

Simple  friendly



**Kawasaki Robot Controller
D Series**

**AS Language
Reference Manual**

Robot

Kawasaki Heavy Industries, Ltd.

Kawasaki Robot Controller D Series

AS Language Reference Manual

Kind of Manual

☐ Standard Manual ☐ Option Manual ☐ Manual for Specific User ☐ Detailed Manual

24 December, 2002					Optical File
ROBOT DIVISION					
Recognition	Recognition	Examination	Checked	Produce	
				Tanaka	

Delivery to :

☐ Manuf. Section ☐ KMS ☐ KRI ☐ KRUK ☐ KRG ☐ KMSK ☐ Others()

Number of sheets

417 Pages

Robot

Kawasaki Heavy Industries, Ltd.

PREFACE

This manual describes the AS* language used in the Kawasaki Robot Controller D 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 Machine Systems. Refer to the contact information listed on the rear cover of this manual for the nearest Kawasaki Machine Systems office.

NOTE* AS is pronounced [az]

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.

1. 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.
2. Kawasaki reserves the rights to change, revise, or update this manual without prior notice.
3. This manual may not, in whole or in part, be reprinted or copied without the prior written consent of Kawasaki.
4. Store this manual with care and keep it available for use at any time. In the event the manual is lost or damaged severely, contact your Kawasaki agent.
5. Though this manual was prepared to be as thorough and accurate as possible, the authors apologize should any information be found incomplete or erroneous.

All rights reserved. Copyright © 2002 by Kawasaki Heavy Industries Ltd.

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.

CONTENTS

Preface	i
1.0 Overall of AS	1-1
1.1 System Overview	1-2
1.2 Characteristics of the AS System	1-3
1.3 AS System Configuration	1-4
2.0 AS System	2-1
2.1 AS System Status	2-2
2.2 AS System Switches	2-3
2.3 AS System Setup	2-5
2.4 Input and Output Operations	2-6
2.5 Installing Terminal Software	2-8
2.6 Operations from Personal Computer	2-9
3.0 Information Expressions in AS Language	3-1
3.1 Notation and Conventions	3-2
3.2 Pose Information, Numeric Information, and Character Information	3-4
3.3 Variables	3-10
3.4 Variable Names	3-12
3.5 Defining Pose Variables	3-13
3.6 Defining Real Variables	3-19
3.7 Defining String Variables	3-20
3.8 Numeric Expressions	3-21
3.9 String Expressions	3-24
4.0 AS Program	4-1
4.1 Types of AS Programs	4-2
4.2 Creating and Editing Programs	4-4
4.3 Executing Programs	4-9
4.4 Program Flow	4-12
4.5 Robot Motion	4-14
5.0 Monitor Commands	5-1
5.1 Editor Commands	5-2
5.2 Program and Data Control Commands	5-15

5.3	Program and Data Storage Commands	5-27
5.4	Program Control Commands	5-36
5.5	Pose Information Commands	5-45
5.6	System Control Commands	5-50
5.7	Binary Signal Commands	5-86
5.8	Message Display Commands	5-98
6.0	Program Instructions	6-1
6.1	Motion Instructions	6-2
6.2	Speed and Accuracy Control Instructions	6-15
6.3	Clamp Control Instructions	6-23
6.4	Configuration Instructions	6-30
6.5	Program Control Instructions	6-33
6.6	Program Structure Instructions	6-46
6.7	Binary Signal Instructions	6-56
6.8	Message Control Instructions	6-74
6.9	Pose Information Instructions	6-81
6.10	Program and Data Control Instructions	6-92
7.0	AS System Switches	7-1
8.0	Operators	8-1
8.1	Arithmetic Operators	8-2
8.2	Relational Operators	8-3
8.3	Logical Operators	8-4
8.4	Binary Operators	8-6
8.5	Transformation Value Operators	8-7
8.6	String Operators	8-9
9.0	Functions	9-1
9.1	Real Value Functions	9-2
9.2	Pose Value Functions	9-20
9.3	Mathematical Functions	9-36
9.4	String Functions	9-39
10.0	Process Control Programs	10-1
11.0	Sample Programs	11-1
11.1	Initial Settings for Programs	11-2

