some additional program names other than those specified by the SAVE command may appear. These are the names of the subroutines the specified program calls. These subroutines are stored in the same file as the program(s). Programs are stored in the file in alphabetical order regardless of the order in which they were saved.

2. SAVE/L, SAVE/R, SAVE/S

   Stores only the variables used in the specified program(s) and the subroutine(s) called by those program(s). (/L: stores only the pose variables, /R: stores only the real variables, /S: stores only the string variables)

3. SAVE/A

   Stores the auxiliary information.

4. SAVE/SYS

   Stores the system data.

5. SAVE/ROB
   Stores the data pertaining specifically to the robot (robot data).

6. SAVE/ELOG
   Stores the error log. This command cannot be entered together with other SAVE/ commands. For example, SAVE/ELOG/P does not function.

7. SAVE/OPLOG
   Stores the operation logs. This command cannot be entered together with other SAVE/ commands. For example, SAVE/OPLOG/P does not function.

8. SAVE/ALLLOG
   Stores all logs such as the error log, operation log, etc. This command cannot be entered together with other SAVE/ commands. For example, SAVE/ALLLOG/P does not function.

9. SAVE/FULL
   Stores all data that can be saved. This command cannot be entered together with other SAVE/ commands. For example, SAVE/FULL/P does not function.

10. SAVE/STG
   Stores the logging data logged in data storage function (option). This command cannot be entered together with other SAVE/ commands. For example, SAVE/STG/P does not function.

11. If /SEL is entered with /P,/L,/R,/S, only the main program and the variables used only in the main program are stored. The subroutines and the variables used in the subroutines are not stored.

   If the specified file name already exists in memory, then the existing file is automatically renamed with a “b” in front of the file extension. (See also SAVE command.)

Example

>SAVE/L file2=pg1, pg2 ↵	The pose variables used in programs pg1 and pg2 are stored under the file name “file2.lc”.

LOAD/Q file name

USB_LOAD/Q file name

Function

LOAD command loads the files in the computer memory into the robot memory. (Use only when PC is connected to the robot controller.) USB_LOAD command loads the files on the USB flash drive.

Parameters

File name

Specifies the file names of a program and a variable. If no file name extension is specified, the extension ".as" is automatically added to the file name. To specify the folder including the file, add "folder name¥" before the file name. For the USB_LOAD command, "USB1¥" to "USB4¥" as a folder name can specify the file on the USB memory of USB#1 to USB#4. USB#1 and USB#2 indicate the controller front USB slot, USB#3 does the controller rear USB slot and USB#4 does the internal USB slot. Also, writing "CFast¥" before the file name can specify the file on the CFast in the controller.

Explanation

This command loads the data (system data, programs, and variables) from the specified file into the robot memory. Attempting to load a program name that already exists in memory results in error, and execution of the LOAD command is aborted.

[NOTE]

When loading a pose variable, real variable, or string variable name that already exists in memory, the data in the memory is overwritten without any warning. (Programs are not overwritten.)

The original data is deleted if LOAD is canceled while overwriting the data in the memory.

For LOAD command with /Q, the following message appears before loading each system data or program:

```
Load? (1:Yes, 0:No, 2:Load all, 3:Exit)
```

The choices are as follows:
1: Loads the data.
0: Does not load the data and goes on to the next data.
2: Loads the data and the remaining data in the file without inquiry.
3: Does not load the data, and ends the LOAD command.

If there is an unreadable or incorrect step in the program, the following message appears: "The step format is incorrect (0: Continue load 1: Delete program and exit)". If the operation is continued by entering "0", use the editor (Edit mode) to correct the step after the program has been loaded.

Example

```
>LOAD   pallet↵            Loads the data in the file "pallet.as" into the memory.
Loading…
System data
Program    a1()
Program    test()
:
Transformation values
Joint interpolation values
Real values
Loading done.

>
>LOAD   f3.pg↵             Loads all programs in file "f3.pg" into the memory.
```

USB_MKDIR folder name

Function

Creates a file folder on the USB flash drive memory by the specified name.

Parameters

Folder name
Creates a folder by this name.

Explanation

Creates a new folder on the USB flash drive memory with the specified name. If a file or folder with the same name already exists on the memory, the message "Could not create folder" is displayed and the folder is not created.
Writing "USB1¥" to "USB4¥" as a folder name can specify the file on the USB memory of USB#1 to USB#4. USB#1 and USB#2 indicate the controller front USB slot, USB#3 does the controller rear USB slot and USB#4 does the internal USB slot. Also, writing "CFast¥" before the file name can specify the file on the CFast in the controller.

5.4 Program Control Commands

SPEED	Sets the monitor speed.
PRIME	Prepares the program for execution.
EXECUTE	Executes the program.
STEP	Executes one step of the program.
MSTEP	Executes one robot motion instruction.
ABORT	Stops execution after the current step is completed.
HOLD	Stops execution.
CONTINUE	Resumes execution of the program.
STPNEXT	Executes the program in step once mode.
KILL	Initializes the execution stack.
DO	Executes a single program instruction.

SPEED monitor speed

Function

Sets the monitor speed in percentage.

Parameter

Monitor speed

Sets the speed in percentage. If this value is 100, then the speed will be 100% of the maximum speed. If it is 50, the speed will be the half of the maximum speed. If the speed restriction removal option is set, the speed will be specified in percentage (%) up to 99999.

Explanation

A product of the monitor speed (set by this command) and program speed (set in the program using the SPEED instruction) determines the robot motion speed. For example, if the monitor speed is set at 50 and the speed set in the program is 60, and then the robot's maximum speed will be 30%.

[**NOTE**]

The maximum speed of the robot is automatically set at 100%, if the product of the monitor speed (set by monitor command or MON SPEED instruction) and the program speed (set by SPEED instruction in program) exceeds 100%.

The default setting of the monitor speed is 10%.

This command will not affect the speed of motion currently in execution. The new speed setting takes effect after the current motion and the planned motion are completed.

Example

If the program speed is set at 100%:

>SPEED 30 ↵	The robot motion speed is set at 30% of the maximum speed.
>SPEED 50 ↵	The robot motion speed is set at 50% of the maximum speed.
>SPEED 100 ↵	The robot motion speed is set at 100% of the maximum speed.

PRIME program name, execution cycles, step number

Function

Prepares the system so that a program can be executed using the A+CYCLE START key. This command alone does not execute the program.

Parameter

Program name

Selects the program to prepare for execution. If not specified, the program last executed or used in prime command will be selected.

Execution cycles

Sets how many times the program is executed. If not specified, 1 is assumed. To execute the program continuously, enter a negative number (-1).

Step number

Selects the step from which to start execution. If not specified, the execution starts from the first step of the program.

Explanation

This command only prepares the system for program execution. It does not execute the program. A program can be executed using the CONTINUE command after the PRIME command prepares the system. The program can also be executed using A+ CYCLE START keys.

[NOTE]

When using this command, the execution stack in the robot memory is initialized; i.e. any information indicating a program is held (e.g. by HOLD command or by an error) will be lost. For example, if program execution is held while in a subroutine (the information is memorized in the stack), and then the subroutine is executed using this command with CYCLE START or CONTINUE command (the stack is initialized), the processing cannot return to the main program as the stack has been initialized.

EXECUTE program name, execution cycles, step number

Function

Executes a robot program.

Parameter

Program name

Selects the program to execute. If not specified, the program last executed (by EXECUTE, PRIME, STEP, or MSTEP command) is selected.

Execution cycles

Sets how many times the program is executed. If not specified, 1 is assumed. To execute the program continuously, enter a negative number (-1). The maximum limit is 32767.

Step number

Selects the step from which to start execution. If not specified, execution starts from the first executable step of the program. If the program is executed more than once, from the second cycle, the program is executed from the first step.

Explanation

Executes a specified robot program from the specified step. The execution is repeated the specified number of cycles.

> **⚠ CAUTION**
>
> **When this command is used, the following conditions are set automatically:**
>
> **SPEED 100 ALWAYS**
>
> **ACCURACY 1 ALWAYS**
>
> **The STOP instruction or the last step of the program marks the end of a cycle.**

Example

>EXECUTE test,-1↵	Executes the program named "test" continuously. (Program execution continues until stopped by commands such as HALT, or when an error occurs.)
>EXECUTE↵	Executes the program last executed. (One cycle only).

STEP program name, execution cycles, step number
MSTEP program name, execution cycles, step number

Function

Executes one step of a robot program.

Parameter

Program name
Selects the program to execute. If not specified, the program currently suspended or the program last executed is selected.

Execution cycles
Sets how many times the program is executed. If not specified, 1 is assumed.

Step number
Selects the step from which to start execution. If not specified, execution starts from the first executable step of the program. If no parameters are specified, the step after the last executed step is selected.

Explanation

This command can be executed without parameters only in the following conditions:

1. after a PAUSE instruction,
2. after the program is stopped by causes other than error,
3. when the previous program instruction was executed using the STEP command.

MSTEP command executes one motion segment (i.e. one motion instruction and the steps before the next motion instruction). STEP command executes only one step of the program (the robot does not necessarily move).

Example

>STEP assembly,,23 ↵	Executes only step 23 of the program "assembly". Entering STEP again without parameter immediately after this executes step 24.

ABORT

Function

Stops execution of the robot program.

Explanation

Stops execution of the robot program after the current step is completed. If the robot is in motion, the execution stops after that motion is completed. Program execution is resumed using the CONTINUE command.

[NOTE]

In AS system, the motion of the robot and the step in execution may not always be the same. Therefore, if the processing of steps is faster than the motion of the robot, the robot may perform one more motion after the current motion before it stops.

HOLD

Function

Stops execution of the robot program immediately.

Explanation

The robot motion is stopped immediately. Unlike EMERGENCY STOP switch, the motor power does not turn OFF. This command has the same effect as when HOLD/RUN state is changed from RUN to HOLD. Program execution is resumed using the CONTINUE command.

CONTINUE NEXT

Function

Resumes execution of a program stopped by PAUSE instruction, ABORT or HOLD command, or as a result of an error. This command can also be used to start programs made ready to execute by PRIME, STEP or MSTEP command.

Parameter

NEXT

If NEXT is not entered, execution resumes from the step at which execution stopped. If it is entered, execution resumes from the step following the step at which execution stopped.

Explanation

The effect that keyword NEXT has on restart of the program differs depending on how the program was stopped.

1. Program stopped during execution of a step or of a motion:
   CONTINUE restarts the program and re-executes the interrupted step.
   CONTINUE NEXT restarts at the step after the step where program stopped.

2. Program execution is stopped after a step or a motion is completed:
   CONTINUE and CONTINUE NEXT restarts program from the step immediately after the completed step, regardless of NEXT.

3. Program suspended by a WAIT, SWAIT or TWAIT instruction:
   CONTINUE NEXT skips the above instructions and resumes execution from the next step.

[NOTE]

The CONTINUE command cannot resume the program execution when:

- The program ended properly
- The program was stopped using the HALT instruction
- The KILL command was used

When the program is executed by CONTINUE command, the last step is displayed when the program is completed normally. On the other hand, the first step of the program is displayed when the program completes normally by A + CYCLE START.

STPNEXT

Function

Executes the next step when the system switch STP_ONCE is ON.

Explanation

When the system switch STP_ONCE is ON, the program can be executed in one step increment. This command advances the execution to the next step in the program.

KILL

Function

Initializes the stack of the robot program.

Explanation

If the program is stopped by PAUSE instruction, ABORT command, or an error, the program stack is kept at the current status. The KILL command is used to initialize the stack. Once the KILL command is used, the CONTINUE command is ineffective, since there is no program on the stack.

DO program instruction

Function

Executes a single program instruction. (Some program instructions cannot be used with this command.)

Parameter

Program instruction

Executes the specified program instruction. If omitted, the program instruction last executed using the DO command is executed again.

Explanation

Program instructions are typically written within the programs and executed as program steps. However the DO command enables execution of a single instruction without having to create a program to run that instruction.

The robot moves in speed equivalent to monitor speed 10% when operated by DO command. When the monitor speed is set below 10%, then the robot moves at that set monitor speed.

Example

>DO JMOVE safe ↵	The robot moves to the pose “safe” in joint interpolation motion.
>DO HOME ↵	The robot moves to the home pose in joint interpolation motion.

5.5 Pose Information Commands

HERE	Assigns the current pose to the specified variable.
POINT	Defines a pose variable.
POINT/X	Sets the X value of a transformation value variable.
POINT/Y	Sets the Y value of a transformation value variable.
POINT/Z	Sets the Z value of a transformation value variable.
POINT/OAT	Sets the OAT values of a transformation value variable.
POINT/O	Sets the O value of a transformation value variable.
POINT/A	Sets the A value of a transformation value variable.
POINT/T	Sets the T value of a transformation value variable.
POINT/7	Sets the seventh axis value for a transformation value variable.
.	
.	
.	
POINT/18	Sets JT18 value of a transformation value variable.
POINT/EXT	Sets the external axis (JT7-JT18) value of a transformation value variable.

HERE pose variable

Function

Assigns the current pose to the pose variable with the specified variable name. The pose may be expressed in transformation values, joint displacement values or compound transformation values.

Parameter

Pose variable

Variable value can be specified in transformation values, joint displacement values, or compound transformation values.

Explanation

Assigns the current pose values to the specified variable in joint displacement values, transformation values, or compound transformation values.

[NOTE]

Only the right most variable in the compound transformation values is defined. (See example below). If the other variables used in the compound values are not defined, this command results in an error.

The values of the variable are displayed on the terminal followed by the message "Change?" The values can be changed by entering the values separating each value with a comma. The value that is not changed may be skipped. Press ↵ after the message "Change?" to finish editing the values.

If the variable is defined in joint displacement values (variable name starting with #), the joint values of the current pose are displayed. If the variable is transformation values, the XYZOAT values are displayed. The XYZ values describe the position of the origin of the tool coordinates with respect to the base coordinates. The OAT values describe the orientation of the tool coordinates.

Example

>HERE #pick ↵	Assigns the robot's current pose to variable "#pick" (joint displacement values)
>HERE place ↵	Assigns the robot's current pose to variable "place" (transformation values)
HERE plate+object ↵	Pose "object " is defined so that its relative pose to the pose "plate" becomes the robot's current pose. Error occurs if "plate " is undefined.

POINT pose variable 1=pose variable 2, joint displacement value variable

Function

Assigns the pose information on the right of "=" to the pose variable on the left side of "=".

Parameter

Pose variable 1

Specifies the name of pose variable to be defined by joint displacement values, transformation values, or compound transformation values.

Pose variable 2

If not specified, the "=" sign is also omitted.

Joint displacement value variable

Specifies a variable defined by joint displacement values. This parameter must be set if the pose variable values on the left are in joint displacement values and the pose variable values on the right are in transformation values (if the parameter on the left is not in joint displacement values, this parameter cannot be set). The joint displacement values specified here expresses the configuration of the robot at the pose. If not specified, the current configuration is used to define the pose variable.

Explanation

Assigns pose values specified by the parameter on the right to the pose variable specified as pose variable 1. When pose variable 2 is not specified, any value already defined for pose variable 1 is displayed on the terminal, and can be edited. If pose variable 1 is undefined, the values displayed will be 0, 0, 0, 0, 0, 0.

Once POINT is executed, the pose values appear followed by the message "Change?" and a prompt. The values can then be edited. Exit by pressing only ↵ at the prompt.

If pose variable 1 is defined by joint displacement values, joint values appear on the display. If the variable is specified by transformation values, the XYZOAT values are displayed. The XYZ values describe the position of the origin of the tool coordinates with respect to the base coordinates. The OAT values describe the orientation of the tool coordinates. When the variable is expressed in compound transformation values, the right most variable in the compound transformation value is defined. If the other variables used in the compound value are not defined, this command results in error.

[NOTE]

When value types on the right and the left side of "=" differ, this command works as follows:

1. POINT transformation values=joint displacement values
   The joint displacement values on the right are transformed into transformation values and assigned to pose variable 1 on the left.
2. POINT joint displacement values=transformation values, joint displacement values
   The transformation values on the right are transformed into joint displacement values and assigned to pose variable 1 on the left. If pose variable 3 is specified, the transformation value of pose variable 2 is transformed with the robot taking the configuration determined by the specified joint displacement values. If not specified, the transformation value is transformed with the robot in its current configuration.

When specifying values, maximum of nine decimal digits can be entered. The accuracy of entries with more than nine digits cannot be guaranteed.

Example

```
>POINT #park
```

Displays the values of joint displacement value variable "#park". (0,0,0,0,0,0 is displayed if it is undefined)

```
JT1      JT2      JT3      JT4      JT5      JT6
10.000   15.000   20.000   30.000   50.000   40.000
Change?(If not, hit RETURN only)
>,,,-15
JT1      JT2      JT3      JT4      JT5      JT6
10.000   15.000   20.000   -15.000  50.000   40.000
Change?(If not, hit RETURN only)
>↵
```

```
>POINT pick1=pick ↵
```

Assigns the transformation values of "pick" as transformation values of "pick1" and displays the values for correction.

```
>POINT pos0=#pos0 ↵
```

Transforms the joint displacement values of variable #pos0 into transformation values and assigns them to variable "pos0".

```
>POINT #pos1=pos1,#pos2 ↵
```

Transforms the transformation values of variable "pos1" into joint displacement values using the robot configuration given by variable #pos2 and assigns the values to variable #pos1.

POINT/ X　transformation value variable 1= transformation value variable 2
POINT/ Y　transformation value variable 1= transformation value variable 2
POINT/ Z　transformation value variable 1= transformation value variable 2
POINT/ OAT　transformation value variable 1= transformation value variable 2
POINT/ O　transformation value variable 1= transformation value variable 2
POINT/ A　transformation value variable 1= transformation value variable 2
POINT/ T　transformation value variable 1= transformation value variable 2
POINT/ 7　transformation value variable 1= transformation value variable 2
:
POINT/ 18　transformation value variable 1= transformation value variable 2
POINT/ EXT　transformation value variable 1= transformation value variable 2

Function

Assigns the components of the transformation values of pose variable 2 to the corresponding components of the transformation values of pose variable 1.　The values will be displayed on the terminal for editing.

Parameter

Transformation value variable 1
Specifies the variable to be defined by transformation values. (variable defined by transformation values or compound transformation values)

Transformation value variable 2
If not specified, the "=" sign can be also omitted.

Explanation

Assigns only the specified components (X, Y, Z, O, A, T, 7 - 18th axes, all external axes) of the transformation values.　Once this command is executed, the values of each component are displayed followed by the message "Change?" and a prompt.　These values can then be edited. Exit by pressing only ↵ key at the prompt.

Example

The following command assigns the OAT values of a1 to a2.　The transformation values of a1 and a2 are as below:

a1 = (1000, 2000, 3000, 10, 15, 30) , a2 = undefined

```
>POINT/OAT a2 = a1
X[mm]   Y[mm]   Z[mm]   O[deg]   A[deg]   T[deg]
   0.      0.      0.     10.      15.      30.
Change? (If not, hit RETURN only)
> ↵
```

5.6 System Control Commands

STATUS	Displays system status.
WHERE	Displays the current pose data for the robot.
IO	Displays the status of the binary signals.
FREE	Displays amount of free memory.
TIME	Displays and sets the current time and date.
LANGUAGE	Sets the display language.
ULIMIT	Sets the upper limit of the robot motion.
LLIMIT	Sets the lower limit of the robot motion.
BASE	Changes the base transformation values.
TOOL	Defines the tool transformation values.
SET_TOOLSHAPE	Sets tool shape data.
ENA_TOOLSHAPE	Enables/ disables speed control by tool shape.
TOOLSHAPE	Sets data for speed control by tool shape.
SETHOME	Sets the home pose.
SET2HOME	Sets the home pose no.2.
ERRLOG	Displays a history of error conditions.
OPLOG	Displays a history of operations.
SWITCH	Displays the system switch setting.
ON	Enable the system switch.
OFF	Disables the system switch.
ZSIGSPEC	Sets and displays the total number of I/O signals.
ZSIGMAP	Sets the signal allocation.
HSETCLAMP	Sets the default clamp specifications.
DEFSIG	Displays or sets software dedicated signals.
ZZERO	Displays or sets the zeroing data.
ERESET	Resets the error condition.
SYSINIT	Initializes the entire system.
HELP	Displays a listing of AS language commands/instructions.

ID	Displays the version information of the software.
WEIGHT	Sets the weight load data.
ENCCHK_EMG	Sets an acceptable deviation range when checking the robot's pose at an emergency stop versus the pose when the robot is restart.
ENCCHK_PON	Sets the acceptable range for the difference in encoder value when the control power is turned ON versus the value when the power was turned OFF the last time.
SLOW_REPEAT	Sets slow repeat mode speed.
REC_ACCEPT	Enables/disables recording and or changing programs.
ENV_DATA	Sets auto servo off timer and teach pendant connect/ disconnect.
ENV_2DATA	Sets terminal connect/disconnect.
CHSUM	Clears check sum error.
TPLIGHT	Turns on teach pendant backlight.
IPEAKLOG	Displays peak current values. (Option)
IPEAKCLR	Resets peak current values. (Option)
OPEINFO	Displays operation information.
OPEINFOCLR	Clears operation information.
REFFLTSET_STATUS	Displays values for moving average span of the command values.
FFSET_STATUS	Displays of robot speed/ acceleration speed feed forward gain.
ENC_TEMP	Displays the lowest and highest temperature of encoder.
SETENCTEMP_THRES	Sets the encoder temperature warning and temperature error thresholds.

STATUS

Function

Displays the status of the system and the current robot program.

Explanation

The system and the robot program status are displayed in the following format:

```
1.…  Robot status:
     REPEAT mode:
2.…  Environment:
     Monitor Speed(%) = 10.0
     Program Speed(%)ALWAYS = 100.0        100.0
     ALWAYS Accu.[mm]=   1.0
3.…  Stepper status:   Program is not running.
4.…  Execution cycles
     Completed cycles:     3
     Remaining cycles:     Infinite
5.…  Program name        Prio      Step number
     test                 0           1          WAIT   sig(1001)
```

1. Robot status

The current robot status is one of the following:

Error state:	An error has occurred; try the error reset operation.
Motor power off:	Motor power is OFF.
Teach mode:	Motor power is ON; the robot is controlled using the teach pendant.
Repeat mode:	Motor power is ON; the robot is controlled by the robot program.

Repeat mode cycle start ON: Motor power is ON; the robot program is running.

Program waiting:	Motor power is ON; the robot program is running and in wait condition (executing a WAIT, SWAIT, or TWAIT instruction).

2. Environment

Displays the conditions of current set speed and accuracy as follows:

Monitor speed (%)	The current monitor speed is displayed.
ALWAYS Program Speed (%)	The rotation speed (see SPEED instruction) is displayed if the program speed specified by ALWAYS and the direction speed option are enabled.
ALWAYS Accu.[mm]	The positioning accuracy range specified by ALWAYS is displayed.

3. Stepper status

The current status of step execution.

4. Execution cycles

Completed cycles: Execution cycles already completed (0 to 32767)

Remaining cycles: Remaining execution cycles. If a negative number was specified for execution cycles in the EXECUTE command, “infinite” is displayed.

5. Program name

The name of the program or step currently being executed or in wait condition.

WHERE display mode

Function

Displays the current robot pose.

Parameter

Display mode

Selects the mode in which the data is displayed. There are 16 modes as shown below (modes 7 to 16 are options). If the mode is not specified, transformation values of the TCP in the base coordinates and the joint angles (JT1, JT2, …, JT3) are displayed. The display mode does not change until ↵ is pressed again.

WHERE …… Displays the current robot pose in transformation values in the base coordinates and the joint angles.
WHERE 1 …… Displays the current pose by joint angles (deg).
WHERE 2 …… Displays the current pose in XYZOAT in the base coordinates (mm, deg).
WHERE 3 …… Displays the current command values (deg).
WHERE 4 …… Displays deviations from the command values (bit).
WHERE 5 …… Displays encoder values of each joint (bit).
WHERE 6 …… Displays speed of each joint (deg/s).
WHERE 7 …… Displays the current pose including the external axis. (Option)
WHERE 8 …… Displays current pose in the fixed workpiece coordinates. (Option)
WHERE 9 …… Displays the instructed value of each joint for transformation values.
WHERE 10 …… Displays the motor current.
WHERE 11 …… Displays the motor speed.
WHERE 12 …… Displays the current transformation values expressed in base coordinates of

another robot. (Option)

WHERE 13 …… Displays the current transformation values expressed in tool coordinates of another robot. (Option)

WHERE 14 …… Displays the instructed value of the motor current.

WHERE 15 …… Displays the original data of the encoder.

WHERE 16 …… Displays the speed of TCP.

WHERE 35 …… Displays the motor load arrival rate (%) for each joint.

WHERE 43 …… Displays the encoder temperature (ºC) for each joint.

Example

```
>WHERE ↵
JT1      JT2      JT3      JT4      JT5      JT6
9.999    0.000    0.000    0.000    0.000    0.000
X[mm]    Y[mm]    Z[mm]    O[deg]   A[deg]   T[deg]
15.627   88.633   930.000  -9.999   0.000    0.000
```

IO/E　signal number

Function

Displays the current status of all the external and internal I/O signals.

Parameter

Signal number

1………Displays 1–32, 1001–1032, 2001–2032

2………Displays 33–64, 1033–1064, 2033–2064

3………Displays 65–96, 1065–1096, 2065–2096

4………Displays 97–128, 1097–1128, 2097–2128

If not specified……Displays 1–32, 1001–1032, 2001–2032

Explanation

If the system switch DISPIO_01 is OFF, “o” will be displayed for signals that are ON, “x” is for signals that are OFF.　Dedicated signals are displayed in uppercase letters (“O” and “X”).　If the system switch DISPIO_01 is ON, “1” is displayed for signals that are ON and “0” for those that are OFF.　“-” is displayed for external I/O signals that are not installed.

If “/E” is entered with the command, signal numbers 3001 and above are displayed along with the signals numbered 1–, 1001–, 2001–.　(Option)

The display updates continuously until the display is terminated with the ↵ key.
(See 7.0 DISPIO_01　system switch)

Example

When DISPIO_01 is OFF

```
>IO ↵
  32 -    1    xxxx  xxxx  xxxx  xxxx  xxxx  xxxx  xxxo  xxxo
 1032 - 1001   xxxx  xxxx  xxxx  xxxx  xxxx  xxxx  xxxx  oxxx
 2032 - 2001   xxxx  xxxx  xxxx  xxxx  xxxx  xxxx  xxxx  xxxx
>
```

```
>IO/E ↵
  32 -    1   xxxx  xxxx  xxxx  xxXX  xxxx  XXXX  XXXO  XXXO
1032 - 1001   xxxx  xxxx  xxxx  xxXX  xxxx  XXXX  XXXO  XXXO
2032 - 2001   xxxx  xxxx  xxxx  xxxx  xxxx  xxxx  xxxx  xxxx
3032 - 3001   xxxx  xxxx  xxxx  xxxx  xxxx  xxxx  xxxx  xxxx
>
```

When DISPIO_01 is ON

```
>IO ↵
  32 -    1   0000  0000  0000  0000  0000  0000  0001  0001
1032 - 1001   0000  0000  0000  0000  0000  0000  0000  1000
2032 - 2001   0000  0000  0000  0000  0000  0000  0000  0000

>
```

FREE

Function

Displays the size of the memory currently not used in percentages and bytes.

Explanation

Displays the size of the memory which can be used for programs, etc. in percentages and bytes.

Example

```
>FREE ↵
Total memory 8192 kbytes
Available memory size 8191 kbytes (99 %)
```

TIME　year – month - day　hour: minute: second

Function

Sets and displays the current time and date.

Parameter

year – month –day　hour: minute: second

Sets the time and date in the format described below.　When setting "hour: minute: second", the parameter "year – month –day" cannot be omitted.　The values set are displayed followed by the message "Change?"　When all the parameters are omitted, the current time and date are displayed.

Explanation

This command sets the calendar within the robot.　The range of values for each element are as below:

Year	(00 - 99) Set the last two digits of the year.
Month	(01 - 12)
Day	(01 - 31)
Hour	(0 - 23)
Minute	(0 - 59)
Second	(0 - 59)

The current time or the value input is displayed followed by the message "Change?"　To change the data, enter new values.　Press the ↵ key to terminate the command.

Example

```
>TIME　02-04-29 09:45:46↵

>TIME
        Current time 02-04-29 09:47:33
 Change? (If not , hit RETURN only)
 02-05-17
        Current time 02-05-17 09:47:50
 Change? (If not , hit RETURN only)
    >↵
```

LANGUAGE language number

Function

Sets the display language.

Parameter

Language number

Japanese	1	English	2	Italian	3
French	4	German	5	Chinese	6
Korean	7	Polish	9	Spanish	10
Dutch	11				

Explanation

Specifying parameters and pressing ↵ key displays the language corresponding to the number. If the parameter is omitted, a list of languages which can be displayed is displayed in English and a message "LANGUAGE?" appears. Specifying parameters and pressing ↵ key displays the language corresponding to the number.

Example

>LANGUAGE 1 ↵	The display language is changed to Japanese.
>LANGUAGE ↵	The list of language which can be displayed is displayed in English. Specify a language number.
1:JAPANESE	
2:ENGLISH	
3:ITALIAN	
..	
10:SPANISH	
11:DUTCH	
>LANGUAGE ? 1 ↵	The display language is changed to Japanese.

ULIMIT joint displacment value variable
LLIMIT joint displacment value variable

Function

Sets and displays the upper/lower limits of the robot motion range.

Parameter

Joint displacement value variable

Specifies a variable defined by joint displacement values.　Sets the software limit (upper or lower) in joint displacement values.　If this parameter is not specified, the current values are displayed.

Explanation

If the parameter is specified, the values of the specified pose variable are displayed followed by the message “Change?”.　Enter the desired values after this message, as done in the POINT command. To end the command, press the ↵ key.

If the parameter is not specified, the values of the limit currently set are displayed, followed by the message “Change?”.

Example

```
>ULIMIT                                   Displays the current setting.
               JT1     JT2     JT3     JT4      JT5     JT6
  Maximum   120.00   60.00   60.00  190.00   115.00  270.00  (The maximum allowable limit)
  Current    30.00   15.00   25.00  -40.00    60.00   15.00  (Current setting)
  Change? (If not , hit RETURN only)
>110,50
               JT1     JT2     JT3     JT4      JT5     JT6
  Maximum   120.00   60.00   60.00  190.00   115.00  270.00
  Current   110.00   50.00   25.00  -40.00    60.00   15.00
  Change? (If not , hit RETURN only) ↵
```

>ULIMIT #upper ↵　　Sets the upper software limit to the pose defined as variable “#upper”.

>LLIMIT #low ↵　　Sets the lower software limit to the pose defined as variable“#low”.

BASE transformation value variable

Function

Defines the base transformation values, which specifies the pose relation between the base coordinates and the null base coordinates.

Parameter

Transformation value variable

Specifies a pose variable defined by transformation values or compound transformation values. Defines the new base coordinates. The pose variable here describe the pose of the base coordinates with respect to the null base coordinates, expressed in null base coordinates. If not specified, the current base transformation values are displayed.

Explanation

If “NULL” is designated for the parameter, the base transformation values are set as “null base” (XYZOAT=0, 0, 0, 0, 0, 0,). When the system is initialized, the base transformation values are set automatically as the null base.

After a new base transformation value is set, the values (XYZOAT) and the message “Change?” are displayed. To change the values, enter new values separated by commas and press ↵. If no parameter is specified, the current values are displayed.

When the robot moves to a pose defined by transformation values or is manually operated in base mode, the system automatically calculates the robot pose taking in consideration the base transformation values defined here.

When a pose variable is used as the parameter and if that pose variable is redefined, note that the base transformation must also be redefined using the BASE command and the newly defined pose as the parameter. The change made in the pose variable will then be reflected to the base transformation.

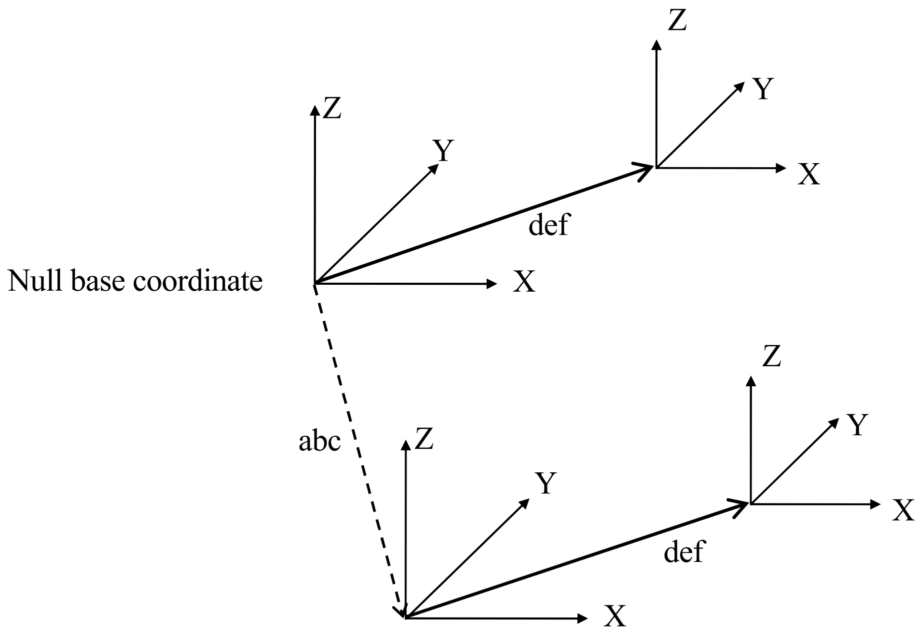
The BASE command has no effect on poses defined by joint displacement values.

[NOTE]

```
BASE abc ↵
DO JMOVE def ↵
```

```
BASE NULL ↵
DO JMOVE def ↵
```

Even if the pose information “def” are the same in both above examples, the destination of the robot will differ according to the base transformation. See the diagram below.



Example

>BASE ↵ Displays the current base transformation values.

X[mm]	Y[mm]	Z[mm]	O[deg]	A[deg]	T[deg]
-300.	0.	0.	0.	0.	0.

Change? (If not , hit RETURN only) ↵

>BASE NULL ↵ Changes the base transformation value to the null base.
(X, Y, Z, O, A, T)=(0, 0, 0, 0, 0, 0)

>BASE abc ↵ Changes the pose of the base coordinates to the pose described by the base transformation value variable "abc".

TOOL transformation value variable, tool shape number

Function

Defines the tool transformation values, which specify the pose relation between the tool coordinates and the null tool coordinates.

Parameter

Transformation value variable

Specifies a pose variable defined by transformation values or compound transformation values. Defines the new tool coordinates. The pose variable here describe the pose of the tool coordinates with respect to the null tool coordinates, expressed in null tool coordinates. If no pose variable is specified , the current tool transformation values are displayed.

Tool shape number

Specifies the tool shape to use for speed control in teach and check mode.

Explanation

If “NULL” is designated for the parameter, the tool transformation values are set at “null tool” (XYZOAT=0, 0, 0, 0, 0, 0,). The null tool coordinates have their origin at the center of the tool mounting flange and the axes are parallel to the axes of the robot’s last joint. When the system is initialized, the tool transformation values are set automatically at the null tool.

After a new tool transformation is set, the values (XYZOAT) and the message “Change?” are displayed. To change the values, enter the new values separated by commas and press ↵. If no parameter is specified, the current values are displayed.

When the robot moves to a pose defined by transformation values or is manually operated in base mode or tool mode, the system automatically calculates the robot pose taking in consideration the tool transformation values defined here.

When a pose variable is used as the parameter and if that pose variable is redefined, note that the tool transformation must also be redefined using the TOOL command and the newly defined pose as the parameter. The change made in the pose variable will then be reflected to the tool transformation. See 11.4 Tool Transformation.

Example

>TOOL grip ↵	Changes the pose of the tool coordinates to the pose described by the pose variable “grip”.
>TOOL NULL ↵	Changes the tool transformation values to null tool. (X, Y, Z, O, A, T)=(0, 0, 0, 0, 0, 0)
>TOOL tool1,1 ↵	Selects 1 for the tool shape number for the tool with tool transformation values “tool1”.

SET_TOOLSHAPE tool shape no. = transformation value variable 1, transformation value variable 2, ..., transformation value variable 8

Function

Registers the tool shape used to control speed in teach mode and check mode.

Parameter

Tool shape no.

Specifies the number of tool shape to register. Setting range: 1 to 9.

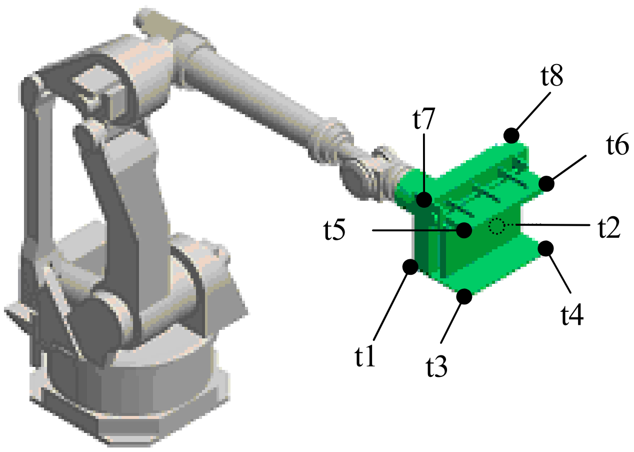
Transformation value variables 1-8

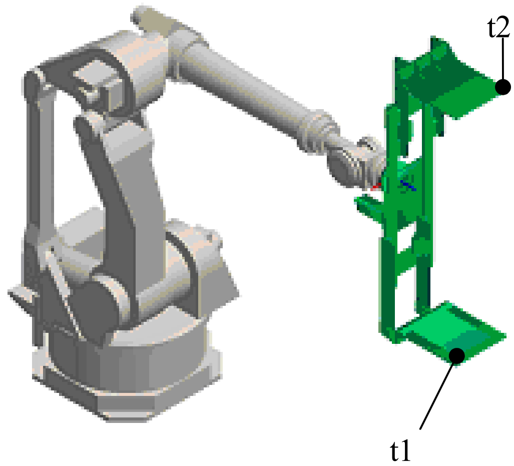
Specifies the points on the tool shape using transformation value variables. Maximum of 8 points can be specified. The points are specified in transformation values as seen from the center of the flange surface. However, only the X,Y, Z values of the transformation values are used for the tool shape registration.

Explanation

Defines the tool shape used for speed control in teach and check modes by maximum 8 points specified in pose variables defined by transformation values (t1 to t8 in the figure below). Speed should be controlled using tool shape in such cases where the tip of the tool is further away from the flange surface than the TCP, or when the shape of the workpiece attached to the tool should be put into consideration.

For tools registered via Aux. function 304, the tool shape can be registered via the screen that is displayed when pressing <Tool Shape> on the same Aux. function 304 screen.





Example

> SET_TOOLSHAPE 1=t1,t2, ↵	Specifies tool edge positions of tool shape no.1 by transformation value variables t1 and t2.
> TOOL tool1,1 ↵	Restricts the speed of edge points of tool shape no.1.

ENA_TOOLSHAPE tool shape no. = TRUE/ FALSE

Function

Enables/disables speed control in teach and check mode.

Parameter

Tool shape number

Specifies in integers from 1 to 9, the number of the tool shape to set enable/ disable.

TRUE/FALSE

Specify TRUE to enable speed control by the specified tool shape. Specify FALSE to disable the speed control.

Explanation

Selects if speed control in teach and check modes are done by the specified tool shape or not. FALSE is selected for all tool shapes as default setting. If TRUE is selected for a tool shape number with not even one point specified, error E1356 Tool shape not set occurs when the robot is operated in teach or check mode. To avoid this, always set at least one tool point via SET_TOOLSHAPE command or change from TRUE to FALSE via this command an d then execute TOOL or TOOLSHAPE command specifying the relevant tool shape number. (Once set to TRUE, the setting will not be changed to FALSE unless TOOL/TOOLSHAPE command is executed.)

TOOLSHAPE tool shape no.

Function

Selects the tool shape used to control speed in teach mode and check mode.

Parameter

Tool shape no.
Specifies the number of the tool shape used for speed control. Setting range: 1 to 9.

Explanation

To enable speed control in teach mode and check mode, the function must be enabled by ENA_TOOLSHAPE command/instruction (ENA_TOOLSHAPE n =TRUE). Error E1356 Tool shape not set occurs if a tool shape with no point registered (all points set to 0) is selected.

Example

> TOOL tool1 ↵ > TOOLSHAPE 1 ↵	Specifies the tool transformation values for the relevant tool as "tool1" and controls the speed using the tool points registered for tool shape 1.

SETHOME accuracy, HERE
SET2HOME accuracy, HERE

Function

Sets and displays the HOME pose.

Parameter

Accuracy

Sets the accuracy range of the HOME pose in millimeters. The robot is at the HOME pose when it nears HOME by the distance specified here. If not specified, the default value 1 mm is assumed.

HERE

Sets the current pose as HOME.

Explanation

If no parameters are entered, the current values are displayed followed by the message “Change?” Enter the desired value and press the ↵ key. If no change is made, press only ↵.

Two HOME poses (HOME1 and HOME2) can be set in the AS system. HOME 1 is set using SETHOME command, HOME 2 using SET2HOME command.

Example

```
>SETHOME 2↵
    JT1   JT2    JT3    JT4    JT5    JT6   accuracy[mm]
    0.    0.     0.     0.     0.     0.    2.
    Change? (If not , hit RETURN only)
    ,90,-90
    JT1   JT2    JT3    JT4    JT5    JT6   accuracy[mm]
    0.    90.    -90.   0.     0.     0.    2.
    Change? (If not , hit RETURN only)↵
    >
```

Sets the accuracy at 2 mm, and changes the HOME pose by entering the new values.

```
>SETHOME 10,HERE↵
```

Sets the current pose as the HOME pose. The accuracy is set at 10 mm; i.e. the dedicated signal HOME will be output when the robot reaches the range of 10 mm from the HOME pose.

ERRLOG

Function

Displays the error log.

Explanation

Displays the last one hundred errors. When the display reaches the end of the screen, press the Spacebar to continue viewing. Errors are listed in chronological order.
(Auxiliary function 0702)

Example

```
>ERRLOG ↵

1-[02/07/17 09:55:45 (SIGNAL:00)
   (D1016)
:
```

OPLOG

Function

Displays the operation log.

Explanation

Displays the last one hundred operations in the format shown below. When the display reaches the end of the screen, press the Spacebar to continue viewing.
(Auxiliary function 0703)

Example

```
          >OPLOG ↵
1-[02/07/17 10:04:46](SIGNAL:00) [ PNL ]
```

SWITCH switch name, ……., switch name = ON
SWITCH switch name, ……., switch name = OFF

Function

Displays and changes the system switches and their setting.

Parameter

Switch name

Displays the specified switch. If not specified, all the switches are displayed. More than one switch name can be entered separating each switch name by commas.

ON or OFF

Turns ON or OFF the specified system switch. If this parameter is not entered, the switch setting is displayed.

Explanation

Displays the setting state of the specified system switch with ON (enabled) or OFF (disabled). For the switches with * at the beginning of the names, only the display of the setting state is possible. The settings cannot be changed.

Example

```
>SWITCH ↵                 Displays all system switches and their setting.
*POWER          ON        *REPEAT                   ON
*RUN            ON        *CS             OFF
*RGSO           OFF       *ERROR          OFF
*TRIGGER        ON        *TEACH_LOCK     OFF
CHECK.HOLD      OFF        CP             ON
CYCLE.STOP      OFF        OX.PREOUT      ON
PREFETCH.SIGINS           OFF       QTOOL           OFF
REP_ONCE                  OFF       RPS             OFF
STP_ONCE                  OFF       AFTER.WAIT.TMR          ON
MESSAGES                  ON        SCREEN                  ON
AUTOSTART.PC  OFF          AUTOSTART2.PC         OFF
AUTOSTART3.PCOFF           ERRSTART.PC    OFF
DISPIO_01       OFF        HOLD.STEP      OFF
FLOWRATE        OFF        SPOT_OP        OFF
>↵
```

```
>SWITCH    SCREEN, MESSAGE = OFF ↵    Turns OFF SCREEN and MESSAGE.
SCREEN       OFF
MESSAGE      OFF
>↵
```

switch name, ……. ON

Function

Turns ON the specified system switch.

Parameter

Switch name

Turns ON the switch specified here. More than one switch name can be entered separating each switch name with a comma.

The current setting of the switch can be checked using the SWITCH command.

Example

>MESSAGES ON ↵	Turns ON the switch MESSAGES.
>SCREEN, MESSAGES ON ↵	Turns ON the switches MESSAGES and SCREEN.

switch name, ……. OFF

Function

Turns OFF the specified system switch.

Parameter

Switch name

Turns OFF the switch specified here. More than one switch name can be entered separating each switch name with a comma.

The current setting of the switch can be checked using the SWITCH command.

Example

>MESSAGES OFF ↵	Turns OFF the switch MESSAGES.
>SCREEN, MESSAGES OFF ↵	Turns OFF the switches MESSAGES and SCREEN.

ZSIGSPEC

Function

Displays and sets max. number of external I/O signals.

Explanation

The current number of signals and message are displayed. (See Example)
FB1 gives the signal numbers for the FB1 port, and FB2 for the FB2 port.
When not changing the setting, just press ↵.

1. ZSIGSPEC sets the value only in the software. Setting is invalid unless hardware corresponds to it.
2. Set the signal numbers in multiples of 16.
3. Max. number of external I/O signals is 960:
   Input signals DI + FB1 + FB2 ≤ 960
   Output signals DO + FB1 + FB2 ≤ 960
4. Turn controller power OFF, then ON to activate the number of I/O signals set as I/O data length for the physical fieldbus interface.

[NOTE]

Ensure that the max. number of signals set by this command is consistent with the setting in [Aux. 0608 Signal Allocation]. If not, the max. number of signals cannot be set by this command.

Example: When increasing the number of FB1 and FB2 port signals

```
>ZSIGSPEC ↵
            DO,   DI,   INT,   FB1,   FB2
            64    64    960     32     32
Change? (If not, Press RETURN only)
, , , 112, 64
            DO,   DI,   INT,   FB1,   FB2
            64    64    960    112     64
Change? (If not, Press RETURN only)
```

ZSIGMAP type, AS signal start number, port signal start number, number of signals

Function

Specifies the signal allocation setting.

Parameter

1. Type
   Specifies the port type.
   0: DIO port
   1: FB1 port
   2: FB2 port

2. AS signal start number
   1 to AS max. number of signals

3. Port signal start number
   1 to port max. number of signals

4. Number of signals
   1 to max. number of signals of each port

Explanation

Specifies the signal allocation setting.
When the AS system switch, SIGMAPD, is OFF, a setting equivalent to Aux. 0608 setting is specified.
When a setting that cannot be achieved with a standard signal allocation setting is specified in Advanced Signal Allocation setting, error message P4706, "Cannot use this data by Signal Allocation Setting.," appears.
Specifying the ZSIGMAP/Z identifier will overwrite the setting in Advanced signal allocation setting.

Example 1: When allocating 1 to 480 of FB1 port to 1 to 480 of AS signal

```
ZSIGMAP 1,1,1,480
```

Example 2: When allocating 1 to 480 of FB2 port to 481 to 960 of AS signal

```
ZSIGMAP 2,481,1,480
```

HSETCLAMP

Function

Assigns signal numbers to operate material handling clamps.

Example

In the example below, clamp 3 is set as a double solenoid.

```
>HSETCLAMP ↵
                    Clamp 1     Clamp 2     Clamp 3     Clamp 4
                    Spot weld   Handling    Not used    Not used
'ON'out.signal        10           0          24          24
'OFF'out.signal        9          11           0           0
                    Clamp 5     Clamp 6     Clamp7      Clamp 8
                    Not used    Not used    Not used    Not used
'ON'out.signal        24          24          24          24
'OFF'out.signal        0           0           0           0
Clamp number (1~8, ENTER only:No change, CTRL+C:Exit)  3        ;Select clamp number
Define as Handling clamp ?
  (1:Defined as Handling clamp, 0:Not used, ENTER only:No change, CTRL+C:Exit)1
                                                   ;Enter 1 to define as handling clamp.
  For single solenoid valve, define one signal.
  For double solenoid valve, define both.
'ON' out. signal
(0:Not used, ENTER only:No change, CTRL+C:Exit) Change ? 12;    Set so that ch12 is output
                                                                if ON
'OFF' out. signal
(0:Not used, ENTER only:No change, CTRL+C:Exit) Change ?; 13    Set so that ch13 is output
                                                                if OFF

                    Clamp 1     Clamp 2     Clamp 3     Clamp 4
                    Spot weld   Handling    Handling    Not used
  'ON'out.signal      10           0          12          24
  'OFF'out.signal      9          11          13           0
                    Clamp 5     Clamp 6     Clamp7      Clamp 8
                    Not used    Not used    Not used    Not used
  'ON'out.signal      24          24          24          24
  'OFF'out.signal      0           0           0           0
Clamp number (1~8, ENTER only:No change, CTRL+C:Exit) ↵
```

[NOTE]

Always use the clamps in order from one to eight. For example clamp 5 cannot be used without using clamp 4.

[NOTE]

Ctrl + C (Exit) cannot be used from the teach pendant keyboard screen.

DEFSIG INPUT
DEFSIG OUTPUT

Function

Displays and changes the current setting of the software dedicated signals.

Parameter

INPUT, OUTPUT

OUTPUT (or only O) displays the output signals, INPUT (or only I) the INPUT signals. The setting can be changed when this parameter is entered. If this parameter is not entered, the signals currently used as dedicated signals are displayed in a list.

Explanation

The signals in the table below can be used as dedicated signals.
For details on dedicated signals, refer to the External I/O Manual.

Software Dedicated Input Signal	Software Dedicated Output Signal
EXT. MOTOR ON	MOTOR_ON
EXT. ERROR RESET	ERROR
EXT. CYCLE START	AUTOMATIC
EXT. PROGRAM RESET	CYCLE START
Ext. prog. select (JUMP_ON, JUMP_OFF RPS_ON, RPSxx)	TEACH MODE HOME1, HOME2
EXT_IT	POWER ON
EXT. SLOW REPEAT MODE	RGSO
	Ext. prog. select (JMP_ST, RPS_ST)

The following codes can be used with this command.
L: Go back to the previous signal.
N: Go to the next signal.
Q: Cancel operation (the data input are ignored)
E: Exit (Input data is enabled)

[NOTE]

1. External program selection
   (1) When selecting JMP as a dedicated signal, signals JMP-ON, JMP-OFF, JMP-ST are also automatically set as dedicated. JMP-ST is an output signal but is set under DEFSIG INPUT command.

   (2) When selecting RPS signal as a dedicated signal, signals RPS-ON, RPS-ST are also automatically set as dedicated signals. RPS-ST is an output signal but is set under DEFSIG INPUT command.

2. RPS code
   If at least one of the following signals is selected as dedicated signal, a prompt appears and an RPS code must be input.
   JMP, RPS or EXT. PROGRAM RESET

3. Signal numbers
   Signals can be set within the following range:
   Dedicated output signals: 1 to number of signals installed
   Dedicated input signals: 1001 to number of signals installed

4. Others
   If a signal number is already assigned to a dedicated signal, it cannot be assigned to another dedicated signal or used as a general purpose signal.

Example

The following example displays the currently selected software dedicated signals.

```
>DEFSIG
Dedicated signals are set

 EXT. MOTOR ON = 1032
 EXT. ERROR RESET = 1031
 EXT. CYCLE START = 1030
 MOTOR ON = 32
 ERROR = 31
 AUTOMATIC = 30
   Condition : Panel switch in RUN.
   Condition : Panel switch in REPEAT.
   Condition : Repeat continuous.
   Condition : Step continuous.
```

```
  CYCLE START = 29
  TEACH MODE = 28
  HOME1 = 27
>↵
```

The following example resets the selection of the software dedicated output signal MOTOR_ON, changes the signal number of AUTOMATIC to 30, selects TEACH MODE as dedicated signal and sets the signal number 3.

```
>DEFSIG OUTPUT
MOTOR ON     Dedication cancel? (Enter 1 to cancel.)1 ↵
ERROR     Dedication cancel? (Enter 1 to cancel.) ↵
    Signal number    31    Change ?   (1 - 32 )   ↵
AUTOMATIC     Dedication cancel? (Enter 1 to cancel.) ↵
    Signal number    2    Change ?   (1 - 32 )   30   ↵
CYCLE START     Dedication cancel? (Enter 1 to cancel.) ↵
    Signal number    29    Change ?   (1 - 32 )   ↵
TEACH MODE     Dedication set? (Enter 1 to set.) 1 ↵
    Signal number    0    Change ?   (1 - 32 )   3   ↵
HOME1     Dedication cancel? (Enter 1 to cancel.) ↵
>
```

ZZERO joint number

Function

Sets the encoder value corresponding to the mechanical origin of each axis of a robot as zeroing data. Also, the current zeroing data and the value of encoder rotation counter can be displayed using this command.

Parameter

Joint number

1. To reset the encoder rotation counter:
   Enter the joint number plus 100. For example, to reset the encoder rotation counter on joint two, enter:
   ```
   ZZERO   102↵
   ```

   If “100” is entered as the joint number, all the encoder counters are reset.

2. To set zeroing data:
   To set the encoder zeroing data for joint two, enter:
   ```
   ZZERO   2↵
   ```

   If “0” is entered as the joint number, all the joints are zeroed.

   If no joint number is specified, the current encoder data and the zeroing data are displayed.

[NOTE]

Reset the encoder rotation counter before setting the zeroing data.

DANGER

Use this command only for the following purposes:

1. To check if the zeroing data has changed when the position of the arm is abnormal.
2. To correct the zeroing data when it has changed unexpectedly.

When the zeroing data is changed, the values detected for robot poses also change. Therefore be aware that the same program ends in different destination pose and trajectory before and after the zeroing data is changed.

Example

1. The following command displays the zeroing data:

>ZZERO ↵

	JT1	JT2	JT3	JT4	JT5	JT6
Set data	268435456	268435456	268435456	268435456	268435456	268435456
Current data	268435456	268435456	268435456	268435456	268435456	268435456

Change?(If not , hit RETURN only)↵

	JT1	JT2	JT3	JT4	JT5	JT6
OFFSET	0	0	0	0	0	0

Change? (If not , hit RETURN only)↵

>

2. The following command resets the encoder rotation counter of all the joints.

>ZZERO 100

** Encoder rot. counter reset (all joints) **

Are you sure? (Enter 1 to execute) 1 ↵

Setting complete.

>

3. The following command resets the encoder rotation counter of joint 2 with the joint value specified for [Current angle] as the current value.

>ZZERO 102

** Encoder rot. counter reset (joint 2) **

Current angle (deg, mm) ? 0 ↵

Are you sure? (Enter 1 to execute) 1 ↵

Setting complete.

>

4. The following command sets the zeroing data of all the joints simultaneously. The current value will become 0°.

>ZZERO 0

Set current values of all joints as zeroing data? (Enter 1 to set.)1 ↵

Setting complete.

>

5. Resets the zeroing data of joint 2 with the value specified for [Current angle] as the current value.

```
>ZZERO 2 ↵
  Current angle (deg.mm)?    0 ↵
  Change? (If not, Press RETURN only.)  ↵
  Encoder value? (Current=268435456, Enter 1to set current value) 1 ↵
  Zeroing value=268435456 (268419072-268451840) OK?  (Enter 0 to change) ↵
  Setting complete.
 >
```

ERESET

Function

Resets the error condition. Identical to the ERROR RESET button on the operation panel.

Explanation

When the ERESET command is executed, the ERROR_RESET signal is output. However this command is ineffective when an error occurs continuously.

SYSINIT

Function

Deletes all program and data in the memory and initializes defined parameters.

Explanation

Initializes the system and deletes all programs, pose variables, numeric variables, and string variable.

[NOTE]

All programs and variables are deleted from the memory when this command is executed.

HELP alpha character
HELP/ M alpha character
HELP/ P alpha character
HELP/ F alpha character
HELP/ PPC alpha character
HELP/ MC alpha character
HELP/ DO alpha character
HELP/ SW alpha character

Function

Displays a list of AS Language commands and instructions.

Parameter

Specifies with which letter in the alphabet the command, instruction, etc. begins. If omitted, all the commands, instructions are displayed.

For example, entering HELP command followed by an alpha character displays the monitor commands or program instructions starting with that alphabet. Entering HELP/F command followed by an alpha character command displays the functions starting with that alphabet.

Explanation

Entering HELP only, displays a list of monitor command and program instructions.
HELP/M lists the monitor commands.
HELP/P lists the program instructions.
HELP/F lists functions.
HELP/PPC lists program instructions usable in PC programs. (Option)
HELP/MC lists monitor commands usable with the MC instruction. (Option)
HELP/DO lists program instructions usable with DO command. (Option)
HELP/SW displays a list of system switches. (Option)

For some commands and instructions, the parameters are also displayed.

Example

```
>HELP/M↵
    ABORT   BASE        BITS    BATCHK  CONTINUE  COPY    DEFSIG
    DELETE  DIRECTORY   DLYSIG  DO      EDIT      ERESET  ERRLOG

>HELP/F↵
    #DEST    #PPOINT   $CHR    $DECODE    $ENCODE    $ERROR
    $LEFT    $MID      $RIGHT  $SPACE     ABS        ASC
```

ID

Function

Displays the version information of the software installed in the robot controller.

Explanation

Displays the following information.

Robot name:	the name of the robot arm controlled by AS software
Joint number:	number of joints of the controlled robot arm
Serial number:	serial number of the controlled robot arm
Number of signals:	maximum number of output, input, and internal signals available in this system
Clamp number:	maximum number of clamps available in this system
Motion type:	motion type currently set
Servo type:	type of servo software currently set
Acceleration speed change per load:	enable/ disable of acceleration speed change function per load
Servo specification:	control specification of the servo software
Software version:	version number of the AS software

[NOTE]

If the above information does not match the actual robot, contact us immediately. Do not turn ON the motor power nor make the robot do any motion operations.

Example

```
>ID ↵
        Robot name: RS010N-A001     Num of axes: 6          Serial No. 1
        Number of signals: output=32          input=32          internal=960
        Clamp number: 2     MOTION TYPE: 2     SERVO TYPE: 2
        ACC. & DEC. VARIABLE BY WEIGHT : OFF
        Servo Spec: 5
    [SOFT VERSION]
=== AS GROUP ===          : ASF_010000005 2016/09/16 11:33
USER IF AS                : UASF010000005 2016/09/16 11:33
USER IF TP                : UTPF010000005 2016/09/16 11:31
ARM CONTROL AS            : AASF010000005 2016/09/16 11:32
```

```
USER IF AS MESSAGE FILE : MASF0100005EN 2016/09/14 18:00
USER IF TP MESSAGE FILE : MTPF0100005EN 2016/09/14 18:00
ARM DATA FILE           : ARMF010000005 2016/09/16 11:31
KERNEL                  : _KNL000610000 2016/08/26
DRIVER                  : _DRV100500001 2016/08/26
RFS                     : _RFS100500001 2016/08/25
=== SERVO GROUP ===     : SVF_010000003 2016/09/14 18:42
ARM CONTROL SERVO       : SVSF010000003 2016/09/14 18:21
SRV DATA FILE           : SVPF010000003 2016/09/14 18:42
ARM CONTROL SERVO FPGA: SVFF010000002 2016/09/14 18:08
```

[**NOTE**]

Displayed software version may vary from above according to the selected option.

WEIGHT **load mass, center of gravity X, center of gravity Y, center of gravity Z, inertia moment ab. X axis, inertia moment ab. Y axis, inertia moment ab. Z axis**

Function

Sets the load mass data (weight of tool and workpiece).　The data is used to determine the optimum acceleration of the robot arm.

Parameter

Load mass
The mass of the tool and workpiece (in kilograms).　Range: 0.0 to the maximum load capacity (kg).

Center of gravity (unit = mm)
X: the x value of the center of gravity in tool coordinates
Y: the y value of the center of gravity in tool coordinates
Z: the z value of the center of gravity in tool coordinates

Inertia moment about X axis, inertia moment about Y axis, inertia moment about Z axis (Option)
Sets the inertia moment around each axis.　Unit is $kg \cdot m^2$.　The inertia moment about each axis is defined as the moment around the coordinates axes parallel to the null tool coordinates with the center of rotation at the tool's center of gravity.

Explanation

If no parameters are specified, the current value is displayed followed by the message "Change?"

> **⚠ DANGER**
>
> **Always set the correct load mass and center of gravity.　Incorrect data may weaken or shorten the longevity of parts or cause overload/deviation errors.**

ENCCHK_EMG

Function

Sets an acceptable deviation range when checking the robot's pose at an emergency stop versus the pose when the robot is restarted.

Explanation

Deviation ＝|(pose after motor power is reapplied) –(pose after the emergency stop)|

The acceptable deviation range can be set at each joint. If 0.0 is set as the range, deviation check is not performed. Setting too small of a range may trigger an error when motor power is reapplied after an emergency stop even if the robot is operating within performance specifications.

ENCCHK_PON

Function

Sets the acceptable range for the difference in encoder value when the control power is turned ON versus the value when the power was turned OFF the last time.

Explanation

Acceptable range = | (value when control power is turn ON) – (value when the control power was turned OFF the last time)|

The acceptable range can be set at each joint. Setting too small of a range may trigger an error when motor power is reapplied after an emergency stop even if the robot is operating within performance specifications.

SLOW_REPEAT

Function

Sets the repeat speed in slow repeat mode.

Explanation

```
>SLOW REPEAT
 SLOW REPEAT MODE Speed  (1-25%)
(Enter only: No change ^C:Exit): Now   10   Change ?
```

If no change is made, press only ↵. To change the speed, enter the new value and press ↵.

> **[NOTE]**
>
> Ctrl + C (Exit) cannot be used from the teach pendant keyboard screen.

REC_ACCEPT

Function

Enables or disables RECORD and PROGRAM CHANGE functions.

Explanation

```
>REC ACCEPT ↵
RECORD(0:Enable, 1:Disable)
          (Enter only: No change ^C:Exit): Now    0 Change ?
PROGRAM CHANGE(0:Enable, 1:Disable)
          (Enter only: No change ^C:Exit): Now    0 Change ?
```

Enter 0 to enable RECORD or PROGRAM CHANGE option. Enter 1 to disable the options.

> **[NOTE]**
>
> 1. Ctrl + C (Exit) cannot be used from the teach pendant keyboard screen.
> 2. If PROGRAM CHANGE is disabled, the following message appears when EDIT command is executed: "Program change inhibited. Set ACCEPT and operate again." REC_ACCEPT command cannot be used in EDIT mode.

ENV_DATA

Function

Sets hardware environmental data. (Auto servo OFF timer and status of teach pendant installation)

Explanation

```
>ENV DATA ↵
 AUTO SERVO OFF TIMER(0:Servo not off)
    (Enter only: No change ^C:Exit): Now    0 Change ?
```

If no change is to be made, press only ↵. Enter 0 to disable the auto servo OFF timer. To enable the timer, enter after how much time (in seconds) the servo turns OFF.

Next, a prompt for the teach pendant is displayed.

```
 TEACH PENDANT(0:Connect, 1:Disconnect)
    (Enter only: No change ^C:Exit): Now    0 Change ?
```

If no change is made, press only ↵. To operate the robot without connecting the teach pendant, enter 1. Connect the short circuit plug after disconnecting the teach pendant.

[NOTE]

Ctrl + C (Exit) cannot be used from the teach pendant keyboard screen.

ENV2_DATA

Function

Sets software environmental data.

Explanation

```
>ENV2 DATA
PANEL (0:Connect, 1:Disconnect)
         (Enter only: No change ^C:Exit): Now    0 Change ?
```

If no change is made, press only ↵. Next, a prompt for the terminal setting is displayed.

```
TERMINAL (0:Connect, 1:Disconnect)
         (Enter only: No change ^C:Exit): Now    0 Change ?
```

If no change is made, press only ↵. Usually the personal computer is set as the terminal.

[NOTE]

Ctrl + C (Exit) cannot be used from the teach pendant keyboard screen.

CHSUM

Function

Enables or disables the resetting of abnormal check sum error.

Explanation

```
>CHSUM
  CLEAR CHECK SUM ERROR(0:Ineffect, 1:Effect)
      (Enter only: No change ^C:Exit): Now    0 Change ?
```

If “0” is entered, error cannot be reset. If “1” is entered the error is reset. The default value is “0”. CHSUM is reset to “0” when the control power is turned OFF.

The following message appears if the error cannot be reset:

```
>CHSUM
Cannot clear check sum error. Check the following command or auxiliary data.
   ZZERO
   DEFSIG
   :
   :
>
```

If any data still contains an abnormal check sum, the message in the first example (CLEAR CHECK SUM ERROR) does not appear. Instead, a message (as in the second example) is displayed identifying additional troubleshooting.

[NOTE]

Ctrl + C (Exit) cannot be used from the teach pendant keyboard screen.

TPLIGHT

Function

Turns on the teach pendant backlight.

Explanation

If the backlight of the teach pendant screen is OFF, then this command turns ON the light. If this command is executed when the backlight is ON, the light stays on for the next 600 seconds.

IPEAKLOG

Option

Function

Displays the peak current value for each joint.

Explanation

Displays the program name, step number, effective current value [Arms], and the ratio of peak value to mechanical limit or motor limit when the motor torque is at its highest for each joint.

Example

```
>IPEAKLOG↵
Log since   00/1/24   13:44:23
```

Joint	Program	Step	Effective current		Date
JT1	pg223	5	4.1[Arms]	29.5[%]	00/1/24 13:44
JT2	pg223	13	1.4[Arms]	9.8[%]	00/1/24 13:44
JT3	pg223	10	13.7[Arms]	98.2[%]	00/1/24 13:44
JT4	pg223	1	2.1[Arms]	55.5[%]	00/1/24 13:45
JT5	pg223	7	2.4[Arms]	64.8[%]	00/1/24 13:45
JT6	pg223	4	1.0[Arms]	27.8[%]	00/1/24 13:45

IPEAKCLR

Option

Function

Clears the peak current value log and restarts logging values.

Explanation

The logged values are reset using the IPEAKCLR command, the SYSINI command, or by initializing the system by turning on No.8 dip switch on 1TA board.

Example

```
>IPEAKCLR↵
Are you sure? (Yes:1,No;0) 1↵
>
```

OPEINFO robot number: joint number

Function

Displays the operation information.

Parameter

Robot number

Specifies the robot if more than one robot is controlled by one controller.

Joint number

Specifies for which joint the information is to be displayed. If not specified, the data on all the joints are displayed.

Example

```
        >OPEINFO↵
Operation Info. (02/1/14   9:54:1 - )(FILE LOAD   02/1/14)        ;describes from which hour data
Control ON          0.2 [H]                                         accumulation started
Servo ON            0.1 [H]
Motor ON            4 times
Servo ON            10 times
Emergency stop (while in motion)     2 times
JT1
Operation hour      0.1 [H]
Operation distance  301.28 [x100 deg, mm]
JT2
Operation hour      0.1 [H]
Operation distance  193.84 [x100 deg, mm]
:
```

[NOTE]

Limits on data accumulation

1. Hours[H]:69 years
2. Number of operation [times]: ab. 2000 million times (1176 years if operated 10000 times a day)
3. Distance [deg, mm]: 12.5 years if operated at 500 mm/s

OPEINFOCLR

Function

Resets the operation information to 0.

REFFLTSET_STATUS

Function

Displays the default and current values for moving average span of the command values.

Explanation

This instruction allows checking the default and current values of the moving average span modified by REFFLTSET instruction.