11.2 Palletizing	11-3
11.3 External Interlocking.....	11-5
11.4 Tool Transformation.....	11-8
11.5 Relative Poses.....	11-11
11.6 Using Relative Poses with FRAME Functions.....	11-14
11.7 Setting Robot Configuration	11-16
Appendix 1 Error Codes.....	A-1
Appendix 2 AS Language List.....	A-19
Appendix 3 ASCII Code	A-33
Appendix 4 Signal Limitation.....	A-36
Appendix 5 Euler's O,A,T Angles	A-37

1.0 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 System Overview

1.2 Characteristics of the AS System

1.3 AS System Configuration

1.1 OVERVIEW OF THE AS SYSTEM

In the AS system, you can place commands or execute programs using AS language. The AS system is written in the nonvolatile memory in the robot control unit. When the control 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 or the robot pose (location), saving data in external memory devices, and writing/editing programs.

1.2 CHARACTERISTICS OF THE AS SYSTEM

In the AS system, the robots are controlled and operated based on a program that describes the necessary tasks and which is made prior to the 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.

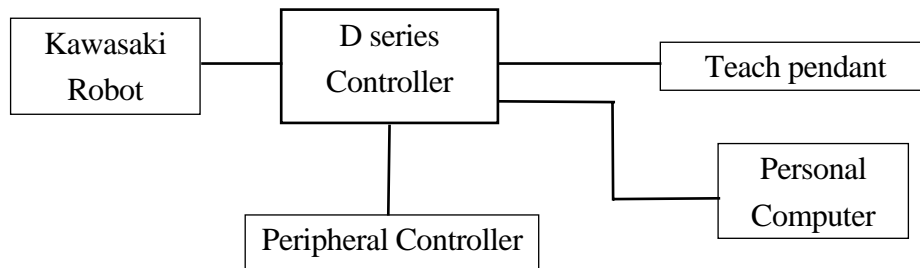
In this manual, a monitor command is referred to as a command, a program instruction as an instruction.

AS is unique in the following ways:

1. The robot can be moved along a continuous path trajectory. (CP motion: Continuous Path motion)
2. Two coordinate systems, base coordinates and tool coordinates, are provided, for more precise control of the robot's movements.
3. The coordinates can be shifted or rotated corresponding to the pose changes of the work.
4. When teaching locations or repeating executions, the robot can be moved along a linear path while keeping the tool posture.
5. Programs can be named freely and saved without limits in numbers.
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) can be executed simultaneously with a robot control program.
9. Programs and pose (location) data can be displayed on terminals and saved in devices such as the PC card.
10. Programming can be done using a personal computer loaded with the terminal software (KRterm or KCwin32) provided by Kawasaki. (Off-line programming)

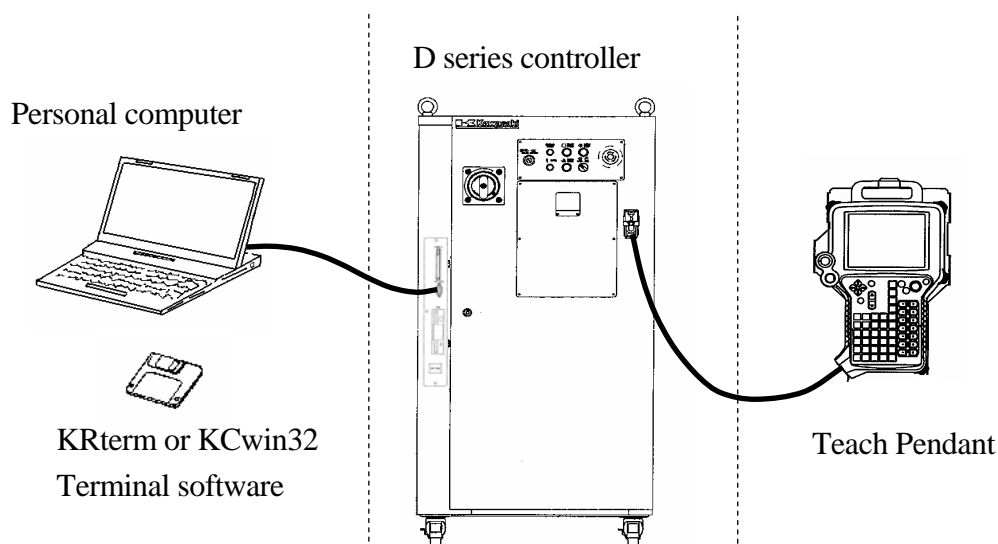
1.3 AS SYSTEM CONFIGURATION

Kawasaki Robot controller D series is composed of the following components:



By connecting a personal computer loaded with the terminal software KRterm or KCwin32 to a D series controller, the following operations can be done:

- Writing AS commands
- Saving and loading to and from the personal computer



Personal computer	Controller	Teach Pendant
<ul style="list-style-type: none"> • Enters AS commands • Creates AS programs • Saves/loads programs 	<p>Daily operations</p>	<ul style="list-style-type: none"> • Selects program • Displays program names and steps • Manually controls the robot • Monitors signals • Sets repeating conditions • Teaches pose (location) data • Teaches auxiliary data (block teaching)

[NOTE]

Monitor software for PC operates with 95/98/Me/2000/XP.
Please prepare the appropriate OS.

2.0 AS SYSTEM

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

2.1 AS System Status

2.2 AS System Switches

2.3 AS System Setup

2.4 Input and Output Operations

2.5 Installing Terminal Software

2.6 Operations from Personal Computer

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. Monitor commands are executed in this mode. Editor or Playback Mode is accessed from this mode.

2. Editor Mode

This mode enables you to create a new program or 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. Computations for robot motion control are performed and commands entered from the terminal are processed during unoccupied CPU time. Some monitor commands cannot be executed in this 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 respond to the EXECUTE, DO, STEP, MSTEP, and CONTINUE commands when the **HOLD/RUN** switch is in the HOLD position.

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 ON or to turn OFF **CYCLE START** 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, allowing for earlier signal output at the memory change instead of at the accuracy setting.

6. PREFETCH.SIGINS

Sets the timing for signal output in AS language programs, and has the same effect on signal timing as OX.PREOUT.

7. QTOOL

When in TEACH mode, determines whether or not to change the tool transformation according to the tool number taught in block instructions.

8. REP_ONCE (Repeat Once)

When this switch is ON, the program runs one time. When it is OFF, the program runs continuously.

9. RPS (Random Program Selection)

Enables or disables the selection of programs based on the binary status of 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 to correspond to a robot's known mechanical position. 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 if it is consistent 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 function 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.

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
EXT_IT	HOME1, HOME2
EXT. SLOW REPEAT MODE	POWER ON
	RGSO
	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 128, 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 below commands. These are called the terminal control commands. The **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, PC card, floppy disk, or computer hard disk.

1. Initializes memory. (CARD_FORMAT, FD_FORMAT)
2. Displays the contents. (CARD_FDIR, FD_FDIR)
3. Saves the data on the robot memory to the disk files. (SAVE*, CARD_SAVE, FD_SAVE)
4. Loads the data on the disk file to the robot memory. (LOAD, CARD_LOAD, FD_LOAD)
5. Deletes the disk files. (DELETE, CARD_FDEL, FD_FDEL)

Commands with CARD_ refer to PC cards and FD_ refers to floppy disks.

Note* SAVE 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

The robot can be controlled from a personal computer using the AS language. To do so, load KCwin32 or KRterm terminal software on to a PC and connect the PC to a D series controller. KCwin32 and KRterm may be installed on computers running Windows 95/98/Me/2000/XP.

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. Terminal software KCwin and KCwin32 can also be used by connecting to the RS-232C port.

Follow the below procedure to install the terminal software on to the PC.

1. Install the software to your hard disk

Copy the following files in the KRterm floppy disk on any directory of your computer hard disk.

KRTERM.EXE

KRTERMJ.HLP (Japanese version help file)

KRTERM.E.HLP (English version help file)