Example

In the sample program below, the default value of the command value moving average span is 48 ms. The screen display will show as below if REFFLTSET_STATUS instruction is executed while the robot is moving from #p1 to #p2.

Program example

```
1 JMOVE #p1
2 REFFLTSET 128
3 JMOVE #p2
```

Example of screen display (REFFLTSET_STATUS instruction is executed while step 3 is being executed):

```
>REFFLTSET_STATUS
 Default value [ms] = 48 48 48 24
 Current value [ms] = 128 128 128 64
```

FFSET_STATUS

Function

Displays the default and current values of robot speed/ acceleration speed feed forward gain.

Explanation

Allows checking the default and current values of speed/acceleration feed forward gain changed via FFSET instruction.

Example

In this example, the below sample program is executed with the default value of 0.7 for speed/ acceleration speed feed forward gain. The screen display shows as below when FFSET_STATUS instruction is executed while the robot is moving from #p1 to #p2.

Program example

```
1 JMOVE #p1
2 FFSET 0.5
3 JMOVE #p2
```

Example of screen display (FFSET_STATUS instruction is executed while step 3 is being executed):

```
>FFSET_STATUS
Default value: KVFF = 0.70, 0.70, 0.70, 0.70, 0.70, 0.70
Current value: KVFF = 0.50, 0.50, 0.50, 0.50, 0.50, 0.50
Default value: KAFF = 0.70, 0.70, 0.70, 0.70, 0.70, 0.70
Current value: KAFF = 0.50, 0.50, 0.50, 0.50, 0.50, 0.50
```

ENC_TEMP robot number: joint number

Function

Displays the encoder temperature information of each joint.

Parameter

Robot No.

Specifies a robot number when one controller is controlling multiple robots.

Joint number

Specifies the joint number to display encoder temperature information. Specifying 0 resets the encoder temperature information of all joints. Displays encoder temperature information of all controlled joints when omitted.

Explanation

Displays each joint's encoder temperature information (current temperature, lowest temperature, detection date/time of lowest temperature, highest temperature, detection date/time of highest temperature), as well as the ability to reset encoder temperature information.

Example

```
>ENC_TEMP ↵
Robot No.1
JT   Current temp.  Lowest temp.  Lowest temp.           Highest temp.  Highest temp.
     (ºC)           (ºC)          Detection date/time   (ºC)           Detection date/time
1    35.000         25.000        [19/01/01 09:00:00]   60.000         [19/01/01 15:00:00]
2    35.000         25.000        [19/01/01 09:00:00]   60.000         [19/01/01 15:00:00]
3    35.000         25.000        [19/01/01 09:00:00]   60.000         [19/01/01 15:00:00]
4    35.000         25.000        [19/01/01 09:00:00]   60.000         [19/01/01 15:00:00]
5    35.000         25.000        [19/01/01 09:00:00]   60.000         [19/01/01 15:00:00]
6    35.000         25.000        [19/01/01 09:00:00]   60.000         [19/01/01 15:00:00]
```

SETENCTEMP_THRES [/N] robot number: joint number, warning threshold, error threshold

Function

Performs the threshold setting of encoder temperature warning and temperature error of each joint.

Parameter

/N

Specifies the presence/absence of inquiries. Specifying /N means no inquiry.

Robot No.

Specifies a robot number when one controller is controlling multiple robots.

Joint number

Specifies the joint number to set the encoder temperature warning and temperature error thresholds. Sets encoder temperature warning and temperature error thresholds for all joints by joint when omitted.

Warning threshold

Sets the encoder temperature warning threshold. Warning threshold can be set between 0 and 125ºC. Specify -1 to reset to default value.

Error threshold

Sets the encoder temperature error threshold. Error threshold can be set between 0 and 125ºC. Specify -1 to reset to default value.

Explanation

Allows the setting and resetting of the encoder temperature warning and temperature error thresholds of each joint.

[NOTE]

Encoder temperature warning thresholds and temperature error thresholds set by this command may be reset to default values due to encoder replacement.

Temperature warning: (W1085) “Encoder temperature exceeded limit.(jt XX) (XX deg C)”
Temperature error: (E1564) “Encoder temperature exceeded limit.(jt XX) (XX deg C)”

CAUTION

Do not raise threshold temperatures above default values, as any rise of temperature may affect arm components such as the encoder.

Example

When setting the encoder temperature warning threshold as 80ºC and encoder temperature error threshold as 90ºC for JT2:

```
>SETENCTEMP_THRES 2↵
Robot No.1
JT2 Encoder temperature threshold
Warning[deg C]
  95 (DEFAULT SETTING: 95[deg C])
Change? (If not, Press RETURN only.)
80
Warning[deg C]
  80 (DEFAULT SETTING: 95[deg C])
Change? (If not, Press RETURN only.)

Error[deg C]
  95 (DEFAULT SETTING: 95[deg C])
Change? (If not, Press RETURN only.)
90
Error[deg C]
  90 (DEFAULT SETTING: 95[deg C])
Change? (If not, Press RETURN only.)
```

5.7 Binary Signal Commands

The robot controller uses two types of binary signals: external I/O signals between the robot and external devices, and internal I/O signals used within the robot. Internal I/O signals are used between robot and PC programs, or as test signals to check programs before actually connecting external devices.

The binary signals are controlled or defined using the following commands.

RESET	Turns OFF all external I/O signal.
SIGNAL	Turns ON (or OFF) output signals.
PULSE	Turns ON output signal for the specified amount of time.
DLYSIG	Turns ON output signal after the specified time has passed.
BITS	Sets signals to be equal to the specified value (up to 16 signals).
BITS32	Sets signals to be equal to the specified value (up to 32 signals).
SCNT	Outputs counter signal upon reading a specified value.
SCNTRESET	Clears the counter signal number.
SFLK	Turns ON/OFF the flicker signal.
SFLP	Turns ON/OFF signals with SET/RESET signals.
SOUT	Outputs signal when specified condition is met.
STIM	Turns ON timer signal when specified signal is ON.
SETPICK	Sets the time to start clamp close control.
SETPLACE	Sets the time to start clamp open control.
HSENSESET	Starts monitoring of specified sensor signal. (Option)
HSENSE	Reads the data stored by HSENSESET. (Option)
RSIGRANGE	Sets the signal range to be used with RSIGPOINT instruction.

RESET

Function

Turns OFF all the external output signals. Dedicated signals, clamp signals and antinomy signals for multifunction OX/WX are not affected by this command.

By using the optional setting, the signals used in the Interface Panel screen are not affected by this command. (Option)

[NOTE]

Beware that this command turns OFF all the signals other than those mentioned above even in repeat mode.

SIGNAL signal number, ……

Function

Turns ON (or OFF) the specified external or internal I/O signal.

Parameter

Signal number

Selects the number of external output signal or internal signal. Positive number turns ON the signal, negative number turns OFF.

Explanation

The signal number determines if the signal is an external or internal signal.

Acceptable Signal Numbers

External output signal	1 – actual number of signals
Internal signal	2001–2960
External input signal	Cannot be defined

If the signal number is positive, the signal is turned ON; if negative, the signal is turned OFF. If "0" is given, all the signals are turned OFF. This command does not take effect upon dedicated signals, clamp signals, and multipurpose double OX/WX signals.

Example

>SIGNAL -1,4,2010 ↵ External output signal1is OFF, 4 is ON, Internal signal 2010 is ON.

>SIGNAL -reset,4 ↵	If the value of the variable "reset" is positive, the output signal determined by that value is turned OFF, and output signal 4 is turned ON.

PULSE signal number, time

Function

Turns ON the specified signal for the given period of time.

Parameter

Signal number

Selects the number of the external output signal or internal signal (only positive values). Error occurs if the signal number is already used as a dedicated signal.

Acceptable Signal Numbers

External output signal	1 – actual number of signals
Internal signal	2001–2960

Time

Sets for how long the signal is output (in seconds). If not specified, it is automatically set at 0.2 seconds.

DLYSIG signal number, time

Function

Outputs the specified signal after the given time has passed.

Parameter

Signal number

Selects the number of the external output signal or internal signal. If the signal number is positive, the signal is turned ON; if negative, the signal is turned OFF. Error occurs if the signal number is already used by a dedicated signal.

Acceptable Signal Numbers

External output signal	1 – actual number of signals
Internal signal	2001–2960

Time

Specifies the time to hold the output of the signal in seconds.

BITS starting signal number, number of signals = value

Function

Arranges a group of external output signals or internal signals in a binary pattern. The signal states are set ON/OFF according to the binary equivalent of the specified value. If the value is not specified, the current signal states are displayed.

Parameter

Starting signal number

Specifies the first signal to set the signal state.

Number of signals

Specifies the number of signals to be set ON/OFF. The maximum number allowed is 16. To set more than 16 signals, use BITS32 command, explained below.

Value

Specifies the value used to set the desired ON/OFF signal states. The value is transformed into binary notation and each bit of the binary value sets the signal state. The least significant bit corresponds to the signal with the smallest signal number, and so on. If the binary notation of this value has more bits than the number of signals, only the state of the given number of signals (starting from the specified signal number) is set and the remaining bits are ignored.

If this parameter is omitted, the current signal states are displayed.

Explanation

Sets (or resets) the signal state of one or more external output signals or internal signals according to the given value.

Acceptable Signal Numbers

External output signal	1 – actual number of signals
Internal signal	2001–2960

Specifying a signal number greater than the number of signals actually installed results in error. Selecting a dedicated signal also results in error.

Example

>BITS 2001,3 ↵	Displays the values of internal signals 2001-2003. (3 bits starting from signal number 2001).
BITS 2001,3 = 5	When internal signals 2001, 2003 are ON and 2002 is OFF, displays 5 of decimals in binary representation of 101.
>BITS 1,8=100 ↵	External output signals 1−8 are set to output 01100100 (the binary notation of 100).

BITS32 starting signal number, number of signals = value

Function

Arranges a group of external output signals or internal signals in binary pattern. The signal states are set ON/OFF according to the binary equivalent to the specified value. If the value is not specified, the current signal states are displayed.

Parameter

Starting signal number
Specifies the first signal to set the signal state.

Number of signals
Specifies the number of signals to be set ON/OFF. The maximum number allowed is 32.

Value
Specifies the value used to set the desired ON/OFF signal states. The value is transformed into binary notation and each bit of the binary value sets the signal state. The least significant bit corresponds to the signal with the smallest signal number, and so on. If the binary notation of this value has more bits than the number of signals, only the state of the given number of signals (starting from the specified starting signal number) is set and the remaining bits are ignored.

If this parameter is omitted, the current signal states are displayed.

Explanation

Sets (or resets) the signal state of one or more external output signals or internal signals according to the given value.

Acceptable Signal Numbers

External output signal	1 – actual number of signals
Internal signal	2001–2960

Specifying a signal number greater than the number of signals actually installed results in error. Selecting a dedicated signal also results in error.

Example

>BITS32 1,32=^H7FFFFFFF ↵	External output signals 1–32 are set to correspond to binary notation of 7FFFFFFF. External output signals 1–31 turn ON.
>BITS32 2001,32 ↵	Displays the numerical values represented by internal signals 2001 to 2032 (32 bits).
BITS32 2001,32 = 2147483647	When internal signals 2001 to 2031 are ON and 2032 is OFF, displays the decimal display value of hexadecimal 7 FFFFFFF.

SCNT counter signal number = count up signal, count down signal, counter clear signal, counter value

Function

Outputs counter signal when the specified counter value is reached.

Parameter

Counter signal number

Specifies the signal number to output. Setting range for counter signal numbers: 3097 to 3128.

Count up signal

Specified by signal number or logical expressions. Each time this signal changes from OFF to ON, the counter counts up by 1.

Count down signal

Specified by signal number or logical expressions. Each time this signal changes from OFF to ON, the counter counts down by 1.

Counter clear signal
Specified by signal number or logical expressions. If this signal is turned ON, the internal counter is reset to 0.

Counter value
When the internal counter reaches this value, the specified signal is output. If “0”is given, the counter signal is turned OFF.

Explanation

If the count up signal changes from OFF to ON when the SCNT command is executed, then the internal counter value increases by 1. If the countdown signal changes from OFF to ON, the internal counter value decreases by 1. When the internal counter value reaches the value specified in the parameter (counter value), the counter signal is output. If the counter clear signal is output, value of the internal counter is set at 0. Each counter signal has its own individual counter value. To force reset of the internal counter to 0, use SCNTRESET command.

To check the states of signals 3001 to 3128, use the IO/E command. (Option)

SCNTRESET counter signal number

Function

Resets the internal counter value of the specified counter signal number to 0.

Parameter

Counter signal number
Select the number of the counter signal to reset. Setting range for counter signal numbers: 3097 to 3128.

SFLK signal number = time

Function

Turns ON/OFF (flickers) the specified signal in specified time cycle.

Parameter

Signal number

Specifies the number of signal to flicker. Setting range: 3065 to 3096.

Time

Specifies the time to cycle ON/OFF (real values). If a negative value is set, flickering is canceled.

Explanation

The process of ON/ OFF is considered one cycle, and the cycle is executed in the specified time.

SFLP output signal = set signal expression, reset signal expression

Function

Turns ON/OFF an output signal using a set signal and a reset signal.

Parameter

Output signal

Specifies the signal number of the signal to output. A positive number turns ON the signal; a negative number turns OFF. Only the signal numbers for output signals can be specified (from 1 to actual number of signals).

Set signal expression

Specifies the signal number or logical expression to set the output signal.

Reset signal expression

Specifies the signal number or logical expression to reset the output signal.

Explanation

If the set signal is ON, the output signal is turned ON. If the reset signal is ON, the output signal is turned OFF. The output signal is turned ON or OFF when the SFLP command is executed, and not when the set signal or the reset signal is turned ON.

SOUT signal number = signal expression

Function

Outputs the specified signal when the specified condition is set.

Parameter

Signal number

Specifies the signal number of the signal to output. Only the signal numbers for output signals can be specified. (1 to actual number of signals).

Signal expression

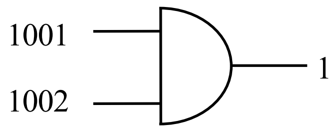
Specifies a signal number or a logical expression.

Explanation

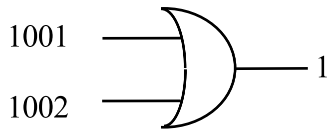
This command is for logical calculation of signals. Logical expressions such as AND and OR are used. The specified signal is output when that condition is set.

Example

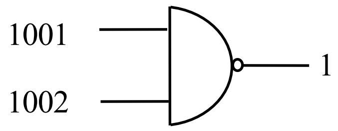
SOUT 1 = 1001 AND 1002



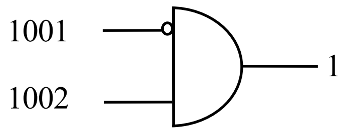
SOUT 1 = 1001 OR 1002



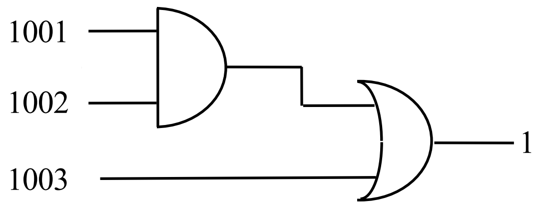
SOUT -1 = 1001 AND 1002
SOUT 1 = NOT(1001 AND 1002)



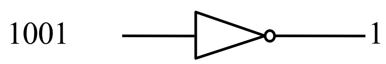
SOUT 1 = -1001 AND 1002



SOUT 1 = (1001 AND 1002) OR 1003



SOUT -1 = 1001 or SOUT 1= -1001 or
SOUT 1 = NOT(1001)



STIM timer signal = input signal number, time

Function

Turns ON the timer signal if the specified input signal is ON for the given time.

Parameter

Timer signal
Selects the signal to turn ON. Acceptable signal numbers are from 3001 to 3064.

Input signal number
Specifies in integers the input signal number or logical expression to monitor as a condition for turning ON the timer signal. The value cannot exceed the number of signals actually installed.

Time
Specifies in real numbers the time (sec) the input signal must be ON before turning ON the timer signal.

Explanation

The monitored input signal has to be ON continuously in order for the timer signal to be turned ON. If the input signal turns OFF before the given time passes, the time count restarts when that signal turns ON again. If the input signal turns OFF, the timer signal turns OFF immediately. However, the input signal affects the timer signal only when STIM is executed. Unless STIM is executed, the timer signal remains ON even when the input signal turns OFF.

To check the state of signals 3001 to 3064, use the IO/E command.

Example

STIM 3001 = 1,5 ↵	sig2 turns ON if sig1 is ON for 5 seconds.
SOUT 2 = 3001 ↵	
>PCEXECUTE ↵	

SETPICK time1, time2, time3, time4, …, time8
SETPLACE time1, time2, time3, time4, …, time8

Function

Sets the time to start clamp close control (SETPICK) or clamp open control (SETPLACE) for each of the 8 clamps.

Parameter

Time 1 to 8

Sets the control time to open/close clamps 1 to 8 in seconds. Setting range: 0.0 to 10.0 seconds.

Explanation

See CLAMP instruction for more details.

HSENSESET no. = input signal number, output signal number, signal output delay time

Option

Function

Declares the starting of signal detection to AS system. When this instruction is executed, AS system starts to watch the sensor signal and accumulates the data such as pose, etc., into the buffer memory at signal transaction. The data saved in the buffer memory can be read using HSENSE instruction. Buffer memory can save up to 20 data.

Parameter

No.

Specifies the number for the monitoring results. Up to 2 input signals can be monitored. Command for each signal is written as HSENSESET 1 or HSENSESET 2. Acceptable range is 1 or 2.

Input signal number

Set the signal number to monitor. Setting zero (0) terminates the monitoring.

Output signal number

Set the number of the signal to be output after system acquires the joint angle. The specified signal turns ON for 0.2 seconds. This may be omitted.

Signal output delay time

Set the time to delay the output of signal after acquiring the pose data. Acceptable range is 0 to 9999 ms. This may be omitted.

[NOTE]

Even when the control power becomes OFF during watching, buffer memory keeps the read data. It is possible to read the kept data by HSENSE instruction after turning ON the control power again. However, watching does not restart automatically, so HSENSESET should be executed again

Example

>HSENSESET 1 = wx_sensor	Starts watching for input signal wx_sensor.

HSENSE no. result variable, signal status variable, pose variable, error variable, memory usage variable

Option

Function

Reads the data saved in the buffer memory by HSENSESET instruction.

Parameter

No.

Sets the monitoring number. To read data saved by HSENSESET 1, specify HSENSE1. To read data saved by HSENSESET 2, specify HSENSE 2.

Result variable

Specifies the name of the real variable to which the watch result is assigned. After executing HSENSE instruction, numerical value is assigned to this variable. Zero (0) is assigned to this variable when AS system does not detect the signal transaction. –1 is assigned to this variable when AS system detects the signal transaction.

Signal status variable

Specifies the name of the real variable to which the status of signal transaction is assigned. After executing HSENSE instruction, a numerical value is assigned to this variable. When the signal(s) is turned from OFF to ON, ON (-1) is assigned. When the signal(s) is turned from ON to OFF, OFF (0) is assigned to this variable.

Pose variable

Specifies the name of the pose variable to which the joint values at time of HSENSE signal input are assigned.

Error variable

Specifies the name of the real variable to which the buffer overflow error result is assigned. When no error occurs, 0 is assigned to this variable. When buffer memory overflows, a numerical value (other than 0) is assigned to this variable. The buffer memory overflows after accumulating data from more than 20 transactions.

Memory usage variable

Specifies the name of the real variable to which the number of used memory in the buffer is assigned. The value assigned to this variable shows the number of memory in the buffer that is already used. When only one memory is used, 0 will be assigned to the variable. When all the memories are used, the value of the variable will be 19.

RSIGRANGE

Function

Configures the display and settings of signal range to be used by RSIGPOINT instruction.

Explanation

Sets signal range by segments of every 32 points.

Specified value

Setting range: 1 to 30 (see below signal number correspondence table)

Specified value	Output signals	Signal range
1	External output signals	1 to 32
2	External output signals	33 to 64
:	:	:
29	External output signals	897 to 928
30	External output signals	929 to 960

After this command is entered, the current set value and message "Change?" are displayed. When using the RSIGPOINT instruction to change the signal range to be used, input the new value. Enter the ↵ key only for no changes.

Example

>RSIGRANGE	
[Now: 1]	The current set value is displayed.
Change? (If not, Press RETURN only.)	
2↵	Specifies 2 when using external output signals of 33 to 64 as shutter signals.

5.8 Message Display Commands

PRINT	Displays data.
TYPE	Displays data.
IFPWPRINT	Displays specified character string in a display window.
IFPWOVERWRITE	Displays by overwriting character string in a display window.
IFPLABEL	Sets the labels for the icons on interface panel.
IFPTITLE	Sets the title for the specified page of the interface panel.
IFPDISP	Displays the specified page of the interface panel.
SETOUTDA	Sets analog output environment. (Option)
OUTDA	Outputs voltage at set condition. (Option)

PRINT device number: print data, …….
TYPE device number: print data, …….

Function

Displays on the terminal the print data specified in the parameter.

Parameter

Device number

Select the device for displaying the data:

0: All terminals that are connected

1: Personal computer

2: Teach pendant

3: - 5: Terminals connected via Ethernet

If not specified, the data is displayed on the currently selected device.

Print data

Select one or more from below. Separate the data with commas when specifying more than one.

(1) character string	e.g.	"count ="
(2) real value expressions (the value is calculated and displayed)	e.g.	count
(3) Format information (controls the format of the output message)	e.g.	/D, /S

A blank line is displayed if no parameter is specified.

Explanation

If "2" is entered for device number, the teach pendant screen changes automatically to keyboard screen.

The following codes are used to specify the output format of numeric expressions. The same format is used until a different code is specified. In any format, if the value is too large to be displayed in the given width, asterisks (*) will fill the space. In this case, change the number of characters that can be displayed. The maximum number of characters displayed in one line is 128. To display more than 128 characters in a line, use the /S code explained on the following page.

[NOTE]

If the MESSAGES switch is OFF, no message appears on the terminal screen.

Format Specification Codes

/D Uses the default format. This is the same as specifying the format as /G15.8 except that zeros following numeric values and all spaces but one between numeric values are removed.

/Em.n Displays the numeric values in scientific notation (e.g. -1.234E+02). "m" describes the total number of characters shown on the terminal and "n" the number of decimal places. "m" should be greater than n by five or more.

/Fm.n Displays the numeric values in fixed point notation (e.g. -1.234). "m" describes the total number of characters shown on the terminal and "n" the number of digits in the fraction part.

/Gm.n If the value is greater than 0.01 and can be displayed in Fm.n format within m digits, the value is displayed in that format. Otherwise, the value is displayed in Gm.n format.

/Hn Displays the values as a hexadecimal number in the n digit field.

/In Displays the values as a decimal number in the n digit field.

The following parameters are used to insert certain characters between character strings.

/Cn Inserts line feed n times in the place where this code is entered, either in front or after the print data. If this code is placed within print data, n-1blank lines are inserted.

/S The line is not fed.

/Xn Inserts n spaces.

/Jn Displays the value as a hexadecimal number in the n digit field. Zeros are displayed in place of blanks. (Option)

/Kn Displays the value as a decimal number in the n digit field. Zeros are displayed in place of blanks. (Option)

/L This is the same as /D except that all the spaces are removed with this code. (Option)

Example

In this example the value of real variable “i” is 5, the fifth element of array variable “point” is 12.66666.

```
>PRINT  "point", i, "=" , /F5.2, point [i] ↵
```

point 5 = 12.67	The display should look like this.
point 5 = *****	If the value of point[5] is 1000 (1000.00), the display should look like this. The value is too large to display (i.e. the number of digits is greater than 5).

In the following example code /S is used to display the data without changing the lines after the data.

```
>PRINT “ABC”
>PRINT/S, “DEF”
>PRINT “GHI” ↵
```

```
| ABC
| DEFGHI            |
```

The display should look like this, with “GHI” displayed on the same line as “DEF”.

IFPWPRINT window number, row, column, background color, label color = "character string", "character string", ……

Function

Displays the specified character string in the string window set by Auxiliary Function 0509 (Interface panel screen).

Parameter

Window number

Corresponds to the window number specified in Auxiliary Function 0509 as the window specification used to display the string. Select from 1to 8 (standard).

Row

Specifies the row in the window for displaying the string. Acceptable number is from 1 to 4, though it depends on the window size. If not specified, 1 is assumed.

Column

Specifies the column in the window for displaying the string. Acceptable number is from 1 to 70, though it depends on the window size. If not specified, 1 is assumed.

Background color

Selects the color of the background of the selected window. Acceptable numbers are from 0 to 15. If not specified, the background is white.

No.	Color	No.	Color	No.	Color	No.	Color
0	Grey	4	Green	8	Pink	12	Navy
1	Blue	5	Pale Blue	9	White	13	Reddish Brown
2	Red	6	Yellow	10	Black	14	Deep Green
3	Orange	7	White	11	Cyan	15	Lavender

Label color

Selects the color of the characters displayed. Acceptable numbers are from 0 to 15 (See chart above). If not specified, the characters are displayed in black.

Character string

Specifies the character string to display. All strings after the first string are displayed on the next row starting at specified column. Execution of IFWPRINT clears all items, except the specified character strings, from the specified window.

Explanation

IFPWPRINT command can be used only when the interface panel is available for use. If the parameters are not specified, the last setting of that particular window is selected (for first time use, the above default values are set). If the character string does not fit in one row, its display overflows to the next line (indenting to the selected column). Strings that extend beyond the size of the window are not displayed. Control characters in the string are displayed as blanks.

IFPWOVERWRITE mode window number, row, column, background color, label color = “character string”, “character string”, ……

Function

Displays by overwriting the specified character string in the string window set by Auxiliary Function 0509 (Interface panel screen).

Parameter

Mode

(None) Overwrites the existing character string in unit of line.

/CUT Displays the character string by truncating the characters that do not fit in one line of the string window, without starting a new line. However, if the target window number is not allocated for the interface panel, the character string is saved to the full extent of the window and is displayed when allocated.

/CHAR Overwrites the existing character string in unit of character.
For two-byte characters, as a result of truncation or overwriting, are displayed within the correctly-displayable range.

Window number

Corresponds to the window number specified in Auxiliary Function 0509 as the window specification used to display the string. Select from 1to 8 (standard).

Row

Specifies the row in the window for displaying the string. Acceptable number is from 1 to 4, though it depends on the window size. If not specified, 1 is assumed.

Column

Specifies the column in the window for displaying the string. Acceptable number is from 1 to 78, though it depends on the window size. If not specified, 1 is assumed.

Background color

Selects the color of the background of the selected window. Acceptable numbers are from 0 to 15. If not specified, the background is white.

No.	Color	No.	Color	No.	Color	No.	Color
0	Grey	4	Green	8	Pink	12	Navy
1	Blue	5	Pale Blue	9	White	13	Reddish Brown
2	Red	6	Yellow	10	Black	14	Deep Green
3	Orange	7	White	11	Cyan	15	Lavender

Label color
Selects the color of the characters displayed. Acceptable numbers are from 0 to 15 (See chart above). If not specified, the characters are displayed in black.

Character string
Specifies the character string to display. All strings after the first string are displayed on the next row starting at specified column.

Explanation

IFPWOVERWRITE command can be used only when the interface panel is available for use. If the parameters are not specified, the last setting of that particular window is selected (for first time use, the above default values are set). If the character string does not fit in one row, its display overflows to the next line (indenting to the selected column). Strings that extend beyond the size of the window are not displayed. Control characters in the string are displayed as blanks. Unlike IFPWPRINT command/ instruction, the rows other than the row specified for display are displayed unchanged as before executing IFPWOVERWRITE.

Example

The figures below show the screens displayed when executing IFPWPRINT and IFPWOVERWRITE command/ instruction from the screen showing “This is a pen”. The left figure is the figure after executing ”IFPWPRINT 1,3,1,7,10=”my”. The characters on lines 1, 2, and 4 disappear and line 3 shows ”my”. On the other hand, the right figure shows the screen after executing “IFPWOVERWRITE 1,3,1,7,10=”my”. The characters on lines 1, 2, and 4 are displayed as they were before executing the instruction and only line 3 has changed to ”my”.

Displays
"This is a pen"

Screen after executing
IFPWPRINT 1,3,1,7,10 ="my"

Screen after executing
IFPWOVERWRITE 1,3,1,7,10 ="my"

IFPLABEL position, "label 1", "label 2", "label 3", "label 4"

Function

Sets and modifies the label of the icon at the specified position on the interface panel.

Parameter

Position

Specifies the display position on the interface panel of the icon to set/ modify the label.
Setting range: 1 – 112.

"Label 1", "Label 2"…

Specifies the character string to display on the interface panel as the label of the specified icon. Omitted label will not be changed.

Explanation

Sets and modifies the label for the icons displayed on the interface panel. When a position with no icon set or when an icon with no label is specified, nothing occurs.

Example

```
>IFPLABEL 10, , ,"label 3"
```

Icon at position 10 on the interface panel is changed to "label 3" (label 1, label 2 is not changed because the parameters are omitted.)

IFPTITLE page no., "title"

Function

Sets and modifies the title for the specified page of the interface panel.

Parameter

Page no.

Specifies the page of the interface panel to change the title. Setting range: 1- 4.

“Title”

Specifies the character string to display on the page as the title. Default setting is “Interface Panel”. When NULL string (“”) is specified, this default setting is also displayed.

Explanation

Sets and modifies the title for the specified page of the interface panel.

Example

>IFPTITLE 1, "Page 1"	The title of the first page of the interface panel is changed to “Page 1”.
>IFPTITLE 2, ""	The title of the second page of the interface panel is changed to “Interface Panel”.

IFPDISP Page no.

Function

Displays the specified page of the interface panel.

Parameter

Page no.

Specifies the page number of the interface panel to be displayed.

Explanation

Executing this command allows the display of the specified page of the interface panel. To switch pages of the interface panel, choose <Prev Page> or <Next Page> in the teach pendant's pull-down menu.

Example

>IFPDISP 2	Displays the second page of the interface panel.

SETOUTDA channel No. = LSB, No. of bits, logic, max. voltage, min. voltage

Option

Function

Specifies the analog output environment including: channel number and LSB, number of bits and logic voltage for signal output, maximum and minimum voltage.

Parameter

Channel No.

Sets the analog output channel number. (Setting range: integers between 1 and 16; first 1TW/1UR board: 1 to 4; second 1TW/1UR board: 5 to 8; third 1TW/1UR board: 9 to 12; fourth 1TW/1UR board: 13 to 16)

LSB

Specifies the first analog output signal number for D/A conversion as an integer. Setting range: OUT1 to OUT125, 2001 to 2125, 3000 (first channel of 1TW/1UR board), 3001 (second channel of 1TW/1UR board), and subsequent 1TW/1UR board analog output channels can be specified. Default value is 3000. Previous setting remains in effect if not specified.

Number of bits

Sets the number of bits of analog output signals for D/A conversion as an integer. Setting range: 4 to 16 bits. Sets to 12 bits when 3000 onward [3000 + analog output channel number on 1TW/1UR board] is chosen for parameter LSB above. Default value is 8 bits. Previous setting remains in effect if not specified.

Logic

Sets the logic to either positive (1) or negative (0). Default value is 0 (negative). Previous setting remains in effect if not specified. Please set logic to positive for 1TW/1UR board.

Maximum voltage

Sets the maximum voltage of hardware (D/A output). Setting range: -15.0 to +15.0 V. Unit: V. Default value is 10 V. The value should be rounded off to the first decimal place. Previous setting remains in effect if not specified.

Minimum voltage

Sets the minimum voltage of hardware (D/A output). Setting range: -15.0 to +15.0 V. Unit: V. Default value is 0 V. The value should be rounded off to the first decimal place. Previous setting remains in effect if not specified.

See also 6.8 SETOUTDA, OUTDA instructions.

[NOTE]

1. Actual voltage output depends on the hardware used.
2. Error occurs if the value for maximum voltage is set lower than the minimum voltage.

OUTDA voltage, channel number

Option

Function

Outputs the voltage at set conditions from the specified analog output channel.

Parameter

Voltage
Sets the analog output voltage. Setting range: -15.0 to +15.0 V. Unit: V. The value should be rounded off to the first decimal place.

Channel number
Specifies the analog output channel number. Setting range: integers between 1 and 16. If not specified, 1 is assumed.

[NOTE]

Confirm that command voltage and actual output voltage are the same by setting output environment to correspond with the hardware settings via SETOUTDA instruction (command).

6 Program Instructions

This chapter groups the program instructions in the following categories, and describes each instruction in detail. A program instruction consists of a keyword expressing the instruction and parameter(s) following that key word, as shown in the example below.

Example

Keyword Parameter

JMOVE pose variable, clamp number

Parameters marked with [highlighted] can be omitted.

Always enter a space between the keyword and the parameter.

↵ in the examples represent the Enter key.

6.1 Motion Instructions

JMOVE	Moves robot in joint interpolated motion.
LMOVE	Moves robot in linear interpolated motion.
DELAY	Stops robot motion for specified time.
STABLE	Stops robot motion for specified time after the axes coincide.
JAPPRO	Approach the destination in joint interpolated motion.
LAPPRO	Approach the destination in linear interpolated motion.
JDEPART	Leaves the current pose in joint interpolated motion.
LDEPART	Leaves the current pose in linear interpolated motion.
HOME	Moves to the home pose.
DRIVE	Moves in the direction of a single axis.
DRAW	Moves the specified amount in the direction of the X, Y, Z, axis of the base coordinates.
TDRAW	Moves the specified amount in the direction of the X, Y, Z, axis of the tool coordinates.
ALIGN	Aligns the tool Z axis with the base coordinate axis.
HMOVE	Moves in linear interpolated motion (the wrist joint moves in joint interpolated motion.)
XMOVE	Moves in linear movement to the specified pose.
C1MOVE	Moves in circular interpolated motion. (Option)
C2MOVE	Moves in circular interpolated motion. (Option)

JMOVE pose variable, clamp number
LMOVE pose variable, clamp number

Function

Moves the robot to the specified pose.
JMOVE: Moves in joint interpolated motion.
LMOVE: Moves in linear interpolated motion.

Parameter

Pose variable
Specifies the destination pose of the robot. (Can be in transformation values, compound transformation values, joint displacement values or pose information functions.)

Clamp number
Specifies the clamp number to open or close at the destination pose. Positive number closes the clamp, and negative number opens it. Any clamp number can be set, up to the maximum number set via HSETCLAMP command (or auxiliary function 0605). If omitted, the clamp does not open or close.

Explanation

The robot moves in joint interpolated motion when JMOVE instruction is executed. The robot moves so that the ratios of distance traveled to the total distance are equal at all joints throughout the movement from the starting pose to the end pose.

The robot moves in linear interpolated motion when LMOVE instruction is executed. The origin of the tool coordinates (TCP) moves along a linear trajectory.

Example

JMOVE #pick	Moves to pose described by joint displacement values “#pick” in joint interpolated motion.
LMOVE ref+place	Moves to the pose described by the compound transformation values “ref + place” in linear interpolated motion.
LMOVE #pick,1	Moves to the pose described by joint displacement values “#pick” in linear interpolated motion. Upon reaching the pose, clamp 1 is closed.

DELAY time

Function

Stops the robot motion for the specified time.

Parameter

Time

Specifies in seconds for how long the robot motion is stopped.

Explanation

In AS system, DELAY instruction is considered as a motion instruction that "moves to nowhere".

Even if the robot motion is stopped by DELAY instruction, all the program steps before the next motion instruction are executed before stopping.

Example

DELAY 2.5	Stops the robot motion for 2.5 seconds.

STABLE time

Function

Postpones execution of next motion instruction until the specified time elapses after the axes coincide. (Waits until the robot is stable.)

Parameter

Time

Specifies in seconds for how long the robot motion is kept stable.

Explanation

If coincidence of the axes fails while the robot is stopped by this command, the time is counted from when the axes coincide again.

JAPPRO pose variable, distance
LAPPRO pose variable, distance

Function

Moves in tool Z direction to a specified distance from the taught pose.

JAPPRO: Moves in joint interpolated motion.

LAPPRO: Moves in linear interpolated motion.

Parameter

Pose variable

Specifies the end pose (in transformation values or joint displacement values)

Distance

Specifies the offset distance between the end pose and the pose the robot actually reaches on the Z axis direction of the tool coordinates (in millimeters). If the specified distance is a positive value, the robot moves towards the negative direction of the Z axis. If the specified distance is a negative value, the robot moves towards the positive direction of the Z axis.

Explanation

In these commands, tool orientation is set at the orientation of the specified pose, and the position is set at the specified distance away from the specified pose in the direction of the Z axis of the tool coordinates.

Example

JAPPRO place,100	Moves in joint interpolated motion to a pose 100 mm away from the pose “place” in the direction of the Z axis of the tool coordinates. Pose “place” is described in transformation values.
LAPPRO place, offset	Moves in linear interpolated motion to a pose away from the pose “place”, described in transformation values, at the distance defined by the variable “offset” in the direction of the Z axis of the tool coordinates.

JDEPART distance
LDEPART distance

Function

Moves the robot to a pose at a specified distance away from the current pose along the Z axis of the tool coordinates.

JDEPART : Moves in joint interpolated motions.

LDEPART : Moves in linear interpolated motions.

Parameter

Distance

Specifies the distance in millimeters between the current pose and the destination pose along the Z axis of the tool coordinates. If the specified distance is a positive value, the robot moves “back” or towards the negative direction of the Z axis. If the specified distance is a negative value, the robot moves “forward” or towards the positive direction of the Z axis.

Example

JDEPART 80	The robot tool moves back 80 mm in −Z direction of the tool coordinates in joint interpolated motion.
LDEPART 2∗offset	The robot tool moves back 2∗offset (200 mm if offset = 100) in −Z direction of the tool coordinates in linear interpolated motion.

HOME home pose number

Function

Moves in joint interpolated motion to pose defined as HOME or HOME2.

Parameter

Home pose number

Specifies the home pose number (1 or 2). If omitted, HOME 1 is selected.

Explanation

Two home poses can be set (HOME 1 and HOME 2). This instruction moves the robot to one of the home poses in joint interpolated motion. The home pose should be defined beforehand using the SETHOME or SET2HOME command/ instruction. If the home pose is not defined, the null origin (all joints at 0°) is assumed as the home pose.

Example

HOME	Moves to the home pose defined by SETHOME command/ instruction in joint interpolated motion.
HOME 2	Moves to the home pose defined by SET2HOME command/ instruction in joint interpolated motion.

DRIVE joint number, displacement, speed

Function

Moves a single joint of the robot.

Parameter

Joint number

Specifies the joint number to move. (In a six-joint robot, the joints are numbered 1 to 6, starting from the joint furthest from the tool mounting flange.)

Displacement

Specifies the amount to move the joint, as either a positive or negative value.

The unit for this value is the same as the value that describes the pose of the joint; i.e. if the joint is a rotational joint, the value is expressed in degrees (°), and if the joint is a slide joint, the value is expressed in distance (mm).

Speed

Specifies the speed for this motion. As in regular program speed, it is expressed as a percentage of the monitor speed. If not specified, 100 % of the monitor speed is assumed.

Explanation

This instruction moves only one specified joint.

The motion speed for this instruction is combination of the speed specified in this instruction and the monitor speed. The program speed set in the program does not affect this instruction.

Example

DRIVE 2,-10,75	Moves joint 2 (JT2) –10° from the current pose. The speed is 75 % of the monitor speed.

**DRAW X translation, Y translation, Z translation
X rotation, Y rotation, Z rotation, speed**

**TDRAW X translation, Y translation, Z translation
X rotation, Y rotation, Z rotation, speed**

Function

Moves the robot in linear movement from the current pose and at the specified speed, the distance specified in the direction of the X, Y, Z axes and rotates the specified amount around each axis. DRAW instruction moves the robot based on the base coordinates, TDRAW instruction moves the robot based on the tool coordinates.

Parameter

X translation

Specifies the amount to move on the X axis in mm. If not specified, 0 mm is entered.

Y translation

Specifies the amount to move on the Y axis in mm. If not specified, 0 mm is entered.

Z translation

Specifies the amount to move on the Z axis in mm. If not specified, 0 mm is entered.

X rotation

Specifies the amount to rotate around the X axis in deg. Acceptable range is less than ±180°. If not specified, 0 deg is entered.

Y rotation

Specifies the amount to rotate around the Y axis in deg. Acceptable range is less than ±180°. If not specified, 0 deg is entered.

Z rotation

Specifies the amount to rotate around the Z axis in deg. Acceptable range is less than ±180°. If not specified, 0 deg is entered.

Speed

Specifies the speed in %, mm/s, mm/min, cm/min, or s. If not specified, the robot moves at the program speed.

Explanation

The robot moves from the current pose to the specified pose in linear movement.

Example

DRAW 50,,-30	Moves from the current pose in linear motion 50 mm in the direction of the X axis and –30 mm in the direction of the Z axis of the base coordinates.

ALIGN

Function

Moves the Z axis of the tool coordinates to be parallel with the closest axis of the base coordinates.

Explanation

In each application, if the reference motion direction is set along the tool Z direction, DO ALIGN enables easy alignment of the tool direction to the base coordinates before teaching the pose data.

HMOVE pose variable, clamp number

Function

Moves the robot to the specified pose. The robot moves in hybrid motion: major axes in linear interpolation, and the wrist joints in joint interpolation.

Parameter

Pose variable

Specifies the destination of the robot motion. (Can be in transformation values, compound transformation values, joint displacement values or pose information functions.)

Clamp number

Specifies the clamp number to open or close at the destination pose. Positive number closes the clamp, and negative number opens it. Any clamp number can be set up to the maximum number set via HSETCLAMP command (or the auxiliary function 0605). If omitted, the clamp does not open or close.

Explanation

This instruction moves the robot in linear interpolated motion. The origin of the tool coordinates draws a linear trajectory. However, the wrist joints move in joint interpolation. This instruction is used when the robot is to be moved in linear motion but the angles of the wrist joints change greatly between the beginning and end of the motion.

XMOVE mode pose variable TILL signal number

Function

Moves the robot towards the specified pose in linear movement, stops motion when the specified signal condition is set even if the pose has not been reached, and skips to the next step.

Parameter

Mode

(Not specified)

Monitors for the rising or trailing edge of the specified input signal. Positive signal number monitors rising edge, and negative number monitors trailing edge.

/ERR (Option)

Returns an error if the signal condition is already set when the monitoring starts.

/LVL (Option)

Immediately skips to the next step if the signal condition is already set when the monitoring starts.

Pose variable

Specifies the destination pose of the robot motion. (Can be in transformation values, compound transformation values, joint displacement values or pose information functions.)

Signal number

Specifies the number of external input signal or internal signal.

Acceptable signal numbers

External input signal	1001 to actual number of installed signals or 1256 (the smaller of the two).
Internal signal	2001 to 2960

[NOTE]

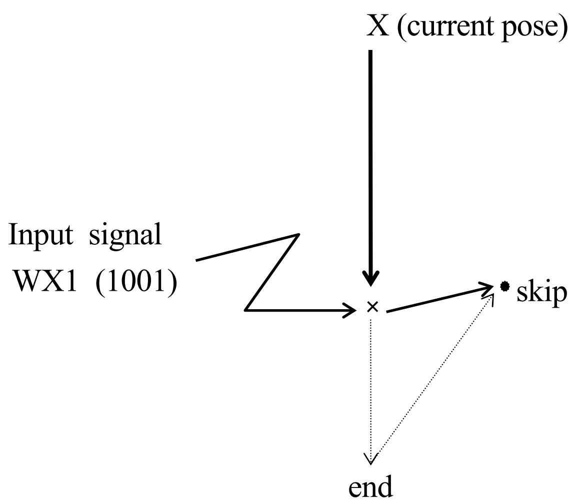
When monitoring the rising and trailing edge of the signal, the program branches only when there is a change in the signal status. Therefore, if the rising edge of signal is monitored and the signal is ON at the time XMOVE is executed, the program will not be interrupted until that signal turns OFF and then ON again.

The input signal should be stable for at least 50 msec for accurate monitoring.

Example

```
XMOVE   end   TILL   1001
LMOVE   skip
```

Moves from the current pose to pose “end” in linear motion. As soon as the input signal 1001 is turned ON, the program execution skips to the next step (LMOVE skip) even if the robot has not reached “end”.



C1MOVE　pose variable, clamp number
C2MOVE　pose variable, clamp number

Option

Function

Moves the robot to the specified pose following a circular path.

Parameter

Pose variable

Specifies the destination of the robot motion.　(Can be in transformation values, compound transformation values, joint displacement values or pose information functions.)

Clamp number

Specifies the clamp number to open or close at the destination pose.　Any clamp number can be set, up to the maximum number set via HSETCLAMP command.　Positive number closes the clamp, and negative number opens it.　If omitted, the clamp does not open or close.

Explanation

C1MOVE instruction moves to a point midway on the circular trajectory, C2MOVE instruction moves to the end of the trajectory.

To move the robot in a circular interpolated motion, three poses must be taught.　The three poses differ in C1MOVE and C2MOVE instructions.

C1MOVE:
1. Pose of the latest motion instruction.
2. Pose to be used as the parameter of C1MOVE instruction.
3. Pose of the next motion instruction. (C1MOVE or C2MOVE instruction)

C2MOVE:
1. Pose of the latest C1MOVE instruction.
2. Pose of the motion instruction before C1MOVE instruction.
3. Pose of C2MOVE instruction.

[NOTE]

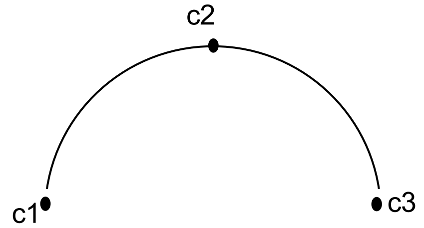
The following motion instructions are needed before the C1MOVE instruction:
ALIGN, C1MOVE, C2MOVE, DELAY, DRAW, TDRAW, DRIVE, HOME, JMOVE, JAPPRO, JDEPART, LMOVE, LAPPRO, LDEPART, STABLE, XMOVE

C1MOVE instruction must be followed by C1MOVE or C2MOVE instruction.

C1MOVE instruction must precede a C2MOVE instruction.

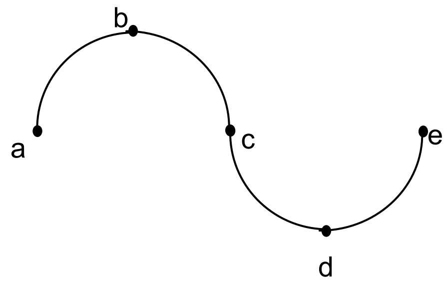
Example

```
JMOVE    c1
C1MOVE c2
C2MOVE c3
```

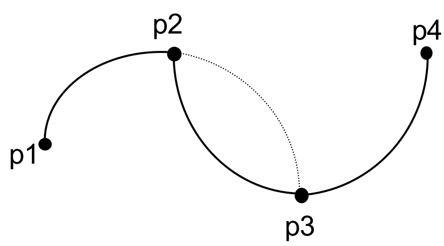


The robot moves in joint interpolated motion to c1 and then moves in a circular interpolated motion following the arc created by c1, c2, c3.

```
JMOVE    #a      ]
C1MOVE #b        ]  arc a,b,c
C2MOVE #c        ]
C1MOVE #d        ]  arc c,d,e
C2MOVE #e        ]
```




```
LMOVE    #p1     ]  arc p1,p2,p3
C1MOVE #p2       ]
C1MOVE #p3       ]  arc p2,p3,p4
C2MOVE #p4       ]
```



6.2 Speed and Accuracy Control Instructions

SPEED	Sets the motion speed (program speed).
MON_SPEED	Sets the monitor speed.
ACCURACY	Sets the accuracy range.
ACCEL	Sets the acceleration.
DECEL	Sets the deceleration.
BREAK	Holds execution of the next step until the current motion is completed.
BRAKE	Stops the current motion and skips to the next step.
BSPEED	Sets the block speed. (Option)
REFFLTSET	Specifies the moving average for robot command values.
REFFLTRESET	Resets the robot's moving average span.
FFSET	Sets the speed/ acceleration feed forward gain.
FFRESET	Resets the robot speed/ acceleration feed forward gain.

SPEED speed, rotational speed, ALWAYS

Function

Specifies the robot motion speed.

Parameter

Speed

Specifies the program speed. Usually it is specified in percentages between 0.01 and 100 (%). Absolute speed can be set by specifying the speeds with these units: MM/S and MM/MIN. The unit S (seconds) specifies the motion time. The input range for motion time is 0.1 S to 3601 S. If the unit is omitted, it is read as percent (%).

Rotational speed (Option)

Specifies the rotational speed of the tool orientation in linear and circular interpolated motions. Usually it is specified in percentages between 0.01 and 100 (%). Absolute speed can be set by specifying the speed with these units: DEG/S and DEG/MIN. If the unit is omitted, it is read as percent (%). If this parameter is omitted, the rotational speed is set at 100%.

ALWAYS

If this parameter is entered, the speed set in this instruction remains valid until the next SPEED instruction is executed. If not entered, the speed is effective only for the next motion instruction.

Explanation

The actual speed of the robot motion is determined by the product of the speed specified by this instruction and the monitor speed specified by the SPEED command or MON_SPEED instruction (Monitor speed × Program speed). However, full speed is not guaranteed in cases such as below:

1. when the distance between the two taught poses is too short,
2. when a linear motion exceeding the maximum speed of axis rotation is taught.

The motion speed is determined differently in joint interpolated motion and linear movement. In joint interpolated motion, the motion speed is determined as a percentage of the maximum speed of each axis. In linear movement, the motion speed is determined as a percentage of the maximum speed at the origin of the tool coordinates.

When the speed is specified in distance per unit time or in seconds, the speed in linear movement at the origin of the tool coordinates is set. When moving in joint interpolated motions, set the speed in percent. (Even if the speed is set in absolute speed or in motion time, the robot will not move in the set speed. Instead, the speed is processed as a percentage of the given value to the maximum speed.)

The absolute speed expressed in values with MM/ S and MM/MIN, and time specified speed expressed in values with S, describe the speed when the monitor speed is 100%. If the monitor speed is decreased, these speeds decrease in the same proportion.

[NOTE]

Even if the product of program speed and the speed set by SPEED command or MON_SPEED instruction (monitor speed) exceeds 100% the actual motion speed does not exceed 100%.

The rotational speed cannot be set without the rotational speed control option ON. If the option is not ON, error occurs.

Example

The speed is set as follows when the monitor speed is 100%:

SPEED 50	Sets the speed of the next motion to 50% of the maximum speed.
SPEED 100	Sets the speed of the next motion to 100% of the maximum speed.
SPEED 200	Sets the speed of the next motion to 100% of the maximum speed (speed over 100% is considered 100%).
SPEED 20MM/S ALWAYS	The speed of the origin of the tool coordinate (TCP) is set at 20mm/sec until it is changed by another SPEED instruction (when the monitor speed is 100%).
SPEED 6000 MM/MIN	Sets the speed of the next robot motion to 6000 mm/min. (The speed of linear motion of the origin of the tool coordinates).
SPEED 5 S	Sets the speed of the next robot motion so that the destination is reached in 5 seconds. (The speed of linear motion of the origin of the tool coordinates).

MON_SPEED monitor speed

Function

Sets or changes the monitor speed.

Parameter

Monitor speed

Specifies the speed to be set or changed as percentage of the maximum speed (unit in %). It is the normal maximum speed if this value is 100, and 1/2 of the maximum speed if it is 50. Percentage of the range up to 99999 can be specified, if the speed limit release option is enabled.

Explanation

The speed of the robot is determined by the product of the speed set by this program instruction and the speed set by the SPEED instruction.
For example, if the monitor speed has a value of 50 and program speed is set at 60, the maximum speed of the robot will be 30%.

[NOTE]

If the sum of the speed specified by this program instruction and the speed specified by SPEED command or MON_SPEED instruction exceeds 100%, motion speed of the robot will be forcibly set at 100%. (Only when speed limit option is released.)

The monitor speed is automatically set to 10% in default settings.

Robot operating instructions that are already running will not be affected by this instruction. The newly set speed will be effective after the current or the next motions are completed.

Example

When the program speed is 100%:

>MON_SPEED 30 ↵	Robot speed is set to 30% of the maximum speed.
>MON_SPEED 50 ↵	Robot speed is set to 50% of the maximum speed.
>MON_SPEED 100 ↵	Robot speed is set to 100% of the maximum speed.

ACCURACY distance ALWAYS FINE

Function

Sets the accuracy when determining the robot pose.

Parameter

Distance

Specifies the distance of accuracy range in millimeters.

ALWAYS

If this parameter is entered, the accuracy setting remains valid until the next ACCURACY instruction is executed. If not entered, the accuracy setting is valid only for the next motion instruction.

FINE

If this parameter is entered, the robot pose is determined only when the current values match the taught pose. If omitted, the pose is determined as if the command value matches the taught pose.

Explanation

When the parameter ALWAYS is entered, all the proceeding motions are controlled by the accuracy set by this instruction. The default accuracy setting is 1 mm.

There is a limit to the effect of the accuracy setting, since in AS system the accuracy check is not started until the robot decelerates as it approaches the taught pose. (See also 4.5.4 Relation between CP Switch and ACCURACY, ACCEL, DECEL Instructions.)

[NOTE]

When the accuracy is set at 1 mm, the robot sets the pose after each motion instruction, coming to a pause in between the motion segments. To assure CP motion, set the accuracy range greater.

Setting the accuracy range too small may result in non-coincidence of the axes.

The accuracy set by this instruction is not the accuracy for repetition but for positioning the robot; therefore do not set values of 1 mm or less.

Example

ACCURACY 10 ALWAYS	The accuracy range is set at 10 mm for all motion instructions after this instruction.

ACCEL　acceleration ALWAYS
DECEL　deceleration ALWAYS

Function

Sets the acceleration (or deceleration) of the robot motion.

Parameter

Acceleration (ACCEL) / deceleration (DECEL)

Specifies the acceleration or deceleration in percentages of the maximum acceleration (deceleration).　Acceptable range is from 0.01 to 100.　Values over this limit are assumed as 100, values below the limit are assumed as 0.01.

ALWAYS

If this parameter is entered, the acceleration (or deceleration) here is valid until the next ACCEL (or DECEL) instruction.　If not entered, this instruction affects only the next motion instruction.

Explanation

ACCEL instruction sets the acceleration when the robot starts a motion as a percentage of the maximum acceleration.　DECEL instruction sets the deceleration when the robot is at the end of a motion as a percentage of the maximum deceleration.

Example

ACCEL　80　ALWAYS	The acceleration is set at 80% for all motions after this instruction.
DECEL　50	The deceleration for the next motion instruction is set at 50%.

BREAK

Function

Holds execution of the next step in the program until the current robot motion is completed.

Explanation

This instruction has the following two effects:

1. Holds the execution of the program until the robot reaches the destination of the current motion instruction.
2. The CP motion from the current motion to the next motion is interrupted. The robot comes to a stop in between the motion segments.

BRAKE

Function

Stops current robot motion.

Explanation

Stops current robot motion immediately and skips to the next step in program.

BSPEED speed

Option

Function

Sets the robot's motion speed (block speed). The robot motion speed is calculated by monitor speed × program speed × block speed.

Parameter

Speed

Sets the speed (acceptable range: 1 to 1000%). The speed set by this instruction is valid until the next BSPEED instruction is executed.

Explanation

The robot motion speed is calculated by monitor speed × program speed × block speed. However, the total speed cannot exceed 100%. Values up to 1000 can be entered for each speed, but if the total speed exceeds 100%, it is automatically cut down to 100%. For example, if the monitor speed is 100% and the program speed 50%, the motion speed is calculated by 100%×50%×block speed. If the block speed is less than 200%, the speed varies following the result of the above expression, but if it is over 200%, the motion speed always becomes 100%.

[NOTE]

1. When the program selection is reset, and a new program is executed (e.g. via EXECUTE command or by program selection via the teach pendant), the block speed is set at the default value of 100%. When the program is selected externally, the block speed is set at the default value if the program is selected by external program reset, but not with RPS and JUMP signals.

2. Note that the robot may not move in the specified program speed if the program is not executed from the beginning of the program or when the steps are skipped. In the example below, the robot is stopped while in step 3 and the motion is resumed after jumping to step 25. Then, the block speed at step 25 will be the speed of block 1.

```
Step 1    BSPEED  block1       ; Sets the speed for block 1.
Step 2    Joint  Speed 9……
Step 3    Linear  Speed 9……
   :
Step 12   BSPEED  block2       ; Sets the speed for block 2.
Step 13   Joint  Speed 9……
Step 14   Linear  Speed 9……
   :
Step 24   BSPEED  block3       ; Sets the speed for block 3.
Step 25   Joint  Speed 9……
Step 26   Linear  Speed 9……
```

Example

Write the program as follows so that the speed is changed by 4 bits from an external signal.

```
a=BITS(first signal for external speed selection,4)
BSPEED block1[a]
```

The following program enables selecting speed from an external device:

```
BSPEED block1                             ; Sets the default value for the block.
IF SIG(External_speed ON)THEN             ; Determines if external speed selection is
                                          enabled.
a=BITS(first signal for external speed selection,4)  ; Acquires the number used for external
                                                       selection
IF(a<11)THEN                              ; Setting not possible if a is 11+
BSPEEDblock11[a]                          ; Sets the selected block speed.
END
END
Joint   Speed 9……                         ; Moves in selected block speed.
Joint   Speed 9……
```

Real number variable “block 1” must be defined in advance.

```
block1=50
block11[0]=10
block11[1]=20
block11[2]=30
block11[3]=40
```

For example, if the first signal for external program selection is 1010, and the signals are inputs as:

```
1010･･･OFF
1011･･･ON
1012･･･OFF
1013･･･OFF
```

then a = 2, therefore block 11[2] is chosen and the motion speed becomes 30%.

REFFLTSET joint value moving average span, position moving average span, orientation moving average span, signal moving average span

Function

Specifies the moving average for robot command values. This instruction is valid only when the moving average option for command value is enabled.

Parameter

Joint value moving average span
Specifies the moving average span for the joint angles when the robot is moving in joint interpolation motion. Unit: [ms]. Acceptable range: integer between 1 through 254. The specified value is rounded up depending on the AS system control cycle. (This is the same for parameters 2-4).

Position moving average span
Specifies the moving average span for position values when the robot is moving in linear interpolation or circular motion. Unit: [ms]. Acceptable range: integer between 1 through 254. When omitted, the same value as joint value moving average span is set.

Orientation moving average span
Specifies the moving average span for orientation values when the robot is moving in linear interpolation or circular motion. Unit: [ms]. Acceptable range: integer between 1 through 254. When omitted, the same value as position moving average span is set.

Signal moving average span (Option)
Specifies the moving average span for signal outputs. Unit: [ms]. Acceptable range: integer between 1 through 254. When omitted, this parameter value is set by multiplying the ratio between the default value of joint value moving average span and its current setting to the default value of the signal moving average span.

[NOTE]

The relative relation between the above four parameters are balanced for the default setting. Therefore, when making any modifications, it is usually only necessary to specify the parameter for the joint value moving average span. This way, the relative relation between the parameters will be kept adequate.

The default and current values for each parameter can be checked via REFFLTSET_STATUS command.

Explanation

Moving average span is set to make the robot reach the specified pose smoothly. When this span is set longer, the robot's vibration is reduced and the robot makes a smoother motion. However, the cycle time will also become longer and there is a tendency that the robot takes a greater shortcut than the taught path.

On the other hand, when the span is set shorter, the cycle time is shortened and the robot follows a trajectory that is more precise to the taught path. However, the robot vibration tends to be greater.

[NOTE]

Normally do not use this instruction, because using this instruction changes the dynamic characteristics. When using this instruction, follow the below procedure:

- Gradually change the values, in about 8 ms increments, and confirm the robot motion after making the changes.
- When checking the robot motion, start at a low monitor speed of about 20% and gradually raise the speed.

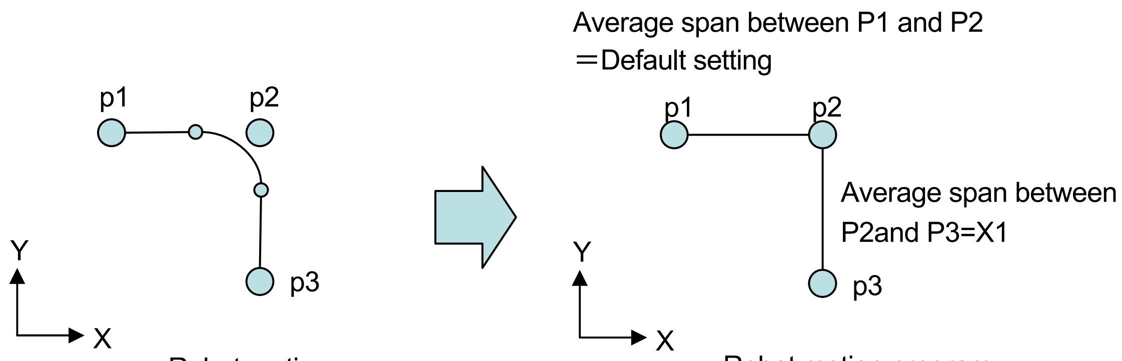
[NOTE]

The values set by executing this instruction apply to all robot motions, as is with SPEED ALWAYS instruction. To reset to the default value, use REFFLTRESET instruction.

The continuous path motion between the current motion and the next motion instruction is interrupted when REFFLTSET instruction is executed. That is, the two motions will not be followed in a one consecutive motion, but the robot will stop once at the end of the first motion before entering the second motion.

Example

REFFLTSET 64	Sets the robot's joint value, position, and orientation average span to 64 ms. The signal average span is set to the default value.
REFFLTSET 64, 64, 64, 32	Sets the robot's joint value, position, and orientation average span to 64 ms and sets the signal average span to 32 ms.



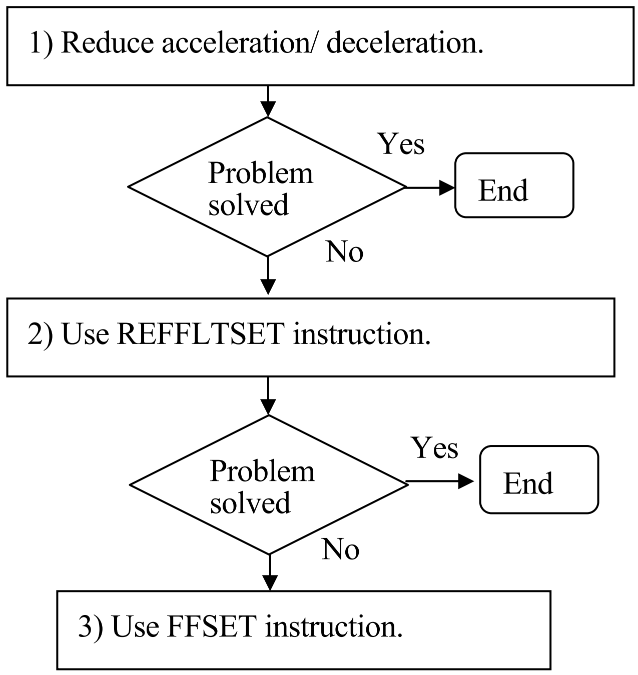
Robot motion program
JMOVE　p1
LMOVE　p2
LOMVE　p3

Robot motion program
JMOVE　p1
LMOVE　p2
REFFLTSET　X1
LOMVE　p3

Take the countermeasures for vibration following the below procedure:

1) Reduce the vibration by reducing the acceleration and deceleration.
   Use AS instructions ACCEL/DECEL to change them.
   For block teaching, change them via Aux. 0301 Acceleration/ Deceleration setting.　(Aux.0301 can be used only when [ACCEL and DECEL] setting in Aux. 0399 is set to [Enable].)

   When the vibration is not reduced or the cycle time is too long, reset them to the original setting and perform the next adjustment.



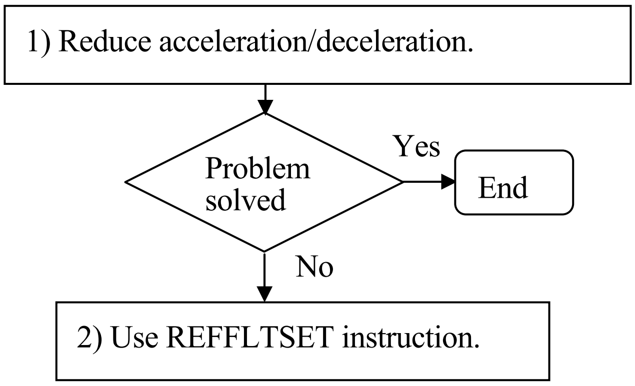
2) Smooth the robot motion via REFFLTSET instruction.
   Set the moving average span larger via REFFLTSET instruction.
   This is effective when the section where the average span is set larger (i.e. the section between REFFLTSET and REFFLTRESET) satisfies the below condition:

- There is more than one step.
- Few steps that require axis coincidence with the current pose.
- There are many steps with short distance below 50 mm.
- The accuracy setting is small in the motion step before REFFLTSET/REFFLTRESET instruction.

It is recommended to adjust the acceleration and deceleration together with the moving average span. Executing REFFLTSET instruction changes the robot's path, so be careful when confirming the motion.

When the vibration is not reduced or the cycle time is too long after using REFFLTSET instruction and changing the acceleration and deceleration, reset the REFFLTSET instruction and acceleration/ deceleration setting to the original setting and perform the next adjustment using FFSET instruction. FFSET instruction is explained later in this section.

Take the countermeasures for accuracy following the below procedure:



1) Confirm the path accuracy at a low speed.
   Use AS instruction SPEED or speed instruction in block teaching to change the speed.

   If the path accuracy does not improve or the speed setting does not match with the application conditions, set the speed back to the original setting and perform the next adjustment.

2) Improve the path accuracy via REFFLTSET instruction.
   Use REFFLTSET instruction to reduce the moving average span and improve the path accuracy.

   The vibration tends to increase when this setting is done, so be careful when checking the robot motion.

REFFLTRESET

Function

Resets the robot's moving average span to the default value.

Parameter

Resets to the default value the moving average span changed by REFFLTSET instruction.

As with the REFFLTSET instruction, the continuous path motion between the current motion and the next motion instruction is interrupted when this instruction is executed. That is, the two motions will not be followed in a one consecutive motion, but the robot will stop once at the end of the first motion before entering the second motion.

Example

In the sample program below, the moving average span from p2 to p3 is X1, and the moving average span between p3 to p1 is set at the default value.

```
Robot motion program
JMOVE    p1
LMOVE    p2
REFFLTSET    X1
LOMVE    p3
REFFLTRESET
JMOVE p1
```

FFSET JT1 gain, JT2 gain, JT3 gain, JT4 gain, JT5 gain, JT6 gain, JT7 gain, JT8 gain, JT9 gain

Function

Sets the speed/ acceleration feed forward gain for when the robot starts moving.

Parameter

JT 1 gain, JT 2 gain, JT 3 gain, JT 4 gain, JT 5 gain, JT 6 gain, JT 7 gain, JT 8 gain, JT 9 gain
Specifies the speed/ acceleration feed forward gain for each axis in real values. Acceptable range: 0 -1 (valid to the third decimal place). Values for the axes other than JT1 can be omitted. The values will be set as follow when omitted.

When parameters for robot axes are omitted:	Sets the same value as the setting for JT1.
When parameters for external axes are omitted:	The ratio between the default and current value of JT1 is multiplied to the default value for the omitted external axis.

[NOTE]

The gains for the robot axes should be set equal to JT1. The default and current setting for each parameter can be checked via FFSET_STATUS command.