KRTERM.INI

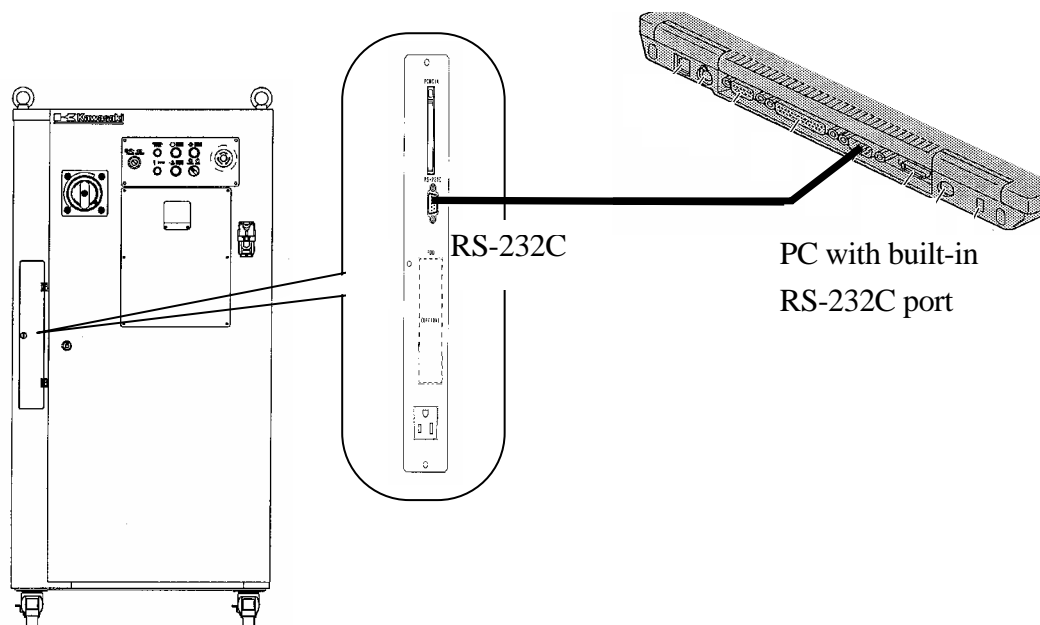
2. It is recommended that you make a shortcut on the desktop or in the start menu for easier startup of the KRterm software.

2.6 OPERATIONS FROM PERSONAL COMPUTER

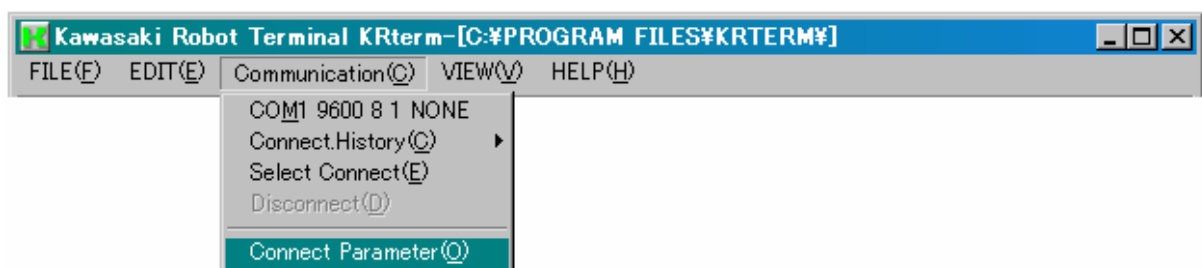
2.6.1 SYSTEM STARTUP

2.6.1.1 CONNECTING TO RS-232C PORT

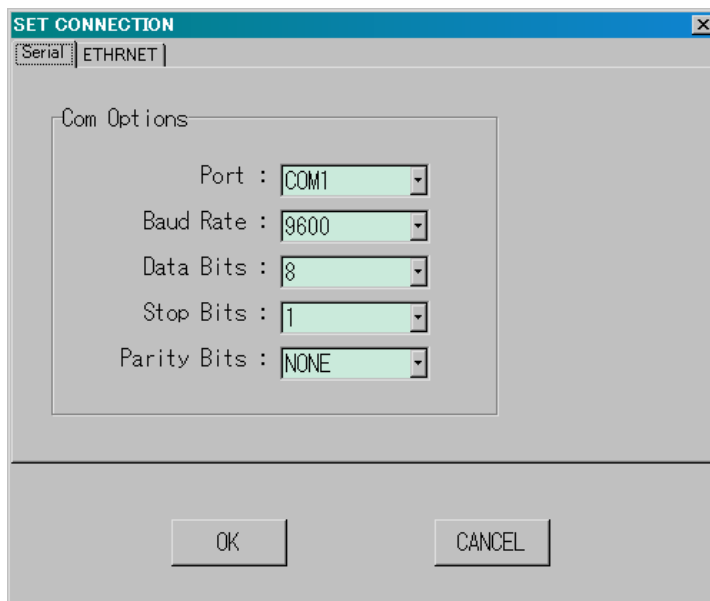
1. Connect the personal computer with the controller using the RS-232C cable. Make sure the CONTROL POWER on the controller and the computer power are both turn off.



2. Turn on the computer, and start the terminal software (KRterm or KCwin32, diagrams below are from KRterm).
3. When the software opens, select the type of connection to use. Select from the menu bar, [Communication (C)] → [Connect Parameter (O)].