Explanation

When the speed/acceleration feed forward gain is set smaller, the robot's vibration is reduced and the robot makes a smoother motion. However, the cycle time will also become longer and there is a tendency that the robot takes a greater shortcut than the taught path.

On the other hand, when the gain is set greater, the cycle time is shortened and the robot follows a trajectory that is more precise to the taught path. However, the robot vibration tends to be greater.

[NOTE]

Normally do not use this instruction, because using this instruction changes the dynamic characteristics. When using this instruction, follow the below procedure:

- Gradually change the values, in about 0.1 increments, and confirm the robot motion after making the changes.
- When checking the robot motion, start with a low monitor speed of about 20% and gradually raise the speed.

This instruction changes the dynamic characteristics; therefore confirm the robot speed does not change suddenly at the beginning and end of the modification.

[NOTE]

The values set by executing this instruction apply to all robot motions, as is with SPEED ALWAYS instruction. To reset to the default value, use FFRESET instruction.

Example

FFSET 0.5	Sets all the speed/ acceleration feed forward value to 0.5.
FFSET 0.5,0.49	Sets the speed/ acceleration feed forward value to 0.5 for all the axes except for JT2, and sets the speed/ acceleration feed forward value to 0.5 for JT2 to 0.49.

In the sample robot motion program below, the speed/acceleration feed forward gain is changed to 0.5 after axis coincidence in #p1.

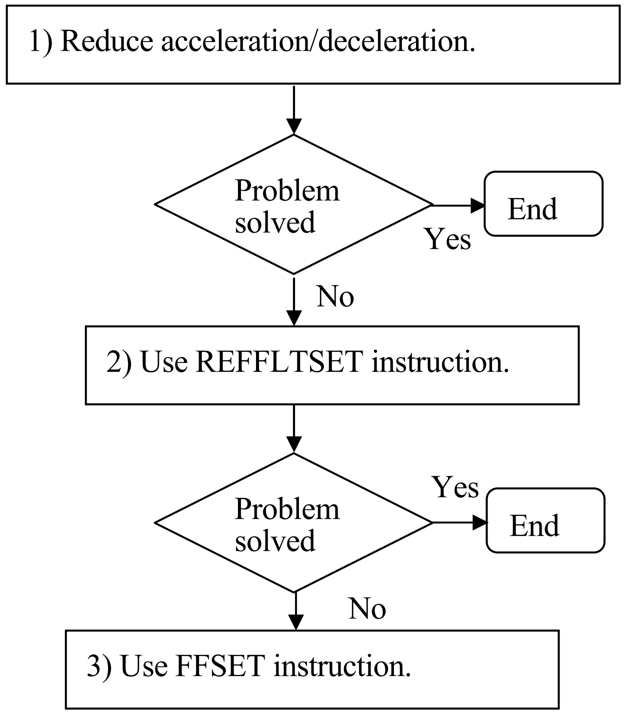
```
JMOVE  #p1
FFSET  0.5
JMOVE #p2
FFRESET
JMOVE #p3
```

Take the countermeasures for vibration following the below procedure:

1) Reduce the vibration by reducing the deceleration.
   Follow the procedures explained for REFFLTSET instruction.

2) Smoothen the motion via REFFLTSET instruction.
   Follow the procedures explained for REFFLTSET instruction.

3) Smoothen the motion via FFSET instruction.
   Set the speed/acceleration feed forward gain via FFSET instruction to smoothen the robot motion.
   Using FFSET instruction changes the robot path so carefully confirm the robot motion after executing the instruction.



FFRESET

Function

Resets the robot speed/acceleration feed forward gain to the default value.

Explanation

Resets the robot speed/acceleration feed forward gain changed by FFSET instruction to the default value. This works as setting the default value via FFSET instruction.

Example

In the sample robot motion program below, the speed/ acceleration feed forward gain is changed to 0.5 after axis coincidence in #p1. The feed forward gain is reset to the default value after axis coincidence in #p2.

```
JMOVE  #p1
FFSET  0.5
JMOVE #p2
FFRESET
JMOVE #p3
```

6.3 Clamp Control Instructions

OPEN	Outputs clamp open signal when next motion instruction begins.
OPENI	Outputs clamp open signal when current motion instruction is completed.
CLOSE	Outputs clamp close signal when next motion instruction begins.
CLOSEI	Outputs clamp close signal when current motion instruction is completed.
RELAX	Turns OFF clamp signals when next motion instruction begins.
RELAXI	Turns OFF clamp signals when current motion instruction is completed.
OPENS	Output clamp open signal during execution of motion instruction.
CLOSES	Output clamp close signal during execution of motion instruction.
RELAXS	Turns OFF clamp signals during execution of motion instruction.
GUNON	Turns ON gun signal and controls gun output timing by distance. (Option)
GUNOFF	Turns OFF gun signal and controls gun output timing by distance. (Option)
GUNONTIMER	Controls gun output ON timing by timer. (Option)
GUNOFFTIMER	Controls gun output OFF timing by timer. (Option)

OPEN　clamp number
OPENI　clamp number

Function

Opens robot clamps (outputs clamp open signal).

Parameter

Clamp number

Specifies the number of the clamp.　If omitted, 1 is assumed.

Explanation

This instruction outputs signals to the control valve of pneumatic hand to open the clamp.

With the OPEN instruction, the signal is not output until the next motion starts.

The timing for signal output using the OPENI instruction is as follows:

1. If the robot is currently in motion, the signal is output after that motion is completed.　If the robot is moving in CP motion, the CP motion is suspended (BREAK).
2. If the robot is not in motion, the signal is sent immediately to the control valve.

Example

OPEN	The clamp open signal is sent to the control valve of clamp 1 when the robot starts the next motion.
OPENI 2	The clamp open signal is sent to the control valve of clamp 2 as soon as the robot completes the current motion.

CLOSE clamp number
CLOSEI clamp number

Function

Closes robot clamps (outputs clamp close signal).

Parameter

Clamp number

Specifies the number of the clamp. If omitted, 1 is assumed.

Explanation

This instruction outputs signals to the control valve of pneumatic hand to close the clamp.

With the CLOSE instruction, the signal is not output until the next motion starts.

The timing for signal output using the CLOSEI instruction is as follows:

1. If the robot is currently in motion, the signal is output after that motion is completed. If the robot is moving in CP motion, the CP motion is suspended (BREAK).
2. If the robot is not in motion, the signal is sent immediately to the control valve.

Example

CLOSE 3	The clamp close signal is sent to the control valve of clamp 3 when the robot starts the next motion.
CLOSEI	The clamp close signal is sent to the control valve of clamp 1 as soon as the robot completes the current motion.

RELAX clamp number
RELAXI clamp number

Function

Turns OFF the pneumatic solenoid valves for both OPEN and CLOSE signals (turns the clamp signal OFF. In double solenoid specification, both clamp open and close signals are turned OFF).

Parameter

Clamp number
Specifies the clamp number. If omitted, 1 is assumed.

Explanation

With the RELAX instruction, the signal is not output until the next motion starts.

The timing for signal output using the RELAXI instruction is as follows:
1. If the robot is currently in motion, the signal is output after that motion is completed. If the robot is moving in CP motion, the CP motion is suspended (BREAK).
2. If the robot is not in motion, the signal is sent immediately to the control valve.

OPENS clamp number
CLOSES clamp number
RELAXS clamp number

Function

Turns ON/OFF the open and close signals of the pneumatic solenoid valves.

Clamp number

Specifies the number of the clamp. If omitted, 1 is assumed.

Explanation

This instruction is different from the OPEN/CLOSE/RELAX and OPENI/CLOSEI/RELAXI instruction in the following ways:

1. OPEN/CLOSE/RELAX instructions:
   The signal is output when the next motion starts.
2. OPENI/CLOSEI/RELAXI instructions:
   If the robot is in motion, the signal is output when that motion is completed. The CP motion is interrupted (BREAK).
3. OPENS/CLOSES/RELAXS instructions:
   The signal is output immediately after this instruction is executed.

This instruction is not affected by the PREFETCH.SIGINS switch.

GUNON gun number, distance
GUNOFF gun number, distance

Option

Function

Turns ON/OFF the gun signal and controls the gun output timing by the specified distance.

Parameter

Gun number

Specifies gun number 1 or 2.

Distance

Specifies the distance (in mm) to adjust the ON/OFF timing of the GUN. Negative value advances the timing, and the positive value delays the timing. If not specified, 0 is assumed.

Explanation

The gun signal is turned ON/ OFF when the motion instruction after the GUNON/GUNOFF instruction is executed. The output timing is determined by the distance specified in the instruction and the time set by GUNONTIMER/GUNOFFTIMER.

Example

GUNON 2,100 Turns ON the ON signal of gun 2 delaying the ON timing by 100 mm.

GUNONTIMER gun number, time
GUNOFFTIMER gun number, time

Option

Function

Adjusts the timing of the gun output (timing at which gun is turned ON/OFF) by the specified time.

Parameter

Gun number

Specifies gun number 1 or 2.

Time

Specifies the time (in seconds) to adjust the ON/OFF timing of the GUN. Negative value advances the timing, and the positive value delays the timing. If not specified, 0 second is assumed.

Explanation

The adjustment time is determined by the environment of the gun system (e.g. the distance from the valve to the tip of the gun, the type of the paint, climate, etc.) so set the timing in the beginning of the program. To change the timing outside the program, use a variable for the time parameter (e.g. when the timing has to be changed due to paint color or viscosity).

This instruction only adjusts the output timing of the gun and does not actually turn the gun ON/OFF.

Example

GUNONTIMER 1,-0.5 Advances the timing of ON signal for gun 1 by 0.5 seconds.

6.4 Configuration Instructions

RIGHTY	Changes configuration so the robot arm resembles a person's right arm.
LEFTY	Changes configuration so the robot arm resembles a person's left arm.
ABOVE	Changes configuration so the elbow joint is in the above position.
BELOW	Changes configuration so the elbow joint is in the below position.
UWRIST	Changes configuration so the angle of JT5 has a positive value.
DWRIST	Changes configuration so the angle of JT5 has a negative value.

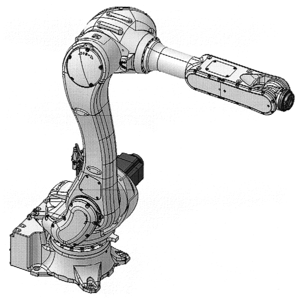
RIGHTY
LEFTY

Function

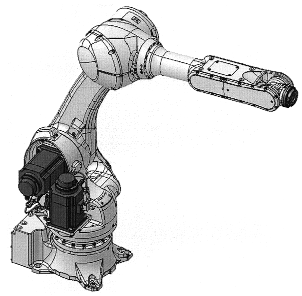
Forces a robot configuration change during the next motion so the robot arm is configured to resemble a person's right (RIGHTY) or left (LEFTY) arm. The configuration may not be changed during a linear interpolated movement, or when the destination of the next motion is expressed in joint displacement values.

See also 11.7 Setting Robot Configuration.

Example



RIGHTY



LEFTY

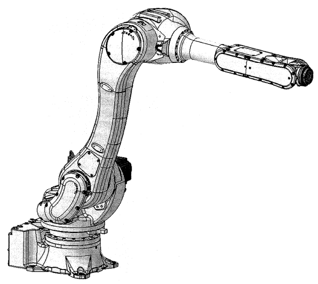
ABOVE
BELOW

Function

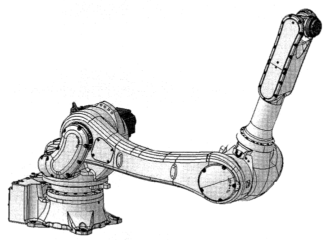
Forces a robot configuration change during the next motion so the "elbow joint" (joint 3) is configured to resemble a person's arm when the elbow is in above or below position relative to the wrist. The configuration may not be changed during a linear interpolated movement, or when the destination of the next motion is expressed in joint displacement values.

See also 11.7 Setting Robot Configuration.

Example



ABOVE



BELOW

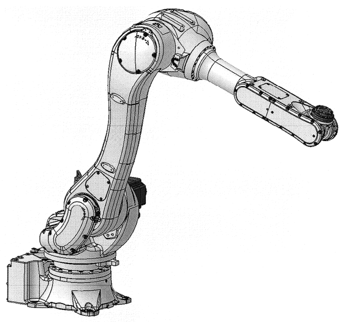
UWRIST
DWRIST

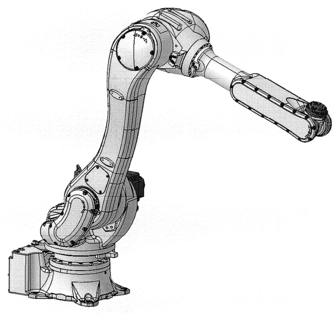
Function

Forces a robot configuration change during the next motion so the angle of joint 5 (JT5) has a positive or negative value. The configuration may not be changed during a linear interpolated movement, or when the destination of the next motion is expressed in joint displacement values.

See also 11.7 Setting Robot Configuration.

Example





UWRIST
(Joint 5 is 90°)

DWRIST
(Joint 5 is -90°)*
NOTE* Joint 4 has rotated 180°.

6.5 Program Control Instructions

GOTO	Jumps to specified label.
IF	Sets condition for GOTO instruction.
CALL	Branches to a subroutine.
RETURN	Returns to the program that called the subroutine.
WAIT	Puts program execution in stand-by until condition is set.
TWAIT	Puts program execution in stand-by until specified time elapses.
MVWAIT	Puts program execution in stand-by until the given distance or time is reached.
LOCK	Changes priority of robot control programs.
PAUSE	Pauses the program execution.
HALT	Stops program execution. (Cannot resume.)
STOP	Stops execution cycle.
SCALL	Branches to a subroutine.
ONE	Calls program when error occurs.
RETURNE	Executes from the step following the step in which the error occurred.
JUMP	Switches the executing program.
SJUMP	Switches the executing program to program specified by character string.
MON_TWAIT	Waits for set time corresponding to speed setting. (Option)

GOTO label IF condition

Function

Jumps to the program step with the specified label.

Parameter

Label

Specifies label of the program step to jump to. The label can be any character string within 15 letters (including alphanumeric characters, periods, underscores) that starts with an integer or alphabetical letter and is followed by a colon (:).

Condition

Specifies the condition to jump. This parameter and the keyword IF can be omitted. If omitted, the program jumps whenever the instruction is executed.

Explanation

Jumps to the step specified by the label. If the condition is specified, the program jumps when the condition is set. If the condition is not set, the execution goes on to the next step after this instruction.

Note that the label and the step number are different. Step numbers are assigned to all program steps automatically by the system. Labels are purposely given to program steps and are entered after the step number.

This instruction functions the same as the IF GOTO instruction when a condition is specified.

Example

GOTO 100	Jumps to label 100, there is no condition. If there is no step labeled 100, error occurs.
GOTO 200 IF n==3	When variable "n" is equal to 3, then the program jumps to label 200. If not, the step after this step is executed.

IF condition GOTO label

Function

Jumps to the step with the specified label when the given condition is set.

Parameter

Condition

Specifies the condition in expressions, e.g. n = = 0, n>3, m + n<0.

Label

Specifies the label of the step to jump to (not the step number). The label must be within the same program.

Explanation

The program jumps to the step specified by the label, when the given condition is set. If the condition is not satisfied, the step after this instruction is executed.

If the specified label does not exist, error occurs.

Example

IF n>3 GOTO 100	If the value of integer variable “n” is greater than 3, then the program jumps to the step labeled 100. If n is not greater than 3, then the step after this step is executed.
IF flag GOTO 25	If the value of integer number variable “flag” is not 0, the program jumps to the step labeled 25. If the value of the variable “flag” is equal to 0, the step after this step is executed. This is the same as writing: IF flag<>0 GOTO 25.

CALL program name

Function

Holds execution of the current program and jumps to a new program (subroutine). When the execution of the subroutine is completed, the processing returns to the original program and executes the step after the CALL instruction.

Parameter

Program name
Specifies the subroutine to execute.

Explanation

This instruction temporarily holds the execution of the current program and jumps to the first step of the specified subroutine.

[NOTE]

The same subroutine cannot be called from a robot control program and a PC program at the same time. Also, a subroutine cannot call itself.

Up to 20 programs can be held while subroutines are called.

Example

CALL sub1	Jumps to the subroutine named “sub1”. When the RETURN instruction in “sub1” is executed, the program execution returns to the original program and executes the program from the step after this CALL instruction.

RETURN

Function

Ends execution of a subroutine and returns to the step after the CALL instruction in the program that called the subroutine.

Explanation

This instruction ends execution of a subroutine and returns to the program that called that subroutine. If the subroutine is not called from another program (e.g. when the subroutine is executed by EXECUTE command) the program execution is ended.

At the end of the subroutine, the program execution returns to the original program even if there is no RETURN instruction. However, the RETURN instruction should be written as the last step of the subroutine (or at any place the subroutine is to be ended).

WAIT condition

Function

Makes program execution wait until the specified condition is set (condition becomes TRUE).

Parameter

Condition

Specifies the stand-by condition. (real number expressions)

Explanation

This instruction holds execution of the program until the specified condition is set. CONTINUE NEXT command resumes the program execution before the condition is set (skips the WAIT instruction being executed).

Example

WAIT SIG (1001, – 1003)	Holds execution of the program until external input signal 1001 (WX1) is ON and signal 1003(WX3) is OFF.
WAIT TIMER(1)>10	Holds program execution until the value of timer1 is over 10 (seconds).
WAIT n>100	Holds program execution until the value of variable “n” exceeds 100. (In this example, suppose variable “n” is a value that is counted up by a PC program or program interruptions.)

TWAIT time

Function

Holds program execution until the specified time elapses.

Parameter

Time

Specifies the time, in seconds, for how long the program execution is held.

Explanation

This instruction holds the program execution until the specified time elapses.

A TWAIT instruction in execution can be skipped using the CONTINUE NEXT command.

WAIT instruction can be used instead of the TWAIT instruction to gain the same result.

Example

TWAIT 0.5	Waits for 0.5 seconds.
TWAIT deltat	Waits until the value of variable “deltat” elapses.

MVWAIT value

Function

Holds program execution until the remaining distance (or time) of the current motion becomes shorter than the specified distance (or time).

Parameter

Value

Specifies the distance or time. The distance is expressed in millimeters (mm) and the time in seconds (S). If the unit is not specified, it is considered as millimeters.

Explanation

This instruction is used to synchronize program execution with the robot motion. However, note that since this instruction monitors the remaining distance (or time) based on the command values, it may be different from the actual remaining distance (or time) due to response lag. When the robot is moving in joint interpolated motion, the distance specified and the actual distance may differ greatly. If the current motion is completed when this instruction is executed, the execution goes on to the next instruction without waiting. CONTINUE NEXT instruction can be used to skip the MVWAIT instruction while it is in execution.

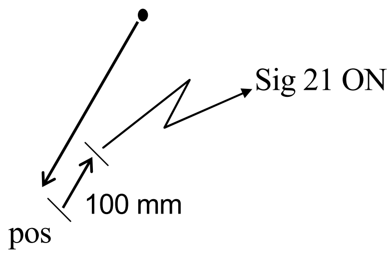
[NOTE]

MVWAIT instruction cannot be used in PC programs. Also, this instruction cannot be used with DO command.

Example

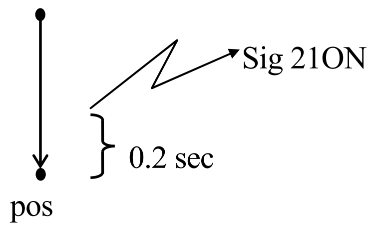
In the diagram below, the robot moves towards pose “pos”, and when coming within 100 mm to “pos”, the signal 21 is turned ON. This is true only when system switch PREFETCH.SIGINS is ON (reads the signal before axes coincidence) and the robot is within the accuracy range.

```
LMOVE pos
MVWAIT 100mm
SIGNAL 21
```



In the diagram below, the robot moves towards pose “pos”, and when the required time to reach “pos” becomes 0.2 seconds, signal 21 is turned ON. This is true only when PREFETCH.SIGINS is ON and the robot is in the accuracy range.

```
LMOVE pos
MVWAIT 0.2S
SIGNAL 21
```



LOCK priority

Function

Changes the priority of the robot program currently selected on the stack.

Parameter

Priority

Specifies the priority in real numbers from 0 to 127.

Explanation

Normally, the priority of robot program is 0. The priority can be changed using this instruction. The greater the number the higher the priority will be.

Example

LOCK 2	Changes the priority to 2.

PAUSE

Function

Temporarily holds (pauses) the program execution.

Explanation

This instruction temporarily holds the program execution and displays a message on the terminal. Execution can be resumed using the CONTINUE command.

This instruction is convenient when checking a program. The values of the variables can be checked while the program is held by the PAUSE instruction.

HALT

Function

Stops the program execution. The program cannot be resumed after this instruction is executed.

Explanation

Stops the program execution regardless of the remaining steps. A message is displayed on the terminal.

Program execution stopped by this instruction cannot be resumed using the CONTINUE command.

STOP

Function

Terminates the current execution cycle.

Explanation

If there are cycles remaining to be completed, execution returns to the first step, otherwise execution ends. This instruction marks the end of the execution path and has a different effect than the HALT instruction.

If there are execution cycles remaining, execution continues with the first step of the main program* (even if STOP instruction was processed during execution of a subroutine or another interrupting program, execution returns to the main program).

NOTE* A main program is the program executed using the EXECUTE, STEP, PCEXECUTE commands. A subroutine is a program called from another program by CALL, ON or ONI instructions.

A RETURN instruction in a main program functions in the same way as a STOP instruction.

Program execution stopped by a STOP instruction cannot be resumed by CONTINUE command.

SCALL string expression, variable

Function

Jumps to the subroutine with the name given by the string expression.

Parameter

String expression

Specifies the subroutine name in the form of a string expression.

Variable

If the subroutine call is executed normally, then the value 0 is assigned to this variable. If some abnormality occurred during the subroutine call, the error code (≠ 0) is assigned. If omitted, the execution comes to an error stop when an abnormality occurs in the subroutine call.

Explanation

This instruction functions the same as the CALL instruction except that the program name is expressed as a string expression. (See CALL instruction).

Example

`$prog="sub1"`	
`SCALL $prog`	Jumps to a subroutine named "sub1".
`num=12`	
`$temp1=$ENCODE(/I2,num)`	Converts into a string expression, the real value given to "num", and jumps to the subroutine named "sub12".
`$temp2=""`	

```
FOR i=1 to LEN($temp1)
$temp3=$MID($temp1,i,1)
IF $temp3<> "" THEN
$temp2=$temp2+$temp3
END
END
SCALL "sub"+$temp2
```

ONE program name

Function

Calls the specified program when an error occurs.

Parameter

Program name
Specifies the name of the program to execute when an error occurs.

Explanation

This instruction calls the specified program when an error occurs. PC programs can be called too.

To return to the original program from the called program, RETURN (or RETURNE) instruction is used. RETURN instruction returns the execution to the step where the error occurred. RETURNE instruction returns the execution to the step after the error. (If neither RETURN nor RETURNE instruction exists within the program, the execution cycle stops at the end of the called program.)

Motion instructions cannot be used in the program called by ONE instruction.

If error occurs in the program called by ONE, the program execution stops there.

[NOTE]

As long as the main program containing the ONE instruction is in execution, the instruction is effective on errors in the subroutines, as well as the main program. When the main program ends execution, ONE becomes ineffective.

When an error arises, the Error lamp does not illuminate if a program is called by the ONE instruction.

RETURNE

Function

Returns to the step after the error.

Explanation

This instruction is commonly paired with the ONE instruction. With ONE instruction, the program jumps to a subroutine when an error occurs. Then, the execution returns to the step after the error in the original program when the RETURNE instruction in the subroutine is executed.

JUMP program name

Function

Ends the current program and moves on to a different program.

Parameter

Program name
Specifies the program to change to.

Explanation

This instruction ends the execution of the current program and moves to the first step of the specified program. After the execution of specified program is completed, it does not return to the original program. However, if this instruction were executed in a subroutine program called by CALL instruction, the execution returns to the next step in the source program after the program execution is completed. The execution cannot jump from a subroutine program to the source program. Error (E0121) “Cannot specify the jump source program as jump destination.” occurs.

SJUMP program name, status variable

Function

Ends the current program and moves on to a different program.

Parameter

Program name

Specifies the program to change to in character string.

Status variable

When the program change is done normally, 0 is written. If not, the error code (≠0) is written. When omitted, the program execution comes to an error stop when switching is not done normally.

Explanation

This instruction ends the execution of the current program and moves to the first step of the specified program by the character string. After the execution of specified program is completed, it does not return to the original program. However, if this instruction were executed in a subroutine program called by CALL instruction, the execution returns to the next step in the source program after the program execution is completed. The execution cannot jump from a subroutine program to the source program. Error (E0121) "Cannot specify the jump source program as jump destination." occurs.

MON_TWAIT time

Option

Function

When the command value path constant move function (option) is valid, holds program execution until the specified time (seconds) times the ratio between monitor speed 100% and speed setting value (monitor speed or check speed) elapses.

Parameter

Time

Specifies the time, in seconds, for how long the program execution is held.

Explanation

When the command value path constant move function (option) is invalid, holds program execution until the specified time (seconds), same as in TWAIT instruction.

When the command value path constant move function (option) is valid, holds program execution until the specified time (seconds) times the monitor speed 100%/speed setting value (monitor speed check speed) elapses.

The wait time is as follows:
When monitor speed is set to 100% in repeat mode, the specified time (seconds).
When monitor speed is set to 10% in repeat mode, 10 times the specified time (seconds).
In check mode, the specified time (seconds) multiplied by the ratio between the maximum speed in straight linear motion and specified check speed.

However, even if the command value path constant move function (option) is valid, the program waits for the specified time ignoring the monitor speed or check speed in cases such as when no motion step in motion exist (i.e. this instruction is used at the beginning of the program), or in check once mode.

The MON_TWAIT instruction in execution can be skipped using CONTINUE NEXT instruction.

Example

When the command value path constant move function (option) is valid

MON_TWAIT 0.5	Waits for 0.5 seconds if monitor speed in repeat mode is 100%.
	Waits for 5 seconds if monitor speed in repeat mode is 10%.

MON_TWAIT deltat	Waits until the value of variable “deltat” elapses if monitor speed in repeat mode is 100%. Waits until 10 times the value of variable “deltat” elapses if monitor speed in repeat mode is 10%.

6.6 Program Structure Instructions

IF……THEN…ELSE……END

WHILE……DO……END

DO……UNTIL

FOR……END

CASE……OF……VALUE……ANY……END

SCASE……OF……SVALUE……ANY……END

IF logical expression THEN
program instructions(1)
ELSE
program instructions(2)
END

Function

Executes a group of program steps according to the result of a logical expression.

Parameter

Logical expression
Logical expression or real value expression. Tests if this value is TRUE (not 0) or FALSE(0).

Program instructions (1)
The program instructions entered here are executed if the above logical expression is TRUE.

Program instructions (2)
The program instructions entered here are executed if the above logical expression is FALSE.

Explanation

This control flow structure executes one of the two groups of instructions according to the value of the logical expression. The execution procedure is as follows:

1. Calculates the logical expression, and jumps to step 4 if the resulting value is 0 (FALSE).
2. Calculates the logical expression, and executes program instructions (1) if the resulting value is 1 (TRUE).
3. Jumps to 5.
4. If there is the ELSE statement, program instructions (2) is executed.
5. Continues program execution from the step after END.

[NOTE]

1. ELSE and END statements each must be entered in a line on its own.
2. The IF…THEN structure must end with END statement.

Example

In the example below, if n is greater than 5, the program speed is set at 10%, if not it is set at 20%.

```
IF n>5 THEN
        sp=10
ELSE
        sp=20
END
SPEED sp ALWAYS
```

The program below first checks the value of variable “m”. If “m” is not 0, the program checks the external input signal 1001(WX1) and displays a different message according to the status of the signal. In this example, the outer IF structure does not have an ELSE statement.

```
IF m THEN
        IF SIG(1001) THEN
                PRINT"Input signal is TRUE"
        ELSE
                PRINT"Input signal is FALSE"
        END
END
```

WHILE condition DO
program instructions
END

Function

While the specified condition is TRUE, the program instructions are executed. When the condition is FALSE, the WHILE statement is skipped.

Parameter

Condition

Logical expression or real value expression. Checks if this value is TRUE (not 0) or FALSE (0).

Program instructions

Specifies the group of instructions to be executed when the condition is TRUE.

Explanation

This control flow structure repeats the given program steps while the specified condition is TRUE. The execution procedure is as follows:

1. Calculates the logical expression, and jumps to step 4 if the resulting value is 0 (FALSE).
2. Calculates the logical expression, and executes program instructions if the resulting value is 1 (TRUE).
3. Jumps to 1.
4. Continues program execution from the step after END.

[NOTE]

Unlike the DO structure, if the condition is FALSE, none of the program steps in the WHILE structure is executed.

When this structure is used, the condition must eventually change from TRUE to FALSE.

Example

In the following example, input signals 1001 and 1002 are monitored and robot motion is stopped based on their condition. When either of the signals from the two parts feeders changes to 0 (feeder is emptied), the robot stops and the execution continues from the step after the END statement (step 27 in this example).

If one of the feeders is empty at the time the WHILE structure begins (external input signal OFF=0), none of the steps in the structure is executed, and processing jumps to step 27.

```
.
.
.
WHILE SIG(1001,1002) DO
        CALL part1
        CALL part2
END
.
.
.
.
```

DO
program instructions
UNTIL　logical expression

Function

Creates a DO loop.

Parameter

Program instructions

These instructions are repeated as long as the logical expression is FALSE.

Logical expression

Logical expression or real value expression.　When the result of this logical expression changes to TRUE, execution of the program instructions in this structure is stopped.

Explanation

This control flow structure executes a group of program instructions while the given condition (logical expression) is FALSE.

The execution procedures are as follows:

1. Executes the program instructions.
2. Checks the value of the logical expression and if the result is FALSE, procedure 1 is repeated. If the result is TRUE, it jumps to procedure 3.
3. Continues program execution from the step after UNTIL statement.

The execution exits the DO structure when the value of the logical expression changes from FALSE to TRUE.

[NOTE]

Unlike the WHILE structure, the program instructions in the DO structure are executed at least once.

The program instructions between DO statement and UNTIL statement can be omitted. If there are no instructions, the logical expression after UNTIL is evaluated repeatedly. When the value of the logical expression changes to TRUE, then the execution exits the loop and goes on to the step after the DO structure.

The DO structure must end with an UNTIL statement.

Example

In the example below, the DO structure controls the following task: a part is picked up, and carried to the buffer. When the buffer becomes full, the binary input signal “buffer.full” is turned ON. When the signal turns ON, the robot stops and starts a different operation.

```
.
.
.
DO
    CALL get.part
    CALL put.part
UNTIL SIG(buffer.full)
.
.
.
```

FOR loop variable = start value TO end value STEP step value
program instructions
END

Function

Repeats program execution.

Parameter

Loop variable

Variable or real value. This variable is first set at an initial value, and 1 is added each time the loop is executed.

Starting value

Real value or expression. Sets the first value of the loop variable.

End value

Real value or expression. This value is compared to the present value of the loop variable and if the value of the loop variable reaches this value, the program exits the loop.

Step value

Real value or expression that can be omitted. This value is added or subtracted to the loop variable after each loop. Enter this parameter when using the STEP statement, unless the loop variable is to increment by 1. If step value is not specified, 1 is added to the loop variable. In this case, the STEP statement can be omitted too.

Explanation

This control flow structure repeats execution of the program instructions between the FOR and END statements. Loop variable is incremented by the given step value each time the loop is executed.

The execution procedures are as follows:

1. The start value is assigned to the loop variable.
2. Calculates the end value and the step value.
3. Compares the value of the loop variable with the end value.
    a. If the step value is positive, and the loop variable is greater than the end value, then jump to procedure 7.
    b. If the step value is negative and the loop variable is smaller than the end value, jump to procedure 7.

    In other cases, goes on to procedure 4.

4. Executes the program instructions after the FOR statement.
5. When the END statement is reached, the step value is added to the loop variable.
6. Returns to procedure 3.
7. Executes the program instructions after the END statement. (The value for the loop variable at the time of the comparison test at procedure 3 above does not change.)

[NOTE]

There must be an END statement for each FOR statement.

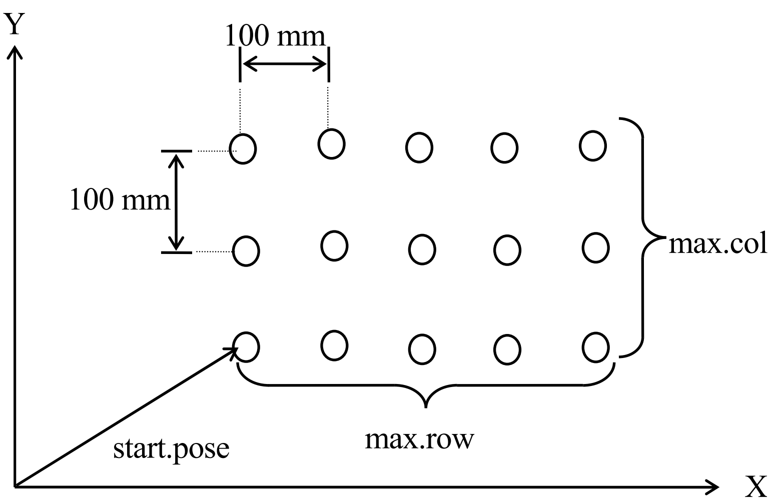
Beware that if the loop variable is greater than the end value (or less if the step value is negative) at the first check, none of the program instructions between FOR and END is executed.

The value for the number of loops (loop variable) must not be changed by other programming (operators, expressions, etc.) within the FOR loop.

Example

The subroutine "pick.place" picks up a part and places it on "hole". The parts are placed as shown in the figure below. (The pallet is placed parallel to X, Y axes of the world coordinates, and the distance between the parts is 100 mm.

```
FOR row = 1 TO max.row
POINT hole = SHIFT (start.pose BY (row-1)*100,0,0)
FOR col = 1 TO max.col
CALL pick.place
POINT hole = SHIFT(hole BY 0,100,0)
END
END
```



CASE　index variable　OF
VALUE　case number 1, ……:
program instructions
VALUE　case number 2, ……:
program instructions
:
VALUE　case number n, ……:
program instructions
ANY　:
program instructions
END

Function

Executes the program according to a particular case number.

Parameter

Index variable

Real value variable or expression. Decides which CASE structure to execute according to the value of this variable.

Program instructions

Executes these program instructions when the value of the index variable equals one of the values after the VALUE statement.

Explanation

This structure enables the program to select from among several groups of instructions and to process the selected group. This is a powerful tool in AS language that provides a convenient method for allowing several alternatives within the program.

The execution procedure is as follows:

1. Checks the value of the index variable entered after the CASE statement.
2. Checks through the VALUE steps and finds the first step that includes the value equal to the value of the index variable.
3. Executes the instructions after that VALUE step.
4. Goes on to the instructions after the END statement.

If there is no value that matches the index variable, the program instructions after the ANY statement are executed. If there is not an ANY statement, none of the steps in the CASE structure is executed.

[NOTE]

ANY statement and its program instructions can be omitted.

ANY statement can be used only once in the structure. The statement must be at the end of the structure as shown in the example below.

The colon ":" after the ANY statement can be omitted. When entering the colon, always leave a space after ANY. Without a space, ANY: is taken as a label.

Both the ANY and END statements must be entered on their own line.

Example

In the program below, if the value of real variable x is negative, the program execution stops after the message is displayed. If the value is positive, the program is processed according to these 3 cases:

1. if the value is an even number between 0 and 10.
2. if the value is an odd number between 1 and 9.
3. if the value is a positive number other than the above.

```
     IF x<0 GOTO 10

     CASE x OF
     VALUE 0,2,4,6,8,10:
     PRINT "The number x is EVEN"
     VALUE 1,3,5,7,9:
     PRINT "The number x is ODD"
     ANY :
     PRINT "The number x is larger than 10"
     END

     STOP

10   PRINT "Stopping because of negative value"
     STOP
```

```
SCASE  index variable  OF
SVALUE  string_1, ……:
program instructions
SVALUE  string_2, ……:
program instructions
:
SVALUE  string_n, ……:
program instructions
ANY  :
program instructions
END
```

Option

Function

Executes program based on condition specified by the character string.

Parameter

Index variable

Specifies character string variable or expression. Decides which SCASE structure to execute according to character string of this variable.

Program instructions

Executes these program instructions when the string of the index variable equals one of the values after the SVALUE statement.

Explanation

Unlike CASE structure described before, the execution condition for SCASE structure is set as character string. See also CASE structure.

If there is no string that matches the string character, the program instructions after the ANY statement are executed. If there is not an ANY statement, none of the steps in the SCASE structure is executed.

[NOTE]

ANY statement and its program instructions can be omitted.

ANY statement can be used only once in the structure. The statement must be at the end of the structure.

The colon “:” after the ANY statement can be omitted. When entering the colon, always leave a space after ANY. Without a space, ANY: is taken as a label.

Both the ANY and END statements must be entered on their own line.

Example

In the program below, if character string variable $str is equal to the string of $a+”c”, the program pc is executed. If character string variable $str is equal to the string of $a+”g”, the program pg is executed.

```
SCASE $str OF
SVALUE $a+”c”:
CALL pc
SVALUE $a+”g”:
CALL pg
SVALUE $a+”c”:
END
```

6.7 Binary Signal Instructions

RESET	Turns OFF all external output signals.
SIGNAL	Turns ON/OFF external I/O signals and internal signals.
PULSE	Turns ON output signal for the specified amount of time.
DLYSIG	Turns signal after the specified time has passed.
RUNMASK	Specifies the signals to mask.
BITS	Sets a group of signals to equal the specified value (Max. 16 signals)
BITS32	Sets a group of signals to equal the specified value (Max. 32 signals)
SWAIT	Suspends program execution until specified condition is set.
EXTCALL	Calls the program selected by external signal.
ON	Sets interruption condition.
ONI	Sets interruption condition.
IGNORE	Cancels ON or ONI instruction.
SCNT	Outputs counter signal at the specified counter value.
SCNTRESET	Clears the counter signal number.
SFLK	Turns ON/OFF the flicker signal in cycle of specified time.
SFLP	Turns ON/OFF signals with SET/RESET signals.
SOUT	Outputs signal when specified condition is set.
STIM	Turns ON timer signal when the specified signal is ON for specified period of time.
SETPICK	Sets the time to start clamp close control. (Option)
SETPLACE	Sets the time to start clamp open control. (Option)
CLAMP	Controls open/close of clamp signals. (Option)
HSENSESET	Starts monitoring of specified sensor signal. (Option)
HSENSE	Reads the data stored in buffer by HSENSESET. (Option)
RSIGPOINT	Sets signal output between motion steps.
RSIGCORRECT	Sets signal output timing of RSIGPOINT instruction.

RESET

Function

Turns OFF all the external output signals.　This command does not have effect on signals used as dedicated signals, clamp signals and antinomy of multifunction OX/WX.

By using the optional setting, the signals used in the Interface Panel screen are not affected by this command. (Option)

SIGNAL　signal number, ……

Function

Turns ON/OFF the specified external output signals (OX) or internal signals.

Parameter

Signal number

Selects the number of external output signal or internal signal.　Selecting a dedicated signal results in error.

Acceptable Signal Numbers

External output signal	1–actual number of signals
Internal signal	2001–2960

See also 5.7 SIGNAL monitor command.

PULSE signal number, time

Function

Turns ON the specified external output signal or internal signal for the given period of time.

Parameter

Signal number

Selects the number of external output signal or internal signal. Selecting a dedicated signal results in error.

Acceptable Signal Numbers

External output signal	1–actual number of signals
Internal signal	2001–2960

Time

Sets for how long the signal is output (in seconds). If not specified, it is automatically set at 0.2 seconds.

See also 5.7 PULSE monitor command

DLYSIG signal number, time

Function

Outputs the specified signal after the given time has passed.

Parameter

Signal number

Selects the number of the external output signal or internal signal. If the signal number is positive, the signal is turned ON; if negative, the signal is turned OFF Selecting a dedicated signal results in error.

Acceptable Signal Numbers

External output signal	1–actual number of signals
Internal signal	2001–2960

Time

Specifies the time to delay the output of the signal in seconds.

See also 5.7 DLYSIG monitor command.

RUNMASK starting signal number, number of signals

Function

Allows signals to be ON only while the program is executing. The signals can be turned ON using the SIGNAL, PULSE or DLYSIG command, but the signal turns OFF when the program execution stops (if this instruction is not used, the signals remain ON once they are turned ON).

Parameter

Starting signal number

Specifies the number of the first external output signal or internal signal in the group of signals to mask. Entering a negative number cancels the mask function for that signal number and the signal does not become OFF when the program stops.

Acceptable Signal Numbers

External output signal	1–actual number of signals
Internal signal	2001–2960

Number of signals

Specifies how many signals are masked. If not specified 1 is assumed.

Explanation

The signals selected by this instruction always turns OFF when the program execution stops. However, dedicated signals are not affected by this instruction.

If the program execution is interrupted, the masked signals turn OFF. When the program is resumed using the CONTINUE command, the signals return to the status they were in when the program was running. The same occurs with DO command or STEP command. (Restarting program via EXECUTE command nullifies the RUNMASK instruction.)

Example

RUNMASK 5,2 Masks the external output signal 5 and the next signal 6, specified by 2 bits. While the program is running these signals can be turned ON by SIGNAL, PULSE, or DLYSIG command. They are turned OFF when the program execution stops.

BITS starting signal number, number of signals = decimal value

Function

Arranges a group of external output signals or internal signals in a binary pattern. The signal states are set ON/OFF according to the binary equivalent of the decimal value specified.

Parameter

Starting signal number

Specifies the first signal to set the signal state.

Acceptable Signal Numbers

External output signal	1–actual number of signals
Internal signal	2001–2960

Number of signals

Specifies the number of signals to be set ON/OFF. The maximum number allowed is 16. To set more than 16 signals, use BITS32 instruction, explained next.

Decimal value

Specifies the decimal value used to set the desired ON/OFF signal states. The decimal value is transformed into binary notation and each bit of the binary value sets the signal state starting from the least significant bit. If the binary notation of this value has more bits than the number of signals, only the state of the given number of signals (starting from the specified signal number) is set and the remaining bits are ignored.

See also 5.7 BITS monitor command.

BITS32 starting signal number, number of signals = value

Function

Arranges a group of external output signals or internal signals in binary pattern. The signal states are set ON/OFF according to the binary equivalent to the specified value.

Parameter

Starting signal number
Specifies the first signal to set the signal state.

Number of signals
Specifies the number of signals to be set ON/OFF. The maximum number allowed is 32.

Decimal value
Specifies the value used to set the desired ON/OFF signal states. The value is transformed into binary notation and each bit of the binary value sets the signal state. The least significant bit corresponds to the smallest signal number, and so on. If the binary notation of this value has more bits than the number of signals, only the state of the given number of signals (starting from the specified signal number) is set and the remaining bits are ignored.

Explanation

Sets (or resets) the signal state of one or more external output signals or internal signals according to the given value.

Acceptable Signal Numbers

External output signal	1−actual number of signals
Internal signal	2001 – 2960

Specifying a signal number greater than the number of signals actually installed results in error. Selecting a dedicated signal also results in error.

See also 5.7 BITS, BITS32 commands.

SWAIT signal number, ……

Function

Waits until the specified external I/O or internal signal meets the set condition.

Parameter

Signal number

Specifies the number of the external I/O or internal signal to monitor. Negative numbers indicate that the conditions are satisfied when the signals are OFF.

Acceptable Signal Numbers

External output signal	1–actual number of signals
Internal signal	2001–2960

Explanation

If all the specified signals meet the set conditions, this instruction is ended and the program executes the next step. If the conditions are not satisfied, the program waits in that step until they are set.

SWAIT instruction in execution can be skipped using CONTINUE NEXT command.

The same result can be gained using the WAIT instruction.

Example

SWAIT 1001,1002	Waits until the external input signals 1001(WX) and 1002 (WX2) are turned ON.
SWAIT 1,-2001	Waits until external output signal 1(OX1) is ON and the internal signal 2001(WX1) is OFF.

EXTCALL

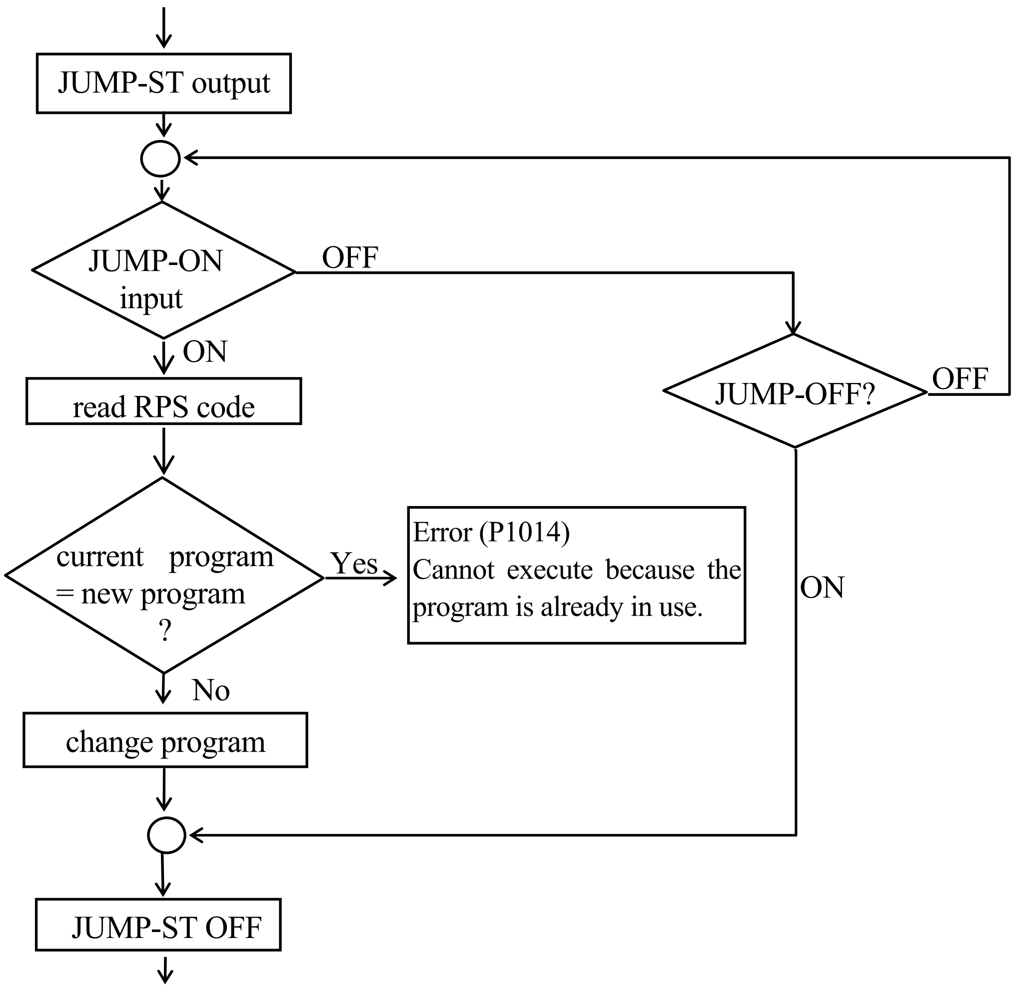
Function

Calls the program selected by the external input signal.

Explanation

EXTCALL instruction is processed as shown in the following procedure:

1. Outputs JUMP-ST signal, allowing input at an external program.
2. Waits for JUMP-ON signal to be input.
3. When JUMP-ON is input, the program number input by RPS-CODE is read. If the number input is 100 or higher, programs pgxxx are called. If the number is 99 – 10, programs pgxx are called and if the number is smaller than 9, pgx.



This instruction can be skipped by entering the CONTINUE NEXT command when waiting for JUMP_ON signal.

[NOTE]

This instruction can be skipped by entering the CONTINUE NEXT command when waiting for JUMP_ON signal.

This instruction is effective only when RPS mode is ON and the RPS signal is set as software dedicated signal. An error occurs if this instruction is executed when RPS is not defined as a dedicated signal.

If RPS mode is OFF, this instruction is ignored.

EXTCALL is used to call a subroutine. After the completion of this subroutine (or when a RETURN instruction is processed in the subroutine), the execution returns to the original program.

ON mode signal number CALL program name, priority
ON mode signal number GOTO label, priority
ONI mode signal number CALL program name, priority
ONI mode signal number GOTO label, priority

Function

Monitors the specified external input signal or internal signal and upon input of the signal, branches to the specified subroutine (CALL) or jumps to the specified label (GOTO).

ONI stops the current motion instruction, while ON waits for the current motion to be completed before jumping to the subroutine or label.

Parameter

Mode

(not specified)	Monitors the rising and trailing edges of the specified signal.
/ERR (option)	Returns an error if the status of the signal already meets the set condition when monitoring starts.
/LVL (option)	Immediately jumps to the specified subroutine or label if the status of the signal already meets the set condition when monitoring starts.

Signal number

Specifies the number of the signal to monitor.

If the number is positive, the rising edge of signal or the change from OFF to ON is monitored. If the number is negative, the trailing edge or the change from ON to OFF is monitored.

Acceptable signal numbers

External input signal	1001–actual number of signals
Internal signal	2001–2256

Program name

Specifies the name of the subroutine to branch to when the specified signal is input. If omitted, the program goes on to the next step in the program and does not branch to a subroutine.

Label

Specifies which label to jump to when the specified signal is input.

Priority
Specifies the priority of the program, setting range: 1 to 127. If not specified, 1 is assumed. The greater the number is, the higher the priority becomes. Priority is ignored when a label is entered as the destination.

Explanation

For ON…CALL instruction, if change is detected in the monitored signal, the program is interrupted and the specified subroutine is executed. This functions the same as CALL instruction after the monitored signal is detected. (See also 6.5 CALL program instruction).

If the RETURN instruction is executed in the called subroutine, the execution returns to the program step after the step that was running before the subroutine was called. (See also 11.3 External Interlock.)

ONI instruction can be used only in robot motion programs and not in PC programs.

Signal monitoring is canceled in any of the following cases:

1. IGNORE instruction is executed for the signal specified in ON and ONI instructions.
2. The ON and ONI instructions are executed and the program has branched to a subroutine.
3. A new ON or ONI instruction specifies the same signal as an earlier ON (ONI) instruction (the older setting is canceled).

[NOTE]

1. When monitoring the rising and trailing edge of the signal, the program branches only when there is a change in the signal state. Therefore, if the leading edge is to be detected, branching does not occur if that signal is already ON when the ON instruction is executed. No branching will occur until the signal is turned OFF then turned ON again.

2. To detect signal changes accurately, the signal must be stable for at least 50 msec.

3. Monitoring starts as soon as the ON (ONI) instruction is executed. Since in the AS system, non-motion instructions are read and executed together with the preceding motion, the monitoring starts at the same time as the motion right before ON (ONI) instruction is executed.

4. The signals are not monitored while the program is not executed.

5. For ON and ONI instruction set in the main program, the signal status is monitored also in the subroutine. However, when the signal signified in the subroutine is input, the execution timing of the interruption process will be right after returning to the main program. To execute the interruption process immediately in the subroutine, set ON and ONI instructions in the subroutine in the same way as in the main program.

6. The robot moves in standard motion type instead of motion type 2 while the signals are monitored by ON instruction. Therefore, the robot motion may differ when monitoring and not monitoring the signals.

Example

ONI -1001 CALL alarm	Monitors external input signal 1001(WX1). As soon as this signal changes from ON to OFF (the signal number is negative so the trailing edge is detected), the motion stops and the program branches to subroutine “alarm”.
ON test CALL delay	Monitors the signal assigned to the variable “test”. If the signal changes as desired (the condition depends on the value of “test”, since it could be negative or positive), the program branches to subroutine “delay” after execution of the current motion step is completed. It returns to the original program when the subroutine “delay” is completed.

IGNORE signal number

Function

Cancels the monitoring of signals set by ON or ONI instruction.

Parameter

Signal numbers

Specifies the number of the signal to cancel monitoring.

Acceptable signal numbers

External input signal	1001–actual number of signals
Internal signal	2001–2960

Explanation

This instruction nullifies the effect of the recent ON or ONI instruction set to the specified signal. (See also 11.3 External Interlock.)

[NOTE]

The ON(ONI) monitoring function is only effective with binary I/O signals actually installed as input signal.

Example

IGNORE 1005 Cancels monitoring of external input signal (Channel 5).

IGNORE test Cancels the monitoring of the signal specified by the value of variable "test".

SCNT counter signal number = count up signal, count down signal, counter clear signal, counter value

Function

Outputs counter signal when the specified counter value is reached.

Parameter

Counter signal number

Specifies the signal number to output. Setting range for counter signal numbers: 3097 to 3128.

Count up signal

Specified by signal number or logical expressions. Each time this signal changes from OFF to ON, the counter counts up by 1.

Count down signals

Specified by signal number or logical expressions. Each time this signal changes from OFF to ON, the counter counts down by 1.

Counter clear signals

Specified by signal number or logical expressions. If this signal is turned ON, the internal counter is reset to 0.

Counter value

When the internal counter reaches this value, the specified counter signal is output. If “0”is given, the signal is turned OFF.

Explanation

If the count up signal changes from OFF to ON when the SCNT command is executed, then the internal counter value increases by 1. If the countdown signal changes from OFF to ON, the internal counter value decreases by 1. When the internal counter value reaches the value specified in the parameter (counter value), the counter signal is output. If the counter clear signal is output, value of the internal counter is set at 0. Each counter signal has its own individual counter value. To force reset of the internal counter to 0, use SCNTRESET command.

To check the states of signals 3001 to 3128, use the IO/E command. (Option)

See 5.7 also SCNT monitor command.

SCNTRESET counter signal number

Function

Resets to 0 the internal counter value corresponding to the specified counter signal.

Parameter

Counter signal number
Selects the number of the counter signal to reset. Setting range for counter signal numbers: 3097 to 3128.

See also 5.7 SCNTRESET monitor command.

SFLK signal number = time

Function

Turns ON/OFF (flicker) the specified signal in specified time cycle.

Parameter

Signal number
Specifies the number of the signal to flicker. Setting range: 3065 to 3096.

Time
Specifies the time to cycle ON/OFF (real values). If a negative value is set, flickering is canceled.

Explanation

The process of ON/ OFF is considered one cycle, and the cycle is executed in the specified time.

See also5.7 SFLK monitor command.

SFLP output signal = set signal expression, reset signal expression

Function

Turns ON/OFF an output signal using a set signal and a reset signal.

Parameter

Output signal

Specifies the number of the signal to output. A positive number turns ON the signal; a negative number turns it OFF. Only output signals can be specified (1 to actual number of signals).

Set signal expression

Specifies the signal number or logical expression to set the output signal.

Reset signal expression

Specifies the signal or logical expression to reset the output signal.

Explanation

If the set signal is ON, the output signal is turned ON. If the reset signal is ON, the output signal is turned OFF. If both the set and reset signal are ON, then the output signal turns OFF. The output signal is turned ON or OFF when the SFLP command is executed, and not when the set signal or the reset signal is turned ON.

See also 5.7 SFLP monitor command.

SOUT signal number = signal expression

Function

Outputs the specified signal when the specified condition is set.

Parameter

Signal number

Specifies the number of the signal to output. Only output signals can be specified (1 to actual number of signals).

Signal expressions

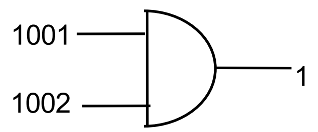
Specifies a signal number or a logical expression.

Explanation

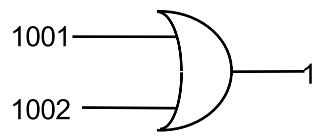
This instruction is for logical calculation of signals. Logical expressions such as AND and OR are used. The specified signal is output when that condition is set. (See also 5.7 SOUT monitor command.)

Example

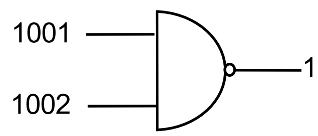
SOUT 1 = 1001 AND 1002



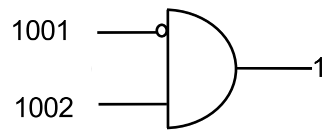
SOUT 1 = 1001 OR 1002



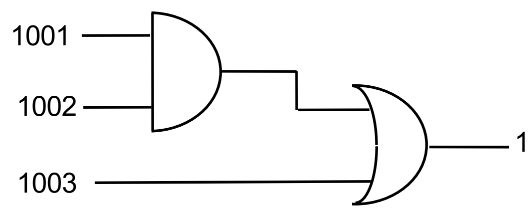
SOUT –1 = 1001 AND 1002
SOUT 1 = NOT(1001 AND 1002)



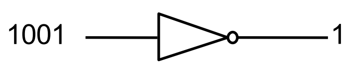
SOUT 1 = –1001 AND 1002



SOUT 1 = (1001 AND 1002) OR 1003



SOUT –1 = 1001 or SOUT 1= –1001 or
SOUT 1 = NOT(1001)



STIM timer signal = input signal number, time

Function

Turns ON the timer signal if the specified input signal is ON for the given time.

Parameter

Timer signal

Selects the signal number to turn ON. Setting range: 3001 to 3064.

Input signal number

Specifies in integers the input signal number or logical expression to monitor as a condition to turn ON the timer signal. The value cannot exceed the number of signals actually installed.

Time

Specifies in real values the time (sec) the input signal is to be ON.

See also 5.7 STIM monitor command.

SETPICK time1, time2, time3, time4, …, time8
SETPLACE time1, time2, time3, time4, …, time8

Option

Function

Sets the time to start clamp close control (SETPICK) or clamp open control (SETPLACE) for each of the 8 clamps.

Parameter

Time 1 to 8

Sets the control time to open/close clamps 1 to 8 in seconds. Setting range: 0.0 to 10.0 seconds.

Explanation

See also CLAMP instruction.

CLAMP clamp number 1, clamp number 2, clamp number 3, clamp number 4, ……, clamp number 8

Option

Function

Outputs clamp signal for opening/closing the hand specified by the parameter clamp number x. The output timing is set by the SETPICK/SETPLACE instruction; i.e. the signal is output x seconds before the current motion is completed.

Parameter

Clamp number 1 to 8
Specifies the clamp number. If the number is positive, the robot hand is opened. If the number is negative, the robot hand is closed.

Explanation

This instruction outputs signals to the control valve to open and close the pneumatic hand. The signal is output immediately if the robot is not in motion, or if the remaining motion time is less than the time set by SETPICK/SETPLACE instructions. The signal is output when the axes coincide if the superposing of the next motion begins before the time set by SETPICK/SETPLACE instructions is reached. If an irrational setting such as “CLAMP 1, -1” is made, the latter clamp number will be valid.

Example

```
12 SETPICK 4, 3, 2, 1
13 SETPLACE 0.2, 0.4, 0.6, 0.8
14 LMOVE a
15 CLAMP -1, 2, 3, -4
```

By executing the above program, the robot will move as follows:

Closes clamp 2, 3 seconds before reaching pose a.
Closes clamp 3, 2 seconds before reaching pose a.
Opens clamp 4, 0.8 seconds before reaching pose a.
Opens clamp 1, 0.2 seconds before reaching pose a.

HSENSESET no. = input signal number, output signal number, signal output delay time

Option

Function

Declares the starting of signal detection to AS system. When this instruction is executed, AS system starts to watch the sensor signal and accumulates the data such as pose, etc., into the buffer memory at signal transaction. The data saved in the buffer memory can be read using HSENSE instruction. Buffer memory can save up to 20 data.

Parameter

No.

Specifies the number for the monitoring results. Up to 2 input signals can be monitored. Instruction for each signal is written as HSENSESET 1 or HSENSESET 2. Acceptable range is 1 or 2.

Input signal number

Set the signal number to monitor. Setting zero (0) terminates the monitoring.

Output signal number

Set the number of the signal to be output after system gets the joint angle. The specified signal turns ON for 0.2 seconds. This may be omitted.

Signal output delay time

Set the time to delay the output of signal after acquiring the pose data. Acceptable range is 0 to 9999 ms. This may be omitted.

[NOTE]

Even when the controller power becomes OFF during watching, buffer memory keeps the read data. It is possible to read the kept data by HSENSE instruction after turning ON the controller power again. However, watching does not restart automatically, so HSENSESET should be executed again.

See also 5.7 HSENSESET monitor command.

Example

HSENSESET 1 = wx_sensor   Starts watching for input signal wx_sensor.

HSENSE no. result variable, signal status variable, pose variable, error variable, memory remainder variable

Option

Function

Reads the data saved in the buffer memory by HSENSESET instruction.

Parameter

No.

Specifies the monitoring number. To read data saved by HSENSESET 1 specify HSENSE 1. To read data saved by HSENSESET 2, specify HSENSE 2.

Result variable

Specifies the name of the real variable to which the watch result is assigned. After executing HSENSE instruction, numerical value is assigned to this variable. Zero (0) is assigned to this variable when AS system does not detect the signal transaction. –1 is assigned to this variable when AS system detects the signal transaction.

Signal status variable

Specifies the name of the real variable to which the status of signal transaction is assigned. After executing HSENSE instruction, a numerical value is assigned to this variable. When the signal(s) is turned from OFF to ON, ON (-1) is assigned. When the signal(s) is turned from ON to OFF, OFF (0) is assigned to this variable.

Pose variable

Specifies the name of the pose variable to which the joint values at time of HSENSE signal input are assigned.

Error variable

Specifies the name of the real variable to which the buffer overflow error result is assigned. When no error occurs, 0 is assigned to this variable. When buffer memory overflows, a numerical value (other than 0) is assigned to this variable. The buffer memory overflows after accumulating data from more than 20 transactions.

Memory remainder variable

Specifies the name of the real variable to which the number of used memory in the buffer is assigned. The value assigned to this variable shows the number of memory in the buffer that is already used. When only one memory is used, 0 will be assigned to the variable. When all the memories are used, the value of the variable will be 19.

Example

In this program, sensor signal wx_sensor is monitored while the robot moves from #p2_1 to #p4_1 and the pose data of JT3 is saved in the array hsens_jt[ ] when the signal is detected. This program uses many local variables (local variable names are written with a period (.) at the beginning of the name).

```
.err = 0                                    ;Initialize
IF SIG(wx_sensor) THEN                      ;When sensor signal keeps ON
  .err=4                                    ;Incorrect starting point
  RETURN
END
;
HSENSESET 1 = wx_sensor                     ;Start watching

JMOVE #p2_1                                 ;Move to starting point
BREAK
SPEED sens_sp
ABS.SPEED ON
JMOVE #p4_1                                 ;Move to finishing point

.num = 0
loop:
HSENSE 1 .stat,hsens_onoff[.num+1],#hsens[.num+1] ,.serr,.rest
IF .serr <> 0 THEN
      .err = 3                              ;Memory buffer over "(HSENS)"
    RETURN
END
IF .stat==ON THEN
  hsens_jt[.num+1] = DEXT(#hsens[.num+1],3)   ;Save pose in Z direction when signal detected
  .num = .num + 1
END
IF .rest GOTO loop                          ;Loop when buffer keeps data
IF DISTANSE(DEST,HERE) > 0.1 GOTO loop
;
HSENSESET 1 = 0                             ;Finish watching
```

RSIGPOINT signal number, output time, distance, correction time

Function

Outputs the specified signals before specified distance from the teaching point of next motion step of this instruction.

Parameter

Signal number

Specifies the external output signal number. Input range: 1 to 960

During instruction execution, if signals other than the signal number specified to output by RSIGRANGE command are specified, or if dedicated signals are specified, an error will occur.

Output time

Specifies the signal output time. (Unit: seconds)

Input range: 0.00 sec to 9.99 sec or -1.00 sec

Pulse output: Specifies between 0.00 sec and 9.99 sec

Level output: -1.00 sec

An error will occur if the output time outside of the input range is specified. When the level output signal is to be turned OFF, execute the step specifying output time as 0.00 sec. Specifying distance from the teaching point can turn the level signal OFF as well. Level signal cannot be turned OFF with the use of SIGNAL related instructions/commands.

Distance

Specifies the distance from the teaching point of next motion instruction to the position where the signals are to be output. (Unit: mm)

Input range: 0 mm to 9999 mm

Omissible. When omitted, the distance will be 0 mm.

Correction time

Using the signal output position set by specified distance as basis, corrects the signal output timing according to this correction time and outputs the signal. (Unit: seconds)

Input range: -9.99 sec to 9.99 sec

Positive: Signal is output ahead of time for the specified time.

Negative: Signal is output with delay for the specified time.

Omissible. When omitted, the correction time will be 0 sec.

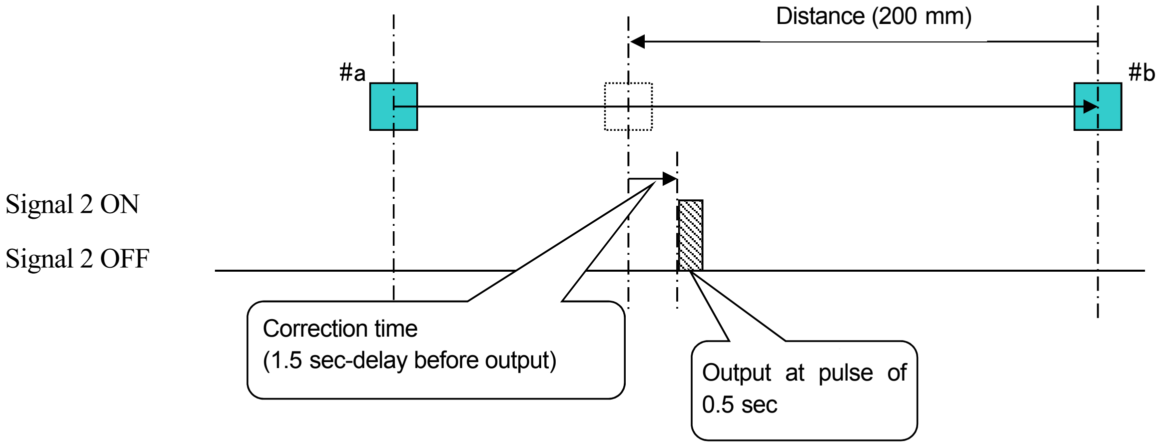
Explanation

Determines the specified position by current instead of command value. Specifying the correction time allows signals to be output ahead or with delay for the specified correction time, using the position of the specified distance as basis. Between the steps of motion instructions, it is possible to use this instruction to teach up to five steps, and to output signals up to five times between two points.

Example

In the following program example, during the linear interpolation operation moving from position #a to position #b, after having reached the position 200 mm before position #b, it delays for 1.5 sec and outputs external output signal 2 at pulse of 0.5 sec.

```
LMOVE #a
RSIGPOINT 2, 0.5, 200, -1.5
LMOVE #b
```



RSIGCORRECT correction time

Function

Advances the output timing of all RSIGPOINT instructions to be executed after the execution of this instruction, by the portion of correction time specified by this instruction.

Parameter

Correction time

Specifies the time to advance the signal output. (Unit: seconds)

Explanation

For example, to correct the time gap between the robot's signal output and the machine's motion, it advances the signal output timing by the portion of correction time only. If the teaching point of the previous step is exceeded in result of the advancement, the teaching point will be the limit. Delaying of timing is not possible.

Input range: 0.00 sec to 9.99 sec

Example

>RSIGCORRECT 1	Advances the output timing of external output signals specified by the subsequent RSIGPOINT instructions by 1 sec.

6.8 Message Control Instructions

PRINT	Displays message on terminal.
TYPE	Displays message on terminal.
PROMPT	Displays message on terminal and waits for input from the keyboard.
IFPWPRINT	Displays specified character string in a display window.
IFPWOVERWRITE	Displays by overwriting character string in a display window.
IFPLABEL	Sets the labels for the icons on interface panel.
IFPTITLE	Sets the title for the specified page of the interface panel.
IFPDISP	Displays the specified page of the interface panel.
SETOUTDA	Sets analog output environment. (Option)
OUTDA	Outputs voltage at set conditions. (Option)

PRINT device number: print data
TYPE device number: print data

Function

Displays on the terminal the print data specified in the parameter.

Parameter

Device number

Select the device to display the data from below:

0: All terminals that are connected
1: Personal computer
2: Teach pendant
3: - 5: Terminals connected via Ethernet

If not specified, the data will be displayed on the currently selected device.

Print data

Select one or more from below. Separate the data with commas when specifying more than one.

(1) Character string	e.g.	"count ="
(2) Real value expressions (the value is calculated and displayed)	e.g.	count
(3) Format information (controls the format of the output message)	e.g.	/D, /S

A blank line is displayed if no parameter is specified.

Explanation

See 5.8 PRINT/TYPE Monitor Command.

[NOTE]

If the MESSAGES switch is OFF, no message appears on the terminal screen.

PROMPT device number: character string, variables

Function

Displays the specified character strings on the terminal followed by the prompt ”>” and waits for input from the keyboard.

Parameter

Device number

Select the device to display the data from below:

1: Personal computer

2: Teach pendant

If not specified, the data will be displayed on the currently selected device.

Character string

Specifies the characters to display on the terminal.

Variables

Specifies to which variable the data input from the keyboard is substituted. It can be a series of real variables or a single string variable.

Explanation

The specified character strings are displayed on the terminal and waits for data and ↵ to be input from the keyboard.

The data input is processed in one of the following ways.

1. When PROMPT is used to ask for values for a series of real variables, the system reads the input line as a series of numbers separated by spaces or commas. Each input number is converted into internal expressions according to its notation, and then they are assigned to the variables one by one.
2. If the number of values input is greater than the number of variables, the excess values are ignored. If the number of values input is less than the number of variables, “0” is assigned to the remaining variables. If data other than numeric values are input, an error occurs, and the program stops execution. To avoid confusion and error, it is advisable that one PROMPT instruction is used to assign one value to one variable.

When using a character string variable as the variable parameter for PROMPT, the characters input are read as a single data unit and all the characters are assigned to the character string variable. At the screen prompt, if only the ↵ key or CTRL + C is pressed, “0” is assigned to real variables, and a null string is assigned to character string variables.

If “2” is entered for device number, the teach pendant screen changes automatically to keyboard screen.

Example

The character string in quotations is displayed on the terminal, and asks for data to be input. When the data (number of parts) is input and the ↵ key is pressed, the value entered is substituted to the variable “part.count”. The program execution then proceeds.

```
PROMPT "Enter the number of parts: ", part.count
```

The instruction below asks for the value of a character string variable. Alphanumeric characters can be input without causing an error.

```
PROMPT "Enter the number of parts: ", $input
```

IFPWPRINT window number, row, column, background color, label color, = "character string", "character string", ……

Option

Function

Displays the specified character string in the string window set in Auxiliary Function 0509 (Interface panel screen).

Parameter

Window number

Corresponds to the window number specified in Auxiliary Function 0509 as the window specification used to display the string. Select from 1to 8 (standard).

Row

Specifies the row in the selected window to display the string. Enter from 1 to 4; available rows depend on the window size. If not specified, 1 is assumed.

Column

Specifies the column in the selected window to display the string. Enter from 1 to 70, though available columns depend on the window size. If not specified, 1 is assumed.

Background color

Selects the background color of the selected window. Colors are numbered from 0 to 15. If not specified, the background is white.

No.	Color	No.	Color	No.	Color	No.	Color
0	Grey	4	Green	8	Pink	12	Navy
1	Blue	5	Pale Blue	9	White	13	Reddish Brown
2	Red	6	Yellow	10	Black	14	Deep Green
3	Orange	7	White	11	Cyan	15	Lavender

Label color

Selects the color of the characters displayed. Colors are numbered from 0 to 15 (See chart above). If not specified, the characters are displayed in black.

Character string

Specifies the character string to display. All strings after the first string are displayed on the next row starting at specified column. Execution of IFWPRINT clears the non-display area in the specified window.

Explanation

IFPWPRINT command can be used only when the interface panel is available for use. If the parameters are not specified, the last setting of that particular window is selected (for first time use, the above default values are set). If the character string does not fit in one row, its display overflows to the next line (indenting to the selected column). Strings that extend beyond the size of the window are not displayed. Control characters in the string are displayed as blanks.

IFPWOVERWRITE mode window number, row, column, background color, label color = "character string", "character string", ……

Function

Displays by overwriting the specified character string in the string window set by Auxiliary Function 0509 (Interface panel screen).

Parameter

Mode

(None) Overwrites the existing character string in unit of line.

/CUT Displays the character string by truncating the characters that do not fit in one line of the string window, without starting a new line. However, if the target window number is not allocated for the interface panel, the character string is saved to the full extent of the window and is displayed when allocated.

/CHAR Overwrites the existing character string in unit of character.
For two-byte characters, as a result of truncation or overwriting, are displayed within the correctly-displayable range.

Window number

Corresponds to the window number specified in Auxiliary Function 0509 as the window specification used to display the string. Select from 1to 8 (standard).

Row

Specifies the row in the window for displaying the string. Acceptable number is from 1 to 4, though it depends on the window size. If not specified, 1 is assumed.

Column

Specifies the column in the window for displaying the string. Acceptable number is from 1 to 78, though it depends on the window size. If not specified, 1 is assumed.

Background color

Selects the color of the background of the selected window. Acceptable numbers are from 0 to 15. If not specified, the background is white.

No.	Color	No.	Color	No.	Color	No.	Color
0	Grey	4	Green	8	Pink	12	Navy
1	Blue	5	Pale Blue	9	White	13	Reddish Brown
2	Red	6	Yellow	10	Black	14	Deep Green
3	Orange	7	White	11	Cyan	15	Lavender

Label color
Selects the color of the characters displayed. Acceptable numbers are from 0 to 15 (See chart above). If not specified, the characters are displayed in black.

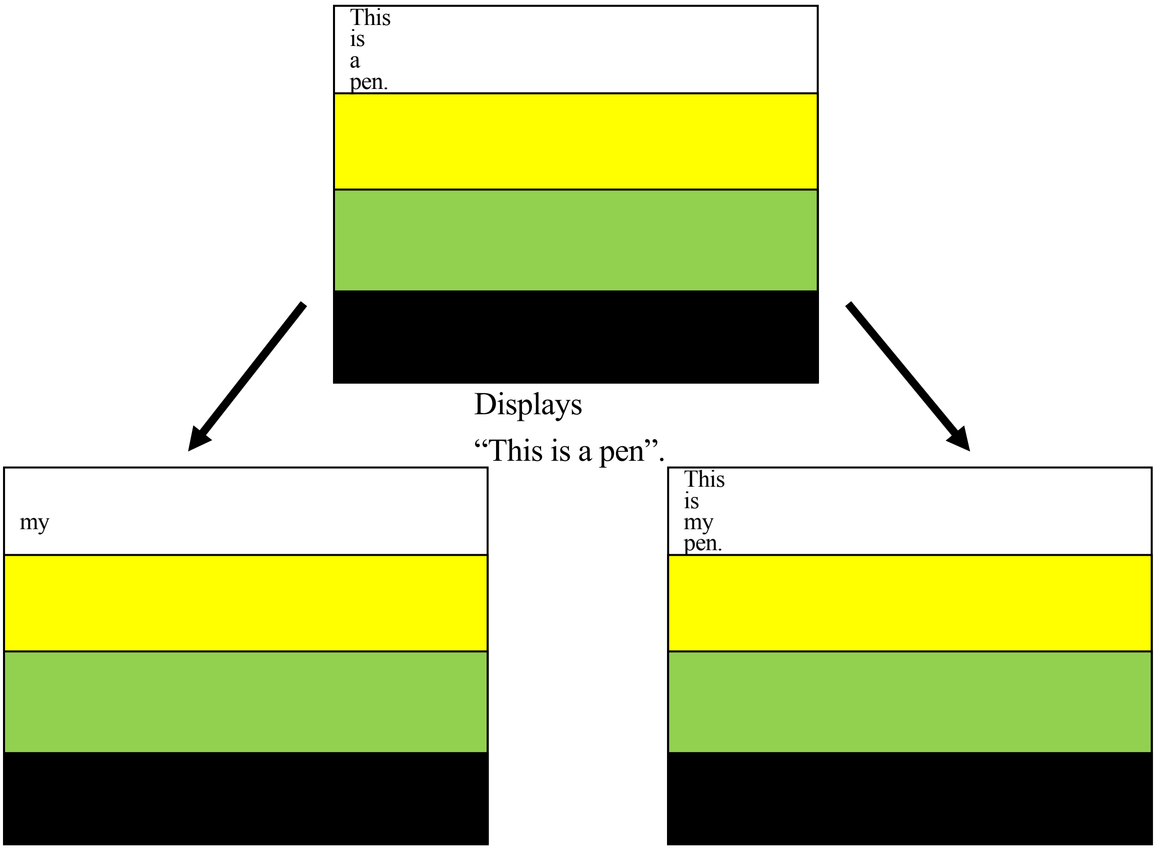
Character string
Specifies the character string to display. All strings after the first string are displayed on the next row starting at specified column.

Explanation

IFPWOVERWRITE command can be used only when the interface panel is available for use. If the parameters are not specified, the last setting of that particular window is selected (for first time use, the above default values are set). If the character string does not fit in one row, its display overflows to the next line (indenting to the selected column). Strings that extend beyond the size of the window are not displayed. Control characters in the string are displayed as blanks. Unlike IFPWPRINT command/ instruction, the rows other than the row specified for display are displayed unchanged as before executing IFPWOVERWRITE.

Example

The figures below show the screens displayed when executing IFPWPRINT and IFPWOVERWRITE command/ instruction from the screen showing “This is a pen”. The left figure is the figure after executing ”IFPWPRINT 1,3,1,7,10=”my”. The characters on lines 1, 2, and 4 disappear and line 3 shows ”my”. On the other hand, the right figure shows the screen after executing “IFPWOVERWRITE 1,3,1,7,10=”my”. The characters on lines 1, 2, and 4 are displayed as they were before executing the instruction and only line 3 has changed to ”my”.



Displays
"This is a pen".

Screen after executing
IFPWPRINT 1,3,1,7,10 ="my"

Screen after executing
IFPWOVERWRITE 1,3,1,7,10 ="my"

IFPLABEL position, "label 1", "label 2", "label 3", "label 4"

Function

Sets and modifies the label of the icon at the specified position on the interface panel.

Parameter

Position

Specifies the display position on the interface panel of the icon to set/ modify the label.
Setting range: 1 – 112.

"Label 1", "Label 2"…

Specifies the character string to display on the interface panel as the label of the specified icon. Omitted label will not be changed.

Explanation

Sets and modifies the label for the icons displayed on the interface panel. When a position with no icon set or when an icon with no label is specified, nothing occurs.

See also 5.8 IFPLABEL monitor command.

IFPTITLE page no., "title"

Function

Sets and modifies the title for the specified page of the interface panel.

Parameter

Page no.

Specifies the page of the interface panel to change the title. Setting range: 1- 4.

"Title"

Specifies the character string to display on the page as the title. Default setting is "Interface Panel". When NULL string ("") is specified, this default setting is also displayed.

Explanation

Sets and modifies the title for the specified page of the interface panel.

See also 5.8 IFPTITLE monitor command.

IFPDISP page no.

Function

Displays the specified page of the interface panel.

Parameter

Page no.

Displays the page number of the interface panel to be displayed.

Explanation

Executing this command allows the display of the specified page of the interface panel.

See also 5.8 IFPDISP monitor command.

SETOUTDA channel No. = LSB, No. of bits, logic, max. voltage, min. voltage

Option

Function

Specifies the analog output environment including: channel number and LSB, number of bits and logic voltage for signal output, maximum and minimum voltage.

Parameter

Channel No.

Sets the analog output channel number. (Setting range: integers between 1 and 16; first 1TW/1UR board: 1 to 4; second 1TW/1UR board: 5 to 8; third 1TW/1UR board: 9 to 12; fourth 1TW/1UR board: 13 to 16)

LSB

Specifies the first analog output signal number for D/A conversion as an integer. Setting range: OUT1 to OUT125, 2001 to 2125, 3000 (first channel of 1TW/1UR board), 3001 (second channel of 1TW/1UR board), and subsequent 1TW/1UR board analog output channels can be specified. Default value is 3000. Previous setting remains in effect if not specified.

Number of bits

Sets the number of bits of analog output signals for D/A conversion as an integer. Setting range: 4 to 16 bits. Sets to 12 bits when 3000 onward [3000 + analog output channel number on 1TW/1UR board] is chosen for parameter LSB above. Default value is 8 bits. Previous setting remains in effect if not specified.

Logic

Sets the logic to either positive (1) or negative (0). Default value is 0 (negative). Previous setting remains in effect if not specified. Please set logic to positive for 1TW/1UR board.

Maximum voltage

Sets the maximum voltage of hardware (D/A output). Setting range: -15.0 to +15.0 V. Unit: V. Default value is 10 V. The value should be rounded off to the first decimal place. Previous setting remains in effect if not specified.

Minimum voltage

Sets the minimum voltage of hardware (D/A output). Setting range: -15.0 to +15.0 V. Unit: V. Default value is 0 V. The value should be rounded off to the first decimal place. Previous setting remains in effect if not specified.

[NOTE]

1. Actual voltage output depends on the hardware used.
2. An error will occur if the value for maximum voltage is set lower than the minimum voltage.

OUTDA voltage, channel no.

Option

Function

Outputs the voltage at set conditions from the specified analog output channel.

Parameter

Voltage

Sets the analog output voltage. Setting range: -15.0 to +15.0 V. Unit: V. The value should be rounded off to the first decimal place.

Channel No.

Specifies the analog output channel number. Setting range: integers between 1 and 16. If not specified, 1 is assumed.

[NOTE]

Confirm that command voltage and actual output voltage are the same by setting output environment to correspond with the hardware settings via SETOUTDA instruction (command).

6.9 Pose Information Instructions

HERE	Assigns the current pose to the specified variable.
POINT	Defines a pose variable.
POINT/X	Sets the X value of a transformation value variable.
POINT/Y	Sets the Y value of a transformation value variable.
POINT/Z	Sets the Z value of a transformation value variable.
POINT/OAT	Sets the OAT values of a transformation value variable.
POINT/O	Sets the O value of a transformation value variable.
POINT/A	Sets the A value of a transformation value variable.
POINT/T	Sets the T value of a transformation value variable.
POINT/7	Sets the value of seventh axis of a transformation value
⋮	
POINT/18	Sets the value of 18th axis of a transformation value variable.
POINT/EXT	Sets the value of external axes (7 – 18th axes) of a transformation value variable.
DECOMPOSE	Assigns the components of pose information to elements of array variable.
BASE	Changes the robots base coordinate system.
TOOL	Defines the pose of TCP.
SET_TOOLSHAPE	Sets tool shape data.
ENA_TOOLSHAPE	Enables/disables speed control by tool shape.
TOOLSHAPE	Sets data for speed control by tool shape.
SETHOME	Sets pose for HOME.
SET2HOME	Sets pose for HOME 2.
LLIMIT	Sets the upper limit of the robot motion.
ULIMIT	Sets the lower limit of the robot motion.
TIMER	Sets the timer.
UTIMER	Sets the timer value of user timer.
ON	Turns ON system switch.
OFF	Turns OFF system switch.
WEIGHT	Sets the weight load data.
MC	Executes monitor commands from PC programs.
TPLIGHT	Turns on teach pendant backlight.

CURLIM	Changes the motor current limit of the external axis.
SETTIME	Sets the time and date.
ENVCHKRATE	Magnification ratio to the threshold value for external axis deviation error.
SETENCTEMP_THRES	Sets the encoder temperature warning and temperature error thresholds

HERE pose variable

Function

Assigns the current pose to the specified variable. The pose may be expressed in transformation values, joint displacement values or compound transformation values.

Parameter

Pose variable
Can be defined in transformation values, joint displacement values, or compound transformation values.

Explanation

Assigns the current pose values to the specified variable in joint displacement values, transformation values, or compound transformation values.

[NOTE]

Only the right most variable in the compound transformation value is defined. If the other variables used in the compound value are not defined, this command results in an error.

See also 5.5 HERE monitor command.

POINT pose variable 1= pose variable 2, joint displacement value variable

Function

Assigns the values of pose variable 2 to pose variable 1.

Parameter

Pose variable 1

Specifies the name of pose variable to be defined (by joint displacement values, transformation values, or compound transformation values). In the case of compound transformation values, POINT specifies the rightmost variable value.

Pose variable 2

Specify the name of variable defined by joint displacement values or transformation values.

Joint displacement value variable

This parameter must be set if the values of pose variable 1 are in joint displacement values and the values of pose variable 2 are in transformation values (if pose variable 1 is not defined by joint displacement values, this parameter cannot be set). The joint displacement values specified here expresses the configuration of the robot at the pose. If not specified, the current configuration is used to define the pose variable.

Explanation

An error is returned if pose variable 2 is not defined.

When pose variable 1 is defined in compound transformation values, the right most variable is defined. If the other variables used in the compound variables are not defined, this command will result in an error.

See also 5.5 POINT monitor command.

POINT/ X transformation value variable 1 = transformation value variable 2
POINT/ Y transformation value variable 1 = transformation value variable 2
POINT/ Z transformation value variable 1 = transformation value variable 2
POINT/ OAT transformation value variable 1 = transformation value variable 2
POINT/ O transformation value variable 1 = transformation value variable 2
POINT/ A transformation value variable 1 = transformation value variable 2
POINT/ T transformation value variable 1 = transformation value variable 2
POINT/ 7 transformation value variable 1 = transformation value variable 2
:
POINT/18 transformation value variable 1 = transformation value variable 2
POINT/EXT transformation value variable 1 = transformation value variable 2

Function

Assigns the specified component(s) of transformation value variable 2 to the corresponding component(s) of transformation value variable 1. The values will be displayed on the terminal for correction.

Parameter

Transformation value variable 1
Specifies a single transformation value variable, or a variable defined by compound transformation values with the last component defined in transformation values.

Transformation value variable 2
Specifies the name of a variable defined by transformation value, compound transformation value or transformation value function. The name of this variable must be defined beforehand.

Explanation

Error occurs if any pose variable on the right side of the "=" is not defined.
If compound transformation value variables are specified for transformation value variable 1, this instruction assigns values to only the rightmost variable. Also, error occurs if any variable other than the rightmost variable is undefined.

Example

POINT/X temp=tempx	Assigns the x component of transformation value variable "tempx" to the x component of transformation value variable "temp".
POINT/EXT temp=tempx	Assigns the external axes components of transformation value variable tempx to the external axes components of transformation value variable temp.

DECOMPOSE array variable [element number] = pose variable

Function

Stores as elements of an array variable, each component of the values of the specified pose variable (X, Y, Z, O, A, T for transformation values; JT1, JT2, JT3, JT4, JT5, JT6 for joint displacement values).

Parameter

Array variable
Specifies the name of the array variable into which the values of each component will be assigned.

Element number
Specifies the first element in which to store the components.

Pose variable
Specifies the name of the pose variable from which to extract each component (transformation values, joint displacement values).

Explanation

This assigns the components of the specified pose information to the elements of the array variable.

In case of transformation values, six elements are assigned from each of the XYZOAT values. In case of joint displacement values, the elements are each assigned from the values of each joint in the robot arm.

Example

DECOMPOSE X[0]=part	Assigns the components of transformation value variable "part" to the first six elements in the array variable "x".
DECOMPOSE angles[4]=#pick	Assigns the components of joint displacement value variable "#pick" to array variable "angles", starting from element number 4.

For example, in the above instruction, if the values of #pick are (10,20,30,40,50,60) then,

angles[4]=10 angles[7]=40
angles[5]=20 angles[8]=50
angles[6]=30 angles[9]=60

BASE transformation value variable

Function

Defines the base transformation values.

Parameter

Transformation value variable

Specifies a transformation value variable or compound transformation value variables. Defines the new base coordinates. The pose variable here describes the pose of the base coordinates with respect to the null base coordinates, expressed in null base coordinates.

Explanation

The CP movement is stopped (BREAK) and the base transformation values are changed to the specified transformation values when this instruction is executed.

See also 5.6 BASE monitor command.

TOOL transformation value variable, tool shape number

Function

Defines the transformation values that shows the pose of the tool tip (tool transformation values) as seen from the tool mounting flange surface (null tool coordinates).

Parameter

Transformation value variable

Specifies a transformation value variable or compound transformation value variable to define the new tool coordinates. The transformation value variable here describes the pose of the tool coordinates with respect to the null tool coordinates, expressed in null tool coordinates. If "NULL" is specified for the parameter, the tool coordinates will be set same as the null tool coordinates.

Tool shape number

Specifies the tool shape to use for speed control in teach and check mode. This may be omitted. If not specified, speed control by tool shape will not be done.

Explanation

The continuous path (CP) movement is stopped (BREAK) and the tool transformation values are changed to the specified transformation values when this instruction is executed.

See also 5.6 TOOL monitor command.

SET_TOOLSHAPE tool shape no. = transformation value variable 1, transformation value variable 2, ..., transformation value variable 8

Function

Registers the tool shape used to control speed in teach mode and check mode.

Parameter

Tool shape no.

Specifies the number of tool shape to register. Setting range: 1 to 9.

Transformation value variables 1-8

Specifies transformation value variables for the points on the tool shape. Maximum of 8 points can be specified. The points are specified in transformation values as seen from the center of the flange surface. However, only the X,Y,Z values of the transformation values are used for the tool shape registration.

Explanation

Defines the tool shape used for speed control in teach and check modes by maximum 8 points specified in pose variables defined by transformation values.

See also 5.6 SET_TOOLSHAPE command.

ENA_TOOLSHAPE tool shape no. = TRUE/ FALSE

Function

Enables/ disables speed control in teach and check mode.

Parameter

Tool shape number

Specifies the tool shape number to set enable/disable in integer from 1 to 9.

TRUE/FALSE

Specify TRUE to enable speed control by the specified tool shape. Specify FALSE to disable the speed control.

Explanation

Selects if speed control in teach and check modes are done by the specified tool shape or not. FALSE is selected for all tool shapes as default setting. If TRUE is selected for a tool shape number with not even one point specified, error E1356 Tool shape not set occurs when the robot is operated in teach or check mode. To avoid this, always set at least one tool point via SET_TOOLSHAPE command or change from TRUE to FALSE via this command an d then execute TOOL or TOOLSHAPE command specifying the relevant tool shape number. (Once set to TRUE, the setting will not be changed to FALSE unless TOOL/TOOLSHAPE command is executed.)

TOOLSHAPE tool shape no.

Function

Selects the tool shape used to control speed in teach mode and check mode.

Parameter

Tool shape no.

Specifies the number of the tool shape used for speed control. Setting range: 1 to 9.

Explanation

To enable speed control in teach mode and check mode, the function must be enabled by ENA_TOOLSHAPE command/ instruction (ENA_TOOLSHAPE n =TRUE). Error E1356 Tool shape not set occurs if a tool shape with no point registered (all points set to 0) is selected.

See also 5.6 TOOLSHAPE command.

SETHOME accuracy, joint displacement value variable
SET2HOME accuracy, joint displacement value variable

Function

Sets HOME pose.

Parameter

Accuracy

Sets the accuracy range of the HOME pose in millimeters.

Joint displacement value variable

Specifies a joint displacement value variable to set the pose for HOME.

Explanation

Sets robot's HOME pose by specifying its joint displacement values (in units of degrees). HOME 1 is set using SETHOME command, HOME 2 using SET2HOME command.

See also 5.6 SETHOME/ SET2HOME command.

Example

SETHOME 10, #place	Records the pose determined by joint displacement value variable #place as HOME. The accuracy is set to the range of within 10 mm of HOME.

ULIMIT joint displacement value variable
LLIMIT joint displacement value variable

Function

Sets and displays the upper/lower limit of the robot motion range.

Parameter

Joint displacement value variable

Specifies a joint displacement value variable for software limit (upper or lower).

Explanation

Sets the upper (lower) limit of the robot motion range in joint displacement values (in degrees).

See also 5.6 ULIMIT/LLIMIT monitor commands.

TIMER timer number = time

Function

Sets the time of the specified timer.

Parameter

Timer number

Specifies the number of the timer to set the time. Acceptable numbers are 1 to 10.

Time

Specifies the time to set to the timer in seconds.

Explanation

When this instruction is executed, the timer is immediately set to the specified time. Check the value of the timer using TIMER function.

Example

The example below holds the program execution for the specified time using the TIMER instruction and TIMER function.

TIMER 1=0	Sets Timer 1 to 0 (seconds).
WAIT TIMER(1)>delay	Waits until the value of Timer 1 is greater than delay.

UTIMER @timer variable = timer value

Function

Sets the default value for the user timer. The user timer can be named freely using the timer variable. More than one user timer can be used at a time.

Parameter

@Timer variable

Specified the variable or array variable name to use as the timer name. Enter @ in front of a integer variable.

Timer value

Specifies the default value for the timer. Acceptable numbers are 0 to 2147483647 (seconds).

switch name, ……ON
switch name, ……OFF

Function

Turns ON (OFF) the specified system switch.

Parameter

Switch name

Specifies the name of the system switch to turn ON (OFF).

Explanation

Turns ON (OFF) the switch specified here. More than one switch name can be entered separating each switch name by commas.

See also 5.6 ON/OFF monitor commands.

The current setting of the switch can be checked using the SWITCH command.

WEIGHT load mass, center of gravity X, center of gravity Y, center of gravity Z, inertia moment ab. X axis, inertia moment ab. Y axis, inertia moment ab. Z axis

Function

Sets the load mass data (weight of the tool and workpiece). The data is used to determine the optimal acceleration/deceleration of the robot arm.

Parameter

Load mass

The mass of the tool and workpiece (in kilograms). Range: 0.0 to the maximum load capacity (kg).

Center of gravity (unit = mm)
X: the x value of the center of gravity in tool coordinates
Y: the y value of the center of gravity in tool coordinates
Z: the z value of the center of gravity in tool coordinates

Inertia moment about X axis, inertia moment about Y axis, inertia moment about Z axis (Option)

Sets the inertia moment around each axis. Unit is kg·m2. The inertia moment about each axis is defined as the moment around the coordinates axes parallel to the null tool coordinates with the center of rotation at the tool's center of gravity.

Explanation

If the parameters are not specified when using WEIGHT as a program instruction, the setting defaults to the maximum load capacity for that robot model.

> **DANGER**
>
> **Always set the correct load mass and center of gravity. Incorrect data may weaken or shorten the longevity of parts or cause overload/deviation errors.**

MC monitor command

Function

Enables execution of monitor commands from PC programs. Monitor commands that can be used with this instruction are: ABORT, CONTINUE, ERESET, EXECUTE, HOLD, and SPEED.

Explanation

This instruction is used in cases when a robot program is executed (EXECUTE command) from program AUTOSTART.PC. MC instruction cannot be used in robot programs.

Example

Robot programs can be executed from AUTOSTART.PC using this command. However, the motor power has to be ON to execute robot programs, so when using MC instruction, add the following steps to check if the power is ON.

```
autostart.pc()
1 10 IF SWITCH(POWER)==FALSE GO TO 10
2 MC EXECUTE pg1
.END
```

TPLIGHT

Function

Turns on the teach pendant backlight.

Explanation

If the backlight of the teach pendant screen is OFF, this instruction turns ON the light. If this instruction is executed when the backlight is ON, the light stays ON for the next 600 seconds.

CURLIM axis number, positive current limit, negative current limit

Function

Changes the motor current limit of the external axis.

Parameter

Axis number

Specifies the number of the external axis. Acceptable range: 7-18.

Positive current limit

Specifies the positive limit of the motor current. The limit is set as percentage of current limit set in the external axis servo parameter. Unit: %. Acceptable range: greater than 0 and less than equal to 100.

Negative current limit

Specifies the negative limit of the motor current. The limit is set as percentage of current limit set in the external axis servo parameter. Unit: %. Acceptable range: greater than 0 and less than equal to 100.

Explanation

This instruction is valid only for external axis with KHI amplifier.

The changed limits are reflected on the current limit values of external axis servo parameters at the following timing:

- When program is executed via EXECUTE command.
- When a new program is reselected and executed via cycle start.
- When program execution is reset.
- When controller power is turned OFF/ON.

SETTIME year, month, day, hour, minute, second

Function

Sets the current time and date.

Parameter

year, month, day, hour, minute, second

Sets the time and date as shown below. Both the date and the time have to be entered as the parameter even when changing only the time. The parameter values must be real values or real variables.

Explanation

Specifying the time and date via this instruction changes the internal calendar of the robot.
Each element for setting the calendar is as below:

Year	(00 - 99)	The last two digits of the year
Month	(01 - 12)	
Day	(01 - 31)	
Hour	(0 - 23)	
Minute	(0 - 59)	
Second	(0 - 59)	

Example

SETTIME 16, 10, 20, 9, 45, 46	Sets the current time to October 20, 2016 9:45:46

ENVCHKRATE　axis number, coefficient

Function

Specifies the magnification ratio to the initial value of the threshold for detection of deviation abnormality in the external axis.

Parameter

Axis number
Specifies the number of the external axis.　Acceptable range: 7-18.

Coefficient
Specifies the desired magnification ratio to the initial value of the deviation abnormality detection threshold.　Acceptable range: 0 - 9.999.　The default value is 1.0.　The deviation abnormality detection threshold is in its initial value when the magnification is set to the default.　Specifying 0 nullifies the deviation abnormality check.

> **⚠ CAUTION**
>
> **When the coefficient is set to greater than 1.0, the deviation error detection may be delayed.　If set to 0, deviation error check is invalidated.　Be careful with the motion of the set axis when setting the coefficient larger than 1.0 or to 0.**

Explanation

This instruction is valid only for external axis with KHI amplifier.　Also, this instruction is executed only in repeat mode.　This instruction is not executed in check mode.

The changed limits return to their initial values at the following timings:

- When program is executed via EXECUTE command.
- When a new program is reselected and executed via cycle start.
- When program execution is reset.
- When the control power is turned OFF then ON.
- When switched to teach mode.

SETENCTEMP_THRES robot number: joint number, warning threshold, error threshold

Function

Performs the threshold setting of encoder temperature warning and temperature error of each joint.

Parameter

Robot No.

Specifies a robot number when one controller is controlling multiple robots.

Joint number

Specifies the joint number to set the encoder temperature warning and temperature error thresholds. Sets encoder temperature warning and temperature error thresholds for all joints by joint when omitted.

Warning threshold

Sets the encoder temperature warning threshold. Warning threshold can be set between 0 and 125ºC. Specify -1 to reset to default value.

Error threshold

Sets the encoder temperature error threshold. Error threshold can be set between 0 and 125ºC. Specify -1 to reset to default value.

Explanation

Allows the setting and resetting of the encoder temperature warning and temperature error thresholds of each joint.

[NOTE]

Encoder temperature warning thresholds and temperature error thresholds set by this command may be reset to default values due to encoder replacement.

Temperature warning: (W1085) “Encoder temperature exceeded limit.(jt XX) (XX deg C)”
Temperature error: (E1564) “Encoder temperature exceeded limit.(jt XX) (XX deg C)”

CAUTION

Do not raise threshold temperatures above default values, as any rise of temperature may affect arm components such as the encoder.

Example

When setting the encoder temperature warning threshold as 80ºC and encoder temperature error threshold as 90ºC for JT2:

>SETENCTEMP_THRES 2,80,90↵

6.10 Program And Data Control Instructions

DELETE	Deletes programs and variables in robot memory.
DELETE/P	Deletes programs in robot memory.
DELETE/L	Deletes pose variables in robot memory.
DELETE/R	Deletes real variables in robot memory.
DELETE/S	Deletes string variables in robot memory.
DELETE/INT	Deletes integer variables in robot memory.
TRACE	Turns ON/OFF the TRACE function.
NLOAD	Reads specified file onto robot memory.
SLOAD	Reads specified file onto robot memory. File is specified in character string.
SHUTDOWN	Executes the data backup to CFast.

DELETE program name, ...
DELETE/P program name, ...
DELETE/L pose variable[array elements] , ...
DELETE/R real variable [array elements] , ...
DELETE/S string variable [array elements] , ...
DELETE/INT string variable [array elements] , ...

Function

Deletes the specified data from the memory.

Parameters

Program name (/P), pose variable (/L), real variable (/R), string variable (/S), integer variable (/INT)
Specifies the type and name of the data to delete.

See also 5.2 DELETE monitor commands.

TRACE stepper number: ON/OFF

Function

Starts (ON) or ends (OFF) logging of robot or PC program to allow program tracing.

Parameters

Stepper number

Specifies the program to log using the following number selection:

1: Robot program
1001: PC program 1
1002: PC program 2
1003: PC program 3
1004: PC program 4
1005: PC program5

If the program is not specified, the program currently in execution is logged.

ON/OFF

Starts/ ends logging.

Explanation

If the necessary memory is not reserved using the SETTRACE command before TRACE ON, the error (P2034) “Memory undefined” occurs. Execute SETTRACE before retrying.

See also 5.2 TRACE/SETTRACE monitor commands.

NLOAD/IF/ARC device number = file name + file name + ···, status variable

Function

Reads the specified file from the USB memory or the CFast stored in the controller and loads it onto robot memory.

Parameter

Device number

Specifies where to display the current processing situation. Specify either 1 or 2:

1: Standard terminal (PC)

2: Teach pendant

When omitted, it is displayed where the program is executed.

File name

Specifies the file to load onto the robot memory. When specifying a folder, specify “folder name¥”before the file name. Indicating “USB1¥” to “USB4¥”in the folder name allows specifying files on the USB memory of USB#1 to USB#4. USB#1 and USB#2 indicate the front of the controller, USB#3 indicates the back and USB#4 indicates the USB slot inside the board. Also, indicating “CFast¥” allows specifying files on the CFast. When loading multiple files, separate the file names using +.

Status variable

Specifying 0 allows continuing program execution even when error occurs. The error code is saved. When omitted, program execution stops at error occurrence.

Explanation

Loads the contents (program and variables) of the specified file onto robot memory.
Specifying a program with the same name as a program in the robot memory will result in such program being overwritten by the loaded program.

1. Specifying NLOAD/IF loads interface panel data.
2. Specifying NLOAD/ARC loads arc welding data.

When there is a step that is not readable in a program being loaded, error message and message “Step format incorrect. (0: Comment-out the step and continue reading, 1: Delete program and terminate) appears. When continuing by selecting zero (0), modify the program via editor after loading is completed.

[NOTE]

Pose variable, real variable, or a string variable with the same name as the variable being read is over-written by the new data without any warning.

Example

NLOAD Pallet	Loads the contents of file pallet.as from USB#1 onto the robot memory (main memory).
NLOAD USB2¥Pallet2	Loads the contents of file pallet2.as from USB#2 onto the robot memory.
NLOAD CFAST¥f3.pg	Loads all the programs in file f3.pg from the CFast onto the robot memory.

SLOAD/IF/ARC device number = character string, status variable

Function

Reads the file name represented by character string from the USB memory or the CFast stored in the controller and loads it onto the robot memory.

Parameter

Device number

Specifies where to display the current processing situation. Specify either 1 or 2:

1: Standard terminal (PC)

2: Teach pendant

When omitted, it is displayed where the program is executed.

Character string

Specifies the file to load onto robot memory. When specifying a folder, specify “folder name¥¥”before the file name. Indicating “USB1¥¥” to “USB4¥¥” in the folder name allows specifying files on the USB memory of USB#1 to USB#4. USB#1 and USB#2 indicate the front of the controller, USB#3 indicates the back and USB#4 indicates the USB slot inside the board. Also, indicating “CFast¥¥” allows specifying files on the CFast. Only one file can be specified.

Status variable

Specifying 0 for this parameter allows continuing program execution even when error occurs. The error code is saved. When omitted, program execution stops at error occurrence.

Explanation

Loads the contents (program and variables) of the specified file onto robot memory.

Specifying a program with the same name as a program already in the robot memory will result in such program being overwritten by the loaded program.

1. Specifying SLOAD/IF loads interface panel data.
2. Specifying SLOAD/ARC loads arc welding data.

When there is a step that is not readable in a program being loaded, error message and message “Step format incorrect. (0: Comment-out the step and continue reading, 1: Delete program and terminate) appears. When continuing by selecting “0”, modify the program via editor after loading is completed.

[NOTE]

Pose variable, real variable, or a string variable with the same name as the variable being read is over-written by the new data without any warning.

Example

SLOAD $str1	When $str1="pallet.as", loads contents of file pallet.as from USB#1.
SLOAD $str2	When $str2="USB2¥¥pallet2.as", loads contents of file pallet2.as from USB#2.
SLOAD $str23	When $str2="CFAST¥¥f3.pg", loads contents of file f3.pg from the CFast.

SHUTDOWN

Function

Executes the data backup to the CFast in the controller.

Explanation

The data backup to the CFast starts when the SHUTDOWN command is executed.
If the data backup is completed normally, the message (D0906) "Data backup to CF is completed. Turn OFF the controller power." appears.
If the data backup is not completed after the specified time (10[sec]), the message (D0907) "Data backup to CF is failed. Turn OFF & ON the controller power." appears.
Executing following operations becomes impossible after the SHUTDOWN command is executed, because the memory of the controller is locked. Turn OFF and ON the controller power.
The SHUTDOWN command cannot be used while a robot motion program is running.
If the SHUTDOWN command is executed while the robot motion program is running, the error (P1012) "Robot is moving now." occurs.

7 AS System Switches

This chapter describes the function of each system switch. For setting ON/OFF or checking the status of switches, refer to the SWITCH, ON and OFF monitor commands/program instructions.

CP	Enables or disables continuous path (CP) function.
CHECK.HOLD	Enables or disables input of commands from the keyboard when HOLD/RUN is in HOLD state.
CYCLE.STOP	Stops cycle with External HOLD.
MESSAGES	Enables or disables terminal output.
OX.PREOUT	Sets the timing of output signal generation.
PREFETCH.SIGINS	Enables or disables early processing of I/O signals in AS programs.
QTOOL	Enables or disables tool transformation during block teaching.
RPS	Enables or disables the random selection of programs.
SCREEN	Control terminal display.
REP_ONCE	Sets if repeat cycle is done once or continuously.
STP_ONCE	Sets the execution as one step at a time or continuous.
AUTOSTART.PC	Enables the PC program to start automatically at controller power ON.
ERRSTART.PC	Determines if the selected PC program is executed when an error occurs.
TRIGGER	Displays the ON/OFF status of TRIGGER switch.
CS	Displays the ON/OFF status of CYCLE START.
POWER	Displays the ON/OFF status of the motor power.
RGSO	Displays the ON/OFF status of servoing.
TEACH_LOCK	Displays the ON/OFF status of TEACH LOCK.
ERROR	Displays if the program is in error status or not.
REPEAT	Displays the status of TEACH/REPEAT switch.
RUN	Displays the current HOLD/RUN state.
PNL_CYCST	Displays the ON/ OFF status of CYCLE START.

PNL_MPOWER	Displays the ON/ OFF status of the motor power.
PNL_ERESET	Displays the ON/ OFF status of ERRORRESET on operation panel.
TPKEY_A	Displays the ON/ OFF status of A key on teach pendant.
DISPIO_01	Changes the display mode of IO command.
HOLD.STEP	Enables display of the step in execution when the program is held. (Option)
WS_COMPOFF	Changes the output timing of WS signal. (Option)
FLOWRATE	Changes from flow rate control mode to speed output mode (and vice versa). (Option)
WS.ZERO	Changes the weld processing that is done when WS=0. (Option)
ABS.SPEED	Enables use of absolute speed.
SLOW_START	Enables or disables the slow start function.
AFTER.WAIT.TMR	Sets how timers begin in block step programs.
STAT_ON_KYBD	Displays status data on the keyboard screen.
WAITREL_AUTO	Displays the wait release popup window.
REP_ONCE.RPS_LAST	Selects the terminating step in repeat once operation.
DEST_CIRINT	Determines the pose where the DEST/#DEST function returns.
PROGDATE	Determines if the date is added to the program information or not.
INSERT_NO_CONFIRM	Determines if confirmation message for inset is displayed or not.
SIGMON_TEACH	Switches between enable/disable of signal operation from signal monitor.
TOUCH.ENA	Switches between enable/disable of touch panel operation of teach pendant repeat condition.
TOUCHST.ENA	Switches between enable/disable of touch panel operation of teach pendant status lamp.
TPSPEED.RESET	Determines if slow speed is set automatically for teach or check speed.
OXZERO	Switches the collective reset function for external output signal (OX).
IFAKEY	In I/F panel, switches the function to allow with A key.
DISP.EXESTEP	Switches the step display at time of program execution.

NO_SJISCONV	Switches enable/disable of the file character code conversion at time of save/load.
CONF_VARIABLE	Determines if configuration change is allowed or not during linear motion.
INVALID.TPKEY_S	Determines whether SHIFT status is invalid when A key is pressed.
DIVIDE.TPKEY_S	Determines whether the information of the two A keys are allocated separately or not.
SIGRSTCONF	Switches the number of signals to reset when signal 0 is output.
SINGULAR	Switches the enable/disable of singular point check function.
USE_ISO8859_5	Switches the ASCII8 bit display font to Cyrillic font.
PCENDMSG_MASK	Switches ON/OFF of message display at the time of PC program completion.
INTERP_FTOOL	Switches between enabling and disabling the function to always select the fixed tool coordinates with the motion coordinates key

CP

Function

Enables or disables continuous path (CP) function.

Explanation

This switch is used to turn ON (enable) or OFF (disable) the CP function. When this switch is changed in a program, the CP function is enabled /disabled starting from the next motion. The default setting for this switch is ON.

Example

`CP OFF` Disables the CP function.

CHECK. HOLD

Function

Enables or disables the use of the keyboard to enter the EXECUTE, DO, STEP, MSTEP, and CONTINUE commands when the HOLD/RUN state is in HOLD.

Explanation

When this switch is ON, the following commands entered from the keyboard are accepted only when HOLD/RUN is in HOLD. (The commands are accepted, but the motion does not start unless HOLD/RUN is changed to RUN.)

EXECUTE, DO, STEP, MSTEP, CONTINUE

When this switch is OFF, the commands above are accepted regardless of the state of HOLD/RUN. Operations not done by keyboard are not affected by this command. (If in RUN state, the robot starts moving as soon as the command is input.)

Default setting is OFF.

CYCLE. STOP

Function

Determines whether or not to continue repeating the execution cycle after an HOLD is applied.

Explanation

When this switch is OFF, the robot stops when the external HOLD signal is input, but CYCLE START remains ON. Therefore, when the external HOLD signal is turned OFF, the robot resumes program execution.

When this switch is ON, the robot stops when the external HOLD signal is input, and CYCLE START is turned OFF. Therefore, even if the external HOLD is released, CYCLE START remains OFF, so the robot does not resume program execution. To resume program execution, follow regular program execution procedures (i.e. press A+CYCLE START on teach pendant).

This switch has effect only on external HOLD and not when HOLD/RUN state is changed to HOLD. If HOLD/RUN is changed to HOLD, robot stops, but CYCLE START remains ON, regardless of the condition of this system switch. The robot resumes program execution when HOLD/RUN is changed to RUN.

Default setting is OFF.

MESSAGES

Function

Enables or disables the output of message on the terminal.

Explanation

When this switch is ON, the messages are displayed on the terminal when the PRINT or TYPE command is used. When this switch is OFF, messages are not displayed on the terminal.

Default setting is ON.

OX. PREOUT

Function

Determines the timing for turning ON/OFF OX signals in block step instruction.

Explanation

When running programs taught by block step instructions and this switch is ON, the OX signal taught for a given pose is turned ON as soon as the robot begins motion toward the pose.

When this switch is OFF, the signal is not turned ON until the axes coincide with the pose taught at that step.

Default setting is ON.

PREFETCH. SIGINS

Function

Enables or disables early processing of signal input and output commands in AS programs.

Explanation

When this switch is OFF, commands for signal input/ output and synchronization listed below are not executed until the axis coincides with the pose taught to the current motion instruction.

SWAIT, SIGNAL, TWAIT, PULSE, DLYSIG, RUNMASK, RESET, BITS

When this switch is ON, all commands including the commands above are executed and the program is processed up to the step before the next motion instruction as soon as the movement towards a taught point starts.

Default setting is OFF.

QTOOL

Function

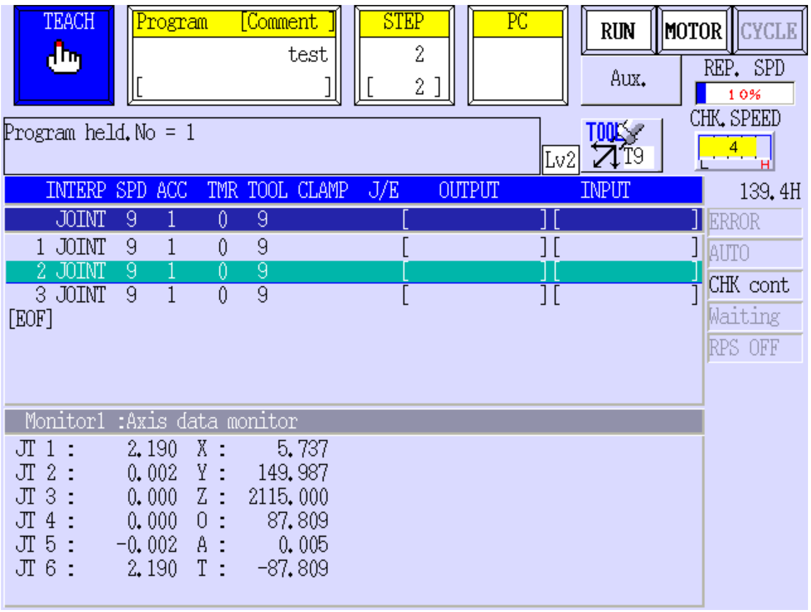
During teach mode, decides whether to convert the tool automatically based on the tool number of the block instruction. When consistently using AS language to create robot program, leave this switch OFF. If this switch is ON, tool settings specified by the TOOL command/instruction are changed automatically to the settings set at Aux. 0304 Tool Coordinates.

Explanation

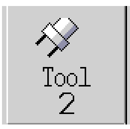
When this switch is ON, the following functions come into effect for the tool coordinates being taught:

1. Automatic selection of the tool coordinates during teaching

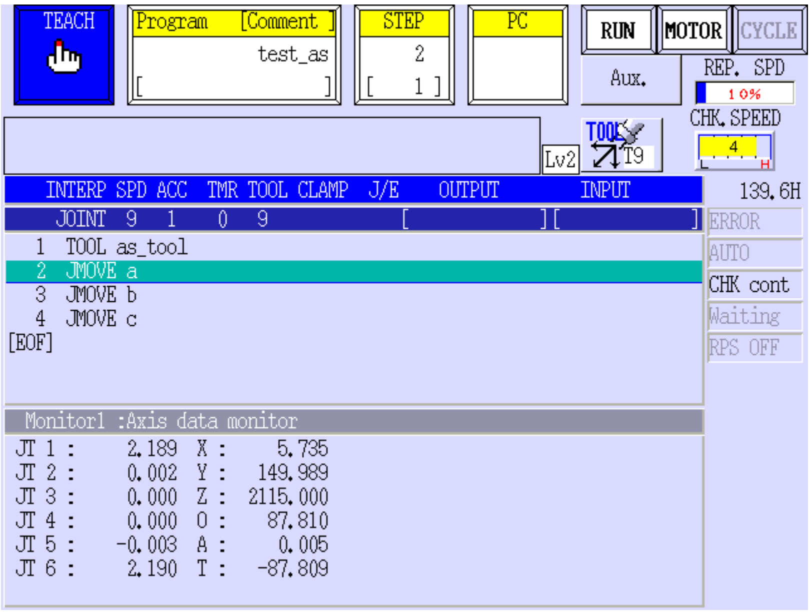
(1) During teaching, if block teaching is being taught at current step, the robot is moved according to the tool data registered for the block teaching tool being taught to that step. For example, when the step that has been taught Tool 9 is selected as below, the robot moves according to the tool data registered for Tool 9.



(2) During teaching, if an AS language instruction is being taught at the current step, the tool changes to the tool specified by the function button below. Therefore, when consistently using AS language to teach, leave this switch OFF.



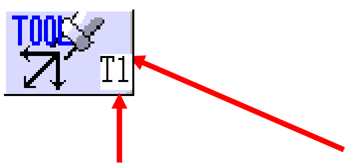
When this switch is ON, even after the TOOL setting is set by the TOOL instruction as below, the tool coordinates are automatically converted to the tool setting specified by the function button if the above function button is operated or if the controller is restarted.

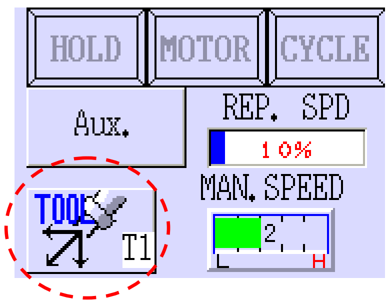


(3) During teaching, if nothing is taught yet at the current step and when the teach pendant is displaying the auxiliary data screen, the robot is moved according to the tool coordinates registered for the block teaching tool currently being shown.

(4) During teaching, if nothing is taught yet at the current step and the teach pendant is not displaying the auxiliary data screen because screen of some other mode is being displayed, the tool coordinates remain unchanged at setting set up to this point.

2. The tool number currently effective is displayed in the status area, on the upper right corner of the teach pendant screen.





The tool number is displayed here after “T”.

When this switch is OFF, the settings will be changed if the TOOL command is executed, or if the tool is newly set by TOOL instruction or block instruction within a program during repeat mode.
Default setting is ON.

RPS

Function

Enables or disables the random selection of programs (JMP, RPS).

Explanation

When this switch is OFF, the EXTCALL instruction and JUMP/ END of the auxiliary data are ignored.

Default setting is OFF.

SCREEN

Function

Enables or disables scrolling of the screen.

Explanation

If the switch is ON, the scrolling of the screen stops when the display is full. Press Spacebar to show the next page.

Default setting is ON.

REP_ONCE

Function

Determines whether the program is run one time or continuously.

Explanation

When this switch is ON, the program runs one time.

When this switch is OFF, the program runs continuously.

Default setting is OFF.

STP_ ONCE

Function

Determines whether the program steps are run one step at a time or continuously.

Explanation

When this switch is ON, the program steps are run one step at a time.

When this switch is OFF, the program runs continuously through all the steps.

Default setting is OFF.

AUTOSTART. PC
AUTOSTART2. PC
AUTOSTART3. PC
AUTOSTART4. PC
AUTOSTART5. PC

Function

Determines if the selected PC program starts automatically when the controller power is turned ON.

Explanation

When this switch is ON, the PC program named AUTOSTART.PC starts automatically when the controller power is turned ON. Program AUTOSTART.PC should be created beforehand. Five different PC programs can be selected to start automatically, each named AUTOSTART.PC and AUTOSTART2.PC to AUTOSTART5.PC. Create a program named accordingly and turn ON each corresponding switch to start the desired program.

Default setting is OFF.

[NOTE]

When this switch is turned ON but the corresponding program does not exist, an error occurs and the message “Program does not exist” appears.

ERRSTART. PC

Option

Function

Determines if the selected PC program is executed or not when an error occurs.

Explanation

When this switch is ON, the PC program named ERRSTART.PC is automatically executed when an error occurs. Create a PC program named ERRSTART.PC in advance.

Default setting is OFF. Once the ERRSTART.PC program is executed, this switch is turned OFF automatically. To reset automatic execution of the ERRSTART.PC program, be sure to set this switch to ON again. Beware that ERRSTART.PC program is executed as the fifth PC program, so if five PC programs are already running, ERRSTART.PC cannot be executed.

[NOTE]

When this switch is turned ON but the corresponding ERRSTART. PC program does not exist, an error occurs and the message “Program does not exist” appears.

SWITCH(TRIGGER)

Function

Displays if TRIGGER (DEADMAN) switch on the teach pendant is ON or OFF.

This function does not turn ON/OFF the switch.

When used with SWITCH function, −1 is returned if the TRIGGER switch is ON, and 0 if it is OFF.

SWITCH (CS)

Function

Displays if CYCLE START is ON or OFF.

This function does not turn ON/OFF the switch.

When used with SWITCH function, −1 is returned if CYCLE START is ON, and 0 if it is OFF.

SWITCH (POWER)

Function

Displays if the motor power is ON or OFF.

This function does not turn ON/OFF the switch.

When used with SWITCH function, −1 is returned if the motor power is ON, and 0 if it is OFF.

SWITCH (RGSO)

Function

Displays if servo motor power is ON or OFF.

This function does not turn ON/OFF the switch.

When used with SWITCH function, −1 is returned if the servo motor power is ON, and 0 if it is OFF.

SWITCH (TEACH_LOCK)

Function

Displays if TEACH LOCK switch is ON or OFF.

This function does not turn ON/OFF the switch.

When used with SWITCH function, −1 is returned if the switch is ON, and 0 if it is OFF.

SWITCH (ERROR)

Function

Displays whether or not an error is currently occurring.

This function does not turn ON/OFF the switch.

When used with SWITCH function, −1 is returned if error is occurring, and 0 if not occurring.

SWITCH (REPEAT)

Function

Displays if TEACH/REPEAT is in TEACH position or in REPEAT position.

This function does not turn ON/OFF the switch.

When used with SWITCH function, −1 is returned if the switch is in REPEAT position, and 0 if it is in TEACH position.

SWITCH (RUN)

Function

Displays if HOLD/ RUN is in RUN state or in HOLD state.

This function does not the state of the switch.

When used with SWITCH function, −1 is returned if the switch is in RUN state, and 0 if it is in HOLD state.

SWITCH (PNL_CYCST)

Function

Displays if CYCLE START is ON or OFF.

The switch cannot be turned ON/OFF with this instruction.

When used with SWITCH function, −1 is returned when the switch is ON, and 0 when it is OFF.

SWITCH (PNL_MPOWER)

Function

Displays if the motor power is ON or not OFF.

The switch cannot be turned ON/OFF with this instruction.

When used with SWITCH function, −1 is returned when the switch is ON, and 0 when it is OFF.

SWITCH (PNL_ERESET)

Function

Displays if ERROR RESET is ON or not (OFF).

The switch cannot be turned ON/OFF with this instruction.

When used with SWITCH function, −1 is returned when the switch is ON, and 0 when it is OFF.

SWITCH (TPKEY_A)

Function

Displays if $\boxed{A}$ switch on the teach pendant is pressed (ON) or not (OFF).

The switch cannot be turned ON/OFF with this instruction.

When used with SWITCH function, −1 is returned when the switch is ON, and 0 when it is OFF.

DISPIO_ 01

Function

Changes how signals (external I/O and internal signals) are displayed with the IO command.

Explanation

If the system switch DISPIO_01 is OFF, “o” is displayed for signals that are ON. “x” is displayed for signals that are OFF. The dedicated signals are displayed in uppercase letters (“O” and “X”).

If the system switch DISPIO_01 is ON, “1” is displayed for the signals that are ON and “0” for those that are OFF. “-” is displayed for external I/O signals that are not installed.

Default setting is OFF. (See also 5.6 IO monitor command.)

Example

When DISPIO_01 is OFF

```
>IO ↵
  32 -    1     xxxx   xxxx   xxxx   xxXX   xxxx   XXXX   XXXO   XXXO
1032 - 1001     xxxx   xxxx   xxxx   xxXX   xxxx   XXXX   XXXX   OXXX
2032 - 2001     xxxx   xxxx   xxxx   xxxx   xxxx   xxxx   xxxx   xxxx
>
```

When DISPIO_01 is ON

```
>IO ↵
32-     0       0000   0000   0000   0000   0000   0000   0001   0001
1032-1001       0000   0000   0000   0000   0000   0000   0000   1000
2032-2001       0000   0000   0000   0000   0000   0000   0000   0000
>
```

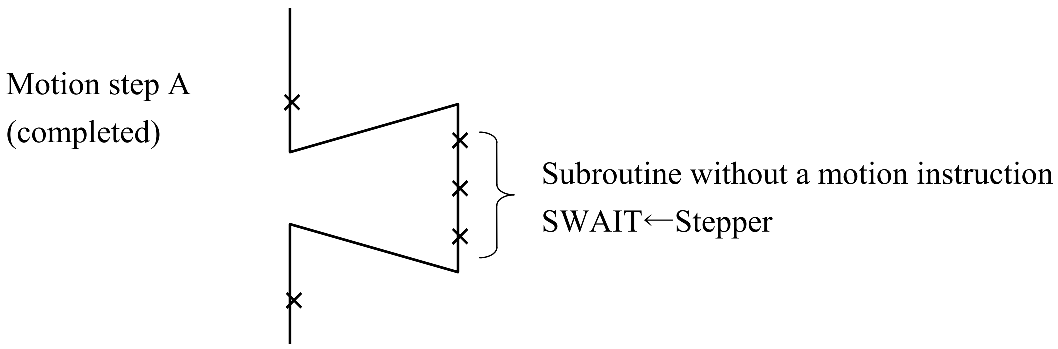
HOLD.STEP

Function

Selects the step to display when the program execution is held.

Explanation

When this switch is ON and program execution is held during execution of a non-motion step, the step in the currently executed stepper is displayed instead of the motion instruction just completed. For example in the situation below, if the switch is ON, the stepper step of SWAIT is displayed (SWAIT can be skipped). If the switch is OFF, the motion step A is displayed.



Default setting is OFF.

WS_COMPOFF

Option

Function

Changes the output condition of the welding signal (WS).

Explanation

When this switch is ON, WS signal is output from the moment memory change occurs until the welding complete signal is input.

When this switch is OFF, WS signal is output from the moment memory change occurs until the next memory change.

Default setting is OFF.

FLOWRATE

Option

Function

Switches between flow rate control mode and speed output mode.

Explanation

When this switch is ON, the flow rate control mode is selected.

When this switch is OFF, the speed output mode is selected.

Default setting is OFF.

WS.ZERO

Option

Function

Changes the operation done when the WS signal is 0.

Explanation

When this switch is ON, the welding is done when WS=0, as well as when WS≠0.
(Pressurizing and welding)

When this switch is OFF, welding is not done when WS=0. (Pressurizing only)

Default setting is OFF.

ABS.SPEED

Function

Enables or disables the use of absolute speed. This function enables execution of motion steps at a low, pre-defined speed setting, taking precedence over the monitor speed setting.

Explanation

When this switch is ON, the robot moves at the absolute speed specified for the program when the below condition is true.

Maximum speed × Monitor Speed > Program Speed

Also, when this switch is ON, the acceleration speed stays at the speed equivalent to monitor speed 100%, independent from the change in monitor speed.

Default setting is OFF.

Example

If Max speed 2400 mm/s, Monitor Speed 10%, Program Speed 100 mm/s, then

$2400 \times 0.1 > 100$

The robot moves at the program speed 100 mm/s.

If Max speed 100%, Monitor Speed 10%, Program Speed 5%, then

$100 \times 0.1 > 5$

The robot moves at the program speed 5%.

If Max speed 2400 mm/s, Monitor Speed 2%, Program Speed 100 mm/s, then

$2400 \times 0.02 < 100$

The robot moves at the monitor speed 48 mm/s.

SLOW_START

Function

Enables (ON) or disables (OFF) the slow start function. If the slow start function is enabled, the robot starts motion at slow repeat speed in the first motion step or during a specified time at the beginning of the motion.

Explanation

When this switch is turned ON, the slow start function is enabled in repeat mode. The slow start function will not be enabled in teach or check mode.

Set the speed in slow repeat mode using SLOW_REPEAT command or Aux. 0508 Slow Repeat.

Also, in Aux. 0508, the time to operate in slow speed can be set (slow speed repeat time at startup). When slow speed repeat time at startup is set as 0, only the first motion step is executed in slow speed.

If a program is stopped and then restarted, the first motion step to be executed will start at the slow repeat speed.

Default setting is OFF.

AFTER.WAIT.TMR

Function

Sets the timing for starting timers in block instructions.

Explanation

When this switch is OFF, the timer starts when the axes coincide. If it is ON, the timer starts when the axes coincide and all the set conditions (e.g. WX, WAIT, RPS ON) are fulfilled.

Default setting is OFF.

STAT_ON_KYBD

Function

Sets if the status information is displayed on the keyboard screen or not.

Explanation

When this switch is OFF, the status information is not displayed and the keyboard screen is displayed at full-screen. If it is ON, the status information is displayed on the top part of the keyboard screen, as in the teach screen. The keyboard screen will be smaller by 4 lines than the full-screen display.

Default setting is ON.

WAITREL_AUTO

Function

Sets if the wait release popup window is displayed or not when the robot comes to a wait in teach or check mode.

Explanation

When this switch is ON, the popup window appears when the robot is in wait status. If it is OFF, the popup window is not displayed.

Default setting is ON.

REP_ONCE.RPS_LAST

Function

Sets on which step to end the program execution when the program is repeated once via RPS function.

Explanation

When this switch is ON with REP_ONCE switch ON (repeat once) the program is repeated once and then ends execution with the program that the END step is taught, without moving on to the next program.

If it is OFF, the program moves on to the next program after executing the END step and stops at the first step of that program.

Default setting is OFF.

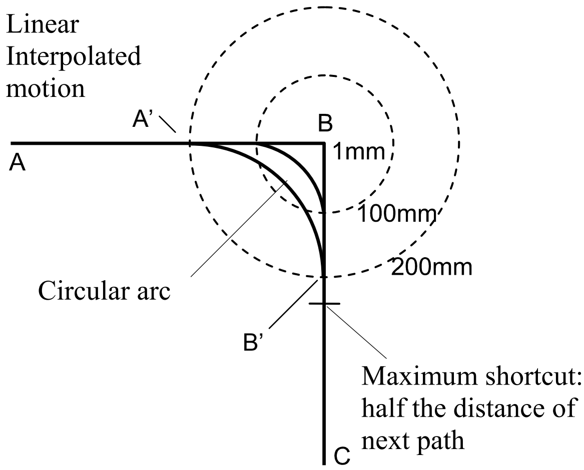
DEST_CIRINT

Function

Determines the pose where the DEST/#DEST function returns on the circular trajectory within the accuracy circle in Motion type 2.

Explanation

The robot's motion trajectory when moving in linear interpolation motion along A→B→C in motion type 2 will be as shown in the figure below. This figure shows the example when the accuracy setting at point B is 1 mm, 100 mm, 200 mm. The robot starts to shortcut to the next path when as soon as it enters the accuracy circle of the set accuracy. The robot will follow a circular trajectory within the accuracy circle. See also "4.5.4 Relation between CP Switch and ACCURACY, ACCEL, and DECEL Instructions".



When DEST/#DEST functions are used in the following two timing, the returned pose can be changed by setting this switch ON or OFF.

1. When the robot is between point A and the point starting circular motion
2. When the robot is following the circular trajectory (within the accuracy circle)

When this switch is ON, at accuracy setting of 200 mm, point A' is returned at timing 1, and point B' is returned for timing 2.

When this switch is OFF, at accuracy setting of 200 mm, point B is returned at timing 1, and point C is returned for timing 2.

Default setting is ON.

PROG.DATE

Function

Determines if the date of program modification is added to the program information or not.

Explanation

When this switch is ON, an item selection button is displayed in the program list screen. Pressing this button will display the total number of execution times and the program modification date.

When this switch is OFF, the item selection button is not displayed on the program list screen. The number of steps in the program and its comment are displayed instead of the program list.

Default setting is OFF.

INSERT_NO_CONFIRM

Function

Selects whether confirmation message is displayed or not when insertion operation is done in teach operation.

Explanation

When this switch is ON, the confirmation message is not displayed when insertion operation is done in teach operation.

When this switch is OFF, the confirmation message is displayed when insertion operation is done in teach operation.

Default setting is OFF.

SIGMON_TEACH

Function

Selects whether operation of hard key in O signal. I/O name and KLogic monitor is nullify or not in teach repeat or teach lock disabled.

Explanation

When this switch is ON, signal operation from monitor screen is not allowed when repeat mode is enabled or teach lock is disabled. Operation can be done only when in teach/check mode or when teach lock is valid.

When this switch is OFF, the robot can be operated both in repeat mode and teach mode.

Default setting is OFF.

TOUCH.ENA

Function

Selects whether touch panel operation of the repeat conditions are enabled or disabled. When set to disable, avoids mistaken operation of the touch panel when the touch panel is touched accidently.

Explanation

When this switch is ON, all the items of repeat conditions can be operated from the touch panel.

When this switch is OFF, touch panel operation is disable except for “speed specification”, “▲+10%”, “▲-10%”. However operations using the cursor keys are possible.

Default setting is ON.

TOUCHST.ENA

Function

Selects whether touch panel operation of status lamps (HOLD/RUN, Motor power, Cycle start) are enabled or disabled. When set to disable, avoids mistaken operation of the touch panel when the touch panel is touched accidently.

Explanation

When this switch is ON, status lamp can be operated from touch panel.

When this switch is OFF, status lamp cannot be operated from touch panel.

Default setting is ON.

TPSPEED.RESET

Function

Selects whether teach speed and check speed is set to slow speed automatically or not.

The timing slow speed is set is as follows:
(1) When switched from repeat mode to teach mode
(2) When Controller power is turned ON
(3) When emergency stop switch is pressed
Sow speed is not set when in inching operation.

Explanation

When this switch is ON, teach and check speeds are automatically changed to slow speed.

When this switch is OFF, teach and check speeds are not automatically changed to slow speed.

Default setting is ON.

OXZERO

Option

Function

Selects whether external output signal (OX) collective reset function is enabled or disabled. When the function is enabled, “OX=0” can be taught and the OX signals of the taught step are reset together.

However, dedicated signals and interface panel signals cannot be reset.

This function is valid only when multi-function OXWX function option is valid.

Explanation

When this switch is ON, collective reset function is enabled.

When this switch is OFF, collective reset function is disabled.

Default setting is OFF.

IFAKEY

Function

Selects whether the press buttons on the interface screen and press buttons with lamp are enabled only with A key or not. Buttons in operation disable status and those set to operation disables cannot be operated.

Explanation

When this switch is ON, allows operation only when pressed together with A key.

When this switch is OFF, allows operation without pressing A key.

Default setting is OFF.

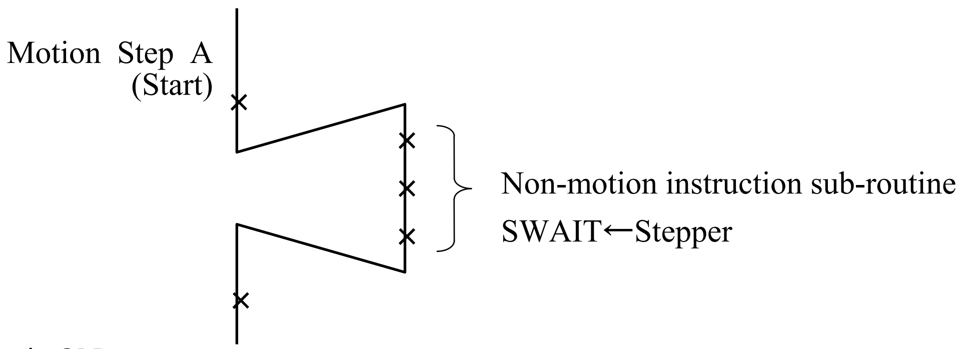
DISP.EXESTEP

Function

Displays the current stepper of the execution step during program execution instead of current motion step.