4. Click the "Serial" tab, check the contents and if it is OK, click <OK>.



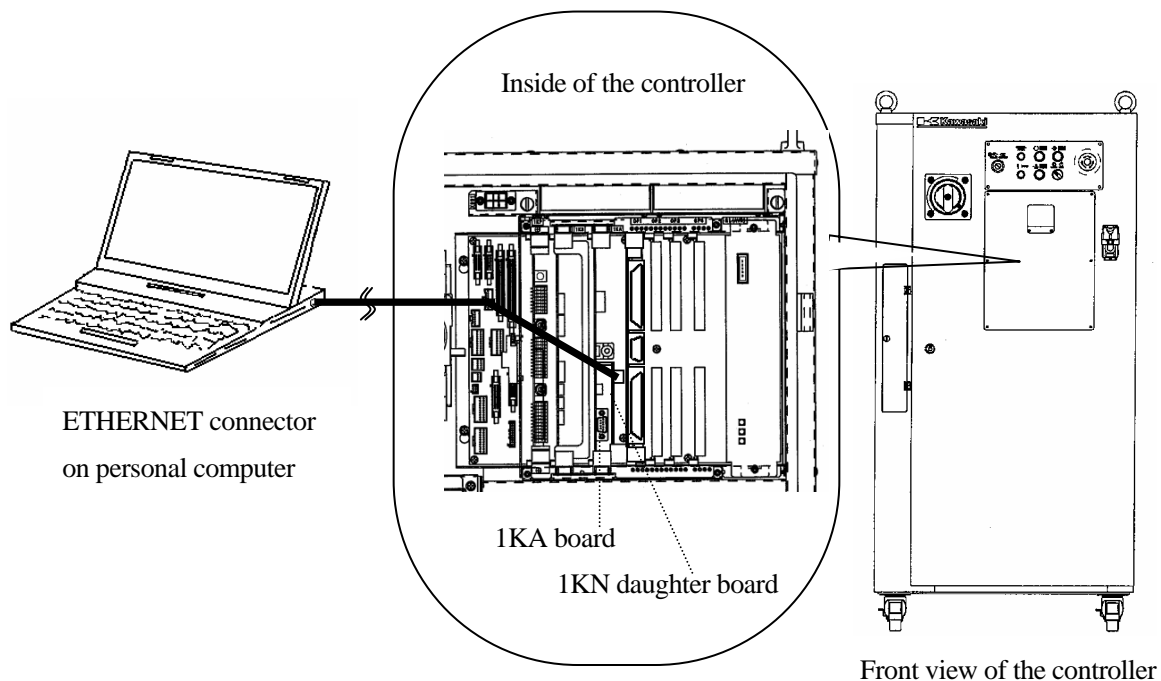
5. Turn ON the **CONTROL POWER** on the controller.
(See “Operation Manual” 3.1 Power ON Procedure).
6. The initial screen of KRterm followed by a prompt “>” will appear on the display.

When the CONTROL POWER is turned ON before connecting the PC and 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

1. Connecting the cables.

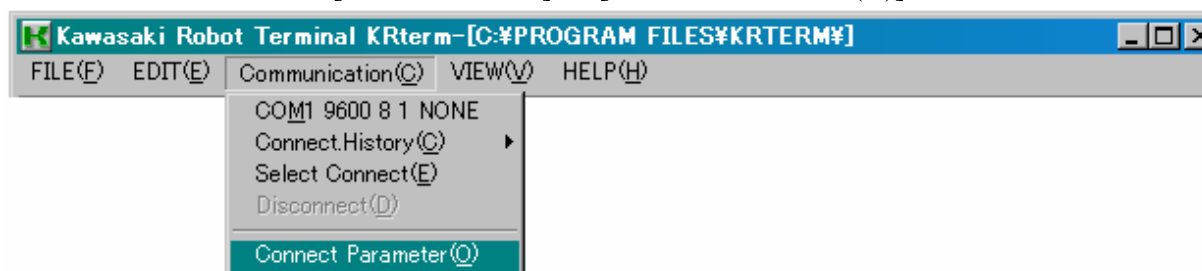
Connect the ETHERNET connector on your personal computer and the connector on optional 1 KN daughter board on 1 KA board in the controller using a LAN cable.