Explanation

For example, if the program is executed in condition as below, when this switch is ON, the stepper step of SWAIT is displayed. (SWAIT can be skipped.) When this switch is OFF, motion instruction A is displayed.



Default setting is ON.

NO_SJISCONV

Function

Selects whether the function to convert the character code between SJIS and EUC at time save/ load is enabled or disabled.

Explanation

When this switch is ON, character code conversion is conducted.

When this switch is OFF, character code conversion is not conducted.

Default setting is ON.

CONF_VARIABLE

Function

Selects whether the configuration change is allowed at time of linear motion or not. Normally, the configuration is kept the same during the linear motion, but when the robot passes a singular point, some axes change greatly in the command value and robot may not be able to move. Configuration can be changed to avoid this and pass the singular point.

Explanation

When this switch is ON, allows configuration during linear motion.

When this switch is OFF, does not allow configuration during linear motion.

Default setting is ON.

INVALID.TPKEY_S

Function

Selects whether or not the shift status is invalidated when A key is pressed.

Explanation

When this switch is ON, pressing A key only turns ON “TPKEY_A” and “TPKEY_S” remains OFF.

When this switch is OFF, pressing A key turns ON both “TPKEY_A” and “TPKEY_S”.

Default setting is OFF.

DIVIDE.TPKEY_S

Function

Selects whether the information of the two A keys on the teach pendant are allocated to switches “TPKEY_A” and “TPKEY_S” or not.

Explanation

When this switch is ON, the information of the two A keys are allocated to the switches as shown below:

Left A key: TPKEY_S
Right A key: TPKEY_A

When INVALID.TPKEY_S switch is ON, TPKEY_S switch is invalidated.

When INVALID.TPKEY_S switch is the information from the two A keys are united to one and allocated to both TPKEY_A and TPKEY_S switches.

Default setting is OFF.

SIGRSTCONF

Function

Changes the number of signals that are reset when signal 0 is output via SIGNAL instruction or manual signal output.

Explanation

When this switch is ON, outputting signal 0 resets 1 to the number of external output signals set by ZSIGSPEC instruction (maximum of 960).

When this switch is OFF, outputting signal 0 resets 1 to 64 signals (1 to 256 when OX/WX signal expansion function is ON).

Default setting is ON.

SINGULAR

Function

Switches between singular point check function enable and disable.

Explanation

When this switch is ON, checks whether the taught point or the command values for JT 5 are within range of 0º to the threshold values (for example 5º) when moving in linear or circular motion. If the value is within the range, error (E6007) “Wrist can't be straightened any more (Singular point 1).” occurs.

Default setting is ON (in some model the default setting is OFF).

USE_ISO8859_5

Function

Changes the display font of ASCII8 bit from Latin font to Cyrillic font.

Explanation

When this switch is OFF, ASCII8 bit (0xA1 - 0xFF) is displayed in Latin font (ISO8859-1). When this switch is ON, the font is displayed in Cyrillic font (ISO8859-5).

Default setting is ON.

PCENDMSG_MASK

Function

Switches the ON/OFF of the message display at the time of the PC program completion.

Explanation

When this switch is OFF, a message “PC program completed.” appears at the time of the PC program completion. When this switch is ON, the message does not appear.

Default setting is OFF.

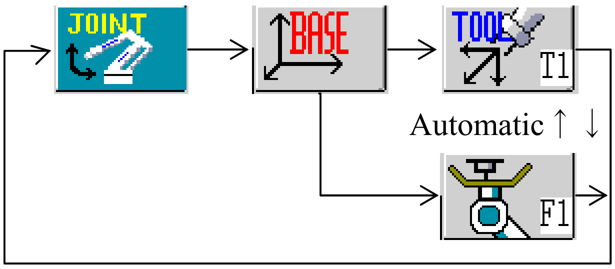
INTERP_FTOOL

Function

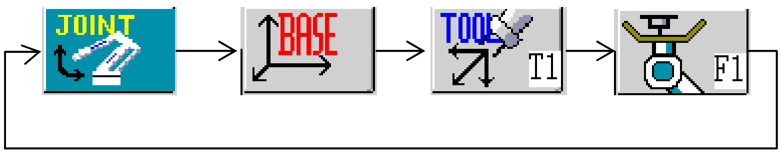
Switches between enabling and disabling the function to always select the fixed tool coordinates with the motion coordinates key.

Explanation

When this switch is disabled (OFF), the motion coordinates key switches to three levels by the pushing of the key button. TOOL or FTOOL is automatically determined by the interpolation type.



When this switch is enabled (ON), the motion coordinates key switches to four levels by the pushing of the key button.



This switch is off upon initialization.

8 Operators

This chapter describes how the operators function in AS language. These operators are used in conjunction with monitor commands and program instructions.

8.1 Arithmetic Operators

Arithmetic operators are used to perform general mathematic calculations.

Operator	Function	Example
+	Addition	i = i + 1
–	Subtraction	j= i – 1
*	Multiplication	i = i * 3
/	Division	i = i/2
MOD	Remainder	i= i MOD 2
^	Power	i = i ^ 3

Example

i = i + 1 The value of i plus 1 is assigned to i; e.g. when i is 5, 6 will be assigned to i as the result of the expression i + 1.

i = i MOD 2 When i is 5, operator calculates 5 ÷ 2 and assigns to i the remainder of 1.

i = i ˆ 3 The value of i^3 is assigned to i. When i is 2, 8 is assigned to i on the left side of the instruction.

In division (/) and MOD, using 0 as the rightmost value of the instruction results in an error.

Example
i = i/0
i = i MOD 0 etc.

8.2 Relational Operators

Relational operators are used with instructions such as IF and WAIT to verify if a condition is set.

Operator	Function	Example
<	TRUE (−1) when left side value is less than right side	i < j
>	TRUE (−1) when left side value is greater than right side	i > j
<=	TRUE (−1) when left side value is less than or equal to right side	i<= j
=<	Same as above	i =< j
>=	TRUE (−1) when left side value is greater than or equal to right side	i >= j
=>	Same as above	i=> j
==	TRUE (−1) when the two sides are equal	i == j
<>	TRUE (−1) when the two sides are not equal	i <> j

Example

IF i <j GOTO 10 — When j is greater than i, (i.e. the instruction i<j is true), the program jumps to the step labeled 10. If not, the program proceeds to the next step.

WAIT INT(t)==INT(5) — When t is 5 (i.e. t==5 is true), the program proceeds to the next step, if not, program execution is suspended until the condition is set.

IF i+j>100 GOTO 20 — When i + j is greater than 100 (i.e. the expression i + j > 100 is true), the program jumps to the step labeled 20. If not, the program proceeds to the next step.

IF $a =="abc" GOTO 20 — When $a is “abc” (i.e. $a == “abc” is true), the program jumps to the step labeled 20. If not, the program proceeds to the next step.

[NOTE]

When comparing real values, do not use “==”. In the AS software, real values are treated as decimal floating point. In decimal floating points, real values cannot be checked is they are equal using “==”. To compare real values and integer values, use INT or ROUND functions.

8.3 Logical Operators

Logical operators are used in Boolean operations such as 0 + 1 = 1, 1 + 1 = 1, 0 + 0 = 0 (logical OR), or 0 × 1 = 0, 1 × 1 = 1, 0 × 0 = 0 (logical AND). There are two types of logical operators in AS language, logical operators and binary operators.

Logical operators are not used for calculating numeric values, but for determining if a value or an expression is true or false. If a numeric value is 0, it is considered FALSE (OFF). All nonzero values are considered TRUE (ON). Take note that the calculation using this operator returns −1 as TRUE.

Operator	Function	Example
AND	Logical AND	i AND j
OR	Logical OR	i OR j
XOR	Exclusive logical OR	i XOR j
NOT	Logical complement	NOT i

Example

i AND j — Evaluates the logical AND between i and j. The variables i and j are generally logical values, but they can also be real number values. In this case, all real number values other than 0 are considered ON (TRUE).

i	j	Result
0	0	0 (OFF)
0	not 0	0 (OFF)
not 0	0	0 (OFF)
not 0	not 0	−1 (ON)

The result is ON (TRUE) only when both values are ON (TRUE).

i OR j — Evaluates the logical OR between i and j.

i	j	Result
0	0	0 (OFF)
0	not 0	−1 (ON)
not 0	0	−1 (ON)
not 0	not 0	−1 (ON)

The result is ON (TRUE) when both or either of the two values are ON (TRUE).

i XOR j Evaluates the exclusive logical OR between i and j.

i	j	Result
0	0	0 (OFF)
0	not 0	−1 (ON)
not 0	0	−1 (ON)
not 0	not 0	0 (OFF)

The result is ON (TRUE) when only one of the two values is ON (TRUE).

NOT i Evaluates the logical complement of i.

i	Result
0	−1 (ON)
not 0	0 (OFF)

In AS, the logical status of a value or expression may be expressed as following:

True: not 0, ON, TRUE
False: 0, OFF, FALSE

8.4 Binary Operators

Binary logical operators perform logical operations for each respective bit of two numeric values. For example, if a number is composed of 4 bits, the values that will be calculated will be 0000, 0001, 0010, 0011, ……, 1111 (In AS, the numeric values are composed of 32 bits).

Binary expression	Decimal expression
0000	0
0001	1
0010	2
0011	3
⋮	⋮
1111	15

Operator	Function	Example
BOR	Binary OR	i BOR j
BAND	Binary AND	i BAND j
BXOR	Binary XOR	i BXOR j
COM	Binary complement	COM i

Example

i BOR j — If i=5, j=9, then the result is 13.

```
i=5   0101
j=9   1001
      ----
      1101 ⇒13
```

i BAND j — If i=5, j=9, then the result is 1.

```
i=5   0101
j=9   1001
      ----
      0001 ⇒  1
```

i BXOR j — If i=5, j=9, then the result is 12.

```
i=5   0101
j=9   1001
      ----
      1100 ⇒ 12
```

COM i — If i=5 then the result is −6.

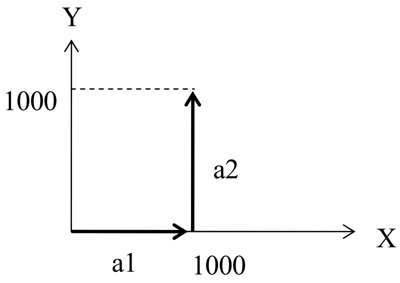
```
i=5 0…  0101
    1…  1010 ⇒ −6
```

8.5 Transformation Value Operators

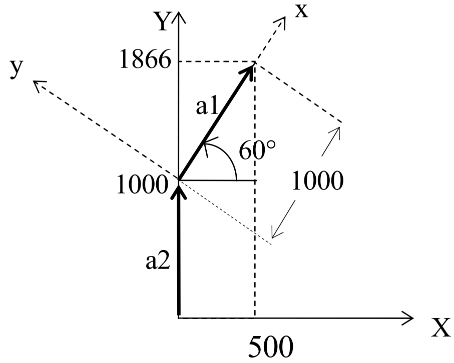
In the AS system, operators + and – are used to determine the compound transformation values (the XYZOAT values). However note that unlike the usual addition or subtraction, the commutative law does not hold true with the transformation operation. Arithmetic expression "a + b" and "b + a" will result the same, but "pose a + pose b" will not necessarily equal "pose b + pose a". This is because in transformation operations, the values of the axes are taken into consideration. An example of this is shown below:

a1 = (1000, 0, 0, 0, 0, 0)
a2 = (0, 1000, 0, 60, 0, 0)

a1 + a2 = (1000, 1000, 0, 60, 0, 0)



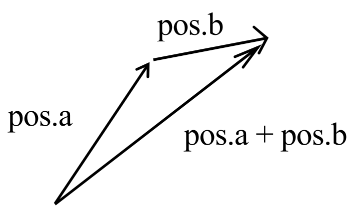
a2 + a1 = (500, 1866, 0, 60, 0, 0)



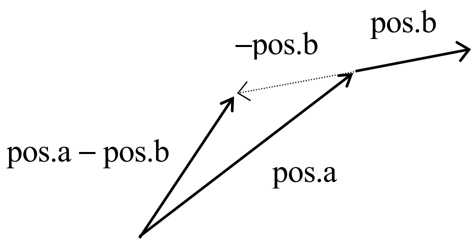
Operator	Function	Example
+	Addition of two transformation values	pos.a + pos.b
–	Subtraction of two transformation values	pos.a – pos.b

Example

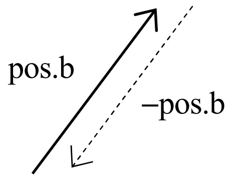
pos.a + pos.b



pos.a – pos.b



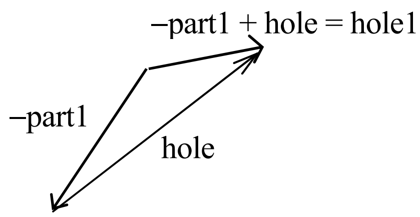
Transformation operator "–" used with a single value (e.g. –x) signifies the inverse value of x. For example, when the transformation value variable pos.b defines the pose of object B relative to object A, then –pos.b defines the pose of object A relative to object B.



In the example below, "hole1" is to be defined relative to "part1" (defined in advance). This can be done using the compound transformation value variable part1+hole.

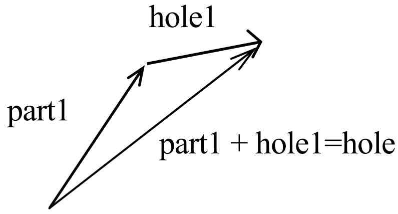
Move the robot to the pose to be defined "hole1" and teach that pose as "hole" using the HERE command. Using this pose ("hole"), "hole1" can be defined.

POINT hole1 = – (part1) + hole



Another way to define "hole1" without using the operator "–" is by writing "hole1" in the left side of the expression in POINT command. The following command also defines "hole1".

POINT part1 + hole1 = hole



8.6 String Operators

Operator	Function	Example
+	Combines two strings	$a = $b + $c

Example

$a=$b + $c Combines $b + $c and assigns that string to $a.

For example, when $b="abc" and $c="123", $a becomes "abc123".

9 Functions

This chapter describes the functions used in AS system.　Functions are generally used in combination with monitor commands and program instructions.　They are expressed in format described below.　The keyword specifies the function, and the parameters entered in parentheses determine the value.

EXAMPLE

Keyword　　　Parameter

SIG (signal number, ……)

Parameters marked by can be omitted.

Always enter a space (blank) after the keyword.

9.1 Real Value Functions

SIG	Returns the logical AND of the specified signal.
BITS	Returns the value corresponding to the bit pattern of the signal (Up to 16 signals).
BITS32	Returns the value corresponding to the bit pattern of the signal (Up to 32 signals).
TIMER	Returns the current value of the timer.
DISTANCE	Returns the distance between two points.
DX, DY, DZ	Returns the displacement value of transformation values. (X, Y, Z)
DEXT	Returns the specified component of a pose.
ASC	Returns ASCII character values.
LEN	Returns string length.
TRUE, ON	Returns the value –1.0, which represents TRUE.
FALSE, OFF	Returns the value 0.0, which represents FALSE.
VAL	Returns the real value in the given string.
INSTR	Returns the value for the starting point of the specified string.
MAXVAL	Compares given values.
MINVAL	Compares given values.
INT	Returns the integer of a value.
MAXINDEX	Returns the value of the largest element in specified dimension.
MININDEX	Returns the value of the smallest element in specified dimension.
SWITCH	Returns the status of system switch.
WHICHTASK	Returns the task number selected by the program or subroutine.
TASK	Returns the status of program execution.
ERROR	Returns the error number.
PRIORITY	Returns the priority of robot program.
UTIMER	Returns the current timer value.
MSPEED	Returns monitor speed of Robot 1.
MSPEED2	Returns monitor speed of Robot 2.
INRANGE	Checks if pose is in motion range.

SYSDATA	Returns AS parameters.
EXISTDATA	Returns if the variable or the program exists or not.
EXISTJOINT	Returns if the specified joint displacement value variable exists or not.
EXISTTRANS	Returns if the specified pose transformation value variable exists or not.
EXISTREAL	Returns if the specified real value variable exists or not.
EXISTCHAR	Returns if the specified character string variable exists or not.
EXISTINTEGER	Returns if the specified integer variable exists or not.
EXISTPGM	Returns if the specified program exists or not.
EXISTLOCALJOINT	Returns if the specified local joint displacement value variable exists or not.
EXISTLOCALTRANS	Returns if the specified local pose transformation value variable exists or not.
EXISTLOCALREAL	Returns if the specified local real value variable exists or not.
EXISTLOCALCHAR	Returns if the specified local character string variable exists or not.
EXISTLOCALINTEGER	Returns if the specified local integer variable exists or not.
STRTOPOS	Returns the pose data for the specified string variable.
STRTOVAL	Returns the real value for the specified string variable.
ROUND	Returns the real value rounded at the first decimal place.
IQARM	Acquires the current value of the specified axis.
TRQNM	Acquires the torque value of the specified axis.
CURLIMM	Acquires the negative limit of external axis current limit
CURLIMP	Acquires the positive limit of external axis current limit
ENVCHKRATE	Acquires the set value for magnification ratio to the initial value of deviation error detection
GETENCTEMP	Specifies the encoder temperature [°C] of the specified axis.
OPEINFO	Returns the operating data corresponding to the data number.
INS_POWER	Returns the instantaneous power corresponding to the power number.

SIG (signal number, ……)

Function

Returns the logical AND of the specified binary signal status.

Parameter

Signal Number

Specifies the number of the external or internal I/O signal.

Explanation

Calculates logical AND of all the specified binary signal states and returns the resulting value. If all the specified signal states are TRUE, −1 (the value of TRUE) is returned. Otherwise 0 (the value of FALSE) is returned. External I/O signals or internal I/O signals as shown below, are specified by their numbers.

Acceptable Signal Numbers

External output signal	1–actual number of signals
External input signal	1001–actual number of signals
Internal signal	2001–2960

Signals specified by positive numbers are considered TRUE when they are ON, while signals specified by negative number are considered TRUE when they are OFF. No signal corresponds with signal number “0”, so it is considered always TRUE.

[NOTE]

There is a timing restriction when evaluating more than one signal at the same time. When more than one signal is input at the same time, note that there is approx. 2 ms difference in stabilization time of each signal.

Example

If the binary I/O signals 1001=ON, 1004=OFF, 20=OFF, then

	Results	
SIG(1001)	−1	TRUE
SIG(1004)	0	FALSE
SIG(−1004)	−1	TRUE
SIG(1001,1004)	0	FALSE
SIG(1001,−1004)	−1	TRUE
SIG(1001,−1004,−20)	−1	TRUE

BITS (starting signal number, number of signals)

Function

Reads consecutive binary signals and returns the decimal value corresponding to the bit patterns of the specified binary signals.

Parameter

Starting signal number
Specifies the first signal to read.

Number of signals
Specifies the number of signals to read. The maximum number accepted is 16. To read more than 16 signals, use BIT32 function, explained next.

Explanation

This function returns the decimal value corresponding to the bit pattern of the specified signals. In the binary expression of the value returned by this function, the least significant bit corresponds to the starting signal number.

Acceptable Signal Numbers

External output signal	1−actual number of signals
External input signal	1001−actual number of signals
Internal signal	2001−2960

[NOTE]

There is a timing restriction when evaluating more than one signal at the same time. When more than one signal is input at the same time, note that there is approx. 2 ms difference in stabilization time of each signal.

Example

If the signal states are as follows, the result of the expression below will be 5.

```
x= BITS(1003,4)
```

The logical values of 4 signals starting from 1003 (i.e. 1003, 1004, 1005 and 1006) are read as a binary representation of the value, and x is 5 or 0101.

Signals:	1008	1007	<u>1006</u>	<u>1005</u>	<u>1004</u>	<u>1003</u>	1002	1001
State:	1	1	0	1	0	1	1	0

BITS32 (starting signal number, number of signals)

Function

Reads consecutive binary signals and returns the decimal value corresponding to the bit patterns of the specified binary signals.

Parameter

Starting signal number
Specifies the first signal to read.

Number of signals
Specifies the number of signals to read. The maximum number accepted is 32.

Explanation

This function reads the signal status of signal numbers specified from the starting signal number, and corresponding to the bit value arranged in ascending order, returns as the integer variable in decimals. In the binary expression of the returned value, the least significant bit corresponds to the starting signal number. When assigning the returned value to a variable, add @ to the front of the variable name to make it an integer variable name.

Example

@x=BITS32(2001, 32) is interpreted as the binary representation of the value with 32 bits starting from 2001 (or signals from 2001 to 2032). The first 32-bit bit represents the sign in two's complement, so the returned value may be negative. Please refer to the below table for values returned for signal statuses.

Internal signals 2032-2001								Returned value
0000	0000	0000	0000	0000	0000	0000	0000	0
0000	0000	0000	0000	0000	0000	0000	0001	1
0000	0000	0000	0000	0000	0000	0000	0011	3
0111	1111	1111	1111	1111	1111	1111	1111	2147483647
1000	0000	0000	0000	0000	0000	0000	0000	-2147483648
1111	1111	1111	1111	1111	1111	1111	1111	-1
1111	1111	1111	1111	1111	1111	1111	1101	-3

TIMER (timer number)

Function

Returns the value of the specified timer in seconds. Expresses timer value of the moment this TIMER function was executed.

Parameter

Timer number
Specifies which timer to read. Acceptable numbers are 0 to10.

Explanation

By using the TIMER function the timer value can be read at any time. Read and returns the time (in seconds) elapsed from the value previously set by the TIMER instruction. If no value has been set by the TIMER instruction, the value of TIMER 0 is returned.

Timer number 0 is for the system clock. The value returned by specifying this timer is the time elapsed since the system start up.

Example

In the below example, TIMER instruction and real value function is used to measure the execution time of a subroutine.

```
TIMER (1)=0                                   Sets Timer 1 to 0.
CALL  test.routine                            Calls the subroutine.
PRINT "Elapsed time=", TIMER(1),"seconds"
```

DISTANCE (transformation value variable 1, transformation value variable 2)

Function

Calculates the distance between two poses that are expressed in transformation values.

Parameter

Transformation values variable 1, transformation values variable 2

Specifies names of the two transformation value variables of which the distance between them is to be calculated.

Explanation

Returns the distance between two poses in millimeters. (The two poses can be entered in any order.)

Example

k=DISTANCE(HERE,part)	Calculates the distance between the current TCP and the pose “part”, and substitutes the result into k.

DX (transformation value variable)
DY (transformation value variable)
DZ (transformation value variable)

Function

Returns the transformation values (X, Y, Z) of the position defined by the specified pose variable.

Parameter

Transformation value variable
Specifies the name of the transformation value variable whose X, Y, or Z component is required.

Explanation

These three functions each returns the X, Y, or Z component of the specified pose.

[NOTE]

Each component of the transformation values can also be obtained using the DECOMPOSE instruction. The values for O, A, and T are obtained using the DECOMPOSE instruction.

Example

If the pose "start" has the transformation values of:

X	Y	Z	O	A	T
125,	250,	–50,	135,	50,	75

x=DX(start)	DX function returns x = 125.00
y=DY(start)	DY function returns y = 250.00
Z=DZ(start)	DZ function returns z = –50.00

DEXT (pose variable, element number)

Function

Returns the specified element of the specified pose.

Parameter

Pose variable

Specifies the name of pose variable defined by joint displacement values or transformation values.

Element number

Specifies the element to be returned in real numbers, as shown in the figure below.

Element number	Pose	
	Transformation values	Joint displacement values
1	X component	JT1
2	Y component	JT2
3	Z component	JT3
4	O component	JT4
5	A component	JT5
6	T component	JT6
7	JT7	JT7

Example

If the transformation values for "aa" are 0, 0, 0, –160, 0, 0, 300, then inputting this function as:

```
type DEXT(aa, 7)
```

This returns 300, the value of JT7.

ASC (string, character number)

Function

Returns the ASCII value of the specified character in a string expression.

Parameter

String

Specifies the string that contains the character for which the ASCII value is required. If the string is a null string, or the number specified for the parameter "character number" exceeds the actual number of characters in the string, −1 is returned.

Character number

Specifies the number of the character counting from the beginning of the string. If not specified, or if 0 or 1 is specified, ASCII value of the first character of the string is returned.

Explanation

The ASCII value is returned in real values.

Example

ASC("sample", 2)	Returns the ASCII value for character "a".
ASC($name)	Returns the ASCII value for the first character of string "$name".
ASC($system, i)	Returns the ASCII value of the ith character in the string variable "$system".

LEN (string)

Function

Returns the number of characters in the specified string.

Parameter

String

Specifies a character string, character string variable, or string expression.

Example

LEN("sample")	Returns the number of characters of the string "sample", which is 6.

. . . **TRUE**. . .
. . . **ON**. . .

Function

Returns the logical value for TRUE (−1).

Explanation

This function is convenient when it is necessary to specify the logical condition TRUE.

The results of functions TRUE and ON are the same. (Choose the function that best fits the needs of the program.)

. . . **FALSE**. . .
. . . **OFF**. . .

Function

Returns the logical value for FALSE (0).

Explanation

This function is convenient when it is necessary to specify the logical condition FALSE.

The results of functions FALSE and OFF are the same. (Choose the function that best fits the needs of the program.)

VAL (string, code)

Function

Returns the real value in the specified string.

Parameter

String

Specifies character string, character string variable, or string expression.

Code

Expressed in real value or expression, specifies the notation of the value returned. If not specified, or if number other than 0, 1, or 2 is specified, 0 (decimal notation) is assumed.

Numbers	Notation
0	Decimal
1	Binary
2	Hexadecimal

[NOTE]

Scientific notation can be used in the string.

Codes that specify the notation (e.g. ^B and ^H) can be added to the beginning of the string.

All characters not read as a numeric value or code for notation are interpreted as characters marking the end of the string.

Example

VAL("123 ")	Returns the real value 123.
VAL("123abc ")	Returns the real value 123.
VAL("12 ab 3 ")	Returns the real value 12.
VAL("1.2E-5")	Returns the real value 0.00001.
VAL("^HFF")	Returns the real value 255. (^H means hexadecimal notation. 16×15+15=255)

INSTR (starting point, string 1, string 2)

Function

Returns the place (in real value) where the specified string starts in the given string.

Parameter

Starting point

Specifies from where in string 1 to search for string 2. If not specified, the search starts from the beginning of string 1.

String 1

Expressed in character string, character string variable, or string expression, specifies the string where string 2 is searched.

String 2

Expressed in character string, character string variable, or string expression, specifies the string to search for. If a null string is specified, the value of the starting point is returned. 1 is returned if this string is not specified.

Explanation

This function returns the value of the starting point of string 2 in string 1, if string 2 is included in string 1.

The value 0 is returned if string 2 is not included in string 1.

If the value specified as the starting point is equal to or smaller than 1, the search starts from the beginning of string 1. If the value of the starting point is larger than the number of characters in string 1, 0 is returned.

Lower and upper case letters are not differentiated.

Example

INSTR("file.ext",".")	Real value 5 is returned.
INSTR("file",".")	Real value 0 is returned.
INSTR("abcdefgh","DE")	Real value 4 is returned.
INSTR(5,"1-2-3-4","-")	Real value 6 is returned.

MAXVAL (real value 1, real value 2, ……)

Function

Compares the given real values and returns the largest among them.

Parameter

Real value 1, real value 2

Specifies the real values to compare.

MINVAL (real value 1, real value 2, ……)

Function

Compares the given real values and returns the smallest among them.

Parameter

Real value 1, real value 2

Specifies the real values to compare.

INT (numeric expression)

Function

Returns the integer of the specified numeric expression.

Parameter

Numeric expression

Returns the integer of the value.

Explanation

Returns the integer (i.e. left side of the decimal point if the value is not in scientific notation). The negative sign remains with the integer unless the integer is 0.

Example

INT(0.123)	0 is returned.
INT(10.8)	10 is returned.
INT(-5.462)	–5 is returned.
INT(1.3125E+2)	131 is returned.
INT(cost+0.5)	The value of “cost” rounded down to the nearest integer is returned.

MAXINDEX (string variable, dimension number)

Function

Returns the value of the largest element in the specified dimension number of an array.

Parameter

String variable

Specifies the name of the array variable.

Dimension number

Specifies the dimension number. (1-3)

If the value is not specified between one and three, an error occurs. If not specified, 1 is assumed.

Explanation

Returns the value of the largest element in the specified dimension number of the array if the array has been already defined. Returns –1 if the variable is not an array. Returns –2 if the variable has not been defined.

Example

Variable #pos is expressed by joint displacement values and is an one-dimensional array from #pos[0] to #pos[100]. The dimension number is omitted.

ret= MAXINDEX (“#pos”) The value for variable ret is 100.

Variable #place is expressed by joint displacement values and is a two-dimensional array from #place[1,1] to #place[1,5].

ret=MAXINDEX (“#place”, 2) The value for variable ret is 5.

Variable #place is expressed by joint displacement values and is a three-dimensional array from #place[2,1,10] to #place[2,1,20].

ret=MAXINDEX (“#place”, 3) The value for variable ret is 20.

This program displays transformation values for variable pos. pos is a one-dimensional array.

```
.PROGRAM index()
max = MAXINDEX("pos",1)
min = MININDEX("pos",1)
FOR j = min TO max
        $val = "pos"+"["+$ENCODE(/I2,j)+"]"
        ret = EXISTTRANS($val)
        IF ret==FALSE THEN
                GOTO continue
        END
        DECOMPOSE x[0] = pos[j]
        TYPE "pos[",j,"] =",/F8.3,x[0],/X3,/F8.3,x[1],/X3,/F8.3,x[2]
continue:
END
.END
```

MINDEX (string variable, dimension number)

Function

Returns the value of the smallest element in the specified dimension number of an array.

Parameter

String variable

Specifies the name of the array variable.

Dimension number

Specifies the dimension number. (1-3)

If the value is not specified between one and three, an error occurs. If not specified, 1 is assumed.

Explanation

Returns the value of the smallest element in the specified dimension number of the array if the array has been already defined.

Returns –1 if the variable is not arrays.

Returns –2 if the variable has not been defined.

Example

Variable #pos is expressed by joint displacement values and is a one-dimensional array from #pos[0] to #pos[100].

ret=MININDEX("#pos", 1) The value for variable ret is 0.

Variable #place is expressed by joint displacement values and is a two-dimensional array from #place[1,1] to #place[1,5].

ret=MININDEX("#place", 2) The value for variable ret is 1.

Variable #place is expressed by joint displacement values and is a three-dimensional array from #place[2,1,10] to #place[2,1,20].

ret=MININDEX("#place", 3) The value for variable ret is 10.

SWITCH (switch name)

Function

Returns the current condition of the specified system switch.

Explanation

−1 is returned if the switch is ON, 0 is returned if it is OFF.

WHICHTASK program name

Function

Returns the task number selected by the specified program (subroutine).

Parameter

Program name

Specifies the name of the program or subroutine in form of string variable.

Explanation

Returns the task number in real values.

1: Robot program (Robot 1)
2: Robot program (Robot 2)

1001: PC program 1	1004: PC program 4
1002: PC program 2	1005: PC program 5
1003: PC program 3	

-1: Executed task does not exist.

Example

task_no=WHICHTASK($pg_name)	Stores the tasks number in variable if the task selected by program $pg_name exists. If it does not exists, task_no= -1.

TASK (task number)

Function

Returns the execution status of the program specified by the task number.

Parameter

Task number

1: Robot 1

2: Robot 2

1001: PC program 1 1004: PC program 4

1002: PC program 2 1005: PC program 5

1003: PC program 3

Explanation

This function returns execution status of a program. For example, this function can be used to monitor the execution status of a PC program from a robot control program. Then the condition of robot operation can be set according to the status of the PC program.

The values returned by this function are:

0	Not in execution
1	Program running.
2	Program execution held.
3	Execution of the stepper completed; waiting for the completion of robot motion.

ERROR

Function

Returns the error code of the current error.

Explanation

Returns the error code when an error is currently occurring. The value 0 is returned when no error is occurring.

Reread the error number as below:

-4xxxx: Dxxxx
-3xxxx: Exxxx
-2xxxx: Wxxxx
-1xxxx: Pxxxx

When -41500 is returned, error D1500 “Encoder misread error. JtXX” is displayed.

Example

type $ERROR (ERROR)	TYPE instruction displays the error message of the error code returned by the function ERROR.

PRIORITY

Function

Returns the priority number of the current robot program.

Explanation

Returns the priority number (in real value) of robot program currently selected on the stack. There is no priority setting among PC programs.

The default value for robot program priority is 0. The priority number can be changed via LOCK instruction.

UTIMER (@timer variable)

Function

Returns the current value of the @timer variable set by UTIMER instruction.

Parameter

@timer variable

Specifies the name of the variable set by UTIMER instruction. An @ sign is added to the beginning of the variable name so that a integer variable can be specified.

MSPEED
MSPEED2

Function

Returns the current monitor speed (0 to 100%).

MSPEED is for Robot 1 and MSPEED2, for Robot 2.

INRANGE (pose variable 1, joint displacement value variable)

Function

Checks if a pose is within the robot's motion range and returns a value depending on the result of this check (see the table below).

Parameter

Pose variable 1

Specifies which pose to check. (Joint displacement values, transformation values, or compound transformation values).

Joint displacement value variable

Specify a pose defined by joint displacement values. This parameter is entered only when the specified pose variable 1 is defined by transformation values or compound transformation values. The robot configuration is calculated by the pose variable 2 defined by joint displacement values. If not specified, the current configuration is used.

Explanation

The values returned by this function are as follows:

0	Out of motion range.
1	JT1 is out of motion range.
2	JT2 is out of motion range.
4	JT3 is out of motion range.
8	JT4 is out of motion range.
16	JT5 is out of motion range.
32	JT6 is out of motion range.
16384	Beyond the collision check range.
32768	Out of reach of robot arm.

[NOTE]

This function checks if the pose is in the motion range but does not check if the path to that pose is within the motion range.

Example

```
IF INRANGE(pos1, #p) GOTO ERR STOP
:
ERR_STOP:
TYPE "pose pos1is out of motion range."
PAUSE
```

Jumps to label ERR_STOP if pose pos1 is out of motion range, displays the message, and stops.

SYSDATA (keyword, opt1, opt2)

Function

Returns specified parameters in the AS system according to the given keyword.

Parameter

Keyword, opt1, opt2

M.SPEED

Returns monitor speed (in percentage). If no motion step is being executed, −1 is returned.

Opt 1: Robot number (1 to number of robots). If not entered, 1 is assumed.

Opt 2: Not used.

MSTEP

Returns the step number of the motion step in execution or the last executed motion step in the program in execution. If no such step exists, −1 is returned.

Opt 1: Robot number (1 to number of robot). If not entered, 1 is assumed.

Opt 2: Not used.

STEP

Returns the step number of the motion step in execution or the last executed motion step in the program in execution. If no such step exists, −1 is returned.

Opt 1: Robot number (1 to number of robot) or PC task number (1001 to number of PC programs). If not entered, 1 is assumed.

Opt 2: Not used.

P.SPEED

Returns the motion speed (in percentage) of the current motion or the next motion executed. If the speed is set in seconds, −1 is returned.

Opt 1: Robot number (1 to number of robot). If not entered, 1 is assumed.

Opt 2: Not used.

P.SPEED.M

Returns the motion speed (in MM/S) of the current motion or the next motion executed. If the speed is set in seconds, −1 is returned.

Opt 1: Robot number (1 to number of robot). If not entered, 1 is assumed.

Opt 2: Not used.

P.ACCEL

Returns the acceleration (in percentage) of the motion step in execution or the last executed motion step in the program in execution. If the motion speed is set in time (unit: S), –1 is returned.

Opt 1: Robot number (1 to number of robots). If not entered, 1 is assumed.

Opt 2: Not used.

P.DECEL

Returns the deceleration (in percentage) of the motion step in execution or the last executed motion step in the program in execution. If the motion speed is set in time (unit: S), –1 is returned.

Opt 1: Robot number (1 to number of robots). If not entered, 1 is assumed.

Opt 2: Not used.

MTR.RPM

Returns the rpm value for the motor speed (actual value) of the specified axis.

Opt 1: Robot number (1 to number of robots). If not entered, 1 is assumed.

Opt 2: Axis number. JT 1 is selected when omitted.

MTR.RPM.CMD

Returns the rpm value for the motor speed (command value) of the specified axis.

Opt 1: Robot number (1 to number of robots). If not entered, 1 is assumed.

Opt 2: Axis number. JT 1 is selected when omitted.

TOOL.VEL.CMD

Returns the mm/s value for the tool center point speed (command value) of the specified axis.

Opt 1: Robot number (1 to number of robots). If not entered, 1 is assumed.

Opt 2: Not used.

JT.VEL.CMD

Returns the deg/s value for rotation axis or mm/s value for linear axis for the speed (command value) of the specified axis.

Opt 1: Robot number (1 to number of robots). If not entered, 1 is assumed.

Opt 2: Axis number. JT 1 is selected when omitted.

NUMROBOT

Returns the number of robots connected.

Opt1: Not used.

Opt2: Not used.

ZROB.MGFNO

Returns the robot number.

opt1: Robot number (1 to number of robot). If not entered, 1 is assumed.

opt2: Not used.

ZROB.NOWAXIS

Returns the number of axis of the robot.

Opt1: Robot number (1 to number of robot). If not entered, 1 is assumed.

Opt2: Not used.

SIG.DO

Returns the number of external output signal.

Opt1: Not used.

Opt2: Not used.

SIG.DI

Returns the number of external input signal.

Opt1: Not used.

Opt2: Not used.

SIG.INT

Returns the number of internal signal.

Opt1: Not used.

Opt2: Not used.

LANGUAGE

Returns the number of the language selected for display. The language numbers are as follows.

Japanese	1	English	2	Italian	3
French	4	German	5	Chinese	6
Korean	7	Polish	9	Spanish	10
Dutch	11				

Opt1: Not used.

Opt2: Not used.

MEM.FREE

Returns the size of the memory currently available in percentage.

Opt1: Not used.

Opt2: Not used.

CONT.NO

Returns the controller number.

Opt1: Not used.

Opt2: Not used.

MTR.CURR.CMD

Returns the current value (command value) as Arms value.

opt1: Robot number (1 to number of robots). Robot 1 is assumed if omitted.

opt2: Joint number. JT1 is assumed if omitted.

MTR.CURR

Returns the current value (feed-back) as Arms value.

opt1: Robot number (1 to number of robots). Robot 1 is assumed if omitted.

opt2: Joint number. JT1 is assumed if omitted.

POWER

Returns the integral power of operating data as kWh value.

opt1: 0: Integral power of consumption
1: Integral power of supply (power regeneration compatible models only)
2: Integral power of regeneration (power regeneration compatible models only)

opt2: Not used.

EXISTDATA　("variable name or program name", type)

Function

Checks if the variable or program of specified name exists in specified type and if it is set with an AND value.

Parameter

"Variable name or program name"
Specifies the variable name or program name to be checked whether it exists or not.

Type
Specifies the data type of variable or program to be checked whether it exists or not.
Data types are as follows:

Specification	Data type
P	Joint displacement value
T	Converted value
R	Real-value
S	Character string
I	Integer
G	Program

Explanation

Returns -1 if the specified variable or program exists, and 0 (zero) if doesn't.

[NOTE]

Please be noted that array specifications have the following limitations:

1. Omission is impossible.
2. For the range, specify the small value on the left of a colon (:) and the big value on the right.

For example, when a real value r is three-dimensional and r[1,1,1], r[1,1,2], r[1,1,3] exist:

- When existence check is OK　　r[1,1,1:3]
- When existence check is not OK　　r[1,1,1:]　　; omitted
  r[1,1,*]　　; omitted
  r[1,1,3:1]　　; range specification from large to small

Example

ret=EXISTDATA("#data", P)	If the joint displacement value variable #data exists, ret is -1. If it doesn't exist, ret is 0 (zero).
ret=EXISTDATA("data", T)	If the converted value variable data exists, ret is -1. If it doesn't exist, ret is 0 (zero).
ret=EXISTDATA("data", R)	If the real variable data exists, ret is -1. If it doesn't exist, ret is 0 (zero).
ret=EXISTDATA("$data", S)	If the character string variable $data exists, ret is -1. If it doesn't exist, ret is 0 (zero).
ret=EXISTDATA("@data", I)	If the integer variable @data exists, ret is -1. If it doesn't exist, ret is 0 (zero).
ret=EXISTDATA("data", G)	If the program data exists, ret is -1. If it doesn't exist, ret is 0 (zero).

EXISTJOINT ("name of joint displacement value variable")

Function

Checks if the specified pose variable exists as variable defined by joint displacement values.

Parameter

"Name of joint displacement value variable"
Specifies the name of joint displacement value variable in form of character string. The variable name should be enclosed in quotations. Start the name with #.

Explanation

If the variable exists, returns –1. If it does not exist, returns 0.

[NOTE]

There are restrictions as described below when specifying array variables.
1. Specify the area of array elements. This cannot be omitted.
2. The minimum value is written to the left of the colon ":" and the largest values is written to the right.

For example, if real variable r is a three-dimensional array and elements r[1,1,1], r[1,1,2], and r[1,1,3] exist,

Check will result OK	r[1,1,1:3]	
Check will result NG	r[1,1,1:]	;Area not specified
	r[1,1,*]	;Area not specified
	r[1,1,3:1]	;Area specified in wrong order (from largest to smallest)

Example

ret=EXISTJOINT("#pos")	If joint displacement value variable #pos exists, ret= –1. If not, ret=0.

The following shows a case where joint displacement value variable #place is a two-dimensional array from #place[1,1] to #place[1,5].

ret=EXISTJOINT("#place")	ret will equal –1.
ret=EXISTJOINT("#place[1,1]")	#place[1,1]exists, so ret=-1.
ret=EXISTJOINT("#place[1,1:5]")	#place[1,1] to #place[1,5] exists, so ret=-1.
ret=EXISTJOINT("#place[1,1:6]")	#place[1,6] does not exist, so ret = 0.

EXISTTRANS　(“name of transformation value variable”)

Function

Checks if the specified pose variable exists as variable defined by transformation values.

Parameter

“Name of transformation value variable”

Specifies the name of transformation value variable in form of character string.　The variable name should be enclosed in quotations.

Explanation

If the variable exists, returns –1.　If it does not exist, returns 0.

See EXISTJOINT for restrictions and examples for specifying array variable.

Example

ret=EXISTTRANS(“pos1”)	If transformation value variable pos1 defined by transformation values exists, ret=-1.　If not, ret=0.

EXISTREAL　(“real variable name”)

Function

Checks if the specified variable exists as real variable.

Parameter

“Real variable name”

Specifies the name of real variable in form of character string.　The variable name should be enclosed in quotations.

Explanation

If the variable name exists, returns –1.　If it does not exist, returns 0.

See EXISTJOINT for restrictions and examples for specifying array variable.

Example

ret=EXISTREAL(“pp”)	If real variable pp exists, ret=-1.　If not, ret=0.

EXISTCHAR　(“string variable name”)

Function

Checks if the specified variable exists as string variable.

Parameter

“String variable name”

Specifies the name of string variable in form of character string.　The variable name should be enclosed in quotations.　Start the name with $.

Explanation

If the string variable exists, returns –1.　If it does not exist, returns 0.

See EXISTJOINT for restrictions and examples for specifying array variable.

Example

ret=EXISTCHAR(“$val”)	If string variable $val exists, ret=-1. If not, ret=0.

EXISTINTEGER　(“integer variable name”)

Function

Checks if the specified variable exists as integer variable.

Parameter

“Integer variable name”

Specifies the name of integer variable in form of character string.　The variable name should be enclosed in quotations.　Start the name with @.

Explanation

If the integer variable exists, returns –1.　If it does not exist, returns 0.

See EXISTJOINT for restrictions and examples for specifying array variable.

Example

ret=EXISTINTEGER(“@abc”)	If integer variable @abc exists, ret=-1.　If not, ret=0.

EXISTPGM ("program name")

Function

Checks if the specified program exists or not.

Parameter

"Program name"

Specifies program (or subroutine) name in form of string variable.

Explanation

If the specified program or subroutine exists, returns –1. If it does not exist, returns 0 (zero).

Example

ret=EXISTPGM("pg1") If program pg1 exists, ret=-1. If not, ret=0.

EXISTLOCALJOINT ("name of local joint displacement value variable")

Function

Checks if the specified local pose variable exists as variable defined by joint displacement values.

Parameter

"Name of local joint displacement value variable"

Specifies the name of local joint displacement value variable in form of character string. The variable name should be enclosed in quotations. Start the name with #. Error occurs if variable other than local joint displacement value variable is specified.

Explanation

If the variable exists, returns –1. If it does not exist, returns 0.

[NOTE]

There are restrictions as described below when specifying array variables.
1. Specify the area of array elements. This cannot be omitted.
2. The minimum value is written to the left of the colon ":" and the largest values is written to the right.

For example, if real variable r is a three-dimensional array and elements r[1,1,1], r[1,1,2], and r[1,1,3] exist,

Check will result OK	r[1,1,1:3]	
Check will result NG	r[1,1,1:]	;Area not specified
	r[1,1,*]	;Area not specified
	r[1,1,3:1]	;Area specified in wrong order (from largest to smallest)

Example

ret=EXISTLOCALJOINT(".#pos")	If local joint displacement value variable #pos exists, ret= –1. If not, ret=0.

The following shows a case where local joint displacement value variable #place is a two-dimensional array from #place[1,1] to #place[1,5].

ret=EXISTLOCALJOINT(".#place")	.#place does not exist, so ret = 0.
ret=EXISTLOCALJOINT(".#place[1,1]")	.#place[1,1] exists, so ret=-1.
ret=EXISTLOCALJOINT(".#place[1,1:5]")	.#place[1,1] to .#place[1,5] exist, so ret=-1.
ret = EXISTLOCALJOINT(".#place[1,1:6]")	.#place[1,6] does not exist, so ret = 0.

EXISTLOCALTRANS (“name of local transformation value variable”)

Function

Checks if the specified pose variable exists as local variable defined by transformation values.

Parameter

“Name of transformation value variable”

Specifies the name of local transformation value variable in form of character string. The variable name should be enclosed in quotations. Error occurs if variable other than local transformation value variable is specified.

Explanation

If the variable exists, returns –1. If it does not exist, returns 0.
See EXISTLOCALJOINT for restrictions and examples for specifying array variable.

Example

ret=EXISTLOCALTRANS(“.pos1”)	If local variable pos1 defined by transformation values exists, ret=-1. If not, ret=0.

EXISTLOCALREAL (“local real variable name”)

Function

Checks if the specified variable exists as local real variable.

Parameter

“Local real variable name”

Specifies the name of local real variable in form of character string. The variable name should be enclosed in quotations. Error occurs if variable other than local real value variable is specified.

Explanation

If the variable exists, returns –1. If it does not exist, returns 0.
See EXISTLOCALJOINT for restrictions and examples for specifying array variable.

Example

ret=EXISTLOCALREAL(“.pp”)	If local real value variable .pp exists, ret=-1. If not, ret=0.

EXISTLOCALCHAR ("local string variable name")

Function

Checks if the specified local variable exists as string variable.

Parameter

"Local string variable name"

Specifies the name of local string variable in form of character string. The variable name should be enclosed in quotations. Start the name with $. Error occurs if variable other than local string variable is specified.

Explanation

If the string variable exists, returns –1. If it does not exist, returns 0.

See EXISTLOCALJOINT for restrictions and examples for specifying array variable.

Example

ret=EXISTLOCALCHAR(".$val")	If local string variable .$val exists, ret=-1. If not, ret=0.

EXISTLOCALINTEGER ("local integer variable name")

Function

Checks if the specified local variable exists as integer variable.

Parameter

"Local integer variable name"

Specifies the name of local integer variable in form of character string. The variable name should be enclosed in quotations. Start the name with @. Error occurs if variable other than local integer variable is specified.

Explanation

If the integer variable exists, returns –1. If it does not exist, returns 0.

See EXISTLOCALJOINT for restrictions and examples for specifying array variable.

Example

ret=EXISTLOCALINTEGER (".@abc")	If integer variable @abc exists, ret=-1. If not, ret=0.

STRTOPOS (string variable)

Function

Returns the value of the pose variable that is specified by the string variable.

Parameter

String variable

Specifies a character string variable to get the specified pose values.

Explanation

Returns the value of the pose variable if a pose variable has been already assigned to the string variable. If a pose variable has not been assigned, an error occurs.

Example

```
HERE #pos
$A = "#pos"
```

JMOVE STRTOPOS($A)	The string value "$A" specifies "#pos". The robot moves to the destination which was described by joint displacement values "#pos". If "#pos" has not been defined, an error occurs.

STRTOVAL (string variable)

Function

Returns the real value specified by the string variable.

Parameter

String variable

Specifies character string variable to get the specified real value.

Explanation

Returns the real value if a real variable has been already assigned to the string variable. If a real variable has not been assigned, an error occurs.

Example

```
VAR = 5
$VA = “VAR”
```

total = STRTOVAL($VA)+6	The string variable “$VA” specifies the real variable “VAR”. The real variable “total” is eleven as “VAR” is five. If the variable “VAR” has not been defined, an error occurs.

ROUND (numeric value)

Function

Returns the value rounded at the first decimal place.

Parameter

Numeric value
This value is rounded at the first decimal place.

Explanation

Returns the value rounded at the first decimal place of the value specified as the parameter. When the specified value is a negative value, the value is rounded as an absolute value and then, the negative sign is added. The sign of the numeric value specified as the parameter remains unchanged unless the result is 0.

Example

ROUND (0.123)	Returns 0.
ROUND (10.8)	Returns 11.
ROUND (-5.462)	Returns -5.
ROUND (-5.662)	Returns -6.

IQARM (axis number)

Function

Returns the motor current value of the axis with the specified number.

Parameter

Axis number

Specify the number of the axis to acquire the motor current value. Acceptable range: 1- to the number of axes set.

Explanation

Returns the motor current value for the axis with the number specified in the parameter. Unit is in Arms.

Error occurs when used under the below condition:

When axis number of Mitsubishi motor is specified, error "(E1145) Cannot use specified channel, already in use." occurs if this function is used when monitoring of the motor current value is conducted by WHERE command, etc.

Example

a = IQARM(1)	Returns the motor current value of JT1 and substitutes it to a.

TRQNM (axis number)

Function

Returns the torque value of the axis with the specified number.

Parameter

Axis number

Specify the number of the axis to acquire the torque value. Acceptable range: 1- to the number of axes set. This function cannot be used for axis with Mitsubishi motor.

Explanation

Returns the torque value for the axis with the number specified in the parameter. Unit is in N·m.

Example

a = TRQNM(1)	Returns the torque value of JT1 and substitutes it to a.

CURLIMM (axis number)

Function

Acquires the negative limit value for the motor current of the external axis.

Parameter

Axis number
Specify the number of the external axis. Acceptable range: 7- 18.

Explanation

Acquires the limit value for the negative current of the external axis motor set by CURLIM instruction in form of percentage to the servo parameter current limit value. Unit is in %. Range of acquisition: 0 – 100.

This function is valid only for external axis using KHI amplifier.

Refer to CURLIM instruction for setting of current limit value.

Example

CURLIM 7,10,20	Sets the current limit value for JT7.
curm7 = CURLIMM(7)	Acquires the negative value for the set current limit value and stores it in the variable. In this example, 20 is stored in “curm7”.

CURLIMP (axis number)

Function

Acquires the positive limit value for the motor current of the external axis.

Parameter

Axis number

Specify the number of the external axis. Acceptable range: 7-18.

Explanation

Acquires the limit value for the positive current of the external axis motor set by CURLIM instruction in form of percentage to the servo parameter current limit value. Unit is in %. Range of acquisition: 0 – 100.

This function is valid only for external axis using KHI amplifier.

Refer to CURLIM instruction for setting of current limit value.

Example

CURLIM 7,10,20	Sets the current limit value for JT7.
curp7 = CURLIMP(7)	Acquires the positive value for the set current limit value and stores it in the variable. In this example, 10 is stored in “curp7”.

ENVCHKRATE (axis number)

Function

Acquires the set value for the magnification ratio to the initial threshold value to detect the deviation abnormality of the external axis.

Parameter

Axis number
Specify the number of the external axis. Acceptable range: 7- 18.

Explanation

Acquires the value set in ENVCHKRATE instruction for the magnification ratio to the initial threshold value for detection of deviation abnormality in external axis.

This function is valid only for external axis using KHI amplifier.

Refer to ENVCHKRATE instruction for the setting of magnification ratio to the deviation error.

Example

ENVCHKRATE 7, 0.1	Sets the magnification ratio to the deviation error detection threshold in JT7.
env7=ENVCHKRATE(7)	Stores the acquired magnification ratio to the variable.
	In this example, 0.1 is stored in “env 7”.

GETENCTEMP (axis number)

Function

Returns the temperature [°C] of the encoder of the axis of the specified number.

Parameter

Axis number

Specify the number of axis to acquire the encoder temperature. Acceptable range: 1- to the number of axes set.

Explanation

Returns the encoder temperature [°C] of the specified axis.

If the specified axis is disconnected, 0 is returned.

When the axis number is omitted, the specified axis number does not exist, or if the specified axis is an external axis not using KHI amplifier, error occurs and the program stops.

Example

The examples below acquire the encoder temperature for JT4 of Robot 1.

Example of monitor command

```
>x = GETENCTEMP(4)
>PRINT x
>55.75
```

Example of program

```
.PROGRAM enctemp.pc()
  X = GETENCTEMP(4)
  TYPE X
.END
```

OPEINFO (data number, robot number, joint number)

Function

Returns operating data corresponding to data numbers.

Parameter

Data number

Specifies the acquiring operating data by data numbers.

Data numbers and correspondence of operating data are described in the below table.

Robot number

When one controller is controlling multiple robots, specifies the robot number of operating data to be acquired.

When omitted, 1 is assumed.

Joint number

Specifies the joint number of operating data to be acquired. When omitted, 1 is assumed.

Data number	Operating data	Robot number	Joint number	Unit
1	Hour meter	Not used	Not used	H
2	Controller power-ON time	1 -	Not used	H
3	Servo-ON time	1 -	Not used	H
4	Frequency of motor-ON	1 -	Not used	Number of times
5	Frequency of servo-ON	1 -	Not used	Number of times
6	Frequency of emergency stop (moving)	1 -	Not used	Number of times
7	Integral power of consumption	1 -	Not used	kWh
8	Integral power of supply	1 -	Not used	kWh
9	Integral power of regeneration	1 -	Not used	kWh
10	Joint moving time	1 -	1 -	H
11	Joint total displacement	1 -	1 -	X1000 deg, mm
12	Joint total displacement (+)	1 -	1 -	X1000 deg, mm
13	Joint total displacement (-)	1 -	1 -	X1000 deg, mm

Explanation

This function returns the operating data displayed by the OPEINFO command as a real value.

Example

OPEINFO(10,1,1) Returns the joint moving time of JT1, robot number 1.

INS_POWER (power number)

Function

Returns the instantaneous power corresponding to the power number.

Parameter

Power number

Specifies the instantaneous power to be acquired by power number. Power numbers and correspondence of instantaneous power are described in the table below.

Power number	Instantaneous power	Unit
0	Instantaneous power of consumption	kW
1	Instantaneous power of supply	kW
2	Instantaneous power of regeneration	kW

Explanation

Returns the instantaneous power [kW] corresponding to the power number specified by the parameter. For power regeneration-incompatible models, 0 is returned for instantaneous power of regeneration.

Example

power = INS_POWER(0) Returns instantaneous power of consumption to power.

9.2 Pose Value Functions

DEST	Returns the destination pose as transformation values.
#DEST	Returns the destination pose as joint displacement values.
FRAME	Returns the transformation values for the frame coordinates.
NULL	Returns the null transformation values.
HERE	Returns the transformation values of the current pose.
#HERE	Returns the joint displacement values of the current pose.
TRANS	Returns the transformation values composed of the given components.
RX, RY, RZ	Returns the transformation value expressing the rotation around an axis.
#PPOINT	Returns the joint displacement values composed of the given components.
SHIFT	Returns the transformation value generated by shifting the original pose.
AVE_TRANS	Returns the average transformation values of two poses.
BASE	Returns the base transformation values.
TOOL	Returns the tool transformation values.
TRADD	Returns the value of the X component with the value of the traverse axis added. (Option)
TRSUB	Returns the value of the X component with the value of the traverse axis subtracted. (Option)
#HOME	Returns the joint displacement value of the home pose.
CCENTER	Returns the transformation values for the center of the circle described by the given poses. (Option)
CSHIFT	Returns the transformation values of the pose shifted towards the center of the circle described by the given poses. (Option)

DEST

#DEST

Function

DEST: Returns the transformation values of the destination of current robot motion.
#DEST: Returns the joint displacement values of the destination of current robot motion.

Explanation

By using these functions, the robot destination can be found out after the robot motion is interrupted for some reason. These functions can be used with all robot motions.

[NOTE]

The pose where the robot stops and the pose returned by DEST/#DEST functions are not always the same. For example, if the HOLD/RUN state is changed from RUN to HOLD, the robot stops immediately, but the pose returned by DEST/#DEST functions describes the pose the robot was heading for at that moment.

Example

POINT #old=#DEST	Stores the destination as a pose variable called #old.
dist=DISTANCE(DEST,HERE)	Calculates the distance between current position and destination, and assigns to dist.

FRAME (transformation value variable 1, transformation value variable 2, transformation value variable 3, transformation value variable 4)

Function

Returns the transformation values of the frame (relative) coordinates with respect to the base coordinates. Note that only the translational components of transformation values of the pose variables are used as positional information to determine the frame coordinates.

Parameter

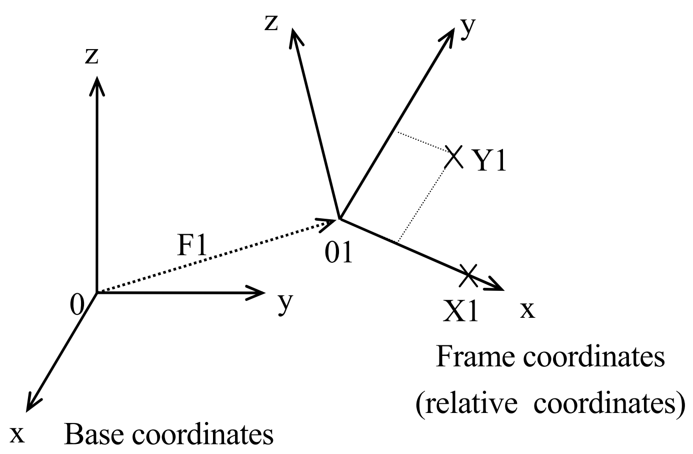
Transformation value variable1, transformation value variable 2
Specifies transformation value variables to determine the direction of the X axis. The X axis of the frame coordinates is set so that it passes through these two poses. The positive direction of the X axis is set in the direction from pose determined by transformation values variable 1 to pose determined by transformation value variable 2.

Transformation value variable 3
Specifies a transformation value variable to determine the direction of the Y axis. The Y axis of the frame coordinates is set so that the three points, pose1, pose 2, and pose 3, each determined by transformation value variables 1, 2 and 3, are on the XY plane. Also, pose 3 is set so it has the positive Y value.

Transformation value variable 4
Specifies a transformation value variable to specify the origin of the frame coordinates, which equals the values of X,Y,Z returned by this function.

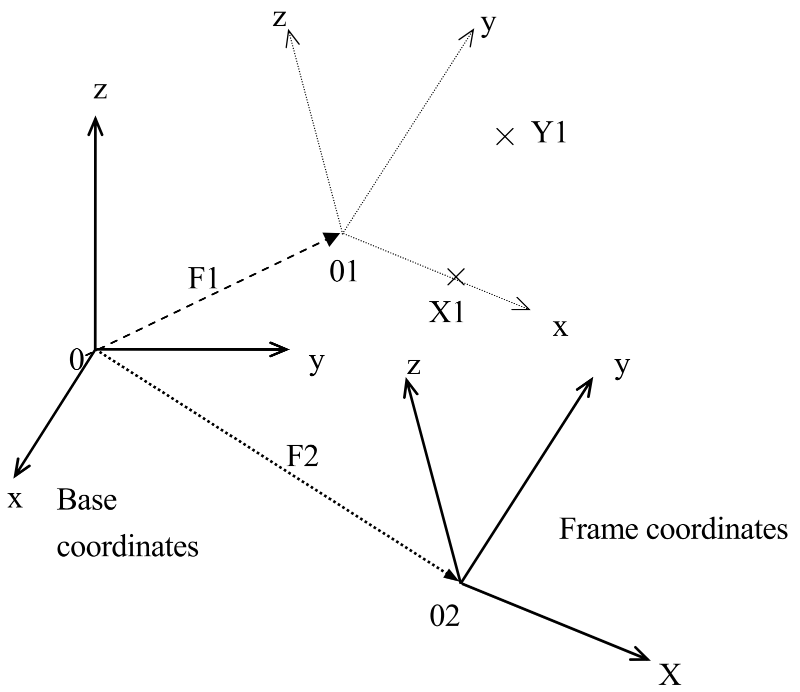
Explanation

POINT F1=FRAME(O1, X1, Y1, O1) Sets frame coordinates as in the diagram below.



If the poses in the frame coordinates are taught as F1+A, then only F1 needs re-teaching if the coordinates change, as when the parts station is moved. (See 11.6 Relative Pose Using the Frame Coordinates.)

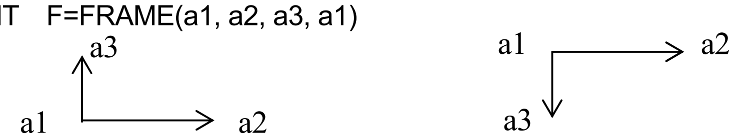
POINT F2=FRAME(O1, X1, Y1, O2) Sets frame coordinates as shown in the diagram below.



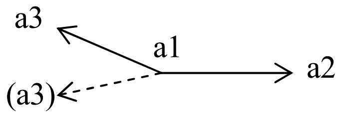
[NOTE]

Pay attention to the following points when defining frame coordinates.

POINT F=FRAME(a1, a2, a3, a1)



Note that the Y and Z axes in the above two frame coordinates face opposite directions, depending on where a3 is taught. Therefore, when a3 is taught close to the X axis (see diagram below), the direction of the Z axis may not result in the desired direction, so always check the frame coordinates before using the coordinates in any programs.



The three points a1, a2, a3 defines the position of the tool coordinates origin. When redefining the frame coordinates, the tool transformation must be the same as when a1, a2, a3 were first taught.

For better accuracy, teach the three points a1, a2, a3 as far apart as possible. Especially, a3 should be a point near the Y axis but as far away from the X axis as possible.

When teaching a1, a2, a3, the origin of the tool coordinates should be defined at a point that is easy to see, e.g. the tip of the tool, etc.

With some tools, it is difficult to determine the origin of the tool coordinates even when it has been moved by tool transformation. In such case, a1, a2, a3 are taught at null tool, but note that in this case, the origin of the tool coordinates is at the center of the flange surface.

NULL

Function

Returns the null transformation values.

Explanation

Returns the transformation values in which both the translational components and the rotational components are all 0 (X=Y=Z=0, O=A=T=0). This function is convenient when used with the SHIFT function enabling easy redefinition of transformation values. Coordinates can be shifted in translation movement without any change in rotational components (OAT).

Example

```
POINT    new=SHIFT(NULL BY x.shift,y.shift,z.shift)+old
```

Defines the variable “new” by shifting the pose defined as “old” a specified distance along the base coordinates.

```
dist=DISTANCE(NULL, test.pos)
```

Calculates the distance between the pose “test.pos” and the null origin of the robot (0,0,0,0,0,0), and assigns that value to dist.

HERE

#HERE

Function

HERE: Returns the transformation values which describe the current pose of the tool coordinates.

#HERE: Returns the joint displacement values which describe the current pose of the tool coordinates.

Explanation

The encoder values at the moment this function is executed are read. Therefore, note that the values returned by this function represent the pose of the robot when the function was executed.

[NOTE]

The name “here” cannot be used as a program name or a variable name.

Example

dist=DISTANCE(HERE, pos1)	Calculates the distance between the pose “pos1” and the current pose and assigns the value to “dist”.

TRANS (X component, Y component, Z component, O component, A component, T component)

Function

Returns the transformation value that has the specified translational and rotational components.

Parameter

X component, Y component, Z component

Specifies the translation components X, Y, Z. If not specified 0 is assumed.

O components, A component, T component

Specifies the rotation components O, A, T. If not specified 0 is assumed.

Explanation

The transformation values are calculated from the values specified in each parameter. The new transformation values can be used then to define pose variables, in compound transformation values, or in motion instructions. This function is convenient when used with DECOMPOSE instruction (see the example for #PPOINT function).

Example

```
POINT temp.pos=TRANS(v[0], v[1]+100, v[2], v[3], v[4], v[5])
Array variable v[0] –
```

RX (angle)
RY (angle)
RZ (angle)

Function

Returns the transformation values that represent the rotation around the specified axis.

Parameter

Angle

Specifies the value of the rotation in degrees.

Explanation

The X, Y, Z in this function represents the axes of coordinates. The rotational value around the specified axis is returned. The translational values are not returned by this function (X, Y, Z = 0).

Example

POINT x_rev =RX(30)	Returns the transformation value that represent 30° rotation around the X axis and assigns the value to "x_rev".

#PPOINT (jt1, jt2, jt3, jt4, jt5, jt6)

Function

Returns the specified joint displacement values.

Parameter

jt1 jt2 jt3 : jt6	Specifies the value of each angle (in degrees for rotational joints) If not specified, 0 is assumed.

Explanation

This function returns the specified joint displacement values. The values represent the displacement of each joint, from joint 1 to the last joint (not necessarily six).

[NOTE]

#PPOINT function is only processed in joint displacement values. Therefore, always enter "#" at the beginning of the function.

Example

In the program below, joints 2 and 3 of a six-joint robot are moved the specified amount from the current pose.

HERE #ref	Stores the current pose in memory.(#ref)
DECOMPOSE x[0]=#ref	Decomposes each component into elements of array variable x[0],……x[5].

```
JMOVE #PPOINT (X[0], x[1]+a, x[2]-a/2, x[3], x[4], x[5])
```

These two instructions result in the same pose as the program above, but unlike the program, the two joints do not move at the same time.

DRIVE 2, a, 100	Move jt2 by a°.
DRIVE 3, -a/2, 100	Move jt3 by – a/2°.

SHIFT (transformation value variable BY X shift, Y shift, Z shift)

Function

Returns the transformation values of the pose shifted by the distance specified for each base axis (X,Y,Z) from the specified pose.

Parameter

Transformation value variable

Specifies a transformation value variable for the pose to be shifted.

X shift, Y shift, Z shift

Specifies the shift amount in X, Y, Z directions of the base coordinates. If any value is omitted, 0 is assumed.

Explanation

The X shift, Y shift, Z shift amounts are added to each of the X, Y, Z component of the specified transformation value variable. The result is returned in transformation values.

Example

If the values of the transformation value variable x are (200, 150, 100, 10, 20, 30), then

```
POINT   y=SHIFT(x BY 5, -5, 10)
```

“x” is shifted by the specified values to (205, 145, 110,10, 20, 30) and those values are assigned to transformation value variable “y”.

AVE_TRANS (transformation value variable 1, transformation value variable 2)

Function

Returns the average values of the two transformation value variables 1 and 2.

Parameter

Transformation value variable 1, transformation value variable 2
Specifies the two transformation value variables to calculate the average between them.

Explanation

This function calculates the transformation values of the pose which defines the midpoint for each of the components of transformation value variables 1 and 2.

This function is commonly used for calculating the average of pose information gained from sensor checks.

[NOTE]

> For the XYZ components, the average is given by adding each of the components and dividing them by 2. However, for the OAT components, the average is not necessarily given in that manner.

Example

```
POINT x = AVE_TRANS(p,q)
JMOVE AVE TRANS(p,q)
```

BASE

Function

Returns the current base transformation values.

Example

point a = BASE Assigns to variable “a” the current base transformation values.

TOOL

Function

Returns the current tool transformation values.

Example

point aa = TOOL Assigns to variable “aa” the current tool transformation values.

TRADD (transformation value variable)

Option

Function

Returns the sum of the traverse axis value and the X component of the specified transformation value variable.

Parameter

Transformation value variable
Specifies the name of the transformation value variable to whose X component the traverse axis value is added.

TRSUB (transformation value variable)

Option

Function

Returns the value gained by subtracting traverse axis value from the X component of the specified transformation value variable.

Parameter

Transformation value variable
Specifies the name of the transformation value variable from whose X component the traverse axis is subtracted.

#HOME (home pose number)

Function

Returns the currently set home pose.

Parameter

Home pose number
Specifies the home pose number.
1: Specifies home pose 1 set by SETHOME command.
2: Specifies home pose 2 set by SET2HOME command.
If not specified, 1 is selected.

Explanation

Returns the pose of the currently set home pose in joint displacement values.

[NOTE]

> #HOME function is only processed in joint displacement values. Therefore, always enter "#" at the beginning of the function.

Example

In the program below, the robot does not move in the direct path but first moves to the same height as the home pose (in the direction of the Z axis only), and then it moves to the home pose.

```
POINT homepos = #HOME(1)
IF DZ(homepos) > DZ(HERE) THEN
HERE tmp
POINT/Z tmp = homepos
LMOVE tmp
END
HOME
```

CCENTER (transformation value variable1, transformation value variable 2, transformation value variable 3, transformation value variable 4)

Option

Function

Returns the center of the arc created by the three specified pose.

Parameter

Transformation value variable 1, transformation value variable 2, transformation value variable 3
Specifies pose variables defined by transformation values to determine the three points on the arc.

Transformation value variable 4
Specifies the pose variable to determine the orientation of the robot.

CSHIFT (transformation value variable1, transformation value variable 2, transformation value variable 3, transformation value variable 4 BY shift amount)

Option

Function

Returns the pose that was shifted the specified amount from the object pose.

Parameter

Transformation value variables 1, 2, 3
Specifies transformation value variables to determine the three points on the arc.

Transformation value variables 4
Specifies a transformation value variable to specify the object pose in transformation values.

Shift amount
Specifies the amount to shift in real values.

9.3 Mathematical Functions

ABS	Returns the absolute value of a numerical expression.
SQRT	Returns the square root of a numerical expression.
PI	Returns the constant π.
SIN	Returns the sine value.
COS	Returns the cosine value.
ATAN2	Returns the arctangent value.
RANDOM	Returns a random number.

ABS(real value)

SQRT(real value)

PI

SIN(real value)

COS(real value)

ATAN2(real value1, real value 2)

RANDOM

Keyword	Function	Example
ABS	Returns the absolute value of a numerical expression.	ABS(value)
SQRT	Returns the square root of a numerical expression.	SQRT(value)
PI	Returns the constant π(3.1415······).	PI
SIN	Returns the sine value of a given angle.	SIN(value)
COS	Returns the cosine value of a given angle.	COS(value)
ATAN2	Returns the values of an angle (in degrees) whose tangent equals v1/v2.	ATAN2(v1,v2)
RANDOM	Returns a random number from 0.0 to 1.0.	RANDOM

Example

x=ABS(y)	substitutes the value |y| into x
x=SQRT(y)	substitutes the square root of y into x
en=2∗PI∗r	substitutes the result of 2πr into en
Z=(SIN(x) ^ 2)+(COS(y) ^ 2)	substitutes the result of $(\sin(x))^2+(\cos(y))^2$ into z
slope=ATAN2(rise,run)	substitutes the result of $\tan^{-1}$(rise/run) into slope
r=RANDOM∗10	substitutes a random number from 0 to 10 into r

9.4 String Functions

$CHR	Returns the ASCII characters of the specified values.
$SPACE	Returns the specified number of blanks.
$LEFT	Returns the leftmost characters in the string.
$RIGHT	Returns the rightmost characters in the string.
$MID	Returns the specified number of characters.
$DECODE	Extracts characters separated by specified characters.
$ENCODE	Returns the string created by the print data.
$ERRORS	Returns the error message of the specified error code.
$ERROR	Returns the error message of the error code specified by a negative number.
$DATE	Returns the system date.
$TIME	Returns the system time in a string.
$TIME_MS	Returns the system time including milliseconds in a string.
$SYSDATA	Returns character strings for parameters in the AS system.
$ERRLOG	Returns character strings for error message of the specified error log.
$STR_ID	Returns character strings for robot information for Robot 1.
$STR_ID2	Returns character strings for robot information for Robot 2.

$CHR (real value)

Function

Returns the ASCII character string corresponding to the specified ASCII value.

Parameter

Real value (or numeric expression)

Specifies the value to change into an ASCII character. Acceptable range: 0 to 255.

Explanation

Returns a corresponding ASCII character string if the specified ASCII value is between 0 and 255.

Example

$CHR(65)	Returns “A” for ASCII value 65.
$CHR(^H61)	Returns “a” for ASCII value 97 (16×6+1).

$SPACE (number of blanks)

Function

Returns the specified number of blanks.

Parameter

Number of blanks

Specifies the number of blanks. Specify 0 or a positive value.

Example

Type ”a” + $SPACE(1) + “dog”	Displays “a dog” (1 space is entered between “a” and “dog”.

$LEFT (string, number of characters)

Function

Returns the specified number of characters starting from the leftmost character of the specified string.

Parameter

String

Character string, string variable, or string expression.

Number of characters

Specifies how many characters to return counting from the leftmost (or first) character of the entered string. If 0 or a negative number is specified, blank is returned. If the number specified is larger than the number of characters in the string, the whole string is returned.

Explanation

Returns the character strings for the number of characters counting from the left of the string.

Example

$LEFT ("abcdefgh",3)	Returns the string "abc".
$LEFT ("*1*2*3*4*5",15)	Returns "*1*2*3*4*5" (the whole string).

$RIGHT (string, number of characters)

Function

Returns the specified number of characters starting from the rightmost character of the specified string.

Parameter

String

Character string, string variable, or string expression.

Number of characters

Specifies how many characters to return counting from the rightmost (or last) character of the entered string. If 0 or a negative number is specified, blank is returned. If the number specified is larger than the number of characters in the string, the whole string is returned.

Explanation

Returns the character strings for the number of characters counting from the right of the string.

Example

$RIGHT("abcdefgh",3) Returns the string "fgh".

$MID (string, real value, number of characters)

Function

Returns the specified number of characters from the specified string.

Parameter

String

Character string, string variable, or string expression.

Real values (or numeric expression)

Specifies the starting position of the string is to be taken.

Number of character

Specifies the number of characters to extract.

Explanation

If the starting position is not specified, or specified by a value of 1 or less, the characters are extracted from the first character of the string. If the starting position is specified by a value of 0 or less, or if the number is larger than the number of characters in the string, blank is returned.

If the number of characters to extract is not specified, or when it is larger than the number of characters in the string, the characters from the specified starting position to the end of the string is returned.

Example

In the instruction below, the $MID function returns "cd" (two characters starting from the third character in the string "abcdef"). Then the result is substituted into string variable $substring.

```
$substring＝$MID("abcdef",3,2)
```

$DECODE (string variable, separator character, mode)

Function

Returns the string separated by “separator characters”.

Parameter

String variable
Specifies the string from where the characters are taken. Characters extracted as a result of this function are removed from this string.

Separator character
Specifies the character to read as separator. (Any character in the string can be specified as separator.)

Mode
Specifies the real number for operation done by this function.

If the mode is a negative number or 0, or if it is not specified, the characters starting from the first character in the string variable to the separator are returned. The returned string is removed from the string variable. If a positive number specifies the mode, the first separator that appears in the string is returned. The returned separator is removed from the value of the string variable. If more than one separator characters exist in the string consecutively, all the separator characters are returned and removed from the string variable.

Explanation

This function searches the specified string for the separator character and extracts the characters from the beginning of the string to the separator. The extracted characters are returned as the result of the function, and at the same time they are removed from the original string.

The string returned as the result of the function (string removed from the original string) could be either the characters before the separator or the separator itself.

[NOTE]

This function changes the original string at the same time it returns the characters.

The separator character is not case-sensitive.

Example

In the instructions below, the numbers separated by commas or blanks are removed from the string “$input”. The first instruction in the DO structure removes the first set of characters in $input and substitutes them into variable “$temp”. Next the function VAL changes the string gained in the previous instruction into a real value. The real value is then substituted into the array variable “value”. Then, the program execution goes on to the next $DECODE function and searches for the next separator (the separator is removed from $input).

```
i=0                                  ;Resets counter
DO
$temp=$DECODE($input,”,”,0)          ;Extracts characters up to the separator ”,”
value[i]=VAL($temp)                  ;Converts the characters to real values.
if $input ==” ” GOTO 100
$temp = $DECODE($input,”,”,1)        ;Extracts the separator “,”
i=i+1                                ;Counter increment
UNTIL $input==“”                     ;Continues program execution until there are no
100 TYPE “END”                       more characters
```

If the values of $input are as below, each separated by space and comma then the result of the above program are as follows:

```
1234, 93465.2,   .4358, 3458103,

value[0]        1234.0
value[1]        93465.2
value[2]        0.4358
value[3]        3458103.0
```

As the result of executing the program, the value of string variable $input becomes “” (blank).

$ENCODE (print data, print data, ……)

Function

Returns the string created from the print data specified in the parameters. The string is created in the same way as when using TYPE command.

Parameter

Print data

Select one or more from below. Separate the data with commas when specifying more than one.

(1) character string
(2) real value expressions (the value is calculated and displayed)
(3) Format information (controls the format of the output message)

Explanation

This function enables creating strings within programs using the same print data as in the TYPE command. Unlike TYPE, the $ENCODE function does not display the created strings, but instead the results are used as values in programs.

The following codes are used to specify the output format of numeric expressions. The same format is used until a different code is specified. In any format, if the value is too large to be displayed in the given width, asterisks (*) are shown instead of the values.

Format Specification Codes

/D	Uses the default format. This is the same as specifying the format as /G15.8 except that zeros following numeric values and all spaces but one between numeric values are removed.
/Em.n	Expresses the numeric value in scientific notation (e.g. -1.234E＋02). “m” describes the total number of characters shown on the terminal and “n” the number of decimal places. “m” should be greater than “n” by five or more, and smaller than by 32.
/Fm.n	Expresses the numeric values in fixed point notation (e.g. -1.234). “m” describes the total number of characters shown on the terminal and “n” the number of decimal places.
/Gm.n	If the value can be expressed in Fm.n format within “m” digits (including “n” digits after the decimal point), the value is expressed in that format. Otherwise, the value is expressed in Em.n format.

/Hn　　Expresses the values as a hexadecimal number in the “n” digit field.

/In　　Expresses the values as a decimal number in the “n” digit field.

The following parameters are used to insert certain characters between character strings.

/Cn　　Inserts line feed n times in the place where this code is entered, either in front or after the print data.　If this code is placed within print data, n−1 blank lines are inserted.

/S　　The line is not fed

/Xn　　Inserts n spaces.

/Jn　　Expresses the value as a hexadecimal number in the n digit field.　Zeros are used in place of blanks.　(Option)

/Kn　　Expresses the value as a decimal number in the n digit field.　Zeros are used in place of blanks.　(Option)

/L　　This is the same as /D except that all the spaces are removed with this code. (Option)

Example

```
$output = $output + $ENCODE(/F6.2,count)
```

The value of the real variable “count” is converted into a string in the format specified by /F6.2, and added to the end of the string “$output”.　Then the combined string is substituted back in the string variable “$output”.

$ERRORS ("error code")

Function

Returns the error message for the specified error code. The error code is returned as a character string with the error message.

Parameter

Error code

Specifies the error code in the following format: Pxxxx, Wxxxx, Exxxx, or Dxxxx.

$ERROR (error number)

Function

Returns the error message for the specified error code.

Parameter

Error number

Specifies the error number by a negative number (starting with −). The error codes are converted into negative error numbers as shown below:

Dxxxx : −4xxxx
Exxxx : −3xxxx
Wxxxx : −2xxxx
Pxxxx : −1xxxx

$DATE (date form)

Function

Returns the system date in the specified string format.

Parameter

Date form

Specifies by numbers 1 - 3, the date format to be output.

Explanation

The types of date forms are as follows.

$DATE(1) mm/dd/yyyy

(If the date is "October 20, 2016" then the value returned is 10/20/2016).

$DATE(2) dd/mmm/yyyy

(If the date is "October 20, 2016" then the value returned is 20/OCT/2016).

mmm is JAN/FEB/MAR/APR/MAY/JUN/JUL/AUG/SEP/OCT/NOV/DEC in order from January to December.

$DATE(3) yyyy/mm/dd

(If the date is "October 20, 2016" then the value returned is 2016/10/20.)

$TIME

Function

Returns the system time in following string format:
hh:mm:ss
(For example, for the time 18:27:50, it becomes 18:27:50.)
hh is in 24-hour time system.

$TIME_MS

Function

Returns the system time including milliseconds in a string.
In form of hh:mm:ss.xxx
(For example, for the time 18:27:50, it becomes 18:27:50.0000.)
hh is in 24-hour time system. Accuracy of the time acquired by this function is ±2 ms.

$SYSDATA (keyword, opt1, opt2)

Function

Returns specified parameters in the AS system according to the given keyword.

Parameter

Keyword, opt1, opt2

ZROB.NAME

Returns the model name of the robot.

Opt 1: Robot number (1 to number of robots). If not entered, 1 is assumed.

Opt 2: Not used.

$ERRLOG (error log number)/$ERRORLOG (error log number) *1

Function

Returns the character string of error message of the specified log number.

Parameter

Error log number

Specifies the error log number of the log to return.

Example

When the error is recorded in the error log as below,

```
1 - [14/11/17 12:05:02 SIGNAL:00 Monitor speed: 10 Teach mode]
      (E1162) Buffer overflow occurred in the gravity comp. value channel 2.
2 - [14/11/17 12:05:02 SIGNAL:00 Monitor speed: 10 Teach mode]
      (E1352) Jt3 4 5 6 Codes set in software and power block do not match.
3 - [14/10/16 17:19:22 SIGNAL:00 Monitor speed: 10 Teach mode]
      (D2023) Failed to load arm data.
```

When

```
> TYPE  $ERRLOG(2)
```

is input, the error message logged second “(E1352) Jt3 4 5 6 Codes set in software and power block do not match.” is displayed.

*1 $ERRORLOG is used for explosion proof software.

$STR_ID (number)
$STR_ID2 (number)

Function

Returns the character string for the robot information.

$STR_ID Robot 1

$STR_ID2 Robot 2

Parameter

Number

Specifies the robot information to return by integer values 0 to 1.

0 : Robot name Returns the robot name of the controlled robot arm.

1: Machine number: Returns the machine number of the controlled robot arm.

Example

When

Robot 1	Robot name: RS020N-A001	Machine number: 1
Robot 2	Robot name: BX200L-B001	Machine number: 2

then,

$STR_ID(0)	String “RS020N-A001” is returned.
$STR_ID(1)	String “1” is returned.
$STR_ID2(0)	String “BX200L-B001” is returned.
$STR_ID2(1)	String “2” is returned.

10 Process Control Programs

This chapter describes the monitor commands and program instructions used with the Process Control (PC) programs. In parentheses on the right, M indicates monitor commands, and P program instructions. Those with both M and P can be used as either commands or instructions.

PCSTATUS	Displays the status of the specified PC program. (M)
PCEXECUTE	Executes the specified PC program. (M, P)
PCABORT	Stops execution of specified PC program immediately. (M, P)
PCKILL	Initializes the PC program execution stack. (M)
PCEND	Stops execution of specified PC program. (M, P)
PCCONTINUE	Resumes execution of PC program. (M)
PCSTEP	Executes a single step of a PC program. (M)
PCSCAN	Specifies PC program processing time. (P)

Example

Keyword ↓ Parameter ↓

PCSTATUS **PC program number**

Parameters marked with ▭ can be omitted.

Always enter a space between the keyword and the parameter.

↵ represents the Enter key in the examples.

PCSTATUS PC program number:

Function

Displays the status of PC programs. (M)

Parameter

PC program number

Selects the PC program number to display. Acceptable range: 1 to 5. If not specified, 1 is assumed.

Explanation

The PC program status is displayed in the following format.

```
(1)··········PC program status:      Program is not running
             Execution cycles:
(2)··········Completed cycles:       11
(3)··········Remaining cycles:       Infinite
(4)··········Program name            Priority   Step No. ······················(5)
             pc_test0                1          PRINT "step1"
```

(1) Program status

The PC program status as described as one of the following:

Program is not running	Program is not currently running.
Program running	Program is currently running.
Program WAIT	Program is running, but waiting for the condition set in WAIT command to fulfill.

(2) Completed cycles

Displays the number of execution cycles completed.

(3) Remaining cycles

Displays the numbers of cycles not yet executed. If the execution cycle is set as negative number (–1) in PCEXECUTE command, the display will be “infinite”.

(4) Program name

Name of the PC program in execution is displayed.

(5) Step

Displays the number of the step currently being executed and the instruction written in that step.

PCEXECUTE PC program number : program name, execution cycle, step number

Function

Executes PC programs. (M, P)

Parameter

PC Program number

Selects the number of the PC program to execute. Acceptable range: 1 to 5. If not specified, 1 is assumed. Up to 5 PC programs can be executed at the same time. The PC program number is not the order of priority.

Program name

Selects the name of the program to execute at that PC program number. If not specified, the program last executed using the PCEXECUTE command is selected.

Execution cycle

Specifies how many times the PC program is to be executed. If not specified, 1 is assumed. If −1 is entered, the program is executed continuously.

Step number

Selects the step from which to start execution. If not specified, the execution starts from the first step in the program.

Explanation

This command is identical to EXECUTE monitor command, except that this command executes PC programs instead of robot control programs. The PC program currently in execution is displayed with a blinking "* " at the end of its name.

PCEXECUTE can be used as either a monitor command or an instruction in a robot control program.

Example

PCEXECUTE control, -1	The program "control" is executed continuously; i.e. program execution continues until PCABORT command is executed, PAUSE or HALT instruction is executed in the program, or an error occurs.

PCABORT PC program number:

Function

Stops the execution of the currently running program. (M, P)

Parameter

PC program number

Selects the number of the PC program to be stopped. Acceptable range: 1 to 5. If not specified, 1 is assumed.

Explanation

PCABORT is identical to ABORT command except this command stops PC programs instead of robot control programs.

The program currently running is stopped, and the execution can be resumed using PCCONTINUE command.

PCABORT can be used as either a monitor command or an instruction in a robot control program.

PCKILL PC program number:

Function

Initializes the stack of PC programs. (M)

Parameter

PC program number

Selects the number of the PC program to initialize. Acceptable numbers are from 1 to 5. If not specified, 1 is assumed.

Explanation

This command initializes the program stack of PC programs.

When a program is suspended by PAUSE or PCABORT command, or by an error, the program remains in the program stack. As long as the program is in the stack, it cannot be deleted (DELETE command). In this case, first use PCKILL to remove the program from the stack.

PCEND PC program number: task number

Function

Ends execution of the PC program currently running upon execution of the next STOP instruction in that program. (M, P)

Parameter

PC program number
Selects the number of the PC program to end. Acceptable range: 1 to 5. If not specified, 1 is assumed.

Task number
Specifies 1 or −1. If not specified, 1 is assumed.

Explanation

If the task number is not specified or specified as 1, the program execution is stopped as soon as the next STOP or RETURN instruction (or similar instruction) is executed, regardless of remaining cycles. The remaining cycles can be executed using PCCONTINUE.

If −1 is specified as the task number, the PCEND command entered previously is canceled. When a program loop occurs or the program runs infinitely without a STOP instruction, PCEND is ineffective and must be canceled by PCEND −1. (To cancel the loop, PCABORT must be used).

PCEND can be used as either a monitor command or an instruction in robot control programs.

PCCONTINUE PC program number NEXT

Function

Resumes execution of a suspended PC program. Or, skips the WAIT instruction in the PC program. (M)

Parameter

PC program number

Selects the number of the PC program to resume execution. Acceptable range: 1 to 5. If not specified, 1 is assumed.

NEXT

If this parameter is specified, the execution is resumed from the step after the step that was suspended. If not specified, the execution resumes from the same step that was suspended. With the parameter NEXT, this command can be used to skip the WAIT instruction in the currently running PC program and to resume execution of that PC program.

Explanation

PCCONTINUE is identical to CONTINUE command except this command is used to continue execution of PC programs instead of robot control programs.

Execution is resumed from the step where the execution was stopped by PAUSE or PCABORT command, or by an error, and from the step after that when the parameter NEXT is specified.

PCSTEP　PC program number: program name, execution cycles, step number

Function

Executes a single step of a PC program. (M)

Parameter

PC program number

Selects the number of the PC program containing the desired step.　Acceptable range: 1 to 5.　If not specified, 1 is assumed.

Program name

Selects the name of the program to execute at that PC program number.　If not specified, the program currently in execution or the program last executed is selected.

Execution cycle

Specifies how many times the program step is to be executed.　If not specified, 1 is assumed.

Step number

Selects the number of the program step to execute.　If not specified, the first step of the program is selected.　If none of the parameters are specified, the next step is executed.

Explanation

PCSTEP command, like the PCCONTINUE command, can be used without parameters only in the following conditions:

1. when PCSTEP command was used in the last executed step
2. after a PAUSE instruction
3. when the program was suspended by reasons other than error.

Example

>PCSTEP sequence,,23 ↵	Executes step 23 of the PC program no.1 named "sequence" one time.

Enter PCSTEP ↵ after this, and then the next step (step 24) is executed.

PCSCAN time

Function

Sets the cycle time for executing the PC program. (P)

Parameter

Time

Sets how long the program repetition cycle takes. The time is specified in seconds, 0 or greater.

Explanation

This command is used to execute the PC program in the specified cycle time. If the execution time is longer than the specified time, the time specified here is ignored.

Example

```
program
    PCSCAN 1
    IF sig(1) THEN
    SIGNAL -1
    ELSE
    SIGNAL 1
    END
```

If the above program is executed continuously using the PCEXECUTE command (execution cycle: −1), SIGNAL 1 turns ON → OFF every second.

11 Sample Programs

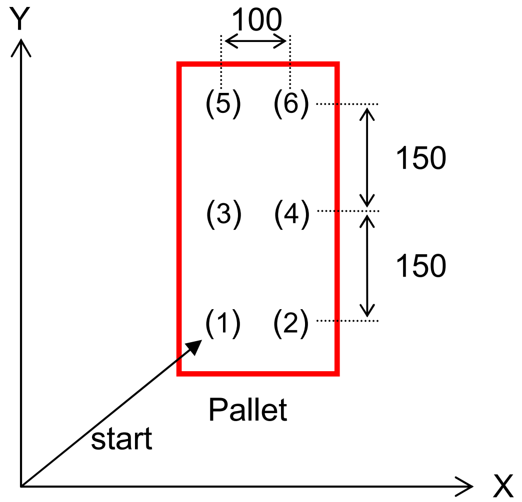
This chapter shows some sample programs using the AS language system.

11.1 Initial Settings for Programs

For easier programming, the settings below are done prior to performing any function on the robot.

- Move the robot to the home pose.
- Define the necessary variables for each task. (e.g. for palletizing, fix the number of parts per pallet)
- Initialize counter, flag, etc.
- Set the tool coordinates to be used in this task.
- Set the base coordinates to be used in this task.

Here is an example of a program initialization routine for a palletizing operation as shown in the figure below.



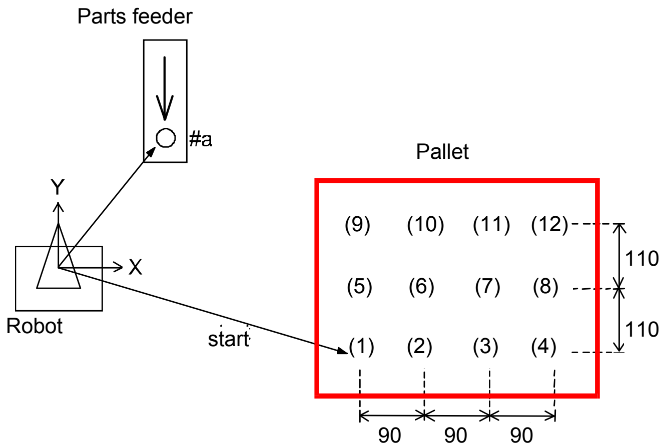
In the above example, parts are palletized in order from (1) to (6). In this case, a program like the following should be used for initial setting. The pallet is set parallel to the robot base coordinates in this example.

```
1  BASE   NULL          ;defines the robot base coordinate (NULL)
2  TOOL   tool1         ;tool transformation (tool1)*
3  row.max=3            ;3 rows
4  col.max=2            ;2 columns
5  xs=100               ;sets the allocation distance in the X coordinate
                        (ΔX=100 mm)
6  ys=150               ;sets the allocation distance in the Y coordinate
                        (ΔY=150 mm)
7  POINT put=start      ;substitutes the value of pose(1) to variable put.
8  OPENI                ;opens the hand of the tool
9  HOME                 ;moves to home pose**
```

NOTE* The tool transformation values (tool 1) should be defined prior to proceeding.
NOTE** The origin (HOME) should be defined prior to proceeding.

11.2 Palletizing

In the example shown here, parts are picked up from a parts feeder and placed on a pallet with three rows (110 mm apart) and four columns (90 mm apart). To simplify the explanation, both the pallet and the parts placed on the pallet are set parallel to the XY plane of the robot base coordinates. Also, the procedure of synchronizing the feeder and the robot using the external I/O signals (SWAIT instruction, SIGNAL instruction, etc.) is omitted.



- The pallet is set parallel to the XY plane of the base coordinates.
- Pose #a (Parts feeder) and pose "start" (where the first part is placed) are to be defined prior to executing the program.

Program Example

```
 .PROGRAM palletize
;          initial setting (3 rows, 4 columns, ∆X=90, ∆Y=110, etc.)
row.max=3
col.max=4
xs=90
ys=110
SPEED 100 ALWAYS
ACCURACY 100 ALWAYS
POINT put=start
OPENI
;
;          Start palletizing
FOR row=1  TO  row.max
FOR  col=1 TO col.max
JAPPRO #a,100
SPEED  30
ACCURACY 1                              Picks up the part from the feeder.
LMOVE  #a
CLOSEI
LDEPART 200
;
JAPPRO put, 200
SPEED  30
ACCURACY 1                              Places the part on the pallet.
LMOVE   put
OPENI
LDEPART 200
;
;          Calculate the pose of part in the next row.
POINT put=SHIFT(put BY xs, 0,0)
END
;
;          Calculate the pose of part in the next column.
POINT put=SHIFT(start by 0,ys*row, 0)
END
.END
```

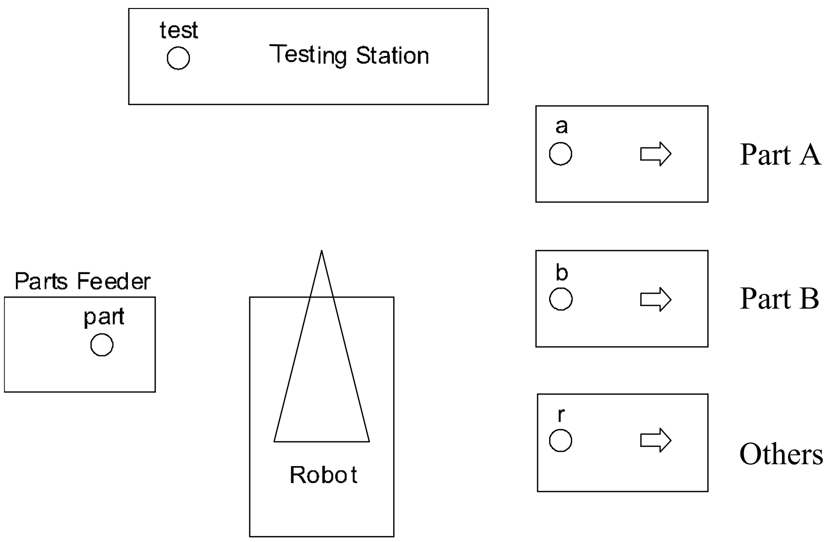
11.3 External Interlocking

This example demonstrates an operation performed synchronously with an external device. This program uses the instructions: SIGNAL, IF, SWAIT, ONI, IGNORE.

1. Two types of parts, A and B, are set in the Parts Feeder in random order. (Input signal for set complete: IN1)
2. The robot picks up a part from the Parts Feeder and sets it at the Testing Station. (Output signal for set complete: OUT1)
3. At the Testing Station, the parts are classified into part A, part B or other than A or B.
   Input signal for testing complete: IN2
   Input signal for parts classification: IN3, IN4
   (IN3, IN4) = (1, 0) : part A
   (IN3, IN4) = (0, 1) : part B
   (IN3, IN4) = (0, 0) or (1, 1) : Others
4. The robot places the parts according to the classification of each part.

If any trouble arises with the Testing Station while the robot picks up the part from the feeder and carries it to the Testing Station, the program immediately halts and branches to the trouble shooting subroutine. The external input signal for trouble occurrence is IN7. The signal IN6 is input when trouble shooting is completed, and the robot resumes execution as soon as this signal is input.

The program to perform the above operation is named MAIN, the trouble shooting subroutine is named EMERGENCY.



Program Example

```
; Define Variables
set.end  =1001                  ;signal for set complete (IN1) is named set.end
test.end =1002                  ;signal for test complete (IN2) is named test.end
a.part   =1003                  ;signal for part A (IN3) is named a.part
b.part   =1004                  ;signal for part B (IN4) is named b.part
retry    =1006                  ;signal for trouble resolved(IN6) is named retry
fault=1007                      ;signal for trouble (IN7) is named fault
test.start= 1                   ;signal for start test (OUT1) is named test.start

.PROGRAM main()
OPENI
10  JAPPRO  part,100
 ONI  fault CALL emergency      ;monitors for signal fault and jumps to emergency
                                 subroutine when it is detected
SWAIT set.end                   ;waits for the part to be set in the feeder
LMOVE       part                ;moves to part (Parts Feeder)
CLOSEI
LDEPART     100
JAPPRO      test,100            ;carries the part to the Testing Station
LMOVE       test
BREAK
;
IGNORE      fault     ;stops monitoring for signal IN7 (fault)
SIGNAL      test.start          ;turns ON the signal test.start
TWAIT 1.0
SWAIT test.end                  ;waits until the testing is completed
JDEPART     100
SIGNAL      -test.start         ;turns OFF the signal test.start
IF SIG(a.part,-b.part) GOTO 20  ;if the part is part A, then jump to label 20
IF SIG(-a.part,b.part) GOTO 30  ;if the part is part B, then jump to label 30
POINT n=r                       ;if it is neither A nor B, then carry the part to r
GOTO 40
20  POINT   n=a                 ;defines the place to put part A
GOTO 40
30  POINT   n=b                 ;defines the place to put part B
40  JAPPRO  n,100               ;carries part to its placing pose
```

```
LMOVE       n
OPENI

LDEPART     100
GOTO 10

.END

.PROGRAM emergency()
PRINT "**ERROR**"            ;outputs error message on the terminal
SWAIT retry                  ;waits until the trouble is resolved
ONI   fault CALL emergency   ;starts monitoring for fault again,
RETURN                       ;returns to the main program
.END
```

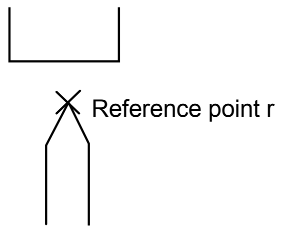
11.4 Tool Transformations

This section explains how to obtain tool transformation values and how to create programs using them.

11.4.1 Tool Transformation Values-1 (When the Tool Size is Unknown)

When the size of the tool is unknown due to awkwardness of the tool, tool transformation values can be calculated as shown below.　The Z axis of the base coordinates is set perpendicular to the ground.

1. Select an object with a sharp point.　Fix the tip of the object pointing up vertically from the ground.　This point will be the reference point “r”.

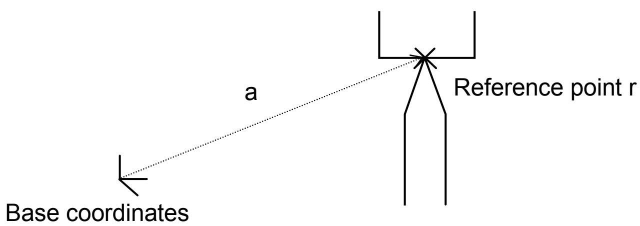


2. Move the robot so that the tool mounting flange faces straight downward.　Then in repeat mode, enter the following commands:

```
>SPEED 10
>TOOL NULL                ;sets the tool coordinates to be null
>DO ALIGN                 ;align the tool Z axis with the base Z axis
```

3. Using the Base Mode on the teach pendant, move the robot so that the center of the flange is perpendicular to the reference point.　Next, move the robot moving only along X, Y, Z of the base coordinates.　Enter as below so that the transformation values for that pose is assigned to variable “a”:

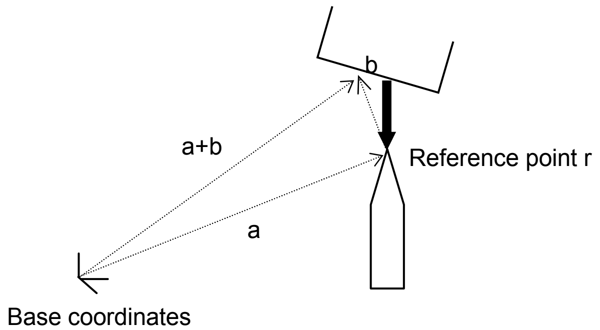
```
>HERE a
```



4. Install the tool to the flange, and move the tool center point (TCP) to the reference point so that the Z axis of the new tool coordinate is perpendicular to the X and Y axes of base coordinates.

   Enter as below to teach the transformation values at that pose as compound values "a+b":

   >HERE a+b ↵



5. From these compound values, the tool transformation values can be found as "-b". Enter:

   >POINT t=-b ↵

   This assigns the values of –b to the variable t.

6. Specify the tool transformation as t.

   >TOOL t ↵

7. To check, enter as following:

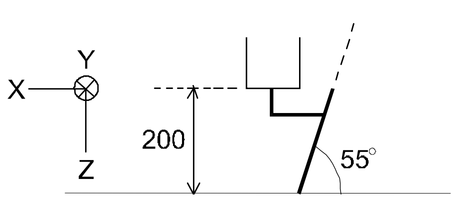
   >DO JMOVE r ↵

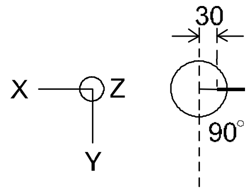
   The tool tip should move to the reference point r.

Once defined, all performances are based on this tool transformation, unless the tool is changed.

11.4.2 Tool Transformation Values-2 (When the Tool Size is Known)

When the tool size is known, the tool transformation values can be obtained as shown below. Values determined by this procedure are generally more accurate than those obtained in the former procedure. (See above 11.4.1.)





The XYZ axes in the above figure express the null tool coordinate. The following procedure sets tool coordinate origin at the tip of the torch and the Z axis in the same direction as the torch.

(1) Define the tool transformation value variable “torch” using the POINT command:

```
>POINT torch ↵
  X     Y     Z     O     A     T
  0     0     0     0     0     0

Change
>-30, 0, 200, 0, 35, 0 ↵
  X     Y     Z     O     A     T
-30     0   200     0    35     0
Change ↵
>
```

(2) Set the tool transformation values using the variable “torch”.

```
>TOOL torch ↵
```

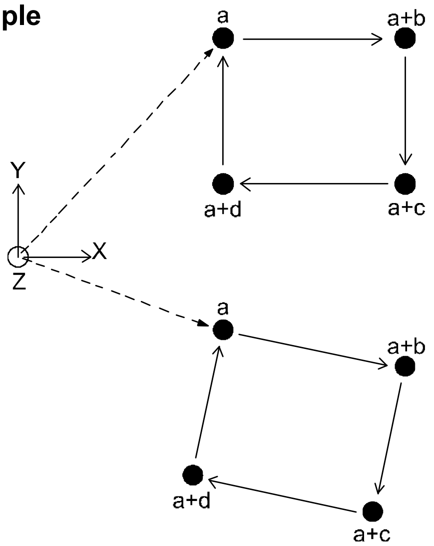
11.5 Relative Poses

11.5.1 Usage of Relative Poses

A pose can be defined relative to a reference point. When defined in this way, the relation between that pose and the reference point remains consistent even if the reference point is redefined.

For example, when the four corners of a table are taught, the pose relation between the robot and the table changes depending on where they are placed, but as long as the shape of the table remains the same, the relation of the four corners does not change. Therefore, if one of the corners is taught as a reference point for specifying the absolute pose relation between the robot and the table, and the other three corners are taught relative to the first corner, then, when the table is relocated, only the reference point has to be redefined.

Example

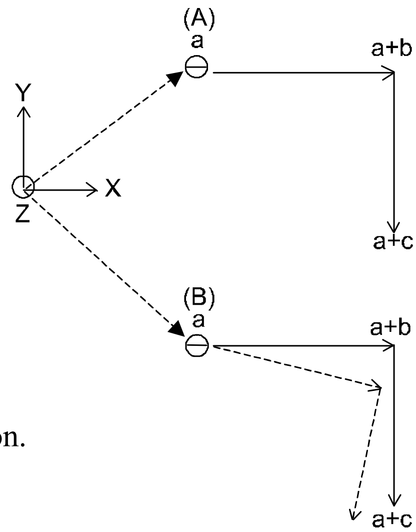


Teaching

```
>HERE   a ↵
>HERE   a+b ↵
>HERE   a+c ↵
>HERE   a+d ↵
```

Program

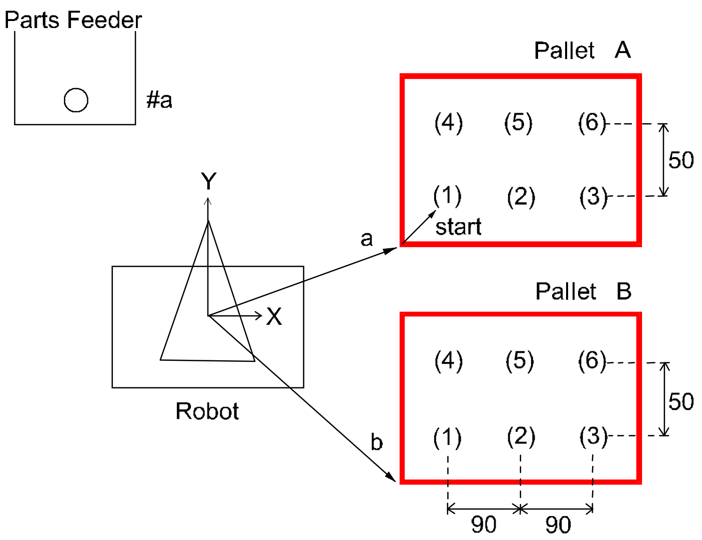
```
JMOVE   a
LMOVE   a+b
LMOVE   a+c
LMOVE   a+d
LMOVE   a
```



In figure (A) on the right, the reference point a and compound transformation values for the other corners are taught. Then, in figure (B) the reference point a is redefined. If the orientation of the robot tool is not reset (i.e. kept at the same orientation as it was in (A)), the robot will move in the trajectory shown in solid line. If the robot is supposed to move along the dotted line, it is necessary to redefine the orientation as well as the position.

11.5.2 Example of Program Using Relative Poses

In this example, the parts are palletized as in the previous example except two pallets are used. The pallets are placed separately but the relation between the reference point and the places the parts are to be put are the same on either pallet. This operation sets the parts from the Parts Feeder on to Pallet A. After six parts are set, the robot goes on to do the same with Pallet B. (The procedure of synchronizing with the Parts Feeder is omitted).



Poses to be taught

#a	: pose where robot picks up parts from the feeder
a	: reference pose on Pallet A
b	: reference pose on Pallet B
start	: pose of the first part on the pallet relative to the reference point

Program example

```
.PROGRAM   relative.test
;            Initial setting (2rows, 3columns, ∆X=90, ∆Y=50, etc.)
row.max=2
col.max=3
xs=90
ys=50
OPENI

flg=0                                              ; flg=0 : Pallet A, flg=1 : Pallet B
POINT   pallet=a
;            start palletizing
10           POINT   put=start
FOR   row1 TO row.max
FOR   col=1 TO col.max
JAPPRO #a,100
LMOVE   #a                                       picks up the part from the feeder
CLOSEI
LDEPART 100
;
POINT     put_pt=pallet+put
JAPPRO   put pt,200
LMOVE     put pt                                 places the part on the pallet
OPENI
LDEPART 200
;
POINT put=SHIFT(put BY xs,0,0)           ;finds the place of the part on the next column
END
;
POINT put=SHIFT(start BY 0,ys*row,0)     ;finds the place of the part on the next row
END
;
IF   flg<>0 GOTO 30     ; goes to finishing procedure when Pallet B is completed (flg=1)
flg=1
POINT pallet=b             ;defines the reference pose of Pallet B
GOTO 10
30           TYPE "*** end ***"
STOP
.END
```

11.6 Relative Pose Using the Frame Function

In the example in 11.5.1, the orientation of the tool had to be corrected when redefining the reference pose. That is not necessary if the FRAME function is used. Teach four points (b, c, d, e) to define the frame transformation value a. Points b and c determine the direction of the X axis, the third point d determines the XY plane, and point e the origin. After the points are taught, enter the following command:

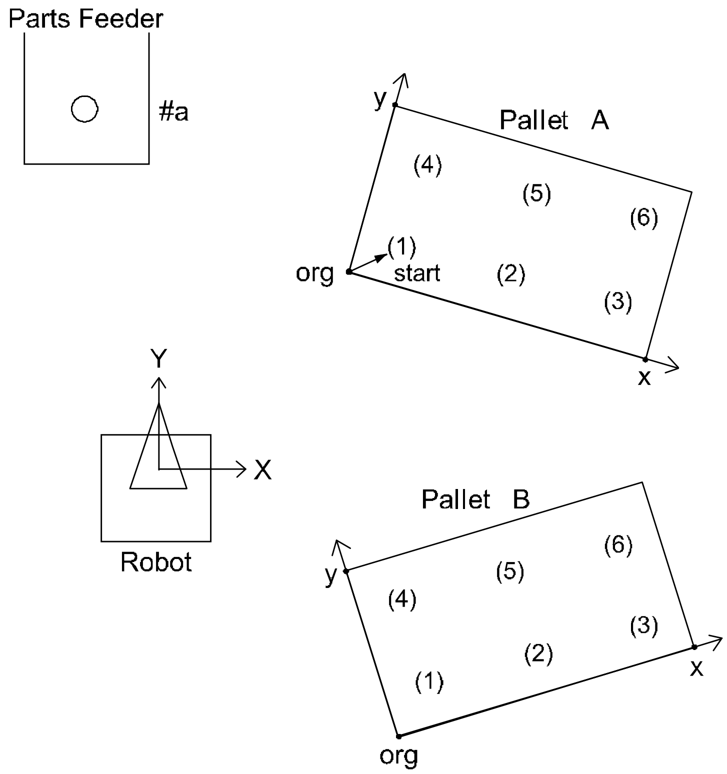
```
POINT a=FRAME(b,c,d,e)
```

Then, the relative coordinates are defined as the variable "a". The XYZ values shows the position of the origin of the relative coordinate and the OAT values show the orientation of the relative coordinates.

Hereafter, all the poses on the relative coordinates can be expressed as pose a+ □. If the place of the pallet changes, teach b, c, d, e again to redefine a in the same manner as above.

The relative coordinates defined using the FRAME function are also called the FRAME coordinates.

In the following sample program, the same operation as in 11.5.2 is performed using the frame coordinates.



The first procedure is to palletize the parts on Pallet A. Three corners of the pallet are taught, one as the origin, another as a point on the X axis and another as a point on the Y axis (see figure above). Execute the program below to palletize on Pallet A (note that after the points are defined, the rest of the program is the same as the previous sample program). To palletize on Pallet B, reteach the three corners and execute the same program. The frame coordinates will be redefined and the parts will be palletized on Pallet B as on Pallet A.

Program Example

```
.PROGRAM   frame.test
;          Initial setting (2 rows,3 columns, ΔX=90, Δ Y=50,etc.)
row.max=2
col.max=3
xs=90
yx=50
OPENI
;
POINT pallet=FRAME(org,x,y,org)          ;defines the frame coordinates of the pallet.
;                                          (3 points: for origin, for X/Y axes)
POINT   put=start                        ; starts palletizing.
FOR   row=1 TO row.max
FOR   col=1 TO col.max
JAPPRO   #a,100                 }
LMOVE    #a                     }
CLOSEI                          }         picks up the part from the parts feeder.
LDEPART 100                     }
;
POINT   put_pt=pallet+put
JAPPRO put_pt,200               }
LMOVE   put_pt                  }         places the part on the pallet.
OPENI                           }
LDEPART 200                     }
;
POINT put=SHIFT(put BY xs,0,0)           ; finds the place of the part on the next column.
END
;
POINT   put=SHIFT(startBY 0,ys*row,0)    ; finds the place of the part on the next row.
END
STOP
.END
```

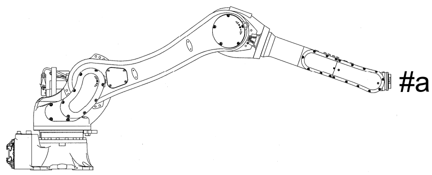
11.7 Setting Robot Configurations

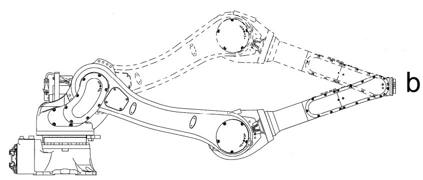
Most robots have six joints, and when a pose is taught using the joint displacement values, the displacement values of each of the six joints are given, so the pose of the robot is defined uniquely. On the other hand, when a pose is defined using the transformation values, there are cases, depending on the arm configuration of the robot, where more than one set of joint values gives the same pose specified by one transformation values. In AS, the robot basically keeps the configuration of the previous action, so no change in the robot configuration is needed. However, in the following cases, the robot's configuration should be specified by a configuration instruction:

1. When the robot moves from a point with unclear configuration to a point taught by transformation values,
2. When the 5th joint (the bent joint) passes through the origin (0°) in a SBS wrist configuration. (SBS: swivel, bend, swivel)

For example in the figure on the right,
if pose #a is defined with the configuration ABOVE
then the result of the instruction JMOVE b will be
ABOVE (dotted line in the figure) even if pose b is
originally defined BELOW.

```
JMOVE #a
JMOVE b
```

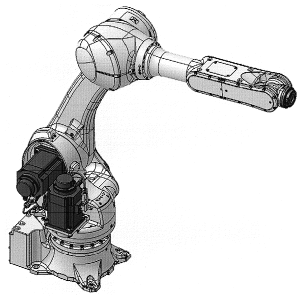


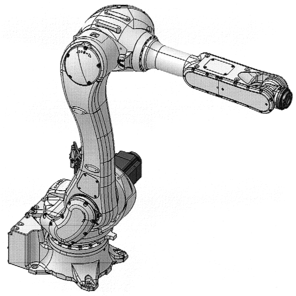


In the same way, if #a is defined UWRIST (JT5>0),
the configuration at pose b will be UWRIST regardless
of the configuration when that pose was taught.

To solve these problems, it is necessary to change the robot's configuration while it is in motion. Do this by executing a configuration instruction whenever a joint motion instruction ends in a point defined with transformation values (joint interpolated motion instructions: JMOVE, JAPPRO, JDEPART, DRIVE etc.). Six configuration instructions are listed here, a program example is given in the note box on the following pages.

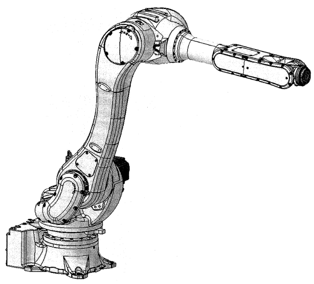
LEFTY, RIGHTY
Sets the configuration of the first three joints (JT1, JT2, JT3) of the robot. LEFTY sets the robot configuration to resemble a person's left arm, RIGHTY to resemble a right arm.

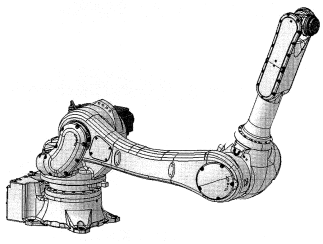

LEFTY


RIGHTY

ABOVE, BELOW

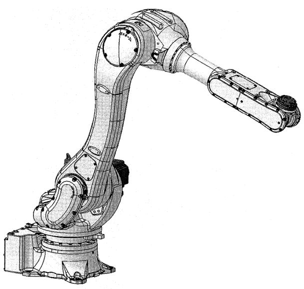
Sets the robot configuration so that the third joint (JT3) is in the above position (ABOVE), or below position (BELOW).

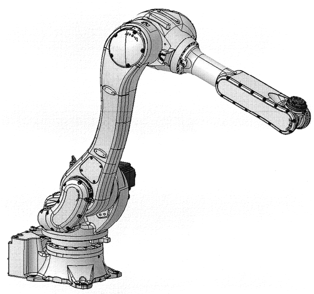

ABOVE


BELOW

UWRIST, DWRIST

Sets the configuration of the robot so that the value of the fifth joint (JT5) is positive (UWRIST) or negative (DWRIST) to acquire the same tool orientation.


UWRIST


DWRIST

[NOTE]

A singular point error occurs if the robot tries to move to a posture that is not structurally controllable.

(E6007) “Wrist can't be straightened any more (Singular point 1).”
When switching point of UWRIST/DWRIST and JT5 become 0 deg
(See system switch SINGULAR)

(E6008) “Wrist can't be bent any more (Singular point 2).”
When the arm is fully retracted

(E6016) “Robot arm stretching out (Singular Point 3).”
When the arm is fully extended

It may be possible to pass through the singular points by using system switch CONF_VARIABLE or option “Singular Point Motion Function”.

[NOTE]

Generally, configuration instructions do not have effect on joint displacement values (poses named with #), and the robot moves to the taught position in taught configuration. However, it results in error (E1089) "Cannot do linear motion in current configuration." when moving the robot in linear interpolated motion between poses where the configuration at the beginning differs from the configuration at the destination.

The robot does not react immediately to a configuration instruction. The configuration changes while executing the next joint interpolated motion (JMOVE, JAPPRO, JDEPART, DRIVE, etc.).

In regular programs, the configuration does not have to be changed except when it is changed on purpose. The configuration instructions are used in the following cases:

(1) When a program does not start with a motion instruction that moves the robot to a pose defined by joint displacement values, a configuration instruction should be written in the beginning of the program to determine the robot's configuration.

(2) When the JAPPRO instruction is used as below, configuration instruction should be used:

```
JMOVE    a
JAPPRO   #b,100
JMOVE    #b
```

The configuration after executing the motion instruction "JAPPRO #b,100" may, in some cases , differ from the configuration at #b. If the configuration of the wrist (the ± of the angle of JT5) is different, executing the next step, "JMOVE #b", may cause JT4 and JT6 to rotate greatly. A way to avoid this is to teach a pose 100 mm above pose #b as #bb and use the JMOVE instruction as follows:

```
JMOVE   a
JMOVE   #bb
JMOVE   #b
   :
```

Another way to avoid the large motion amount of JT4 and JT6 is to specify the wrist configuration using the configuration instruction (UWRIST, DWRIST). In this example, configuration in #b is assumed to be UWRIST (value of JT5 is positive).

```
JMOVE    a
UWRIST
JAPPRO   #b,100
JMOVE    #b
   :
```

The configuration of the wrist changes to UWRIST after "JAPPRO #b, 100" is executed, thus avoiding unnecessary rotation of joints 4 and 6 at execution of "JMOVE #b".

Appendix 1 Limitation of Signal Numbers

No	M, P, F*		Output Signals	Input Signals	Internal Signals
1	BITS	MP	1 to maxsig**	------	2001 to maxsig**
2	BITS	F	1 to maxsig**	1001 to maxsig**	2001 to maxsig**
3	DEFSIG	M	1 to maxsig**	1001 to maxsig**	------
4	DLYSIG	MP	1 to maxsig**	------	2001 to maxsig**
5	ON, ONI	P	------	1001 to 1256***	2001 to 2256***
6	PULSE	MP	1 to maxsig**	------	2001 to maxsig**
7	RUNMASK	P	1 to maxsig **	------	2001 to maxsig**
8	SIGNAL	MP	1 to maxsig**	------	2001 to maxsig**
9	SIG	F	1 to maxsig**	1001 to maxsig**	2001 to maxsig**
10	SWAIT	P	1 to maxsig**	1001 to maxsig**	2001 to maxsig**
11	XMOVE	P	------	1001 to 1256***	2001 to 2960
12					
13					

NOTE* M= monitor command, P= program instruction, F= function

NOTE** maxsig: number of I/O signals installed
16 (standard), maximum 960 (option)

NOTE*** Although being marked as 1001 to 1256, in standard specifications these signals become 1001 to 1016.

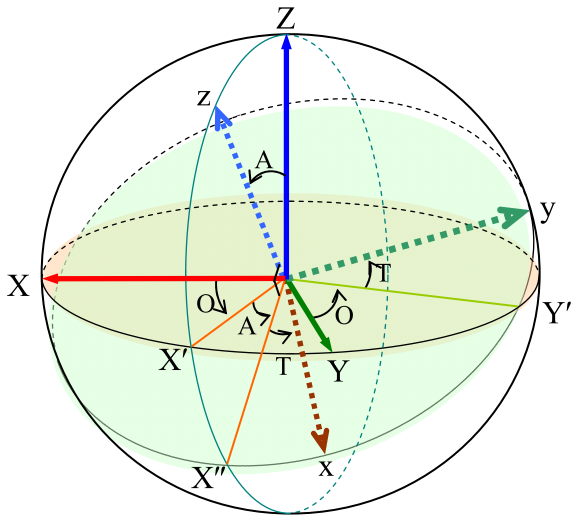
Appendix 2 ASCII Codes

ASCII character	Octal	Decimal	Hexa-decimal	Description
NULL	000	00	00	Null
SOH	001	01	01	Start of heading
STX	002	02	02	Start of text
ETX	003	03	03	End of text
EOT	004	04	04	End of transmission
ENQ	005	05	05	Enquiry
ACK	006	06	06	Acknowledge
BEL	007	07	07	Bell
BS	010	08	08	Backspace
HT	011	09	09	Horizontal tabulation
LF	012	10	0A	Line feed
VT	013	11	0B	Vertical tabulation
FF	014	12	0C	Form feed
CR	015	13	0D	Carriage return
SO	016	14	0E	Shift out
SI	017	15	0F	Shift in
DLE	020	16	10	Data link escape
DC1	021	17	11	Device control 1
DC2	022	18	12	Device control 2
DC3	023	19	13	Device control 3
DC4	024	20	14	Device control 4
NAK	025	21	15	Negative acknowledge
SYN	026	22	16	Synchronous idle
ETB	027	23	17	End of transmission block
CAN	030	24	18	Cancel
EM	031	25	19	End of medium
SUB	032	26	1A	Substitute
ESC	033	27	1B	Escape
FS	034	28	1C	File separator
GS	035	29	1D	Group separator
RS	036	30	1E	Record separator
US	037	31	1F	Unit separator
SP	040	32	20	Space

ASCII character	Octal	Decimal	Hexa-decimal	ASCII character	Octal	Decimal	Hexa-decimal
	040	32	20	0	060	48	30
!	041	33	21	1	061	49	31
”	042	34	22	2	062	50	32
#	043	35	23	3	063	51	33
$	044	36	24	4	064	52	34
%	045	37	25	5	065	53	35
&	046	38	26	6	066	54	36
’	047	39	27	7	067	55	37
(	050	40	28	8	070	56	38
)	051	41	29	9	071	57	39
*	052	42	2A	:	072	58	3A
+	053	43	2B	;	073	59	3B
‘	054	44	2C	<	074	60	3C
–	055	45	2D	=	075	61	3D
.	056	46	2E	>	076	62	3E
/	057	47	2F	?	077	63	3F

ASCII character	Octal	Decimal	Hexa-decimal	ASCII character	Octal	Decimal	Hexa-decimal
@	100	64	40		140	96	60
A	101	65	41	a	141	97	61
B	102	66	42	b	142	98	62
C	103	67	43	c	143	99	63
D	104	68	44	d	144	100	64
E	105	69	45	e	145	101	65
F	106	70	46	f	146	102	66
G	107	71	47	g	147	103	67
H	110	72	48	h	150	104	68
I	111	73	49	i	151	105	69
J	112	74	4A	j	152	106	6A
K	113	75	4B	k	153	107	6B
L	114	76	4C	l	154	108	6C
M	115	77	4D	m	155	109	6D
N	116	78	4E	n	156	110	6E
O	117	79	4F	o	157	111	6F
P	120	80	50	p	160	112	70
Q	121	81	51	q	161	113	71
R	122	82	52	r	162	114	72
S	123	83	53	s	163	115	73
T	124	84	54	t	164	116	74
U	125	85	55	u	165	117	75
V	126	86	56	v	166	118	76
W	127	87	57	w	167	119	77
X	130	88	58	x	170	120	78
Y	131	89	59	y	171	121	79
Z	132	90	5A	z	172	122	7A
[	133	91	5B		173	123	7B
¥	134	92	5C		174	124	7C
]	135	93	5D		175	125	7D
	136	94	5E		176	126	7E
–	137	95	5F	DEL	177	127	7F

Appendix 3 Euler’s OAT Angles



The orientation of a coordinate system Σ(x,y,z) with respect to the base coordinate system Σ(X,Y,Z) is generally expressed using the Euler’s OAT angles. As shown in the figure above, the three angles can be defined as follows. In the figure above, the two coordinate systems Σ(x,y,z)and Σ(X,Y,Z)share the same origin.

O: The angle between Zz plane and XZ plane
A: The angle between z axis and Z axis
T: The angle between x axis and X″ axis
X″ axis is on the Zz plane and the angle between this axis and the z axis is 90°.

These three angles can also be said to represent the angles of rotations necessary for the base coordinate system Σ(X,Y,Z) to coincide with the coordinate system Σ(x,y,z). The order of rotation must not be changed, or else will result differently.

1. O rotation of the coordinate system Σ(X,Y,Z) around Z axis. (This moves Σ(X,Y,Z) to Σ(X′,Y′,Z).)
2. A rotation of the coordinate system Σ(X′,Y′,Z) around Y′ axis. (This moves Σ(X′,Y′,Z) to Σ(X″,Y′,z).)
3. T rotation of the coordinate system Σ(X″,Y′,z) around z axis. (This moves Σ(X″,Y′,z) to Σ(x,y,z).)

Furthermore, this can be considered in terms of polar coordinate values. When point P, which is on the z axis at the distance of d from the origin, is written as (d, A, O), then O and A in these coordinate values are equal to O and A described above. The direction of the z axis is expressed by these two values.

Appendix 4 Error Message List

Code	Error Message
P0100	Illegal input data.
P0101	Too many arguments.
P0102	Input data is too big.
P0103	Illegal PC number.
P0104	Illegal Robot number.
P0105	Illegal program.
P0106	Illegal priority.
P0107	Invalid coordinate value.
P0108	Syntax error.
P0109	Invalid statement.
P0110	Specify full spelling of command.
P0111	Cannot use this command/instruction in current mode.
P0112	Cannot execute with DO command.
P0113	Not a program instruction.
P0114	Illegal expression.
P0115	Illegal function.
P0116	Illegal argument of function.
P0117	Invalid variable (or program) name.
P0118	Illegal variable type.
P0119	Incorrect array suffix.
P0120	Incongruent num. of parenthesis.
P0121	Expected to be a binary operator.
P0122	Illegal constant.
P0123	Illegal qualifier.
P0124	Invalid label.
P0125	Missing expected character.
P0126	Illegal switch name.
P0127	Specified switch name needs full spelling.
P0128	Illegal format specifier.
P0129	Duplicate statement label.
P0130	Cannot define as array.
P0131	No. of dimensions in array exceeds 3.
P0132	Array variable already exists.
P0133	Non array variable exists.
P0134	Array variable expected.

Code	Error Message
P0135	Local variable expected.
P0136	Unexpected suffix.
P0137	Mismatch of arguments at subroutine call.
P0138	Mismatch of argument type at subroutine call.
P0139	Illegal control structure.
P0140	Step:XX Wrong END statement.
P0141	Step:XX Extra END statement.
P0142	Step:XX Cannot terminate DO with END.
P0143	Step:XX No VALUE statement after CASE.
P0144	Step:XX Preceding IF missing.
P0145	Step:XX Preceding CASE missing.
P0146	Step:XX Preceding DO missing.
P0147	Step:XX Cannot find END of XX.
P0148	Step:XX Too many control structures.
P0149	Variable (or program) already exists.
P0150	Variable of different type already exists.
P0151	Internal buffer over due to complicated expression.
P0152	Undefined variable (or program).
P0153	Illegal clock value.
P0154	Missing '='.
P0155	Missing ')'.
P0156	Missing ']'.
P0157	Missing "TO".
P0158	Missing "BY".
P0159	Missing ':'.
P0160	Specify "ON" or "OFF".
P0161	Robot Num. must be specified.
P0162	Cannot modify position data in this instruction.
P0163	Name of program, variable, file, etc. misspecified.
P0164	Illegal Robot network ID.
P0165	Step:XX No SVALUE statement after SCASE.
P0166	Step:XX Preceding SCASE missing.
P1000	Cannot execute program because motor power is OFF.
P1001	Cannot execute program in TEACH mode.
P1002	Cannot execute program because teach lock is ON.
P1003	Cannot execute program because of EXT. HOLD input.
P1004	Cannot execute program being reset.

Code	Error Message
P1005	Cannot execute program because of EXT. START ENABLE.
P1006	Cannot execute program because of EXT. START DISABLE.
P1007	Start signal was not input at a RPS_END step.
P1008	Cannot execute program, HOLD sw. engaged.
P1009	Program is already running.
P1010	Robot control program is already running.
P1011	Cannot continue this program. Use EXECUTE.
P1012	Robot is moving now.
P1013	Cannot execute because in error now. Reset error.
P1014	Cannot execute because program already in use.
P1015	Cannot delete, in use by another command.
P1016	Cannot delete, used in program.
P1017	Cannot delete a program in Editor.
P1018	KILL or PCKILL to delete program.
P1019	PC program is running.
P1020	Cannot operate, teach pendant in operation.
P1021	Cannot execute with DO command.
P1022	Cannot execute with MC instruction.
P1023	Cannot execute in Robot program.
P1024	Statement cannot be executed.
P1025	Cannot be executed, function not set.
P1026	Cannot KILL program that is running.
P1027	Cannot edit program, teach lock is ON.
P1028	Cannot paste.
P1029	Program name not specified.
P1030	Program interlocked by other procedure.
P1031	No free memory.
P1032	No program step.
P1033	Program name already exists.
P1034	This program is not editable.
P1035	Record inhibited. Set [Record Accept] and operate again.
P1036	Program change inhibited. Set [Accept] and operate again.
P1037	Program name cannot be called "calib_load_".
P1038	Program does not exist.
P1039	Teach pendant is not connected.
P1040	Cannot execute this command in I/F panel.
P1041	Auto monitor command failure.

Code	Error Message
P1042	NUM program is running.
P1043	Cannot execute in REPEAT mode.
P1044	Cannot execute on because motor power is ON.
P1045	Set TEACH mode and teach lock ON.
P1046	Turn on trigger switch.
P1047	The disconnected robot cannot select the program/step.
P1048	Cannot operate during execution of brake check.
P1049	Program is locked.
P1050	Exist protected program.
P1051	Cannot unlock protection while program running.
P1052	Because the memory was full, could not copy the program.
P1053	Because the memory was full, the copy of program was suspended.
P1054	Turn off trigger switch.
P1055	Teach the axis lock instruction at the step of clamp ON.
P1056	Teach the axis unlocking instruction at the step of clamp ON.
P2000	Turn OFF motor power.
P2001	Turn HOLD/RUN sw. to HOLD.
P2002	There is no external axis.
P2003	Illegal positioner type.
P2004	Cannot change, user data already exists.
P2005	Graphic area error.
P2006	Option is OFF.
P2007	Cannot execute because executed by other device.
P2008	Device is not ready.
P2009	Illegal file name.
P2010	Disk is not ready.
P2011	Invalid disk format.
P2012	Disk is write-protected.
P2013	Disk full.
P2014	Too many files.
P2015	Cannot write on read-only file.
P2016	Cannot open the file.
P2017	Cannot close the file.
P2018	Storage data logging now.
P2019	ADC function already in use.
P2020	Illegal device number.
P2021	Cannot execute on this terminal.

Code	Error Message
P2022	Cannot use DOUBLE OX.
P2023	In cooperative mode.
P2024	Invalid coordinate value X.
P2025	Invalid coordinate value Y.
P2026	Invalid coordinate value Z.
P2027	Cannot use signal, already used in I/F panel.
P2028	Arm ID board is busy.
P2029	Axis setting data is incorrect.
P2030	Unknown Aux. function number.
P2031	Deleted step was destination step of Jump, Call instruction.
P2032	Incorrect number input as WHERE parameter.
P2033	Logging is running.
P2034	Undefined memory.
P2035	Non data.
P2036	Memory verify error.
P2037	Real time path modulation is already running.
P2038	Matrix calculation error.
P2039	Cannot start cycle from FN instruction.
P2040	Card is not ready.
P2041	Wrong card loaded.
P2042	Card is write-protected.
P2043	Card battery is low voltage.
P2044	Card is not formatted.
P2045	Cannot format this card.
P2046	Card initialization error.
P2047	File is already open.
P2048	File does not exist in card.
P2049	Attempted to open too many files.
P2050	Unexpected error during card access.
P2051	Illegal sequence numbers for file I/O data.
P2052	[LSEQ]Program includes unavailable instruction.
P2053	[LSEQ]Too many steps exist.
P2054	[LSEQ]Invalid type of signal variable.
P2055	[LSEQ]Program is already running.
P2056	[LSEQ]No.of signal is outside specifiable range.
P2057	[SerialFlash]Cannot open file.
P2058	[SerialFlash]Data read error.