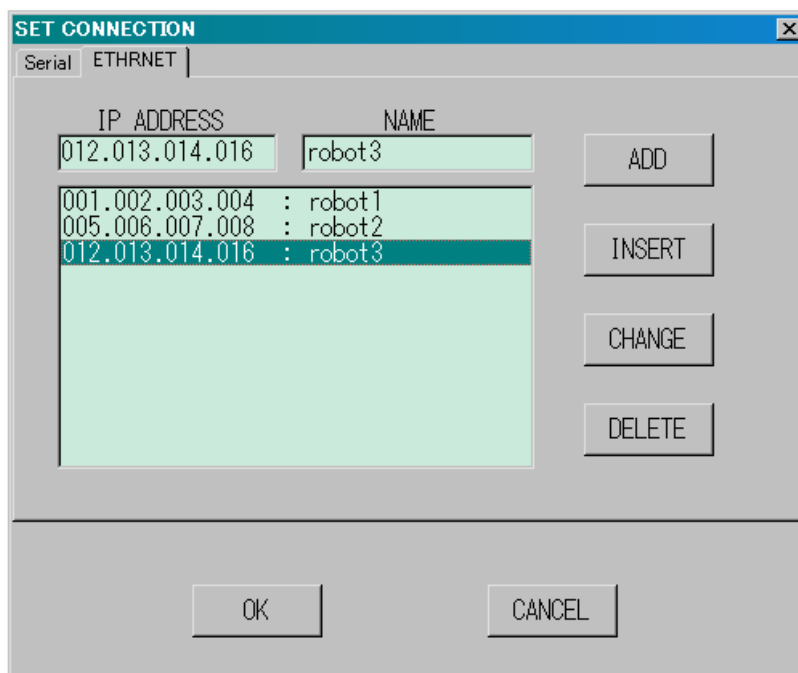
2. Following steps 2 and 3 above, start KRterm and the robot.

3. Set the robot's IP address.

(1) Choose from the menu bar [Communication] → [Connect Parameter (O)].

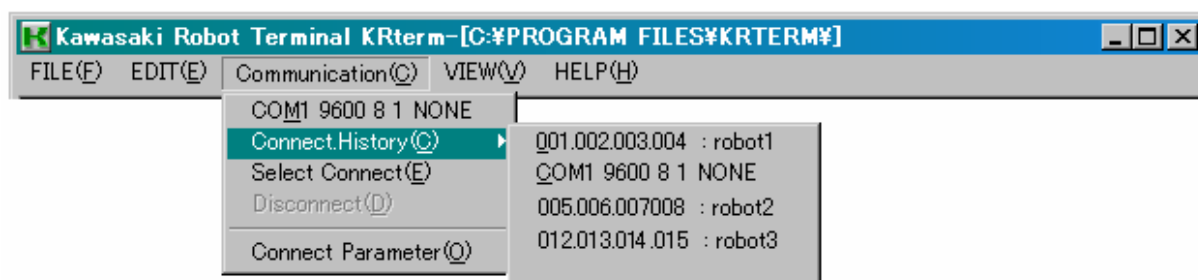


- (2) Enter the IP address and name of the robot you want to connect to the network, and click <ADD>.