Code	Error Message
P2059	[SerialFlash]Data write error.
P2060	[SerialFlash]File or directory doesn't exist.
P2061	File does not exist in floppy.
P2062	[FDD/PC_CARD]Failure in writing data per verify function.
P2063	[FDD/PC_CARD]Faulty response from verify function.
P2064	[FDD]No space available.
P2065	[Multi Disks]Invalid disk was loaded.
P2066	Boot flash state is write-disenable.
P2067	[Serial Flash]File directory error.
P2068	Cannot execute program being edited now.
P2069	[FDD/PC_CARD]Device already in use.
P2070	No more data can be registered.
P2071	C/S switch set to disable.
P2072	[LSEQ]Maximum cycles of execution.
P2073	[LSEQ]Other program is waiting execution.
P2074	Floppy disk is broken.
P2075	Channel number for JtXX is incorrect.
P2076	SAVE/LOAD in progress.
P2077	[Serial Flash]Access error occurred.
P2078	[Serial Flash]Upload or Download was aborted.
P2079	Card full.
P2080	Can not execute because of the channel assigned joint No.
P2083	User log is not created.
P2084	The number of registration of a user log was changed.
P2085	Cannot register user log, no free memory.
P2086	User log data is not registered.
P2087	The kind of user log data and specified data is different.
P2088	Cannot load the improper compensation parameter.
P2090	No servo data of the servo spec.
P2091	[Serial Flash]The file or directory already exists.
P2092	[Serial Flash]The directory is not yet empty.
P2093	[Serial Flash]There is no space to write.
P2094	[Serial Flash]Cannot access the file for read only.
P2095	No response from option CPU board.
P2096	Cannot execute cycle start after palletizing motion aborted.
P2097	Cannot change steps during palletizing motion.
P2098	The axis is not for endless rotation.

Code	Error Message
P2099	Cannot change palletizing state into ON.
P2100	Macro error.
P2101	Nesting is too deep in include file.
P2102	File or folder is missing.
P2103	USB memory is not inserted.
P2104	Failed to download softwares.
P2105	Available USB memory is low.
P2106	Available compact flash memory is low.
P2107	System is now downloading the software.
P2108	There is no software in the USB memory.
P2109	Cannot execute program because of simultaneously operation sig. inputting.
P2110	[USB/CF]File write error.
P2111	Please return spin-axis to the original position.
P2112	File name is too long.
P2113	Cannot start cycle from KI instruction.
P4500	FIELD-BUS)Interface not enabled.
P4501	DEVNET)Node XX not in the scanlist.
P4502	DEVNET)Already in that mode.
P4503	Duplicate signal number.
P4504	FIELD-BUS)signals limit over.(max. XX)
P4505	CC-LINK)Version mismatch.
P4506	EN/IP-M)Already in specified mode.
P4507	FIELD-BUS)Cannot execute with old ANYBUS card firmware.
P4508	FIELD-BUS)Cannot communicate with interface card.
P4509	FIELD-BUS)Wrong interface card type error.
P4510	FIELD-BUS)Initialization of the card is not complete.
P4706	Cannot use this data by Signal Allocation Setting.
P5000	Waiting weld completion.
P5001	Waiting retract or extend pos. input signal.
P5002	Spot sequence is running.
P5003	External-axis type and Gun type data mismatch.
P6000	Shifted location of STEPXX is out of range.
P6001	STEPXX in source program is out of motion range.
P6002	Specified painting data bank does not exist.
P6003	Cannot execute program because of suspend playback.
P6004	Cannot execute because of the Air Purge sequence.
P6005	Cannot execute because robot is disconnected.

Code	Error Message
P6006	Cannot specify circular move to end point of spraying path.
P6007	Number of taught points for spraying path exceeds the limit.
P6008	Number of instructions between points exceeds the limit.
P6009	Shortage in number of taught points for spraying path.
P6010	Selected program other than pgxxx.
P6011	Cannot move, change to joint interpolation or add points.
P6012	Cannot edit program, TEACH LOCK is OFF.
P6500	Cannot generate working line direction.
P6501	Illegal tool posture.
P6502	No weld database.
P6503	Cannot change weld condition.
P6504	Step:XX Preceding L.START missing.
P6505	The axis type is not set as the servo torch.
P6506	Shift function can not be used in CIR interp.
P7000	Cannot program reset, because not at home position 1.
P7001	In force meas. mode, only NOP Interp. avail.
P7002	Cannot change stroke because clamp on now.
P7003	Servo parameter file is not found.
P7500	Turn motor power on.
P7501	Because of repeat mode,wait to teach mode.
P7502	Over the number which can be registered in interruption.
P7503	Cannot execute program in error masking.
P7504	ONC/ONCI channel has already received.
P7505	Cannot execute in saving.
P7506	Cannot accept a record because robot is moving.
P7507	Amount of the data change is too large in repeat operation.
P8400	Cannot execute program because of CLAMP MODE sig. inputting.
P8800	The controller number is duplicated.
P8801	The IL robot number is duplicated.
P8802	The IL server is processing.
P8803	The connection with the IL server is disabled.
P8804	IL server IP address is not yet set.
P8805	Set the mode to Teach.
P8806	Turn off servo.
P8807	ILL)Communication time out error.
P8808	ILL)Time out error for PC server processing.
P8809	ILL)PC server processing demand completion waiting is time out error.

Code	Error Message
P8810	ILL)Error in auto interlock setting system.
P8811	ILL)Could not release operation lock for slave controller.
P8812	ILL)Could not communicate with the PC server.
P8813	The IL robot number is unregistered.
P9000	Unacceptable control-direction.
P9001	Unacceptable control-distance.
P9002	Same data are specified for some reference points.
P9003	Reference points1,2,3 are on a straight line.
P9004	Reference point 4 is out of allowed range.
P9005	Cannot manipulate because teach lock is ON.
W1000	Cannot move along straight line JtXX in this configuration.
W1001	Over maximum joint speed in check. Set low speed.
W1002	Operation log information was cleared.
W1003	Calibration failed. Retry after changing posture.
W1004	JtXX out of motion range. Check operational area.
W1005	Illegal center of gravity, default parameter is set.
W1006	Incorrect load moment. Default parameter is set.
W1007	Application setting changed. Turn OFF & ON the control power.
W1008	Parameter changed. Turn OFF & ON the control power.
W1009	Position envelope error of JtXX at last E-stop.
W1010	RAM battery low voltage.
W1011	PLC alarm. (XX)
W1012	To change the servo parameter, turn the controller power OFF and then ON.
W1013	Encoder battery low voltage. [Servo(XX)]
W1014	Number of axes changed. Reinitialize.
W1015	Possibility of failure.
W1016	The motor torque exceeds the limit value. JTXX
W1017	Encoder battery low voltage.[External axis(XX)]
W1018	Network parameter is changed. Turn OFF & ON the control power.
W1019	The registered value is beyond rated load.
W1020	Error sector was found.
W1021	The optimal posture can't be found at present location.
W1022	Not execute ZRPAADSET command.
W1023	Teach Plug Position wrong or P-N low voltage.[XX]
W1024	Deviation from last stop position exceeds the limit set.
W1025	(SSCNET)Excessive regenerative warning of JtXX.(CodeXX)
W1026	(SSCNET)Motor overload warning of JtXX.(CodeXX)

Code	Error Message
W1027	While lifter is locked, it cannot move.
W1028	The center of gravity for payload is over limit. Reduction gears could be broken.
W1029	The center of gravity for payload is over limit. Use the Jt5 at zero degree only.
W1030	Braking torque of JtXX has decreased.
W1031	Cannot move along straight line unless JtXX value is 0 degree.
W1032	Cannot move straight - the flange faces direction of upper sphere.
W1033	Cannot change orientation.
W1034	Encoder power voltage is low. JtXX
W1035	Encoder battery voltage is low. Check zeroing . JtXX
W1036	Step data is different.
W1037	The axis is not for endless rotation.
W1038	Encoder rotation data is abnormal.(JtXX)
W1039	Encoder response error.(JtXX)
W1040	Encoder communication error.(JtXX)
W1041	Speed error JtXX.
W1042	Encoder rotation speed exceeded limit (JtXX)
W1043	Encoder temperature exceeded limit (JtXX)
W1044	Velocity envelope error in endless rotation axis.(JtXX)
W1045	Abn. curr feedback JtXX. (Amp fail, pwr harness disconnect)
W1046	Encoder ABS-track error.(JtXX)
W1047	Encoder INC-pulse error.(JtXX)
W1048	No. of encoder errors exceeded limit JtXX.
W1049	RSC)TCP communication error occurred.(Code:XX)
W1050	RSC)Command value output communication error occurred. (Code:XX)
W1051	RSC)USB communication initialize error occurred.(Code:XX)
W1052	RSC)RC parameter generation error occurred.(Code:XX)
W1053	(FANXX-XX)Rotational speed of fan is below the limit. (ServoBoardXX)
W1054	AVR reaches the expected lifetime soon.
W1055	Vision cycle timer over.
W1056	[Main CPU board] CPU temperature exceeded the limit. (XX 1/1000°C)
W1057	Cannot move along straight line tool in this configuration.
W1058	Link3 interferes in ground.
W1059	Link5 interferes in base.
W1060	Link6 interferes in base.
W1061	TP changed. Confirm current pose, and operate the robot.
W1062	The TP backlight lighting time exceeded the limit.
W1063	The number of ON/OFF operations of MC relay exceeded the limit.(SrvB'dXX)(MCXX)

Code	Error Message
W1064	Exceeded the limit.(Parts:XX)
W2901	SLOGIC ERROR MESSAGE #1
W2902	SLOGIC ERROR MESSAGE #2
W2903	SLOGIC ERROR MESSAGE #3
W2904	SLOGIC ERROR MESSAGE #4
W2905	SLOGIC ERROR MESSAGE #5
W2906	SLOGIC ERROR MESSAGE #6
W2907	SLOGIC ERROR MESSAGE #7
W2908	SLOGIC ERROR MESSAGE #8
W2909	SLOGIC ERROR MESSAGE #9
W2910	SLOGIC ERROR MESSAGE #10
W2911	SLOGIC ERROR MESSAGE #11
W2912	SLOGIC ERROR MESSAGE #12
W2913	SLOGIC ERROR MESSAGE #13
W2914	SLOGIC ERROR MESSAGE #14
W2915	SLOGIC ERROR MESSAGE #15
W2916	SLOGIC ERROR MESSAGE #16
W2917	SLOGIC ERROR MESSAGE #17
W2918	SLOGIC ERROR MESSAGE #18
W2919	SLOGIC ERROR MESSAGE #19
W2920	SLOGIC ERROR MESSAGE #20
W2921	SLOGIC ERROR MESSAGE #21
W2922	SLOGIC ERROR MESSAGE #22
W2923	SLOGIC ERROR MESSAGE #23
W2924	SLOGIC ERROR MESSAGE #24
W2925	SLOGIC ERROR MESSAGE #25
W2926	SLOGIC ERROR MESSAGE #26
W2927	SLOGIC ERROR MESSAGE #27
W2928	SLOGIC ERROR MESSAGE #28
W2929	SLOGIC ERROR MESSAGE #29
W2930	SLOGIC ERROR MESSAGE #30
W2931	SLOGIC ERROR MESSAGE #31
W2932	SLOGIC ERROR MESSAGE #32
W2933	SLOGIC ERROR MESSAGE #33
W2934	SLOGIC ERROR MESSAGE #34
W2935	SLOGIC ERROR MESSAGE #35
W2936	SLOGIC ERROR MESSAGE #36

Code	Error Message
W2937	SLOGIC ERROR MESSAGE #37
W2938	SLOGIC ERROR MESSAGE #38
W2939	SLOGIC ERROR MESSAGE #39
W2940	SLOGIC ERROR MESSAGE #40
W2941	SLOGIC ERROR MESSAGE #41
W2942	SLOGIC ERROR MESSAGE #42
W2943	SLOGIC ERROR MESSAGE #43
W2944	SLOGIC ERROR MESSAGE #44
W2945	SLOGIC ERROR MESSAGE #45
W2946	SLOGIC ERROR MESSAGE #46
W2947	SLOGIC ERROR MESSAGE #47
W2948	SLOGIC ERROR MESSAGE #48
W2949	SLOGIC ERROR MESSAGE #49
W2950	SLOGIC ERROR MESSAGE #50
W2951	SLOGIC ERROR MESSAGE #51
W2952	SLOGIC ERROR MESSAGE #52
W2953	SLOGIC ERROR MESSAGE #53
W2954	SLOGIC ERROR MESSAGE #54
W2955	SLOGIC ERROR MESSAGE #55
W2956	SLOGIC ERROR MESSAGE #56
W2957	SLOGIC ERROR MESSAGE #57
W2958	SLOGIC ERROR MESSAGE #58
W2959	SLOGIC ERROR MESSAGE #59
W2960	SLOGIC ERROR MESSAGE #60
W2961	SLOGIC ERROR MESSAGE #61
W2962	SLOGIC ERROR MESSAGE #62
W2963	SLOGIC ERROR MESSAGE #63
W2964	SLOGIC ERROR MESSAGE #64
W2965	The max load is XX of a permissible torque.
W2966	Load exceeded permissible torque.
W2967	Load exceeded max torque.
W2968	Please set a group number as XX.
W3800	Encoder battery voltage decrease.
W3801	Because the brake has been released, it is not possible to move.
W3802	Maint. period elapsed, maint. is required.
W3803	Total power ON time exceeded limit, maint. is required.
W3804	Total robot connection time exceeded limit, maint. is required.

Code	Error Message
W3805	Total servo ON time exceeded limit, maint. is required.
W3806	Total JtXX the total travel dist. exceeded limit, maint. is required.
W3807	Total times of MC ON exceeded limit, maint. is required.
W3808	Total times of servo ON exceeded limit, maint. is required.
W3809	Total times of E-stop exceeded limit, maint. is required.
W3810	JtXX total (curr)^3-motion dist. exceeded limit, maint. is required.
W3811	JtXX RMS value of current exceeded limit, maint. is required.
W3812	Power supply(1) for input to NO.XX I/O board is abnormal.
W3813	Power supply(2) for input to NO.XX I/O board is abnormal.
W3814	Power for output from NO.XX I/O board is abnormal or Fuse blown.
W4000	No response from PLC to Break down info writing.
W4001	Break down info cannot write on PLC[EC = XX].
W4002	Break down info writing cannot receive correct answer.
W4500	FIELD-BUS)Slave port OFFLINE.
W4501	FIELD-BUS)Master port OFFLINE.
W4502	CC-LINK)Data link error on Master board. XX
W5000	Release wait cond., in force measurement mode.
W5001	PLC communication error.
W5002	Weld controller XX not connected.
W5003	Weld controller XX no response.
W5004	Weld controller XX response error.
W5005	(Spot welding)No response from RWC XX.
W5006	(Spot welding)RWC response error XX.
W5007	(Spot welding)Weld fault XX.
W5008	(Spot welding)Cable disconnection error XX.
W5009	(Spot welding)Internal leak XX.
W5010	(Spot welding)Main cable exchange alarm XX.
W5011	(Spot welding)No connection with RWC XX.
W5012	Cannot achieve set force.
W5013	Tip wear over the limit. (MOVING SIDE)
W5014	Tip wear over the limit. (FIXED SIDE)
W5015	(Spot welding)Welding current has decreased.
W5016	Weld warning has arisen.(CodeXX)
W6000	Grease reduction gears and motor bearings.
W6001	Replace the robot main cable.
W6002	Replace the cooling fans in the controller.
W6003	Replace the DC power supply in the controller.

Code	Error Message
W6004	Replace the servo power unit.
W6005	Replace the power amplifier for the robot arm.
W6006	Replace the power amplifier for the robot wrist.
W6007	Replace the power amplifier for the traveller.
W6008	Exp interlock is disabled by jumper wiring.
W6009	Not selected Internal pressure Explosion-proof.
W6010	Disable Gun Relative Distance Check (ID:XX).
W6011	Shutter release signal variable logging error.
W7000	Cannot operate excluding the servo welding gun axis. Because the pressure measurement mode.
W7001	Detected board gap quantity error.
W7002	Detected board gap quantity error.
W7003	The foreign body was detected in the Tip Dress.
W7004	Value of auto collect exceeds work abnormality level.
W7500	Can't continue check motion because separated from last pos.
W7501	Cannot execute a program because of LOW voltage.
W8400	Cannot achieve set force of JtXX.
W8800	Command value almost exceeds virtual safety fence.(SphereXX, LineXX)
W8801	Command value almost exceeds virtual safety fence.(SphereXX, ZUpper)
W8802	Command value almost exceeds virtual safety fence.(SphereXX, ZLower)
W8803	Command value almost invades restricted space.(SphereXX, Part.XX LineXX)
W8804	Command value almost invades restricted space.(SphereXX, Part.XX ZUpper)
W8805	Command value almost invades restricted space.(SphereXX, Part.XX ZLower)
W8806	Command value almost exceeds virtual safety fence.(ToolBox, LineXX)
W8807	Command value almost exceeds virtual safety fence.(ToolBox, ZUpper)
W8808	Command value almost exceeds virtual safety fence.(ToolBox, ZUpper)
W8809	Command value almost invades restricted space.(ToolBox, Part.XX)
W8810	Command value almost exceeds virtual safety fence.(LinkXX, LineXX)
W8811	Command value almost exceeds virtual safety fence.(LinkXX, ZUpper)
W8812	Command value almost exceeds virtual safety fence.(LinkXX, ZLower)
W8813	Command value almost invades restricted space.(LinkXX, Part.XX LineXX)
W8814	Command value almost invades restricted space.(LinkXX, Part.XX ZUpper)
W8815	Command value almost invades restricted space.(LinkXX, Part.XX ZLower)
W8851	Detected area interference.
W8852	Detected arm interference.(XX, XX)
W8853	ILL)Detected arm interference.(XX, XX)
W8854	ILL)Communication time out error.
W8855	ILL)Sequence processing demand completion waiting is time out error.
W8856	ILL)Sequence processing completion waiting is time out error.

Code	Error Message
W8857	ILL)Sequence processing system error.
W8858	ILL)Create/Set processing completion waiting is time out error.
W8859	ILL)Inter lock less function system error.
W8860	[ARM CONTROL b'd]Data received from IL server is invalid.
W8900	Can not operate because motion limitation signal was input.
E0001	Unknown error.
E0002	[Servo boardXX]CPU BUS error.
E0100	Abnormal comment statement exists.
E0101	Nonexistent label.
E0102	Variable is not defined.
E0103	Location data is not defined.
E0104	String variable is not defined.
E0105	Program or label is not defined.
E0106	Value is out of range.
E0107	No array suffix.
E0108	Divided by zero.
E0109	Floating point overflow.
E0110	String too long .
E0111	Attempted operation with neg. exponent.
E0112	Too complicated expression.
E0113	No expressions to evaluate.
E0114	SQRT parameter is negative.
E0115	Array suffix value outside range.
E0116	Faulty or missing argument value.
E0117	Incorrect joint number.
E0118	Too many subroutine calls.
E0119	Nonexistent subroutine.
E0120	No destination program.
E0121	Cannot specify the jump source program as jump destination.
E0900	Block step instruction check sum error.
E0901	Step data is broken.
E0902	Expression data is broken.
E0903	Check sum error of system data.
E1000	ADC channel error.
E1001	ADC input range error.
E1002	PLC interface error.
E1003	Built-in PLC is not installed.

Code	Error Message
E1004	INTER-bus board is not ready.
E1005	Spin axis encoder difference error.
E1006	Touch panel switch is short-circuited.
E1007	Power sequence board is not installed.
E1008	Second Power sequence board is not installed.
E1009	No.XX I/O board is not installed.
E1010	Power sequence detects error.
E1011	Built-in sequence board is not installed.
E1012	RI/O board or C-NET board is not installed.
E1013	INTER-BUS board is not installed.
E1014	Dual port memory for communication is not installed.
E1015	Amp Interface board is not installed.(Code=XX)
E1016	No.XX CC-LINK board is not installed.
E1017	PLC error.Error code is Hex.XX.
E1018	INTER-BUS status error.
E1019	Power sequence board for safety unit is not installed.
E1020	External equipment is abnormal.
E1021	Arm ID board error. (CodeXX)
E1022	Power sequence board error. (CodeXX)
E1023	Communication error in robot network.
E1024	EXT.AXIS release sequence error.(CodeXX)
E1025	EXT.AXIS connect sequence error.(CodeXX)
E1026	Main CPU ID mismatch.
E1027	Safety circuit was cut OFF.
E1028	JtXX motor overloaded.
E1029	Encoder rotation data is abnormal.(JtXX)
E1030	Encoder data is abnormal.(JtXX)
E1031	Miscount of encoder data.(JtXX)
E1032	Mismatch ABS and INC encoder data(JtXX).
E1033	Encoder line error of (JtXX).
E1034	Encoder initialize error (JtXX).
E1035	Encoder response error(JtXX).
E1036	Encoder communication error.(JtXX)
E1037	Encoder data conversion error.(JtXX)
E1038	Encoder ABS-track error.(JtXX)
E1039	Encoder INC-pulse error.(JtXX)
E1040	Encoder MR-sensor error. (JtXX)

Code	Error Message
E1041	Limit switch (JtXX) is ON.
E1042	Limit switch signal line is disconnected.
E1043	Teach Plug is abnormal.
E1044	Destination position is out of the specified area.
E1045	(Spot welding)Gun-clamp mismatch.
E1046	Too short distance between start point and end point.
E1047	Axis number is not for conveyor follow mode.
E1048	Offset data of zeroing is illegal value.
E1049	Current position is out of the specified area.
E1050	Encoder and brake power off signal not dedicated.
E1051	Incorrect double OX output.
E1052	Work sensing signal is not dedicated.
E1053	Work sensing signal already input.
E1054	Cannot execute motion instruction.
E1055	Start point position error for circle.
E1056	MASTER robot already exists.
E1057	Check to which robot MASTER/ALONE were instructed.
E1058	SLAVE robot already exists.
E1059	Not an instruction for cooperative motion.
E1060	Cannot execute in check back mode.
E1061	Cannot execute in ONE program.
E1062	Jt2 and Jt3 interfere during motion to start pose.
E1063	Jt2 and Jt3 interfere during motion to end pose.
E1064	Illegal pallet number.
E1065	Illegal work number.
E1066	Illegal pattern number.
E1067	Illegal pattern type.
E1068	Illegal work data.
E1069	Illegal pallet data.
E1070	ON/ONI signal is already input.
E1071	XMOVE signal is already input.
E1072	Home position data is not defined.
E1073	Illegal timer number.
E1074	Over maximum signal number.
E1075	Illegal clamp number.
E1076	Cannot use negative time value.
E1077	No value set.

Code	Error Message
E1078	Illegal signal number.
E1079	Cannot use dedicated signal.
E1080	Not RPS mode.
E1081	Cannot use negative value.
E1082	Out of absolute lower motion range limit.
E1083	Out of absolute upper motion range limit.
E1084	Out of set lower motion range limit.
E1085	Out of set upper motion range limit.
E1086	Start point for JtXX beyond motion range.
E1087	End point for JtXX beyond motion range.
E1088	Destination is out of motion range.
E1089	Cannot do linear motion in current configuration.
E1090	External modulation data is not input.
E1091	External modulation data is abnormal.
E1092	Modulation data is over limit.
E1093	Incorrect motion instruction to execute modulate motion.
E1094	Illegal joint number.
E1095	Cannot execute motion instruction in PC program.
E1096	Incorrect auxiliary data settings.
E1097	Missing C1MOVE or C2MOVE instruction.
E1098	C1MOVE(CIR1)instruction required before C2MOVE.
E1099	Unable to create arc path, check positions of the 3 points.
E1100	Cannot execute in sealing specification.
E1101	Can only execute in sealing specification.
E1102	Option is not set, cannot execute.
E1103	Over conveyer position.
E1104	Too many SPINMOVE instructions.
E1105	Start/Destination point is in protected space.
E1106	Cannot execute in this robot.
E1107	Cannot use SEPARATE CONTROL.
E1108	Duplicate robot network IDs.
E1109	Conveyor I/F board is not installed.
E1110	GROUP is not primed.
E1111	Due to motion restriction, JtXX cannot move.
E1113	Work sensing signal is not detected.
E1114	Interruption in cooperative control.
E1115	Forced termination of cooperative control.

Code	Error Message
E1116	Spin axis is not stopped on every 360 degrees.
E1117	Process time over.
E1118	Command value for JtXX suddenly changed.
E1119	Command value for JtXX beyond motion range.
E1120	Current command causes interference betw Jt2 and Jt3.
E1121	Other robot is already in the interference area.
E1122	Unexpected motor power OFF.
E1123	Speed error JtXX.
E1124	Deviation error of JtXX.
E1125	Velocity envelope error JtXX.
E1126	Command speed error of JtXX.
E1127	Command acceleration error of JtXX.
E1128	Uncoincidence error betw destination and current JtXX pos.
E1129	External axis JtXX moved while holding them.
E1130	JtXX collision was detected.
E1131	JtXX unexpected shock is detected.
E1132	Motor power OFF. Measurement stopped.
E1133	Conveyor has reached max. value position.
E1134	Abnormal work transfer pitch of conveyor.
E1135	Motor power OFF.
E1136	Standard terminal is not connected.
E1137	Cannot input/output to teach pendant.
E1138	Aux. terminal is not connected.
E1139	DA board is not installed.
E1140	No conveyor axis.
E1141	Conveyor transfers beyond sync. zone.
E1142	No traverse axis.
E1143	Conveyor axis number is not set.
E1144	No Arm control board.
E1145	Cannot use specified channel, already in use.
E1146	[LSEQ]Aborted by processing time over.
E1147	Cannot open setting file, so cannot set to shipment state.
E1148	Cannot read setting file, so cannot set to shipment state.
E1149	Cannot open setting data, so cannot set to shipment state.
E1150	Cannot read setting data, so cannot set to shipment state.
E1151	Too much data for setting to the shipment state.
E1152	Name of setting data for shipment state is too long.

Code	Error Message
E1153	Power sequence board detected error.(Code=XX)
E1154	Option SIO port not installed.
E1155	A/D converter is not installed.
E1156	[ARM CONTROL BOARD]Processing time over.
E1157	Arm ID I/F board error. (CodeXX)
E1158	(SSCNET)Servo error in JtXX.
E1159	(SSCNET)Error code for servo is (XX).
E1160	(SSCNET)Servo error and monitor setting error of JtXX.
E1161	Automatic Tool Registration not supported by robot model.
E1162	Buffer overflow occurred in the gravity comp. value channel XX.
E1163	Robot stopped in checking operational area.
E1164	[LSEQ]Program execution error at control power ON.(CodeXX)
E1165	Unable to download ext. axis parameter.(Jt-A)
E1166	Num. not assigned to specified channel.(Jt-A)
E1167	Unable to download ext. axis parameter.(Jt-B)
E1168	Num. not assigned to specified channel.(Jt-B)
E1169	Error in servo parameter change sequence.(CodeXX)
E1170	Slave is not ready.
E1171	CC-LINK communication board is not installed.
E1172	Weld communication board is not installed.
E1173	Servo communication error JtXX.
E1174	AD board No.0 is not installed.
E1175	Offset data of zeroing is illegal value.(RobotXX)
E1176	(SSCNET)Download error of external axis parameter.
E1177	(SSCNET)Joint number is not assigned to the channel.
E1178	Communication error between Arm Control and Arm I/F board.
E1179	The current under bending compensation is too large.(JtXX)
E1180	Download error of external axis parameter.(JtXX)
E1181	Encoder battery low voltage.[Servo(XX)]
E1182	Encoder battery low voltage.[External axis(XX)]
E1183	Because Jt5 is not zero degree, cannot move along straight line.
E1184	Illegal configuration for motion.
E1185	Jt1 and Jt2 is interfered at start location.
E1186	Jt1 and Jt2 is interfered at end location.
E1187	Current command between Jt1 and Jt2 is interfered.
E1188	(SSCNET)Error in servo parameter change sequence.(CodeXX)
E1189	(SSCNET)Regenerative error of JtXX.(CodeXX)

Code	Error Message
E1190	(SSCNET)Speed error of JtXX.(CodeXX)
E1191	(SSCNET)Motor overload of JtXX.(CodeXX)
E1192	(SSCNET)Deviation error of JtXX.(CodeXX)
E1193	(SSCNET)Encoder battery of JtXX is low voltage.(CodeXX)
E1194	(SSCNET)Parameter warning of JtXX.(CodeXX)
E1195	(Dual servo)Deviation error between master joint and slave joint.
E1196	While lifter is locked, it cannot move.
E1197	Compensation LS signal is not dedicated.
E1198	Brake check sequence error.
E1199	Brake check function is not supported by servo software ver.
E1200	(Dual servo)Cannot compensate current error (deviation XX).
E1201	Interference check board is not installed.
E1202	Voice recorder cannot stop.
E1203	LS location is not registered.
E1204	Current stretch over the limit.
E1205	Total stretch over the limit.
E1207	The type of I/O board on arm ID board is wrong.
E1208	Download error of servo parameter.(JtXX)
E1209	Upload error of servo parameter.(JtXX)
E1210	Cannot execute program because unprotected.
E1211	Because the memory was full, could not copy the program.
E1212	Because the memory was full, the copy of program was suspended.
E1213	Jt4 and robot arm interfere during motion to start pose.
E1214	Jt4 and robot arm interfere during motion to end pose.
E1215	Current command causes interference between Jt4 and robot arm.
E1216	Jt5 and Jt6 interfere during motion to start pose.
E1217	Jt5 and Jt6 interfere during motion to end pose.
E1218	Current command causes interference between Jt5 and Jt6.
E1219	Exceeds allowable No. of output instructions for the path.
E1220	Signal output point is out of the path.
E1221	Too many signal numbers are specified.
E1222	Motion instruction to start/end point of path not set.
E1223	No pose data in last/next motion instruction.
E1224	Several signal output points detected at the same point.
E1225	Correction end instruction is missing.
E1228	Jt4 value in the start point is not 0 degree.
E1229	Jt4 value in the target point is not 0 degree.

Code	Error Message
E1230	Flange faces direction of upper sphere in start point.
E1231	Flange faces direction of upper sphere in target point.
E1232	Option CPU board is not installed.
E1233	IJoint/ILinear signal not specified.
E1234	IJoint/ILinear signal not detected.
E1235	Separate control I/O board is not installed.
E1236	Distance is too long to correct.
E1237	Vision recognition error.
E1238	Vision communication error.
E1239	Cannot use this instruction in frame correction mode.
E1240	BASE FRAME is not sent from vision unit.
E1241	Improper parameter for FN481.
E1242	Cannot create more than 99 BASE FRAME.
E1243	Cannot execute because cameraXX is disconnected.
E1244	Jt1 and Jt2 and floor interfere during motion to start pose.
E1245	Jt1 and Jt2 and floor interfere during motion to end pose.
E1246	Current command causes interference between Jt1, Jt2 and floor.
E1247	Calculation of encoder absolute data is not completed.(JtXX)
E1248	EEPROM access flag in encoder is busy.(JtXX)
E1249	Temperature in encoder is over the limit.(JtXX)
E1250	Rotation speed of encoder is over the limit.(JtXX)
E1251	EEPROM access error occurred in encoder.(JtXX)
E1252	Encoder rotation data (internal) is abnormal.(JtXX)
E1253	Uncoincidence error between request and response in encoder. (JtXX)
E1254	Cannot operate because a MC of a group XX is off.
E1255	The motor power of unchosen robot was turned on.
E1256	Internal valve , sensor and error reset I/F board missing.
E1257	MC of groupXX turned off during individual repeat operation.
E1258	MC turned off during operation.
E1259	Invalid Structure of palletizing instruction.
E1260	Cannot execute instruction during palletizing motion.
E1261	Palletising motion aborted.
E1262	Encoder rotation speed exceeded limit. (JtXX)
E1263	Encoder temperature exceeded limit (jt XX)
E1264	Velocity envelope error in endless rotation axis.(JtXX)
E1267	The initial setting of Encoder is abnormal. (JtXX)
E1268	Breakage in the encoder line or faulty setting of encoder baud rate. (JtXX)

Code	Error Message
E1269	The program is for other robot.
E1270	The pose variable is for other robot.
E1271	Interference between arm and floor at start pose.
E1272	Interference between arm and floor at end pose.
E1273	Command causes interference between arm and floor.
E1274	JtXX Over speed in heavy load mode.
E1275	JtXX Beyond the motion range in heavy load mode.
E1276	Start point JtXX is beyond the motion range in heavy load mode.
E1277	End point JtXX is beyond the motion range in heavy load mode.
E1278	Can't slant wrist any more.
E1279	Wrist does not face vertically down at start point.
E1280	Wrist does not face vertically down at end point.
E1281	Command to Jt4 over limit.
E1282	Cannot operate because a MC of a groupXX(JtXX) is off.
E1283	analysis)The E1035 error occurs frequently.JtXX
E1284	analysis)The E1035 and E1029 error occur at the same time.JtXX
E1285	analysis)The E1035 and E1036 error occur at the same time.JtXX
E1286	analysis)The E1035 and E1032 error occur at the same time.JtXX
E1287	Power module error JtXX (UP).
E1288	Power module error JtXX (LOW).
E1289	[Servo boardXX]Synchronous error.(Servo FPGA)
E1290	JtXX The voltage of the current sensor exceeded the upper bound value.
E1291	JtXX Current sensor is disconnected or out of order. (U)
E1292	[Servo boardXX]Input abnormal signal from MCXX.
E1293	[Servo boardXX]FB current setting gain data is abnormal.
E1294	[Servo boardXX]I/O 24V is low.
E1295	[Servo boardXX]24V for internal valve is low.
E1296	[Servo boardXX]Mismatch in safety circuit LS conditions.
E1297	[Servo boardXX]Mismatch in jumper wiring of internal pressure.
E1298	[Servo boardXX]Mismatch in LS override switch.
E1299	[Servo boardXX]Jumper wiring of internal pressure is disconnected.
E1300	[Servo boardXX]DC 24V is abnormal.
E1301	[Servo boardXX]Soft or servo is not compatible with encoder type.
E1302	[MCXX]OFF check is abnormal.(Servo boardXX)
E1303	[MCXX]OFF check of safety relay is abnormal.(Servo boardXX)
E1304	[MCXX]Incorrect operation of K1.(Servo boardXX)
E1305	[MCXX]Incorrect operation of K2.(Servo boardXX)

Code	Error Message
E1306	[MCXX]Incorrect operation of rush control relay.(Servo boardXX)
E1307	[MCXX]Incorrect operation of safety relay KS1.(Servo boardXX)
E1308	[MCXX]Incorrect operation of safety relay KS2.(Servo boardXX)
E1309	[MCXX]Incorrect operation of safety relay KS3.(Servo boardXX)
E1310	[MCXX]Incorrect operation of motor power ON relay.(Servo boardXX)
E1311	[MCXX]Incorrect operation of safety circuit motor OFF relay.(Servo boardXX)
E1312	[MCXX]Mismatch in safety circuit motor OFF relay.(Servo boardXX)
E1313	[MCXX]Mismatch in MC control of safety circuit.(Servo boardXX)
E1314	[MCXX]Thyristor Thermal is abnormal.(Servo boardXX)
E1315	Watchdog error in NoXX I/O board.
E1316	[I/O board(No.XX)]Access Error.[Address:XX][Code:XX]
E1317	[Servo Board(NoXX)]Response from monitor is abnormal. [Code:XX]
E1318	[MCXX]DC 20V is abnormal.(Servo boardXX)
E1319	Internal valve, sensor and error reset I/F board No.2 is not installed.
E1321	[Main CPU board]Servo board(XX) communication error. (CodeXX)
E1322	Setting num. of safety circuits differs betw. powerseq.b'd and MCXX. (Servo b'dXX)
E1323	Setting num. of safety circuits differs betw. servo b'dXX and MCXX.
E1324	Safe circuit disconnected between power sequence board and servo boardXX.
E1325	Safe circuit disconnected between servo boardXX and MCXX.
E1326	Safety fence is open.
E1327	[Power sequence board]Miscompare in motor off relay condition on safety circuit.
E1328	[Power sequence board]Error of motor off relay on safety circuit.
E1329	[Power sequence board]Error in TEACH/REPEAT switch on safety circuit.
E1330	[Power sequence board]IO 24V is low.
E1331	[Power sequence board]Thermal error.
E1332	[Power sequence board]Power error signal was input from servo boardXX.
E1333	Motor power on signal has turned off.(Servo boardXX)(MCXX)
E1334	TEACH/REPEAT switch is abnormal.(Mode differs betw. safety circuit and monitor.)
E1335	Unexpected motor powerOFF.(Servo boardXX)(MCXX)
E1336	[Servo boardXX]Communication error with Main CPU board .
E1337	[MCXX]Brake power is abnormal.(Servo boardXX)
E1338	[MCXX]P-N low voltage.(Servo boardXX)
E1339	[MCXX]P-N high voltage.(Servo boardXX)
E1340	[MCXX]Regenerative time over.(Servo boardXX)
E1341	[MCXX]Regenerative resistor overheat.(Servo boardXX)
E1342	Motor harness disconnected or robot temperature exceeded limit.(MCXX)
E1343	Mismatch in wiring brake and software setting.(JtXX)

Code	Error Message
E1344	JtXX Current sensor is disconnected or out of order. (V)
E1345	[Servo boardXX]Limit switch signal line is disconnected.
E1346	JtXX Failed to get encoder full data.
E1347	[MCXX]Destination spec is incorrect.(Servo boardXX)
E1348	[MCXX]Controlled target is incorrect.(Servo boardXX)
E1349	[MCXX]Explosion proof setting is mismatch.(Servo boardXX)
E1350	[MCXX]MC specification error.[CodeXX](Servo boardXX)
E1351	[MCXX]MC OFF delay specification is incorrect.(Servo boardXX)
E1352	JtXX Codes set in software and power block do not match.
E1353	[Main CPU board]CPU temperature is abnormal.
E1354	[Main CPU board]Temperature in CPU board exceeded the limit.(XX 1/1000 deg C)
E1355	Error in servo I/F command communication.(Code:XX)
E1356	The tool shape is not set.
E1357	Failed to download ext. axis parameter data.(Jt-C)
E1358	Axis No. is not assigned to the specified channel.(Jt-C)
E1359	JtXX axis U phase overcurrent.
E1360	JtXX axis V phase overcurrent.
E1361	JtXX axis W phase overcurrent.
E1362	[Servo boardXX]Speed of tool center point exceeded safety speed.
E1363	[Servo boardXX]Speed of flange center point exceeded safety speed.
E1364	[ARM CONTROL BOARD]Out of command synch, between IF and SV.
E1365	TEACH KEY SWITCH is ON in two or more places.
E1366	Watchdog error in NoXX ANYBUS interface board.
E1367	Improper parameter for KI481.
E3800	JtXX axis amp servo amp heating.
E3801	JtXX axis amp main circuit power supply decrease.
E3802	Encoder harness disconnected. JtXX
E3803	JtXX axis amp speed control error.
E3804	JtXX axis amp velocity feedback error.
E3805	JtXX axis amp position envelope error.
E3806	JtXX axis amp servo ready does not turn on.
E3807	JtXX axis amp IPM overheated.
E3808	Motor power OFF (EXT_EMG).
E3809	Brake release signal error.
E3810	Power sequence ready off.
E3811	JtXX axis amp command value suddenly changed.

Code	Error Message
E3900	Mismatch moving tool data and selected tool data.
E4000	Data communication error.
E4001	Data reading error.
E4002	Data write error.
E4003	Unexpected error in file access.
E4004	Communication retry error.
E4005	Communication process was stopped.
E4006	Receive no data after request.
E4007	Receiving data is too long(MAX=255 characters).
E4008	Abnormal data (EOT) received in communication.
E4009	Communication time out error.
E4010	Terminal already in use.
E4011	Communication port already in use.
E4012	Waiting for input of PROMPT. Connect input device.
E4013	TELNET)SEND error. Code=XX
E4014	TELNET)RECV error. Code=XX
E4015	TELNET)IAC receive error. Code=XX
E4016	TELNET)Close failure. Code=XX
E4017	TELNET)Main socket close failure. Code=XX
E4018	TELNET)System error. Code=XX
E4019	TCPIP)Socket open failure. Code=XX Dst.IP=XX.XX.XX.XX
E4020	TCPIP)Socket close failure. Code=XX Dst.IP=XX.XX.XX.XX
E4021	TCPIP)Communication Error. Code=XX Dst.IP=XX.XX.XX.XX
E4022	TCPIP)Message is too long.
E4023	TCPIP)Cannot reach the Host.
E4024	TCPIP)Communication Time Out. Dst.IP=XX.XX.XX.XX
E4025	TCPIP)Connection aborted.
E4026	TCPIP)No Buffer Space.
E4027	TCPIP)Bad Socket.
E4028	FTP)Data receive error.(Code=XX)
E4029	FTP)Data send error.(Code=XX)
E4030	FTP)Server does not recognize command.(Code=XX)
E4031	FTP)Failed to disconnect with FTP server.(Code=XX)
E4032	FTP)Unregistered OS detected.
E4033	FTP)Failed to connect with server.(Code=XX)
E4034	FTP)Failed to receive HOST OS information.(Code=XX)
E4035	FTP)TCP/IP not initialized.

Code	Error Message
E4036	FTP)FTP service busy now.
E4037	FTP)Failed AUTO-SAVing.
E4050	No response from the FDD/PC_CARD driver board.
E4051	No communication with FDD/PC_CARD driver board.
E4052	[FDD/PC_CARD]Failed to set verify function. Please set again.
E4053	Channel error.
E4054	TCPIP)Cannot execute because Ethernet board not installed.
E4055	TCP)Cannot create a socket.
E4056	TCP)This port is not in LISTEN (SOCK).
E4057	TCP)Illegal Socket ID.
E4058	Failed download to FDD/PC_CARD driver board.
E4059	ASCYCLE communication receive error.(Code:XX)
E4060	[ARM CONTROL BOARD]ASCYCLE communication receive error.(Code:XX)
E4061	Received gauge hole data exceeds allowable range.
E4062	Master/slave data is not registered.
E4063	Reference point data is not registered.
E4064	3D calibration/measurement modes are both ON.
E4065	Unregistered variable specified to receive data.
E4066	Variable specified to receive data is broken.
E4067	Received data is broken.
E4068	Start code is not correct.
E4069	End code is not correct.
E4070	3D camera group No. not specified.
E4071	Incorrect 3D camera group No.
E4072	Communication beginning wait time out error.
E4073	No servo off signal from ARM I/F board.
E4074	[Servo boardXX]No response from MCXX.(Code:XX)
E4075	[Servo boardXX]MCXX communication error.(Code:XX)
E4076	[MCXX]Servo boardXX communication error.(Code:XX)
E4077	[Servo board XX] Communication check error with the main CPU board. (code: XX)
E4078	[Servo boardXX]Error in command communication with the main CPU board.(Code:XX)
E4500	ANYBUS)IN-AREA request timeout.XX
E4501	ANYBUS)OUT/FB.CTRL release timeout.XX
E4510	DN)Master status.XX
E4511	DN)Node status.XX

Code	Error Message
E4512	ABM-DN)Mailbox error.
E4520	ABMA-PDP)Status STOP.XX
E4521	ABMA-PDP)Status OFFLINE. XX
E4522	ABMA-PDP)I/O data Communication error.XX
E4523	ABMA-PDP)Sending of timed out I/O data.XX
E4524	ABMA-PDP)Timeout of receiving I/O data.XX
E4525	ABMA-PDP)Timeout of sending message.XX
E4526	ABMA-PDP)Timeout of receiving message.XX
E4527	ABMA-PDP)Check configuration data.XX
E4528	PROFIBUS)Slave Diag-error response detected.XX
E4529	PROFIBUS)Statistic counter-error response detected.XX
E4530	DN)DeviceNet cable is disconnected.
E4531	CC-LINK)Communication has been disconnected. XX
E4532	CC-LINK)Initial condition setting is incorrect.
E4533	CC-LINK)Watch dog timeout error.
E4534	CC-LINK)Parameter setting error. XX
E4535	CC-LINK)Time out on setting parameter.
E4536	CC-LINK)Master board is abnormal. XX
E4537	CC-LINK)Initialization error on master board . XX
E4538	CANopen)Network is disconnected.
E5000	Connected permission signal has not been turned ON.
E5001	RWC type is not process control type.
E5002	1GS board is not process control type.
E5003	Illegal extend (retract) output signal.
E5004	Weld completion signal already input.
E5005	(Spot weld)Weld schedule setting data is abnormal.
E5006	CLAMP SPEC is not set as PULSE.
E5007	Servo weld gun not connected or wrong gun connected.
E5008	Tip wear measurement (STAGE1) was not executed.
E5009	Work sensing signal(gun_tip touch signal) is not set.
E5010	Servo weld gun mechanical parameter is not set.
E5011	This clamp number already set for servo weld gun axis.
E5012	Cannot change the gun because offset data is abnormal.
E5013	Cannot change multiple guns at the same step.
E5014	Cannot execute, gun connected to another joint.
E5015	Gun status data disagrees with clamp condition.
E5016	Data of SRVPRESS is wrong.

Code	Error Message
E5017	Wear base data is not registered.
E5018	Weld completion signal has not been detected.
E5019	Weld fault signal is detected.
E5020	Retract pos. monitor error.
E5021	Extend pos. monitor error.
E5022	Current gun retract position differs from a destination.
E5023	Wear is abnormal, cannot take measurement.
E5024	Pressurization comp. signal has not been detected.
E5025	Gun opening comp. signal has not been detected.
E5026	(Spot welding)RWC error. XX
E5027	Robot stopped in welding.
E5028	Cannot achieve set force.
E5029	Gun tip stuck.
E5030	Copper plate wear exceeds limit. step=XX
E5031	Weld completion signal is not turned OFF.
E5032	Calibration did not end normally.
E5033	Cannot weld because of abnormal thickness.
E5034	Tip wear exceeds limit. (MOVING SIDE)
E5035	Tip wear exceeds limit. (FIXED SIDE)
E5036	Incorrect gun status data.
E5037	Tip wear exceeds limit. XX
E5038	Arc detection signal did not turn OFF.
E5039	No response from RWC communication I/F board.
E5040	Cannot connect gun because gun is already connected.
E5041	Cannot disconnect gun because gun is already disconnected.
E5042	Gun No is not defined or Gun type is not servo gun.
E5043	Communication error in welder. (CodeXX)
E5044	Failed to get weld data. (timer XX)
E5045	Failed to change weld data. (timer XX)
E5046	Weld error has arisen.
E5047	Receiving weld items now, wait till completion.
E5048	Weld controller is unconnected or weld items are not received. (timer XX)
E5049	Serial number signal error.
E5050	This welder is without Traceability.
E5051	Cannot calibrate because tool change axis is disconnected.
E5052	The pressurizing power measurement value is abnormal.
E5053	The pressurizing power sensor is disconnected or it breaks down.

Code	Error Message
E5054	The selector switch on TP is set to manual operation.
E5055	The selector switch on TP is set to automatic operation.
E5056	No Initialization Weld board.
E5057	Initialization failure in Initialization Weld board.
E5058	Welder(DENGEN COMPANY) not connected. (welder XX)
E5059	Welder(DENGEN COMPANY) response error. (welder XX)
E5060	Initialization Weld board protected. (welder XX)
E5061	Welder(DENGEN COMPANY) data process not execute. (welder XX)
E5062	Welder(DENGEN COMPANY) data process error.(welder XX)
E5063	Weld error has arisen. (CodeXX)
E5064	Welder(DENGEN COMPANY) of weld was aborted. (welder XX)
E5065	Welder(DENGEN COMPANY) error has arisen. (welder XX)
E5066	Waiting weld completion time out.(welder XX)
E5067	Magnet control is abnormal.(welder XX)
E5500	Vision board is not installed.
E5501	(Vision)Camera not connected.
E5502	(Vision)Incorrect parameter.
E5503	(Vision)Incorrect Symbol.
E5504	(Vision)Incorrect name.
E5505	(Vision)Incorrect image memory.
E5506	(Vision)Incorrect histogram data.
E5507	(Vision)Incorrect mode.
E5508	(Vision)Incorrect density(/color).
E5509	(Vision)Incorrect camera input assignment.
E5510	(Vision)Incorrect camera ch.number.
E5511	(Vision)Incorrect Window No.
E5512	(Vision)Incorrect coordinates data.
E5513	(Vision)Incorrect number.
E5514	(Vision)Incorrect image code(binary/multi).
E5515	(Vision)Incorrect threshold.
E5516	(Vision)PROTO(/TEMPLATE) not registered or already exists.
E5517	(Vision)Cal. data not registered.
E5518	(Vision)Graphic cursor is not initialized.
E5519	(Vision)Too many samples from PROTO object.
E5520	(Vision)Too many targets detected.
E5521	(Vision)Vision command not initiated.
E5522	(Vision)System registered with abnormal data.

Code	Error Message
E5523	(Vision)Error in processing image(s).
E5524	(Vision)Sound port assigned another function.
E5525	(Vision)Lack of data storage area.
E5526	(Vision)Incorrect synch. mode.
E5527	(Vision)Vision processing now.
E5528	(Vision)Image capture error.
E5529	(Vision)Time out or Buffer overflow.
E5530	(Vision)Failed to write on flash memory.
E5531	(Vision)Proto data abnormal, so initialized.
E5532	(Vision)Work detection failure.
E5533	(Vision)Initialization error. Code = XX
E5534	(Vision)Vision system error.
E5535	(Vision)Specified motion mode is incorrect.
E5536	(Vision)Inappropriate camera/projector parameters.
E5537	(Vision)Incorrect camera switch assignment.
E5538	(Vision)This plane is assigned to another camera.
E5539	(Vision)Edge was not found.
E5540	(Vision)Inappropriate HSI data.
E5541	(Vision)H data range width is over 128.
E5542	(Vision)Distance image input unit not set for camera.
E5543	(Vision)Cannot calculate the set edge points.
E5544	(Vision)Check color conversion table type in set config.
E5545	(Vision)Incorrect area size.
E5546	(Vision)Slit image does not exist.
E5547	(Vision)Incorrect no. of correlation vectors.
E5548	(Vision)Inappropriate vector data.
E5549	(Vision)X-Fit environment was not set.
E5550	(Vision)Mouse is not initialized.
E5551	(Vision)Camera switcher board is not installed.
E6000	Explosion proof teach pendant is not connected.
E6001	Step after XD(2)START must be LMOVE or HMOVE.
E6002	Signal condition already input.
E6003	Door open detect signal is not dedicated.
E6004	Location data was not detected.
E6005	Incorrect setting of barrier unit.
E6006	Signal not detected.
E6007	Wrist can't be straightened any more (Singular point 1).

Code	Error Message
E6008	Wrist can't be bent any more (Singular point 2).
E6009	Purge air flow is insufficient.
E6010	Out of XYZ MOVING AREA LIMIT.
E6011	Pressure within enclosure is low.
E6012	Relative distance between guns is too near (ID:XX).
E6013	No free memory in program queue.
E6014	No free memory in delayed start queue.
E6015	Special signal is not specialized.
E6016	Robot arm stretching out (Singular Point 3).
E6017	Out of mechanical XYZ motion limits.
E6018	Painting equipment control board error. (CodeXX)
E6019	Painting equipment control board is not installed.
E6020	Monitoring Robot ID is duplicate.
E6021	(Mutual-Wait)There is no response from the other party robot.
E6022	Duplicate Mutual-Wait IDs.
E6023	(Mutual-Wait)Communication error in Mutual-Wait.
E6024	Wrist can't bend any further left/right (Singular Point 1).
E6025	(Conveyer synchronous communications)It is a conveyer position reception error.
E6026	Guns are too near in X direction. (ID:XX)
E6027	Guns are too near in Y direction. (ID:XX)
E6028	Guns are too near in Z direction. (ID:XX)
E6029	[Servo boardXX]Mismatch in internal pressure of safety relay.
E6030	[Servo boardXX]Pressure within enclosure is low.
E6031	Monitoring Robot ID is invalid.
E6032	[Purge control board]Pressure within enclosure is low.
E6033	Painting equipment control process error. (CodeXX)
E6034	Cartridge table rotate command is abnormal.
E6500	No welding Interface board.
E6501	No.2 welding Interface board not found.
E6502	Arc failure.
E6503	Wire stuck.
E6504	Arc start failure.
E6505	Arc weld insulation defect.
E6506	Torch interference.
E6507	Illegal interpolation data.
E6508	No D/A board for polarity ratio control.
E6509	No work detected.

Code	Error Message
E6510	Undefined sensing direction.
E6511	Insufficient num. of sensing points.
E6512	Undefined mother or daughter work.
E6513	Too many sensing points.
E6514	Work specification incorrect.
E6515	Incorrect sensing point specified.
E6516	Wire check failure.
E6517	Incorrect weld condition number.
E6518	No weld condition data set.
E6519	Weld condition data is out of range.
E6520	Laser sensor tracking value exceeded.
E6521	Beyond Laser sensor tracking ability.
E6522	Laser sensor cannot detect welding joint.
E6523	Calibration data between torch and camera is not ready.
E6524	Error in data calculated using Laser sensor.
E6525	Cannot detect weld joint, Laser sensor tracking set already.
E6526	No response from Laser sensor controller.
E6527	Laser sensor communication error. Code is XX.
E6528	Start point not found by Laser sensor.
E6529	Finish point not found by Laser sensor.
E6530	Cannot use circular interp. with Laser sensor function.
E6531	Cannot turn Laser ON because motor power is OFF.
E6532	No communication board to Laser sensor.
E6533	No RTPM board.
E6534	Too many taught points for RTPM.
E6535	RTPM arc sensor error.
E6536	RTPM current deviation error.
E6537	RTPM tracking value is out of range.
E6538	Beyond RTPM tracking ability.
E6539	AVC tracking value is out of range.
E6540	Beyond AVC tracking ability.
E6541	No AVC board.
E6542	AVC voltage deviation error.
E6543	Too many taught points for AVC.
E6544	Hyper Arc tracking value is out of range.
E6545	Beyond Hyper Arc tracking ability.
E6546	Bead end is not found.

Code	Error Message
E6547	Finish end is not found.
E6548	Hyper Arc revolution beyond normal deviation.
E6549	Hyper Arc torch calibration error.
E6550	Hyper Arc Z phase index error.
E6551	No Hyper Arc board.
E6552	Hyper Arc board error. Code is XX.
E6553	Hyper Arc current sensor error.
E6554	Hyper Arc voltage sensor error.
E6555	Hyper Arc current deviation error.
E6556	Hyper Arc amplifier error. Code is XX.
E6557	No Wire feeding Control board.
E6558	Wire feeding control error, code is XX.
E6559	Wire feeding speed deviation error.
E6560	Cannot re-calibrate weld in progress.
E6561	Cannot weld, re-calibration in progress.
E6562	Electric pole stuck.
E6563	KHITS tracking system error. (Code = XX)
E6564	The arc weld instruction sequence is incorrect.
E6565	Arc welding Interface board(1LN) is not installed.(robot XX)
E6566	FN instructions not executed in the correct order.
E6567	KLS tracking system error.(Code=XX)
E6568	Failed in executing command to tracking system.(cmd=XX)
E6569	Taught data exceeds inclination limit for compensation.
E6570	Cannot execute because welding now.
E6571	Cannot execute because wire inching/retracting now.
E6572	Teach point for circular motion is missing.
E6573	The welding machine is abnormal.(Code=XX)
E6574	Sensing of groove: cannot detect edge.
E6575	Sensing of groove: gap error.
E6576	Welder is not ready for operation. (Code=XX)
E6577	Deviation is too large.Reset allows non-correction movement.
E6578	Sensing was interrupted due to power OFF. Restore step and retry.
E6579	KI instructions not executed in the correct order.
E7000	Servo weld gun disconnected.
E7001	Location data includes released gun status data.
E7002	Destination is far from target point.
E7003	The clearance distance of gunXX is set to 0mm.

Code	Error Message
E7004	Gun tip wear change over the limit. (MOVING SIDE)
E7005	Gun tip wear change over the limit. (FIXED SIDE)
E7006	Clamp number or gun number is not servo weld gun.
E7007	Cannot change tip base data in 1 Stg. because tip wear rate is not set.
E7008	Independent Gun control is not completed.
E7009	Current limit for servo welding gun is abnormal.
E7500	JtXX Collision is detected.
E7501	JtXX Unexpected shock is detected.
E7502	AC Fail Process Error = XX
E7503	POWER SEQUENCE setting data incorrect.
E7504	Angle between JtXX is out of range at start location.
E7505	Angle between JtXX is out of range at end location.
E7506	Angle between JtXX is out of range.
E7507	SC1MOVE or SC2MOVE instruction is required after SC1MOVE.
E7508	SC1MOVE instruction is required before SC2MOVE.
E7509	Cannot execute, interpolation conditions are not fulfilled.
E7510	Cannot move with current posture.
E7511	Brake control bit number is duplicated.
E7512	L3C1MOVE or L3C2MOVE instruction is required after L3C1MOVE.
E7513	L3C1MOVE instruction is required before L3C2MOVE.
E7514	Specified parameter is not consistent.
E8200	Not in cooperative mode.
E8201	Unmatch the total of motion instruction in cooperative mode.
E8202	Unmatch step of motion instruction in cooperative mode.
E8203	Cannot use this instruction in cooperative mode.
E8204	Invalid cooperative group No.
E8205	No JMASTER robot.
E8206	TouchSensing in Cooperative mode is no supported.
E8207	JMASTER robot already exists.
E8208	WSLAVE robot already exists.
E8209	Fixed Point Motion in Cooperative mode is no supported.
E8210	No WSLAVE robot.
E8211	Out of sync.
E8212	Cannot continue non-cooperative instruction in cooperative mode.
E8213	No MASTER robot.
E8214	No SLAVE robot.
E8400	Servohand opened in clamp ON step.(CLAMP=XX)

Code	Error Message
E8401	Clamping position of servo Hand is error.(CLAMP=XX)
E8402	Cannot achieve set force of JtXX.
E8403	NC Joints lock signal not off.
E8404	Interpolation other than joint int. is unavailable.
E8405	It tries to move the Matehan axis with the axis locked.
E8600	（FSJ)Processing condition error.XX
E8601	Gap was over the lower pos. limit.
E8602	Reached the Penetration depth within min.processing time.
E8603	It could not reach the set Penetration depth within the appointed period.
E8604	Pressure cable disconnected.
E8605	Please input two set pressure or more.
E8606	Please input data in ascending order.
E8607	FSJ COUNTER ALARM.XX
E8608	(FSJ)FSJ schedule setting data is abnormal.
E8609	Setting tip force is over limit.
E8610	Setting rotation speed is over limit.
E8611	FSW Logging buffer is full.
E8800	Command value almost exceeds virtual safety fence.(SphereXX, LineXX)
E8801	Command value almost exceeds virtual safety fence.(SphereXX, ZUpper)
E8802	Command value almost exceeds virtual safety fence.(SphereXX, ZLower)
E8803	Command value almost invades restricted space.(SphereXX, Part.XX LineXX)
E8804	Command value almost invades restricted space.(SphereXX, Part.XX ZUpper)
E8805	Command value almost invades restricted space.(SphereXX, Part.XX ZLower)
E8806	Command value almost exceeds virtual safety fence.(ToolBox, LineXX)
E8807	Command value almost exceeds virtual safety fence.(ToolBox, ZUpper)
E8808	Command value almost exceeds virtual safety fence.(ToolBox, ZUpper)
E8809	Command value almost invades restricted space.(ToolBox, Part.XX)
E8810	Command value almost exceeds virtual safety fence.(LinkXX, LineXX)
E8811	Command value almost exceeds virtual safety fence.(LinkXX, ZUpper)
E8812	Command value almost exceeds virtual safety fence.(LinkXX, ZLower)
E8813	Command value almost invades restricted space.(LinkXX, Part.XX LineXX)
E8814	Command value almost invades restricted space.(LinkXX, Part.XX ZUpper)
E8815	Command value almost invades restricted space.(LinkXX, Part.XX ZLower)
E8820	Current value exceeded virtual safety fence.(SphereXX, LineXX)
E8821	Current value exceeded virtual safety fence.(SphereXX, ZUpper)
E8822	Current value exceeded virtual safety fence.(SphereXX, ZLower)
E8823	Current value invaded restricted space.(SphereXX, Part.XX LineXX)
E8824	Current value invaded restricted space.(SphereXX, Part.XX ZUpper)
E8825	Current value invaded restricted space.(SphereXX, Part.XX ZLower)

Code	Error Message
E8826	Current value exceeded virtual safety fence.(ToolBox, LineXX)
E8827	Current value exceeded virtual safety fence.(ToolBox, ZUpper)
E8828	Current value exceeded virtual safety fence.(ToolBox, ZLower)
E8829	Current value invaded restricted space.(ToolBox, Part.XX)
E8830	Current value exceeded virtual safety fence.(LinkXX, LineXX)
E8831	Current value exceeded virtual safety fence.(LinkXX, ZUpper)
E8832	Current value exceeded virtual safety fence.(LinkXX, ZLower)
E8833	Current value invaded restricted space.(LinkXX, Part.XX LineXX)
E8834	Current value invaded restricted space.(LinkXX, Part.XX ZUpper)
E8835	Current value invaded restricted space.(LinkXX, Part.XX ZLower)
E8850	Disabled robot motion.
E8851	Detected area interference.
E8852	Detected arm interference.(XX, XX)
E8853	Failed to predict trajectory.
E8854	Detected near miss.(XX, XX)
E8855	No response from interference check board.
E8856	Communication error between interference check board and ARM CONTROL board.
E8857	The number of robots is too many.
E8858	[INTERFERENCE CHECK BOARD]Processing time-out.
E8859	[INTERFERENCE CHECK BOARD]Can not receive data from ARM CTRL BOARD.
E8860	[ARM CTRL BOARD]Cannot receive data from INTERFERENCE CHECK BOARD.
E8861	Communication error between IL server and ARM CONTROL board.
E8862	Cable disconnected between IL server and ARM CONTROL board.
E8900	Detected torque for load presence is abnormal.
E8901	Detected torque for load absence is abnormal.
E8902	Stopped because motion limitation signal was input.
E9000	Joystick of JtXX is disconnected.
E9100	RSC)Watchdog timer overflow.
E9101	RSC)Overvoltage error. (3.3V)
E9102	RSC)Overvoltage error. (5V)
E9103	RSC)Internal processing time time-out.
E9104	RSC)RSC error occurred.(Code:54)
E9105	RSC)Robot number transmission, interprocessor communication error.
E9106	RSC)RSC operation status, interprocessor communication error.
E9107	RSC)I/O output, interprocessor communication error.

Code	Error Message
E9108	RSC)I/O check, interprocessor communication error.
E9109	RSC)Schedule management, timer synchronization error.
E9110	RSC)Main module, interprocessor communication error.
E9111	RSC)Operation part, interprocessor communication error.
E9112	RSC)Tool number input, interprocessor communication error.
E9113	RSC)I/O Port filtering, interprocessor communication error.
E9114	RSC)Robot diagnosis, interprocessor communication error.
E9115	RSC)RSC error occurred.(Code:5F)
E9116	RSC)Ethernet chip writing error.
E9117	RSC)Ethernet chip System. Open failure.
E9118	RSC)RSC error occurred.(Code:62)
E9119	RSC)RSC error occurred.(Code:63)
E9120	RSC)RSC error occurred.(Code:64)
E9121	RSC)Error log addition error.
E9122	RSC)Error log acquisition error.
E9123	RSC)Error log overwrite error.
E9124	RSC)RSC error occurred.(Code:68)
E9125	RSC)RSC error occurred.(Code:69)
E9126	RSC)Current time initialization error.
E9127	RSC)Current time acquisition error.
E9128	RSC)Current time set point error.
E9129	RSC)RSC error occurred.(Code:6D)
E9130	RSC)RSC error occurred.(Code:6E)
E9131	RSC)RSC error occurred.(Code:6F)
E9132	RSC)CPU error.
E9133	RSC)Memory error.
E9134	RSC)CPU status exchange failure.
E9135	RSC)Firmware CRC error.
E9136	RSC)RSC parameter CRC error.
E9137	RSC)RSC error occurred.(Code:75)
E9138	RSC)Mac address CRC error.
E9139	RSC)Initialization failure in "power down backup".
E9140	RSC)RSC error occurred.(Code:78)
E9141	RSC)RSC error occurred.(Code:79)
E9142	RSC)Power-source monitoring process error.
E9143	RSC)Pulse check error.
E9144	RSC)Readback error.

Code	Error Message
E9145	RSC)Relay contact check error.
E9146	RSC)Crosscheck error.
E9147	RSC)Input mismatching check error.
E9148	RSC)First-time encoder data receiving time-out.
E9149	RSC)FPGA operation error.
E9150	RSC)RSC error occurred.(Code:82)
E9151	RSC)RSC error occurred.(Code:83)
E9152	RSC)RSC error occurred.(Code:84)
E9153	RSC)RSC error occurred.(Code:85)
E9154	RSC)RSC error occurred.(Code:86)
E9155	RSC)Command value axes number error.
E9156	RSC)parameter error.(axes number / tool number)
E9157	RSC)"Command value input section" CRC error.
E9158	RSC)Robot number transmitting failure.
E9159	RSC)First command value receive time out.
E9160	RSC)USB communication impossible.
E9161	RSC)command value byte-number corruption.
E9162	RSC)USB Device recognition time-out.
E9163	RSC)command value receive time out.
E9164	RSC)RSC error occurred.(Code:90)
E9165	RSC)RSC error occurred.(Code:91)
E9166	RSC)RSC parameter read failure.
E9167	RSC)Robot number error.
E9168	RSC)Encoder data movement error.
E9169	RSC)Parameter and RC zeroing data mismatch.
E9170	RSC)TCP Communication Retry count over.
E9171	RSC)Rotary switch number error.
E9172	RSC)RSC error occurred.(Code:98)
E9173	RSC)RSC error occurred.(Code:99)
E9174	RSC)Parameter setting range over error.
E9175	RSC)Monitoring area parameter setting error.
E9176	RSC)Parameter error at TOOL monitoring invalid.
E9177	RSC)Operation part internal error.
E9178	RSC)RSC error occurred.(Code:9E)
E9179	RSC)RSC error occurred.(Code:9F)
E9180	RSC)Safety speed over.(TCP)
E9181	RSC)Speed over.(flange point)

Code	Error Message
E9182	RSC)Axis upper limit over.
E9183	RSC)Axis lower limit over.
E9184	RSC)Outside of a SSL area limit.(partial restricted area)
E9185	RSC)Outside of a SSL area limit.(restriction area)
E9186	RSC)Outside of a SSF area limit.
E9187	RSC)Positioning confirmation processing error.
E9188	RSC)TOOL verification error.
E9189	RSC)Distance error between flanges.
E9190	RSC)RSC error occurred.(Code:EA)
E9191	RSC)RSC error occurred.(Code:EB)
E9192	RSC)RSC error occurred.(Code:EC)
E9193	RSC)RSC error occurred.(Code:EE)
E9194	RSC)RSC error occurred.(Code:EE)
E9195	RSC)RSC error occurred.(Code:EF)
E9196	RSC)RSC error occurred.(Code:F0)
E9197	RSC)RSC error occurred.(Code:F1)
E9198	RSC)RSC error occurred.(Code:F2)
E9199	RSC)RSC error occurred.(Code:F3)
E9200	RSC)RSC error occurred.(Code:F4)
E9201	RSC)RSC error occurred.(Code:F5)
E9202	RSC)RSC error occurred.(Code:F6)
E9203	RSC)RSC error occurred.(Code:F7)
E9204	RSC)RSC error occurred.(Code:F8)
E9205	RSC)RSC error occurred.(Code:F9)
E9206	RSC)Encoder receiving timeout error.
E9207	RSC)Encoder receiving timeout error 2.
E9208	RSC)Encoder status error.
E9209	RSC)Encoder data reading Retry count over.
E9210	RSC)RSC error occurred.(Code:FE)
E9211	RSC)RSC error occurred.(Code:FF)
E9300	Cannot rotate JtXX. Because disconnected axis.
E9301	Cannot rotate JtXX. Because invalid axis.
E9302	Rotation speed setting for JtXX is abnormal.
D0001	CPU error.(PC=XX)
D0002	Main CPU BUS error.(PC=XX)
D0003	VME BUS error.(PC=XX)
D0004	[ARM CONTROL BOARD]CPU error.(PC=XX)

Code	Error Message
D0005	[ARM CONTROL BOARD] CPU BUS error.(PC=XX)
D0006	[ARM CONTROL BOARD]Servo control software CPU error. (PC=XX, CodeXX)
D0007	[Servo boardXX]CPU error. (CodeXX)
D0008	[Servo boardXX]Floating point exception. (CodeXX)
D0009	[Servo boardXX]CPU exception. (PC=XX)
D0900	Teach data is broken.
D0901	AS Flash memory sum check error.
D0902	Servo Flash memory sum check error.
D0903	IP board memory error. (XX)
D0904	Memory is locked due to AC_FAIL.
D0906	Data backup to CF is completed. Turn OFF the controller power.
D0907	Data backup to CF is failed. Turn OFF & ON the controller power.
D1000	Read error of servo control software.
D1001	Download error of servo control software.
D1002	Init. error of servo software.
D1003	Init. error of servo control software.
D1004	[ARM CTRL BOARD]Watch dog error of Servo control software.
D1005	Servo board command error. (XX)
D1006	Servo system error.
D1007	Regenerative time over. [XX]
D1008	P-N low voltage. [XX]
D1009	P-N high voltage. [XX]
D1010	Regenerative resistor overheat. [XX]
D1011	As or servo software is not compatible with the robot model.
D1012	Servo type mismatch. Check the settings.
D1013	P-N capacitor is not discharged.
D1014	Servo system error.(Code=XX)
D1015	The servo data file does not exist.
D1016	Data applicable to the robot model not in servo data file.
D1017	Error of download of servo data.
D1018	Servo software version mismatch.
D1019	[ARM CONTROL BOARD]Watchdog error in timer built-in CPU for servo control.
D1020	[ARM CONTROL BOARD]Synchronous error between CPUs.
D1021	Servo FPGA configuration data not found.
D1022	Configuration error in servo FPGA.(CodeXX)

Code	Error Message
D1023	Current mismatch betw. m-plexer&software. JtXX
D1024	[ARM CONTROL BOARD]Servo FPGA detected Watch dog error on ARMSC Software.
D1025	[Servo boardXX]Detected Watch dog error.(Servo FPGA)
D1026	[Servo boardXX]Input abnormal signal from power sequence board.
D1027	[MCXX]Detected Watch dog error.
D1028	[Servo boardXX]DC is abnormal.(Servo FPGA)
D1029	[Servo boardXX]AC primary power is abnormal.(Servo FPGA)
D1030	Cannot start communication with the servo boardXX.
D1031	Read error of servo software.
D1032	[Servo boardXX]Download error of servo software.(CodeXX)
D1033	Connection Port No(XX) and Servo board No(XX) mismatch.
D1034	The servo data file is missing or not acceptable.(CodeXX)
D1035	[Servo boardXX]Init. error of servo software.(CodeXX)
D1036	[Servo boardXX]Download error of servo data.(CodeXX)
D1037	[Servo boardXX]Configuration error in servo FPGA.(CodeXX)
D1038	[Servo boardXX]Upload error of servo software initial data.(CodeXX)
D1039	[Servo boardXX]Download error of servo software initial data.(CodeXX)
D1040	[Servo boardXX]Device check error. (CodeXX)
D1041	JtXX axis brake release circuit is abnormal.
D1500	Encoder misread error. JtXX
D1501	Defective gun changer connection or encoder comm. error.
D1502	Amp overcurrent. JtXX
D1503	Current detector type (XX) mismatch!
D1504	Abn. curr feedback JtXX. (Amp fail, pwr harness disconnect)
D1505	Motor harness disconnected or amplifier overheated.(XX)
D1506	Power module error. JtXX
D1507	AC primary power OFF.
D1508	24VDC power source is too low.
D1509	Primary power source is too high.
D1510	Primary power source is too low.
D1511	+12VDC or -12VDC is abnormal.
D1512	Brake line error for JtXX.
D1513	Brake power is abnormal.(XX)
D1514	I/O 24V fuse is open.
D1515	Mismatch in setting of safety circuit as single/double.
D1516	Mismatch betw hard/software settings for HOLD backup time.

Code	Error Message
D1517	Blown fuse on safety circuit emergency line.
D1518	Mismatch in the Emer. Stop condition on safety circuit.
D1519	Mismatch in safety circuit LS conditions.
D1520	Mismatch in safety circuit TEACH/REPEAT condition.
D1521	Mismatch in safety circuit safety-fence condition.
D1522	Mismatch in cond. of safety circuit enabling device.
D1523	Mismatch in cond. of safety circuit ext.enabling device.
D1524	Incorrect operation of the safety relay.
D1525	Incorrect operation of MC(K1).
D1526	Incorrect operation of MC(K2).
D1527	Incorrect operation of MC(K3).
D1528	Controller temperature is out of range.
D1529	Signal harness disconnected or encoder power error.
D1530	Abnormal current limit of JtXX.
D1531	Heat sink on power block overheated.
D1532	(SSCNET)EnCoder communication error.(JtXX)(CodeXX)
D1533	(SSCNET)Absolute position of JtXX is erased.(CodeXX)
D1534	(SSCNET)Parameter error of JtXX.(CodeXX)
D1535	(SSCNET)Alarm of JtXX.(CodeXX)
D1536	JtXX does not move normally.
D1537	Brake rectifier relay failure.
D1538	DC 24V is abnormal.
D1539	Power supply circuit for PWM signal output malfunctioned.
D1540	Amplifier overheats.(XX)
D1541	Encoder type set in software and arm control board mismatch.
D1542	No Rotation data from multidrop encoder at initialize.
D1543	[Servo boardXX]DC 5V is abnormal.
D1544	[Servo boardXX]DC 3.3V is abnormal.
D1545	[Servo boardXX]DC 12V is abnormal.
D1546	[Servo boardXX]DC 2.5V is abnormal.
D1547	[Servo boardXX]DC 1.2V is abnormal.
D1548	[Servo boardXX]DC 1.0V is abnormal.
D1549	[Servo boardXX]Primary power source is too low.
D1550	[Servo boardXX]Primary power source is too high.
D1551	[Servo boardXX]AC primary power OFF.
D1552	[MCXX]DC 3.3V is abnormal.
D1553	[MCXX]DC 5V is abnormal.

Code	Error Message
D1554	Brake power in servo amplifiers abnormal.
D1555	Amplifier temperature is out of range or regenerative resistor overheats.
D1556	Control power in servo amplifier is abnormal.
D1557	[Power sequence board]DC 3.3V is abnormal.
D1558	[Power sequence board]DC 5V is abnormal.
D1559	[Power sequence board]DC 12V is abnormal.
D1560	[Power sequence board]DC 24V is abnormal.
D1561	[Power sequence board]AC primary power OFF.
D1562	[Power sequence board]AC primary power voltage is too high.
D1563	[Power sequence board]AC primary power voltage is too low.
D1564	[Power sequence board]Remote power off signal was detected.
D1565	Cannot access power sequence board.(CodeXX)
D1566	P-N capacitor has not discharged.(Servo boardXX)(MCXX)
D1567	[Servo boardXX]Primary Power source is error.
D1568	[Servo boardXX]Power supply circuit for PWM signal output malfunctioned.
D1569	Servo amplifier is abnormal.(XX)
D2000	No response from Comm. board for Laser sensor.
D2001	RI/O or C-NET board initialize error.
D2002	No response from the Arm ID board.
D2003	No data in the Arm ID board.
D2004	Mismatch data in the Arm ID board.
D2005	CC-LINK software version mismatch.
D2006	Watch dog error on communication board for Explosion proof TP.
D2007	No response from the built-in sequence board.
D2008	Magnet is Contactor of groupXX is stuck.
D2009	Sensor for detecting pressure in enclosure is abnormal.
D2010	Sync. error between User I/F and Arm control board.
D2011	Parameter download error betw User I/F & Arm control boards.
D2012	Soft Absorber error. Turn OFF & ON the control power.
D2013	Change gain error. Turn OFF & ON the control power.
D2014	Robot network initialize error.
D2016	No response from the Arm control board.
D2017	No response from User I/F board.
D2018	[ARM CTRL BOARD]No response.
D2019	[ARM CTRL BOARD]Servo software no response.
D2020	[ARM CTRL BOARD]Servo control software no response.
D2021	Arm data file is not found.

Code	Error Message
D2022	Arm data is not found.
D2023	Failed to load arm data.
D2024	[ARM CTRL BOARD]Robot type setting failed.
D2025	Robot codes set in software and Arm ctrl board do not match.
D2026	Codes set in software & curr. sensor I/F b'd do not match.
D2027	Codes set in software and power block do not match.
D2028	(SSCNET) Initialization error. (CodeXX)
D2029	Motor codes in software & Arm control b'd mismatch.(Jt-A)
D2030	Codes set in software & curr. sensor I/F b'd mismatch.(Jt-A)
D2031	Codes set in software and on add'l pwr block mismatch.(Jt-A)
D2032	Motor codes set in software and Arm ctrl b'd mismatch.(Jt-B)
D2033	Codes set in software & curr. sensor I/F b'd mismatch.(Jt-B)
D2034	Codes set in software and on add'l pwr block mismatch.(Jt-B)
D2035	Program execution error.
D2036	(SSCNET)System error occurred in 1LP I/F board. (CodeXX)
D2037	Safety unit circuit is abnormal.
D2038	(SSCNET)Interface board is not installed.
D2039	(SSCNET)Communication error of JtXX on initialization.
D2040	(SSCNET)Initialization error of JtXX.(CodeXX)
D2041	Connection of the signal harness is wrong.
D2042	Servo amp and robot arm are mismatched.
D2043	Arm I/F board detects AC-Fail.
D2044	[ARM CONTROL BOARD]No response from Servo FPGA software.
D2045	[ARM CONTROL BOARD]Device check error. (CodeXX)
D2046	Relay error on purge control board. (relay XX)
D2047	Jumper setting error or Safety relay failure on Servo CPU board.
D2048	DC 12V Voltage source error on purge control board.
D2049	Over current error in interlock relay drive circuit(1) for purge control board.
D2050	Over current error in interlock relay drive circuit(2) for purge control board.
D2051	Communication error on purge control board.
D2052	Hardware setting for the external axis amplifier has discrepancy. robot=n
D2053	(FANXX-XX)Rotational speed of fan is abnormal.(Servo boardXX)
D2054	Codes set in software and power block do not match.(Code:XX)
D2055	[Power sequence board]Watchdog error was detected.
D2056	[I/O board(No.XX)]Several boards have same ID address.
D2057	[Servo boardXX]No response from Servo FPGA software.
D2058	[Main CPU board]DC power supply is abnormal.(XX mV)

Code	Error Message
D2059	1SP board is abnormal.(DXX)
D2060	Safety unit is abnormal.(DXX)
D2061	Mother board is abnormal.(DXX)
D2062	1QL board is abnormal.(DXX)
D2063	MC unit is abnormal.(DXX)
D2064	[Purge control board]Pressure within enclosure is low.(during purging)
D2065	Safety relay is abnormal which cut off brake power when inner pressure is low.
D2066	[Purge control board]DC is abnormal.(12V)
D2067	[Main CPU board]Communication with purge control board is abnormal.
D2068	[IO board No. XX]Device check failure.(CodeXX)
D2069	[ANYBUS interface board(No.XX)]Several boards have the same ID address.
D3800	Communication board memory error. (XX)
D3801	JtXX axis amp interface error 1.
D3802	JtXX axis amp interface error 2.
D3803	JtXX axis amp interface error 3.
D3804	JtXX axis amp power element error.
D3805	JtXX axis amp current detector error.
D3806	JtXX axis amp main circuit voltage unmatch.
D3807	JtXX axis amp memory error.(EEPROM error)
D3808	JtXX axis amp inside RAM error.
D3809	JtXX axis amp servo processor error.
D3810	JtXX axis amp parameter error.
D3811	JtXX axis amp initial processing error.
D3812	JtXX axis amp undefinition error 1.
D3813	Amp communication I/F board initialed check error.(XX)
D3814	Amp communication I/F board undefinition error.(XX)
D3815	It is not possible to communicate with JtXX axis amp.
D3816	JtXX axis amp communication frame reception error.
D3817	JtXX axis amp communication frame reception timeout.
D3818	JtXX axis amp communication bank data error.
D3819	JtXX axis amp init timeout.
D3820	JtXX axis amp communication undefinition error.
D3821	Motor harness connection point is error.
D3822	Motor parameter is not consistent with controller. JtXX
D3823	FAN NO. XX in Controller is out of order.
D3824	Fuse NO.XX on IO board NO.1 is open.
D3825	Fuse NO.XX on IO board NO.2 is open.

Code	Error Message
D3826	Robot DC voltage error.
D3828	Controller type error.
D3829	K1 and/or K2 works wrong.
D3830	PN high voltage error.
D3831	PN low voltage error.
D3832	Register over time error.
D3833	Discharge resistor overheated.
D3834	Power board switching circuit is abnormal.
D3835	Power board inrush current limiting circuit is abnormal.
D3836	DC Power voltage is abnormal.(CodeXX)
D3837	JtXX axis amp control power supply error.
D3838	Power board is abnormal.
D3839	Servo control line error.
D3840	FAN NO. XX on power board is out of order.
D3841	RobotXX servo amp is missing.
D3842	Control power supply for a power device is abnormal.
D3843	Brake release setting is abnormal.
D4000	[DIAG]Error is detected in RS232C.(Code:XX)
D4001	[DIAG]Error is detected in Ethernet.(Code:XX)
D4500	Fieldbus interface board is not detected.
D4501	ABMA-PDP)I/F module error. XX
D4502	FIELD-BUS-INIT)Error reply. XX
D4503	FIELD-BUS-INIT)Reply timeout. XX
D4504	ANYBUS)OUT/FB.CTRL request timeout. XX
D6000	Over temperature error in Barrier unit.
D6001	Mutual-Wait initialize error.

Appendix 5 AS Language List (Alphabetical Order)

The abbreviation for each AS language may be changed without prior notice.

The alphabets after the function represent the following:
M: monitor commands, E: editor commands, P: program instructions, S: switches,
F: functions, O: operators, K: other keywords.

Name	Abbreviation	Function		Format (Parameter)	Reference
ABORT	AB	Stops execution	M	ABORT	5.4
ABOVE	AB	Changes elbow joint to above position	P	ABOVE	6.4
ABS	ABS	Returns absolute value	F	ABS (real value)	9.3
ABS.SPEED	ABS.SPEED	Enables use of absolute speed	S	...ABS.SPEED...	7
ACCEL	ACCE	Sets acceleration	P	ACCEL acceleration ALWAYS	6.2
ACCURACY	ACCU	Sets accuracy range	P	ACCURACY distance ALWAYS FINE	6.2
AFTER.WAIT.TIMR	AF	Sets how timers begin in block step programs	S	AFTER.WAIT.TIMR	7
ALIGN	AL	Aligns tool Z axis with base coordinate axis	P	ALIGN	6.1
AND	AND	Logical AND	O	.... AND	8.3
ASC	ASC	Returns ASCII value	F	ASC (string, character number)	9.1
ATAN2	ATAN2	Returns the arctangent value	F	ATAN2 (real value1, real value2)	9.3
AUTOSTART.PC	AUTOSTART.	Starts PC program automatically	S	AUTOSTART.PC	7
AUTOSTART2.PC	AUTOSTART2.	Starts PC program automatically	S	AUTOSTART2.P	7
AUTOSTART3.PC	AUTOSTART3.	Starts PC program automatically	S	AUTOSTART3.PC	7
AUTOSTART4.PC	AUTOSTART4.	Starts PC program automatically	S	AUTOSTART4.PC	7
AUTOSTART5.PC	AUTOSTART5.	Starts PC program automatically	S	AUTOSTART5.PC	7
AVE_TRANS	AVE_TRANS	Returns average value	F	AVE_TRANS (transformation value variable1, transformation value variable2)	9.2
BAND	BAND	Binary AND	O	... BAND ...	8.4
BASE	BA	Defines base transformation values	M	BASE transformation value variable	5.6
BASE	BA	Defines base transformation values	P	BASE transformation value variable	6.9
BASE	BASE	Returns base transformation values	F	BASE	9.2
BELOW	BE	Changes elbow joint to below position	P	BELOW	6.4
BITS	BI	Sets output signals	M	BITS starting signal number , number of signals = value	5.7
BITS	BI	Sets output signals	P	BITS starting signal number , number of signals = decimal value	6.7
BITS	BITS	Returns signal status	F	BITS (starting signal number , number of signals)	9.1
BITS32	BITS32	Sets output signals	M	BITS32 starting signal number , number of signals = value	5.7

Name	Abbreviation	Function		Format (Parameter)	Reference
BITS32	BITS32	Sets output signals	P	BITS 32 starting signal number , number of signals = value	6.7
BITS32	BITS32	Returns signal status	F	BITS32 (starting signal number, number of signals)	9.1
BOR	BOR	Binary OR	O	... BOR ...	8.4
BRAKE	BRA	Stops robot motion immediately	P	BRAKE	6.2
BREAK	BRE	Causes a break in CP motion	P	BREAK	6.2
BSPEED	BSP	Sets block speed	P	BSPEED speed	6.2
BXOR	BX	Binary XOR	O	... BXOR ...	8.4
BY	BY	Shift amount	K	SHIFT (transformation value variable BY X shift, Y shift, Z shift)	9.2
C	C	Changes program to edit	E	C program name, step number	5.1
C1MOVE	C1	Circular interpolated motion	P	C1MOVE pose variable, clamp number	6.1
C2MOVE	C2	Circular interpolated motion	P	C2MOVE pose variable, clamp number	6.1
CALL	CA	Calls subroutine	P	CALL program name	6.5
CASE	CASE	CASE structure	P	CASE index variable OF ... VALUE ... ANY... END	6.6
CCENTER	CCENTER	Returns the center of the arc	F	CCENTER (transformation value variable 1, transformation value variable 2, transformation value variable 3,transformation value variable 4)	9.2
CHECK.HOLD	CH	Enables or disables input of commands from the keyboard when HOLD/RUN is in HOLD	S	CHECK.HOLD	7
$CHR	$CHR	Returns ASCII characters	F	$CHR(real value)	9.4
CHSUM	CH	Enables/disables resetting of abnormal check sum error	M	CHSUM	5.6
CLAMP	CLAMP	Controls open/close clamp signals	P	CLAMP clamp number 1, …, clamp number 8	6.7
CLOSE	CLOSE	Closes clamp hand	P	CLOSE clamp number	6.3
CLOSEI	CLOSEI	Closes clamp hand	P	CLOSEI clamp number	6.3
CLOSES	CLOSES	Turns ON/OFF close clamp signal	P	CLOSES clamp number	6.3
COM	COM	Binary complement	O	... COM ...	8.4
CONF_VARIABLE	CONF_VARIABLE	Configuration change in linear motion	S	CONF_VARIABLE	7
CONTINUE	CON	Resumes execution	M	CONTINUE NEXT	5.4
COPY	COP	Copies programs in robot memory	M	COPY new program name = source program name +source program name+...	5.2
COS	COS	Returns the cosine value	F	COS (real value)	9.3
CP	CP	Continuous path (CP) function	S	CP	7
CS	CS	CYCLE START switch ON/OFF status	S	switch (CS)	7

Name	Abbreviation	Function		Format (Parameter)	Reference
CSHIFT	CSHIFT	Returns the shifted pose	F	CSHIFT (transformation value variable 1, transformation value variable 2, transformation value variable 3, transformation value variable 4 BY shift amount)	9.2
CURLIM	CURLIM	Modify external axis motor current limit	P	CURLIM axis number, positive current limit, negative current limit	6.9
CURLIMM	CURLIMM	Acquire external axis motor current limit value (negative)	F	CURLIMM (axis number)	9.1
CURLIMP	CURLIMP	Acquire external axis motor current limit value (positive)	F	CURLIMP (axis number)	9.1
CYCLE.STOP	CY	Stops cycle with External HOLD	S	CYCLE.STOP	7
D	D	Deletes program steps	E	D number of steps	5.1
$DATE	$DATE	Returns system date	F	$DATE (date form)	9.4
DECEL	DECE	Sets deceleration	P	DECEL deceleration ALWAYS	6.2
$DECODE	$DECODE	Extracts characters	F	$DECODE (string variable, separator character, mode)	9.4
DECOMPOSE	DECO	Extracts components of pose variable	P	DECOMPOSE array variable [element number]= pose variable	6.9
DEFSIG	DEF	Displays and changes software dedicated signals	M	DEFSIG OUTPUT/INPUT	5.6
DELAY	DEL	Stops the robot for a given time	P	DELAY time	6.1
DELETE	DEL	Deletes data in memory	M	DELETE (/P)(/L)(/R)(/S)(/INT) program name/pose variable/real variable, ...	5.2
DELETE	DEL	Deletes data in memory	P	DELETE (/P)(/L)(/R)(/S)(/INT) program name/pose variable/real variable, ...	6.10
#DEST	#DEST	Returns destination in transformation values	F	#DEST	9.2
DEST	DEST	Returns destination in transformation values	F	DEST	9.2
DEST_CIRINT	DEST_CIRINT	Determines the position to return via DEST/ #DEST function	S	DEST_CIRINT	7
DEXT	DEXT	Returns specified element of given pose	F	DEXT (pose variable, element number)	9.1
DIRECTORY	DI	Display in list the specified program/data name	M	DIRECTORY(/P)(/L)(/R)(/S)(/INT) program name/pose variable /real variable, …	5.2
DISP.EXESTEP	DISP.EXESTEP	Change display of step at program execution	S	DISP.EXESTEP	7
DISPIO_01	DIS	Changes the display mode of IO command	S	DISPIO_01	7
DISTANCE	DISTANCE	Returns distance	F	DISTANCE (transformation value variable1, transformation value variable2)	9.1
DIVIDE.TPKEY_S	DIVIDE.TPKEY_S	Switch A key information allocation	S	DIVIDE.TPKEY_S	7
DLYSIG	DL	Outputs signal after delay	M	DLYSIG signal number, time	5.7
DLYSIG	DL	Outputs signal after delay	P	DLYSIG signal number, time	6.7
DO	DO	Executes a single program instruction	M	DO program instruction	5.4
DO	DO	DO structure	P	DO....UNTIL....	6.6

Name	Abbreviation	Function		Format (Parameter)	Reference
DRAW	DRA	Moves the robot by a given amount	P	DRAW X translation, Y translation, Z translation, X rotation, Y rotation, Z rotation, speed	6.1
DRIVE	DRI	Moves a single joint	P	DRIVE joint number, displacement, speed	6.1
DWRIST	DW	Changes wrist configuration	P	DWRIST	6.4
DX	DX	Returns X component	F	DX (transformation value variable)	9.1
DY	DY	Returns Y component	F	DY (transformation value variable)	9.1
DZ	DZ	Returns Z component	F	DZ (transformation value variable)	9.1
E	E	Exits from edit mode	E	E	5.1
EDIT	ED	Enters edit mode	M	EDIT program name, step number	5.1
ELSE	EL	IF structure	P	IF ... ELSE ... END	6.6
ENA_TOOLSHAPE	ENA_TOOLSHAPE	Enables/ Disables speed control using tool shape	M	ENA_TOOLSHAPE tool shape no.=TRUE/FALSE	5.6
ENA_TOOLSHAPE	ENA_TOOLSHAPE	Enables/ Disables speed control using tool shape	P	ENA_TOOLSHAPE tool shape no.=TRUE/FALSE	6.9
ENC_TEMP	ENC_TEMP	Displays encoder's current temperature, lowest temperature, highest temperature	M	ENC_TEMP joint number	5.6
ENCCHK_EMG	ENCCHK_E	Sets deviation range of robot pose at E-stop	M	ENCCHK_EMG	5.6
ENCCHK_PON	ENCCHK_P	Sets acceptable encoder value of robot pose at E-stop	M	ENCCHK_PON	5.6
$ENCODE	$ENCODE	Returns string created by print data	F	$ENCODE (print data, print data, …...)	9.4
END	EN	FOR structure, etc.	P	FOR loop variable=start value TO end value... END	6.6
ENV_DATA	ENV	Sets hardware environmental data	M	ENV_DATA	5.6
ENV2_DATA	ENV2	Sets software environmental data	M	ENV2_DATA	5.6
ENVCHKRATE	ENVCHKRATE	Magnification ratio for initial value of ext. axis deviation error detection threshold value	P	ENVCHKRATE axis number, coefficient	6.9
ENVCHKRATE	ENVCHKRATE	Acquire the set value for magnification ratio for initial value of ext. axis deviation error detection threshold value	F	ENVCHKRATE (axis number)	9.1
ERESET	ERE	Resets error condition	M	ERESET	5.6
ERRLOG	ERR	Displays error log	M	ERRLOG	5.6
$ERRLOG	$ERRLOG	Returns string for error message	F	$ERRLOG (error log number)	9.4
ERROR	ERROR	In error status	S	switch (ERROR)	7
ERROR	ERROR	Returns program error status	F	ERROR	9.1
$ERROR	$ERROR	Returns error messages	F	$ERROR (error number)	9.4
$ERRORS	$ERRORS	Returns error messages	F	$ERRORS ("error code")	9.4
ERRSTART.PC	ERRS	Executes PC program when an error occurs	S	ERRSTART.PC	7
EXECUTE	EX	Executes a robot program	M	EXECUTE program name, execution cycles, step number	5.4

Name	Abbreviation	Function		Format (Parameter)	Reference
EXISTCHAR	EXISTCHAR	Checks if the variable exists or not (string variables)	F	EXISTCHAR ("string variable name")	9.1
EXISTDATA	EXISTDATA	Checks if the variable or program exists or not	F	EXISTDATA ("variable name or program name")	9.1
EXISTINTEGER	EXISTINTEGER	Checks if the variable exists or not (integer variables)	F	EXISTINTEGER ("integer variable name")	9.1
EXISTJOINT	EXISTJOINT	Checks if the variable exists or not (joint displacement values)	F	EXISTJOINT ("name of joint displacement value variable")	9.1
EXISTLOCALCHAR	EXISTLOCALCHAR	Checks if the local variable exists or not (string variables)	F	EXISTLOCALCHAR ("local string variable name")	9.1
EXISTLOCALINTEGER	EXISTLOCALINTEGER	Checks if the local variable exists or not (integer variables)	F	EXISTLOCALINTEGER ("local integer variable name")	9.1
EXISTLOCALJOINT	EXISTLOCALJOINT	Checks if the local variable exists or not (joint displacement values)	F	EXISTLOCALJOINT ("name of local joint displacement value variable")	9.1
EXISTLOCALREAL	EXISTLOCALREAL	Checks if the local variable exists or not (real variables)	F	EXISTLOCALREAL ("local real variable name")	9.1
EXISTLOCALTRANS	EXISTLOCALTRANS	Checks if the local variable exists or not (transformation value variables)	F	EXISTLOCALTRANS ("name of local transformation value variable")	9.1
EXISTPGM	EXISTPGM	Checks if the program exists or not	F	EXISTPGM ("program name")	9.1
EXISTREAL	EXISTREAL	Checks if the variable exists or not (real variables)	F	EXISTREAL ("real variable name")	9.1
EXISTTRANS	EXISTTRANS	Checks if the variable exists or not (transformation values)	F	EXISTTRANS ("name of transformation value variable")	9.1
EXTCALL	EX	Calls program selected by signals	P	EXTCALL	6.7
F	F	Searches for a string	E	F character string	5.1
FFRESET	FFRESET	Resets acceleration feed forward gain.	P	FFRESET	6.2
FFSET	FFSET	Sets acceleration feed forward gain.	P	FFSET JT1 gain, JT2 gain, ..., JT9 gain	6.2
FFSET_STATUS	FFSET_STATUS	Displays the acceleration feed forward gain setting.	M	FFSET_STATUS	5.6
FLOWRATE	FLOWRATE	Changes flow rate control mode	S	FLOWRATE	7
FOR	FOR	FOR structure	P	FOR loop variable=start value TO end value…END	6.6
FRAME	FRAME	Returns the transformation values for frame coordinates	F	FRAME (transformation value varible 1, transformation value varible 2,transformation value varible 3, transformation value varible 4)	9.2
FREE	FR	Displays size of free memory	M	FREE	5.6
GETENCTEMP	GETENCTEMP	Encoder temperature of specified axis [ºC]	F	GETENCTEMP (axis number)	9.1
GOTO	GO	Jumps to label	P	GOTO label IF condition	6.5
GUNOFF	GUNOFF	Turns OFF gun signals	P	GUNOFF gun number, distance	6.3
GUNOFFTIMER	GUNOFFTIMER	Controls gun output OFF timing	P	GUNOFFTIMER gun number, time	6.3
GUNON	GUNON	Turns ON gun signals	P	GUNON gun number, distance	6.3
GUNONTIMER	GUNONTIMER	Controls gun output ON timing	P	GUNONTIMER gun number, time	6.3
HALT	HA	Stops execution	P	HALT alpha character	6.5
HELP	HEL	Displays a list of AS commands and instructions	M	HELP alpha character	5.6

Name	Abbreviation	Function		Format (Parameter)	Reference
HELP/DO	HEL/DO	Displays a list of functions	M	HELP/DO alpha character	5.6
HELP/F	HEL/F	Displays a list of monitor commands	M	HELP/F alpha character	5.6
HELP/M	HEL/M	Displays a list of commands usable with MC instructions	M	HELP/M alpha character	5.6
HELP/MC	HEL/MC	Displays a list of instructions usable with DO command	M	HELP/MC alpha character	5.6
HELP/P	HEL/P	Displays a list of program instructions	M	HELP/P alpha character	5.6
HELP/PPC	HEL/PPC	Displays a list of instructions usable in PC programs	M	HELP/PPC alpha character	5.6
HELP/SW	HEL/SW	Displays a list of system switches	M	HELP/SW alpha character	5.6
HERE	HE	Records current pose	M	HERE pose variable	5.5
HERE	HE	Records current pose	P	HERE pose variable	6.9
#HERE	#HERE	Returns transformation values for current pose	F	#HERE	9.2
HERE	HERE	Returns joint displacement values for current pose	F	HERE	9.2
HMOVE	HM	Moves in hybrid motion	P	HMOVE pose variable, clamp number	6.1
HOLD	HO	Stops execution	M	HOLD	5.4
HOLD.STEP	HOLD.STEP	Enables display of the step in execution when the program is held	S	HOLD.STEP	7
HOME	HO	Moves to home pose	P	HOME home pose number	6.1
#HOME	#HOME	Returns joint displacement values for home pose	F	#HOME (home pose number)	9.2
HSENSE	HSENSE	Reads HSENSESET data	M	HSENSE No. result variable, signal status variable, pose variable, error variable, memory usage variable	5.7
HSENSE	HSENSE	Reads HSENSESET data	P	HSENSE No result variable, signal status variable, pose variable, error variable, memory remainder variable	6.7
HSENSESET	HSENSESET	Starts monitoring of signal	M	HSENSESET No = input signal number, output signal number, signal output delay time	5.7
HSENSESET	HSENSESET	Starts monitoring of signal	P	HSENSESET No = input signal number, signal output delay number	6.7
HSETCLAMP	HS	Assigns signal number to operate clamps	M	HSETCLAMP	5.6
I	I	Inserts new steps	E	I	5.1
ID	ID	Displays version information	M	ID	5.6
IF	IF	Jumps to label when condition is set	P	IF condition GOTO label	6.5
IF	IF	IF structure	P	IF...THEN...ELSE...END	6.6
IFAKEY	IFAKEY	Allow operation while A key is pressed	S	IFAKEY	7
IFPDISP	IFPDISP	Displays the specified page of I/F panel	M	IFPDISP page number	5.8
IFPDISP	IFPDISP	Displays the specified page of I/F panel	P	IFPDISP page number	6.8

Name	Abbreviation	Function		Format (Parameter)	Reference
IFPLABEL	IFPLABEL	Sets label for I/F panel	M	IFPLABEL position, "label1","label2","label3","label4"	5.8
IFPLABEL	IFPLABEL	Sets label for I/F panel	P	IFPLABEL position, "label1","label2","label3","label4"	6.8
IFPTITLE	IFPTITLE	Set title for I/F panel	M	IFPTITLE page no., "title"	5.8
IFPTITLE	IFPTITLE	Set title for I/F panel	P	IFPTITLE page no., "title"	6.8
IFPWOVERWRITE	IFPWOVERWRITE	Overwrite and displays string in string window	M	IFPWOVERWRITE mode window number, row, column, background color, label color = "character string", "character string",…	5.8
IFPWOVERWRITE	IFPWOVERWRITE	Overwrite and displays string in string window	P	IFPWOVERWRITE mode window number, row, column, background color, label color = "character string", "character string",…	6.8
IFPWPRINT	IFPWPRINT	Displays string in string window set by aux. function	M	IFWPRINT window number, row, column, background color, label color = "character string", "character string",…	5.8
IFPWPRINT	IFPWPRINT	Displays string in string window set by aux. function	P	IFWPRINT window number, row, column, background color, label color = "character string","character string", …	6.8
IGNORE	IG	Cancel ON or ONI instruction	P	IGNORE signal number	6.7
INPUT	I	Software dedicated input signals	K	DEFSIG INPUT	5.6
INRANGE	INRANGE	Returns the result of motion range check	F	INRANGE (pose variable1, joint displacement value variable)	9.1
INS_POWER	INS_POWER	Acquires instantaneous power	F	INS_POWER (power number)	9.1
INSERT_NO_CONFIRM	INSERT_NO_CONFIRM	ON/OFF of confirmation message for insert operation in teach operation	S	INSERT_NO_CONFIRM	7
INSTR	INSTR	Returns the starting point of the specified string	F	INSTR(starting point, string 1, string 2)	9.1
INT	INT	Returns the integer value of numeric expression	F	INT (numeric expression)	9.1
INTERP_FTOOL	INTERP_FTOOL	Switches constant selecting of fixed tool coordinates	S	INTERP_FTOOL	7
INVALID.TPKEY_S	INVALID.TPKEY_S	Invalidate SHIFT when A key is pressed	S	….INVALID.TPKEY_S….	7
IO	IO	Displays signal states	M	IO/E signal number	5.6
IPEAKCLR	IPEAKCLR	Displays peak current value for each joint	M	IPEAKCLR	5.6
IPEAKLOG	IPEAKLOG	Displays peak current value log	M	IPEAKLOG	5.6
IQARM	IQARM	Acquires current value for specified axis	F	IQARM(axis number)	9.1
JAPPRO	JA	Approaches pose in joint interpolated motion	P	JAPPRO pose variable, distance	6.1
JDEPART	JD	Withdraws from current pose in joint interpolated motion	P	JDEPART distance	6.1
JMOVE	JM	Starts joint interpolated motion	P	JMOVE pose variable, clamp number	6.1
JUMP	JUMP	Switch executing program	P	JUMP program name	6.5
KILL	KI	Initializes program stack	M	KILL	5.4

Name	Abbreviation	Function		Format (Parameter)	Reference
L	L	Selects the previous step	E	L	5.1
LANGUAGE	LANGUAGE	Sets the display language.	M	LANGUAGE language number	5.6
LAPPRO	LA	Approaches pose in linear interpolated motion	P	LAPPRO pose variable, distance	6.1
LDEPART	LD	Withdraws from current pose in linear interpolated motion	P	LDEPART distance	6.1
$LEFT	$LEFT	Returns the leftmost characters	F	$LEFT(string, number of characters)	9.4
LEFTY	LE	Changes to left-hand configuration	P	LEFTY	6.4
LEN	LEN	Returns number of characters	F	LEN (string)	9.1
LIST	LI	Displays data or program listing	M	LIST (/P)(/L)(/R)(/S)(/INT) program name/pose variable/real variable,…	5.2
LLIMIT	LL	Sets lower limit of robot motion	M	LLIMIT joint displacement value variable	5.6
LLIMIT	LL	Sets lower limit of robot motion	P	LLIMIT joint displacement value variable	6.9
LMOVE	LM	Starts linear interpolated motion	P	LMOVE pose variable, clamp number	6.1
LOAD	LO	Loads contents of PC into robot memory	M	LOAD/Q filename	5.3
LOCK	LO	Changes priority	P	LOCK priority	6.5
LSTRACE	LSTRACE	Displays the logging data	M	LSTRACE stepper number: logging number	5.2
M	M	Modifies characters	E	M/existing characters/new characters	5.1
MAXINDEX	MAXINDEX	Returns the largest element in the specified dimension	F	MAXINDEX (string variable, dimension number)	9.1
MAXVAL	MAXVAL	Returns the largest value	F	MAXVAL (real value1, real value 2, …)	9.1
MC	M	Executes monitor commands from PC programs	P	MC monitor commands	6.9
MESSAGES	ME	Enables or disables terminal output	S	MESSAGES	7
$MID	$MID	Returns characters	F	$MID(string, real value, number of characters)	9.4
MININDEX	MININDEX	Returns the smallest element in the specified dimension	F	MININDEX (string variable, dimension number)	9.1
MINVAL	MINVAL	Returns the smallest value	F	MINVAL (real value1, real value 2, …)	9.1
MM/MIN	MM/M	millimeters per minute	K	... MM/MIN	6.2
MM/S	MM/S	millimeters per second	K	... MM/S	6.2
MOD	MOD	Remainder	O	... MOD ...	8.1
MON_SPEED	MON_SPEED	Sets monitor speed	P	MON_SPEED monitor speed	6.2
MON_TWAIT	MON_TWAIT	Wait time corresponding to speed setting	P	MON_TWAIT time	6.5
MSPEED	MSPEED	Returns the current monitor speed	F	MSPEED	9.1
MSPEED2	MSPEED2	Returns the current monitor speed	F	MSPEED2	9.1
MSTEP	MS	Executes a single robot motion	M	MSTEP program name, execution cycles, step number	5.4
MVWAIT	MVWAIT	Waits until given time or distance is reached	P	MVWAIT value	6.5
NEXT	N	Skips to next step	K	CONTINUE NEXT	5.4

Name	Abbreviation	Function		Format (Parameter)	Reference
NLOAD	NLOAD	Loads files to robot memory	P	NLOAD/IF/ARC device number=file name + file name + ・・・, status variable	6.10
NOT	NOT	Logical NOT	O	... NOT ...	8.3
NO_SJISCONV	NO_SJISCONV	File character code switch at save/ load	S	NO_SJISCONV	7
NULL	NULL	Returns null transformation values	F	NULL	9.2
O	O	Places the cursor in current line	E	O	5.1
OFF	OF	Turns OFF system switches	M	switch name, ... OFF	5.6
OFF	OF	Turns OFF system switches	P	switch name, ... OFF	6.9
OFF	OFF	Returns FALSE value	F	...OFF....	9.1
ON	ON	Turns ON system switches	M	switch name, ... ON	5.6
ON	ON	Sets interruption condition	P	ON mode signal number CALL program name, priority	6.7
ON	ON	Sets interruption condition	P	ON mode signal number GOTO label, priority	6.7
ON	ON	Turns ON system switches	P	switch name, ... ON	6.9
ON	ON	Returns TRUE value	F	.....ON.....	9.1
ONE	ONE	Calls specified program when error occurs	P	ONE program name	6.5
ONI	ONI	Sets interruption condition	P	ONI mode signal number CALL program name, priority	6.7
ONI	ONI	Sets interruption condition	P	ONI mode signal number GOTO label, priority	6.7
OPEINFO	OPEINFO	Displays operation information	M	OPEINFO robot number: joint number	5.6
OPEINFO	OPEINFO	Acquires operating data	F	OPEINFO (data number, robot number, joint number)	9.1
OPEINFOCLR	OPEINFOCLR	Resets operation information	M	OPEINFOCLR	5.6
OPEN	OPEN	Opens clamps	P	OPEN clamp number	6.3
OPENI	OPENI	Opens clamps	P	OPENI clamp number	6.3
OPENS	OPENS	Turns ON/OFF open clamp signal	P	OPENS clamp number	6.3
OPLOG	OP	Displays history of operations	M	OPLOG	5.6
OR	OR	Logical OR	O	... OR ...	8.3
OUTDA	OUTDA	Outputs voltage at set condition.	M	OUTDA voltage, channel number	5.8
OUTDA	OUTDA	Outputs voltage at set condition.	P	OUTDA voltage, channel number	6.8
OUTPUT	O	Software dedicated output signals	K	DEFSIG OUTPUT	5.6
OX.PREOUT	OX	Sets the timing of OUTPUT signal generation	S	OX.PREOUT	7
OXZERO	OXZERO	Switches the collective reset function for external output signal (OX).	S	OXZERO	7
P	P	Displays program steps	E	P number of steps	5.1
PAUSE	PA	Stops execution temporarily	P	PAUSE	6.5
PCABORT	PCA	Stops execution of PC program	M	PCABORT PC program number:	10
PCABORT	PCA	Stops execution of PC program	P	PCABORT PC program number:	10

Name	Abbreviation	Function		Format (Parameter)	Reference
PCCONTINUE	PCC	Resumes execution of PC program	M	PCCONTINUE PC program number: NEXT	10
PCEND	PCEN	Stop execution of PC program	M	PCEND PC program number: task number	10
PCEND	PCEN	Stop execution of PC program	P	PCEND PC program number: task number	10
PCENDMSG_MASK	PCENDMSG_MASK	Hides message in PC program completion	S	PCENDMSG_MASK	7
PCEXECUTE	PCEX	Executes PC program	M	PCEXECUTE PC program number, program name, execution cycle, step number	10
PCEXECUTE	PCEX	Executes PC program	P	PCEXECUTE PC program number, program name, execution cycle, step number	10
PCKILL	PCK	Initializes PC program stack	M	PCKILL PC program number:	10
PCSCAN	PCSC	Sets cycle time for PC program execution	P	PCSCAN time	10
PCSTATUS	PCSTA	Displays status of PC program	M	PCSTATUS PC program number:	10
PCSTEP	PCSTE	Executes single step of a PC program	M	PCSTEP PC program number: program name, execution cycles, step number	10
PI	PI	Returns the constant pi	F	PI	9.3
PNL_CYCST	PNL_CYCST	Turns ON/OFF cycle start switch	S	switch (PNL_CYCST)	7
PNL_ERESET	PNL_ERESET	Turns ON/OFF error reset switch	S	switch (PNL_ERESET)	7
PNL_MPOWER	PNL_MPOWER	Turns ON/OFF motor power	S	switch (PNL_MPOWER)	7
POINT	PO	Defines pose variable	M	POINT pose variable1= pose variable2, joint displacement value variable	5.5
POINT	PO	Defines pose variable	P	POINT pose variable = pose variable2, joint displacement value variable	6.9
POINT/7	PO/7	Assigns the value of joint 7	M	POINT/7 transformation value variable1 = transformation value variable2	5.5
POINT/7	PO/7	Assigns the value of joint 7	P	POINT/7 transformation value variable1 = transformation value variable2	6.9
POINT/18	PO/18	Assigns the value of joint 18	M	POINT/18 transformation value variable1 = transformation value variable2	5.5
POINT/18	PO/18	Assigns the value of joint 18	P	POINT/18 transformation value variable1 = transformation value variable2	6.9
POINT/A	PO/A	Assigns A component value	M	POINT/A transformation value variable1 = transformation value variable2	5.5
POINT/A	PO/A	Assigns A component value	P	POINT/A transformation value variable1 = transformation value variable2	6.9
POINT/EXT	PO/EXT	Assigns external axes component value	M	POINT/EXT transformation value variable1 = transformation value variable2	5.5
POINT/EXT	PO/EXT	Assigns external axes component value	P	POINT/EXT transformation value variable1 = transformation value variable2	6.9
POINT/O	PO/O	Assigns O component value	M	POINT/O transformation value variable1 = transformation value variable2	5.5
POINT/O	PO/O	Assigns O component value	P	POINT/O transformation value variable1 = transformation value variable2	6.9
POINT/OAT	PO/OAT	Assigns O-, A- and T component values	M	POINT/OAT transformation value variable1 = transformation value variable2	5.5

Name	Abbreviation	Function		Format (Parameter)	Reference
POINT/OAT	PO/OAT	Assigns O-, A- and T component values	P	POINT/OAT transformation value variable1 = transformation value variable2	6.9
POINT/T	PO/T	Assigns T component value	M	POINT/T transformation value variable1 = transformation value variable2	5.5
POINT/T	PO/T	Assigns T component value	P	POINT/T transformation value variable1 = transformation value variable2	6.9
POINT/X	PO/X	Assigns X component value	M	POINT/X transformation value variable1 = transformation value variable2	5.5
POINT/X	PO/X	Assigns X component value	P	POINT/X transformation value variable1 = transformation value variable2	6.9
POINT/Y	PO/Y	Assigns Y component value	M	POINT/Y transformation value variable1 = transformation value variable2	5.5
POINT/Y	PO/Y	Assigns Y component value	P	POINT/Y transformation value variable1 = transformation value variable2	6.9
POINT/Z	PO/Z	Assigns Z component value	M	POINT/Z transformation value variable1 = transformation value variable2	5.5
POINT/Z	PO/Z	Assigns Z component value	P	POINT/Z transformation value variable1 = transformation value variable2	6.9
POWER	POWER	MOTOR POWER switch ON/OFF status	S	switch(POWER)	7
#PPOINT	#PPOINT	Returns joint displacement values	F	#PPOINT (jt1, jt2, jt3, jt4, jt5, jt6)	9.2
PREFETCH.SIGINS	PR	Enables or disables early processing of I/O signals	S	PREFETCH.SIGNS	7
PRIME	PRIM	Sets up for program execution	M	PRIME program name, execution cycles, step number	5.4
PRINT	PRIN	Displays data on the terminal	M	PRINT device number: print data,……	5.8
PRINT	PRIN	Displays data on the terminal	P	PRINT device number: print data	6.8
PRIORITY	PRIORITY	Returns priority number	F	PRIORITY	9.1
PROG.DATE	PROG.DATE	Add date to program information	S	PROG.DATE	7
PROMPT	PROM	Displays message on terminal and waits for input	P	PROMPT device number: character string, variable order	6.8
PULSE	PU	Turns ON signal for a given period of time	M	PULSE signal number, time	5.7
PULSE	PU	Turns ON signal for a given period of time	P	PULSE signal number, time	6.7
QTOOL	Q	Tool transformation during block teaching	S	QTOOL	7
R	R	Replaces characters	E	R character string	5.1
RANDOM	RANDOM	Returns random number from 0 to 1	F	RANDOM	9.3
REC_ACCEPT	REC	Enables/disables RECORD/PROGRMA CHANGE function	M	REC_ACCEPT	5.6
REFFLTRESET	REFFLTRESET	Resets moving average span.	P	REFFLTRESET	6.2
REFFLTSET	REFFLTSET	Sets moving average span of command values.	P	REFFLTSET joint value moving average span, position moving average span, orientation moving average span, signal moving average span	6.2
REFFLTSET_STATUS	REFFLTSET_STATUS	Displays the command value moving average span setting.	M	REFFLTSET_STATUS	5.6

Name	Abbreviation	Function		Format (Parameter)	Reference
RELAX	RELAX	Turns OFF clamp signals (close and open)	P	RELAX clamp number	6.3
RELAXI	RELAXI	Turns OFF clamp signals (close and open)	P	RELAXI clamp number	6.3
RELAXS	RELAXS	Turns ON/OFF clamp signal (close and open)	P	RELAXS clamp number	6.3
RENAME	REN	Changes program name	M	RENAME new program name = existing program name	5.2
REP_ONCE	REP	Sets if repeat cycle is done once or continuously	S	REP_ONCE	7
REP_ONCE.RPS_LAST	REP_ONCE.RPS_LAST	Selects the terminating step in repeat once operation.	S	REP_ONCE.PRS_LAST	7
REPEAT	REPEAT	TEACH/REPEAT switch ON/OFF status	S	switch (REPEAT)	7
RESET	RES	Turns OFF all external output signals	M	RESET	5.7
RESET	RES	Turns OFF all external output signals	P	RESET	6.7
RESTRACE	RESTRACE	Releases memory set aside by SETTRACE	M	RESTRACE	5.2
RETURN	RET	Returns to the caller program	P	RETURN	6.5
RETURNE	RETURNE	Returns to step after the error	P	RETURNE	6.5
RGSO	RGSO	Servoing switch ON/OFF status	S	switch(RGSO)	7
$RIGHT	$RIGHT	Returns the rightmost characters	F	$RIGHT(string, number of characters)	9.4
RIGHTY	RI	Changes to right-hand configuration	P	RIGHTY	6.4
ROUND	ROUND	Round the real value at the first decimal place	F	ROUND (numeric value)	9.1
RPS	RP	Calls program selected by signals	S	RPS	7
RSIGCORRECT	RSIGC	Sets signal output timing of RSIGPOINT instruction	P	RSIGCORRECT	6.7
RSIGPOINT	RSIGP	Sets signal output between motion steps	P	RSIGPOINT	6.7
RSIGRANGE	RSIGR	Sets signal range to be used by RSIGPOINT instruction	M	RSIGRANGE	5.7
RUN	RUN	HOLD/RUN switch status	S	switch (RUN)	7
RUNMASK	RU	Masks signals	P	RUNMASK starting signal number, number of signals	6.7
RX	RX	Rotation about X Axis	F	RX (angle)	9.2
RY	RY	Rotation about Y Axis	F	RY (angle)	9.2
RZ	RZ	Rotation about Z Axis	F	RZ (angle)	9.2
S	S	Selects program step	E	S step number	5.1
SAVE	SA	Stores program/variable into a PC	M	SAVE/SEL filename =program name,...	5.3
SAVE/ALLLOG	SA/ALLLOG	Stores all log	M	SAVE/ALLLOG file name	5.3
SAVE/ELOG	SA/ELOG	Stores error log into PC	M	SAVE/ELOG file name	5.3
SAVE/FULL	SA/FULL	Stores all savable data	M	SAVE/ FULL file name	5.3
SAVE/OPLOG	SA/OPLOG	Stores operation log	M	SAVE OPLOG file name	5.3

Name	Abbreviation	Function		Format (Parameter)	Reference
SAVE/P,L,R,S,A	SA/P,...	Stores data into PC	M	SAVE(/P)(/L)(/R)(/S)/(A)/SEL file name=program name,...	5.3
SAVE/ROB	SA/ROB	Stores robot data into PC	M	SAVE/ROB file name	5.3
SAVE/STG	SA/STG	Stores logging data of data storage function into PC	M	SAVE/STG file name	5.3
SAVE/SYS	SA/SYS	Stores system data into PC	M	SAVE/SYS file name	5.3
SCALL	SCA	Jumps to subroutine	P	SCALL string expression, variable	6.5
SCASE	SCASE	SCASE structure	P	SCASE index variable OF ... SVALUE ... ANY... END	6.6
SCNT	SCN	Outputs counter signal when counter value is reached	M	SCNT counter signal number = count up signal, count down signal, counter clear signal, counter value	5.7
SCNT	SCNT	Outputs counter signal when counter value is reached	P	SCNT counter signal number = count up signal, count down signal, counter clear signal, counter value	6.7
SCNTRESET	SCNTR	Resets internal counter value	M	SCNTRESET counter signal number	5.7
SCNTRESET	SCNTR	Resets internal counter value	P	SCNTRESET counter signal number	6.7
SCREEN	SC	Control terminal display	S	SCREEN	7
SET_TOOLSHAPE	SET_TOOLSHAPE	Registration of tool shape	M	SET_TOOLSHAPE tool shape no.=transformation value variable 1, transformation value variable2, …, transformation value variable 8	5.6
SET_TOOLSHAPE	SET_TOOLSHAPE	Registration of tool shape	P	SET_TOOLSHAPE tool shape no.=transformation value variable 1, transformation value variable2, …, transformation value variable8	6.9
SET2HOME	SET2	Defines home pose 2	M	SET2HOME accuracy, HERE	5.6
SET2HOME	SET2	Defines home pose 2	P	SET2HOME accuracy, joint displacement value variable	6.9
SETENCTEMP_THRES	SETENCTEMP_THRES	Sets encoder temperature warning and temperature error thresholds	M	SETENCTEMP_THRES/N joint number, warning threshold, error threshold	5.6
SETENCTEMP_THRES	SETENCTEMP_THRES	Sets encoder temperature warning and temperature error thresholds	P	SETENCTEMP_THRES joint number, warning threshold, error threshold	6.9
SETHOME	SETH	Defines home pose 1	M	SETHOME accuracy, HERE	5.6
SETHOME	SETH	Defines home pose 1	P	SETHOME accuracy, joint displacement value variable	6.9
SET_MAXTOOLSHAPENUM	SET_MAXTOOLSHAPENUM	Sets the maximum tool shape number	M	SET_MAXTOOLSHAPENUM maximum tool shape number	5.6
SETOUTDA	SETOUTDA	Sets analog output environment.	M	SETOUTDA channel No. = LSB, No. of bits, logic, max. voltage, min. voltage	5.8
SETOUTDA	SETOUTDA	Sets analog output environment.	P	SETOUTDA channel No. = LSB, No. of bits, logic, max. voltage, min. voltage	6.8
SETPICK	SETPICK	Sets time to start clamp close control	M	SETPICK time1,…, time8	5.7
SETPICK	SETPICK	Sets time to start clamp close control	P	SETPICK time1,…, time8	6.7
SETPLACE	SETPLACE	Sets time to start clamp open control	M	SETPLACE time1,…, time8	5.7
SETPLACE	SETPLACE	Sets time to start clamp open control	P	SETPLACE time1,…, time8	6.7
SETTIME	SETTIME	Sets date an time	P	SETTIME year, month, day, hour, minute, second	6.9

Name	Abbreviation	Function		Format (Parameter)	Reference
SETTRACE	SETTRACE	Reserves necessary memory to log data	M	SETTRACE number of steps	5.2
SFLK	SFLK	Turns ON/OFF signal in given cycle time	M	SFLK flicker signal number = time	5.7
SFLK	SFLK	Turns ON/OFF signal in given cycle time	P	SFLK flicker signal number = time	6.7
SFLP	SFLP	Turns ON/OFF signal using set/reset signal	M	SFLP output signal = set signal expression, reset signal expression	5.7
SFLP	SFLP	Turns ON/OFF signal using set/reset signal	P	SFLP output signal = set signal expression, reset signal expression	6.7
SHIFT	SHIFT	Returns shifted transformation values	F	SHIFT (transformation value variable BY X shift, Y shift, Z shift)	9.2
SHUTDOWN	SHUTDOWN	Executes the data backup to CFast.	M	SHUTDOWN	5.2
SHUTDOWN	SHUTDOWN	Executes the data backup to CFast.	P	SHUTDOWN	6.10
SIG	SIG	Returns logical AND of signal states	F	SIG (signal number, …)	9.1
SIGMON_TEACH	SIGMON_TEACH	Enable/ disable signal monitor signal operation	S	SIGMON_TEACH	7
SIGNAL	SI	Turns signals ON/OFF	M	SIGNAL signal number, …	5.7
SIGNAL	SI	Turns signals ON/OFF	P	SIGNAL signal number, …	6.7
SIGRSTCONF	SIGRSTCONF	Switches number of signal to reset when signal 0 is output	S	SIGRSTCONF	7
SIN	SIN	Returns the sine value	F	SIN (real values)	9.3
SINGULAR	SINGULAR	Enable/ disable singular point check	S	SINGULAR	7
SJUMP	SJUMP	Switches execution program to program specified in character string	P	SJUMP program name, status variable	6.5
SLOAD	SLOAD	Loads files to robot memory	P	SLOAD/IF/ARC device number= string, status variable	6.10
SLOW_REPEAT	SL	Sets the repeat speed in slow repeat mode	M	SLOW_REPEAT	5.6
SLOW_START	SLOW_START	Enables or disables the slow start function	S	SLOW_START	7
SOUT	SO	Outputs signal when condition is set	M	SOUT signal number = signal expression	5.7
SOUT	SO	Outputs signal when condition is set	P	SOUT signal number = signal expression	6.7
$SPACE	$SPACE	Returns blanks	F	$SPACE (number of blanks)	9.4
SPEED	SP	Sets monitor speed	M	SPEED monitor speed	5.4
SPEED	SP	Sets monitor speed	P	SPEED speed, rotational speed ALWAYS	6.2
SQRT	SQRT	Returns the square root	F	SQRT (real values)	9.3
STABLE	STA	Holds robot motion for a given time	P	STABLE time	6.1
STAT_ON_KYBD	STAT_ON_KYBD	Displays status information on keyboard screen	S	STAT_ON_KYBD	7
STATUS	STA	Displays system status	M	STATUS	5.6
STEP	STE	Executes a single step of a program	M	STEP program name, execution cycles, step number	5.4
STIM	STI	Turns ON timer signal	M	STIM timer signal = input signal number, time	5.7

Name	Abbreviation	Function		Format (Parameter)	Reference
STIM	STI	Turns ON timer signal	P	STIM timer signal = input signal number, time	6.7
STOP	STO	Terminates execution cycle	P	STOP	6.5
STPNEXT	STP	Executes the next step	M	STPNEXT	5.4
STP_ONCE	ST	Sets the execution as one step at a time or continuous	S	STP_ONCE	7
$STR_ID	$STR_ID	Returns robot information for robot 1	F	$STR_ID (number)	9.4
$STR_ID2	$STR_ID2	Returns robot information for robot 2	F	$STR_ID2 (number)	9.4
STRTOPOS	STRTOPOS	Returns the value of specified pose variable	F	STRTOPOS (string variable)	9.1
STRTOVAL	STRTOVAL	Returns the value of specified real variable	F	STRTOVAL (string variable)	9.1
SWAIT	SW	Waits for desired signal state	P	SWAIT signal number, ...	6.7
SWITCH	SW	Sets system switches	M	SWITCH switch name,... switch name =ON (=OFF)	5.6
SWITCH	SWITCH	Returns system switch status	F	SWITCH (switch name)	9.1
SYSDATA	SYSDATA	Returns parameters in AS system	F	SYSDATA (keyword, opt1, opt2)	9.1
$SYSDATA	$SYSDATA	Returns parameters in AS system	F	SYSDATA (keyword, opt1, opt2)	9.4
SYSINIT	SYS	Initialize system	M	SYSINIT	5.6
T	T	Enables teaching by TP in editor mode	E	T pose variable	5.1
TASK	TASK	Returns execution status of program	F	TASK (task number)	9.1
TDRAW	TD	Moves the robot by a given amount of the tool coordinates	P	TDRAW X translation, Y translation, Z translation, X rotation, Y rotation, Z rotation, speed	6.1
TEACH_LOCK	TEACH_LOCK	TEACHLOCK switch ON/OFF status	S	switch (TEACH_LOCK)	7
THEN	THEN	IF structure	K	IF logical expression THEN	6.6
TIME	TI	Sets and displays date and time	M	TIME year-month-day hour:minute:second	5.6
$TIME	$TIME	Returns system date and time	F	$TIME	9.4
$TIME_MS	$TIME_MS	Returns system time string including milliseconds	F	$TIME_MS	9.4
TIMER	TI	Sets timer	P	TIMER timer number = time	6.9
TIMER	TIMER	Returns timer values	F	TIMER (timer number)	9.1
TO	TO	FOR structure	K	FOR ... TO ... END	6.6
TOOL	TOOL	Defines tool transformation values	M	TOOL transformation value variable, tool shape number	5.6
TOOL	TOOL	Defines tool transformation values	P	TOOL transformation value variable, tool shape number	6.9
TOOL	TOOL	Returns tool transformation values	F	TOOL	9.2
TOOLSHAPE	TOOLSHAPE	Sets speed control using tool shape	M	TOOLSHAPE tool shape no.	5.6
TOOLSHAPE	TOOLSHAPE	Sets speed control using tool shape	P	TOOLSHAPE tool shape no.	6.9
TOUCH.ENA	TOUCH.ENA	Touch panel operation of TP repeat condition enable/ disable	S	TOUCH.ENA	7

Name	Abbreviation	Function		Format (Parameter)	Reference
TOUCHST.ENA	TOUCHST.ENA	Touch panel operation of TP status lamp enable/ disable	S	TOUCHST.ENA	7
TPKEY_A	TPKEY_A	Turns ON/OFF A key on TP	S	switch (TPKEY_A)	7
TPLIGHT	TPLIGHT	Turns on the TP backlight	M	TPLIGHT	5.6
TPLIGHT	TPLIGHT	Turns on the TP backlight	P	TPLIGHT	6.9
TPSPEED.RESET	TPSPEED.RESET	Automatic slow speed setting for teach and check speed	S	TPSPEED.RESET	7
TRACE	TRACE	Logs and traces programs	M	TRACE stepper number: ON/OFF	5.2
TRACE	TRACE	Logs and traces programs	P	TRACE stepper number: ON/OFF	6.10
TRADD	TRADD	Returns the sum of traverse axis and transformation values	F	TRADD (transformation value variable)	9.2
TRANS	TRANS	Returns transformation values	F	TRANS (X component, Y component, Z component, O component, A component, T component)	9.2
TRIGGER	TRIGGER	TRIGGER switch ON/OFF status	S	switch (TRIGGER)	7
TRQNM	TRQNM	Acquires the torque value of specified axis	F	TRQNM(axis number)	9.1
TRSUB	TRSUB	Returns the difference of traverse axis and transformation values	F	TRSUB (transformation value variable)	9.2
TWAIT	TW	Wait for a given period of time	P	TWAIT time	6.5
TYPE	TY	Displays data on the terminal	M	TYPE device number: print data,…	5.8
TYPE	TY	Displays data on the terminal	P	TYPE device number: print data	6.8
ULIMIT	UL	Sets lower limit of robot motion	M	ULIMIT joint displacement value variable	5.6
ULIMIT	UL	Sets lower limit of robot motion	P	ULIMIT joint displacement value variable	6.9
UNTIL	UN	DO structure	P	DO ... UNTIL logical expression	6.6
USB_COPY	USB_COPY	Copies files in USB drive	M	USB_COPY new file name = source file name	5.2
USB_FDEL	USB_FDEL	Deletes data in USB drive	M	USB_FDEL file name,,…	5.2
USB_FDIR	USB_FDIR	Displays names of program/variable in USB memory	M	USB_FDIR folder name	5.2
USB_LOAD	USB_LO	Loads contents of USB drive into robot memory	M	USB_LOAD/Q filename	5.3
USB_MKDIR	USB_MKDIR	Creates a folder on the USB drive	M	USB_MKDIR folder name	5.3
USB_RENAME	USB_RENAME	Changes file name in USB drive	M	USB_RENAME new file name = existing file name	5.2
USB_SAVE	USB_SA	Stores program/variable into USB drive	M	USB_SAVE/SEL filename =program name,…	5.3
USB_SAVE/ALLLOG	USB_SA/ ALLLOG	Stores all logs into USB drive	M	USB_SAVE/ALLLOG file name	5.3
USB_SAVE/ELOG	USB_SA/ELOG	Stores error log into USB dirve	M	USB_SAVE/ELOG file name	5.3
USB_SAVE/FULL	USB_SA/FULL	Stores all data into USB drive	M	USB_SAVE/FULL file name	5.3
USB_SAVE/OPLOG	USB_SA/OPLOG	Stores operation log into USB drive	M	USB_SAVE/OPLOG file name	5.3
USB_SAVE/P,L,R,S,A	USB_SA/P, ...	Stores data into floppy disk	M	USB_SAVE(/P)(/L)(/R)(/S)/(A)/SEL file name=program name,..	5.3
USB_SAVE/ROB	USB_SA/ROB	Stores robot data into USB drive	M	USB_SAVE/ROB file name	5.3

Name	Abbreviation	Function		Format (Parameter)	Reference
USB_SAVE/STG	USB_SA/STG	Stores data storage data into USB drive	M	USB_SAVE/STG file name	5.3
USB_SAVE/SYS	USB_SA/SYS	Stores system data into USB drive	M	USB_SAVE/SYS file name	5.3
USE_ISO8859_5	USE_ISO8859_5	Changes ASCII8 bit display font to Cyrillic font	S	USE_ISO8859_5	7
UTIMER	UTIMER	Sets user timer	P	UTIMER @timer variable = timer value	6.9
UTIMER	UTIMER	Returns current user timer values	F	UTIMER (@timer variable)	9.1
UWRIST	UW	Changes wrist configuration	P	UWRIST	6.4
VAL	VAL	Returns real value	F	VAL (string, code)	9.1
WAIT	WA	Waits for specified condition	P	WAIT condition	6.5
WAITREL_AUTO	WAITREL_AUTO	Displays wait release popup window	S	WAITREL_AUTO	7
WEIGHT	WE	Sets load mass data	M	WEIGHT load mass, center of gravity X, center of gravity Y, center of gravity Z, inertia moment ab. X axis, inertia moment ab. Y axis, inertia moment ab. Z axis	5.6
WEIGHT	WE	Sets load mass data	P	WEIGHT load mass, center of gravity X, center of gravity Y, center of gravity Z, inertia moment ab. X axis, inertia moment ab. Y axis, inertia moment ab. Z axis	6.9
WHERE	W	Displays current robot pose	M	WHERE display mode	5.6
WHICHTASK	WHICHTASK	Returns task number of specified program	F	WHICHTASK program name	9.1
WHILE	WH	DO structure	P	WHILE ... DO ... END	6.6
WS.ZERO	WS.ZERO	Changes the weld processing	S	WS.ZERO	7
WS_COMPOFF	WS_COMPOFF	Changes the output timing of WS signal	S	WS_COMPOFF	7
XD	XD	Cuts step and pastes on the buffer	E	XD number of steps	5.1
XFER	XF	Copies and transfers steps	M	XFER destination program name, step number 1= source program name, step number 2, number of steps	5.2
XMOVE	X	Moves until the signal changes	P	XMOVE mode pose variable TILL signal number	6.1
XOR	XOR	Exclusive logical OR	O	...XOR...	8.3
XP	XP	Inserts contents of paste buffer	E	XP	5.1
XQ	XQ	Inserts contents of paste buffer in reverse order	E	XQ	5.1
XS	XS	Displays contents of paste buffer	E	XS	5.1
XY	XY	Copies step and pastes on the buffer	E	XY number of steps	5.1
ZSIGMAP	ZSIGMAP	Signal allocation setting	M	ZSIGMAP type, AS signal start number, port signal start number, number of signals	5.6
ZSIGSPEC	ZSIG	Sets number of installed signals	M	ZSIGSPEC	5.6
ZZERO	ZZ	Sets zeroing data	M	ZZERO joint number	5.6
FALSE	FALSE	Returns FALSE value	F	... FALSE ...	9.1
TRUE	TRUE	Returns TRUE value	F	... TRUE...	9.1

Name	Abbreviation	Function		Format (Parameter)	Reference
-	-	Subtraction	O	.... -...	8.1
-	-	Subtraction of transformation values	O	.... - ...	8.5
*	*	Multiplication	O	.....*.....	8.1
/	/	Division	O	.... / …	8.1
ˆ	^	Power	O	.... ˆ.....	8.1
+	+	Addition	O	.... + ...	8.1
+	+	Addition of transformation values	O	.... + ...	8.5
+	+	Combination of strings	O	.... + …	8.6
<	<	Less than	O	.... < ...	8.2
<=	<=	Less than or equal to	O	.... <= ...	8.2
<>	<>	Not equal to	O	.... <> ...	8.2
=<	=<	Less than or equal to	O	.... =< ...	8.2
==	==	Equal to	O	.....==.....	8.2
=>	=>	Greater than or equal to	O	.... => ...	8.2
>	>	Greater than	O	.... > ...	8.2
>=	>=	Greater than or equal to	O	.... >= ...	8.2

Kawasaki Robot Controller F Series
AS Language Reference Manual

2017-01 : 1st Edition
2019-06 : 2nd Edition

Publication : Kawasaki Heavy Industries, Ltd.
90209-1025DEB