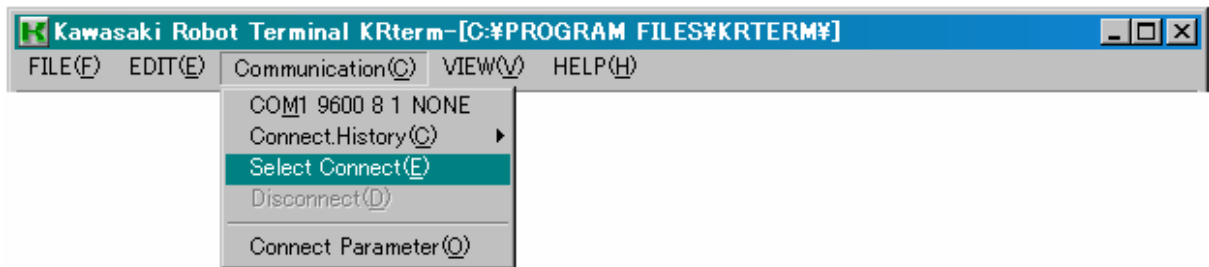


4. To connect with the robot, follow the procedures below.

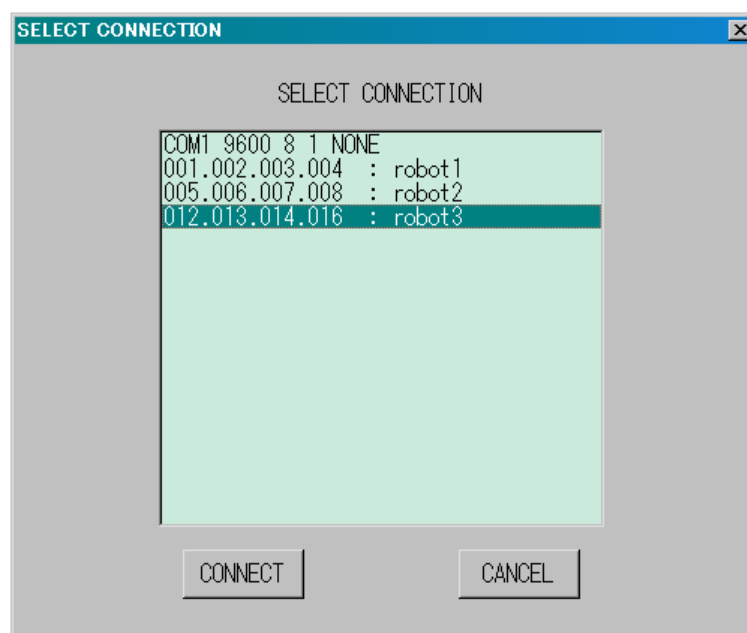
- (1) The robot recently connected is displayed at the top of the drop-down menu when [Communication(C)] is selected from the menu bar. Click the robot name to select the robot.
- (2) Choose from the menu bar [Communication(C)] → [Connect.History(C)] to display the list of recently used robots. Select the robot to connect from the list.



- (3) Choose from the menu bar [Communication(C)] → [Select Connect(E)] if the desired robot does not appear in the above ways..



Choose the desired robot and click <CONNECT>.



If the connection is established, robot information such as its name followed by a prompt “>” appears on the KRterm screen. 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 the KRterm in the computer hard disk.

 >SAVE test.pg¥My Documents This saves the data in the specified file.

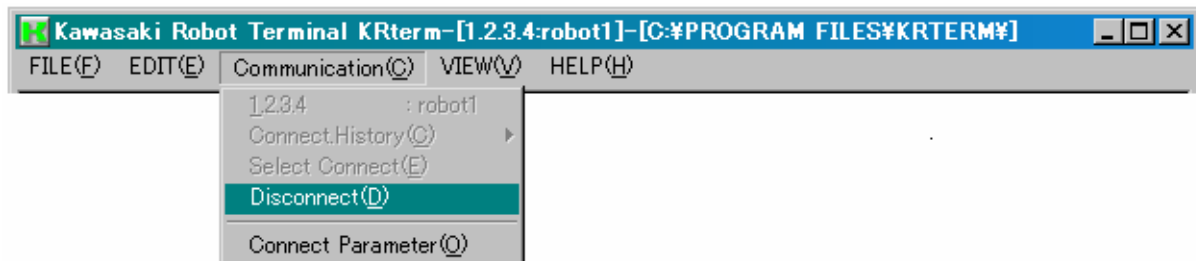
- (2) LOAD command

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 [Communication(C)] → [Disconnect(D)] to disconnect the robot.

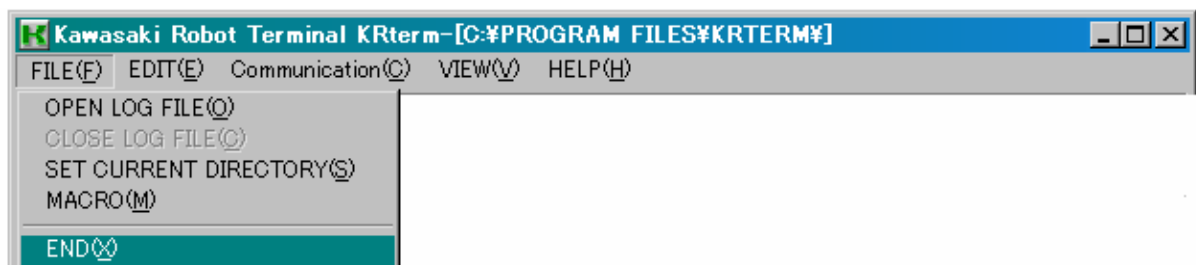


2. Turn off the robot controller. (See “Operation Manual” 3.2 POWER OFF procedure).

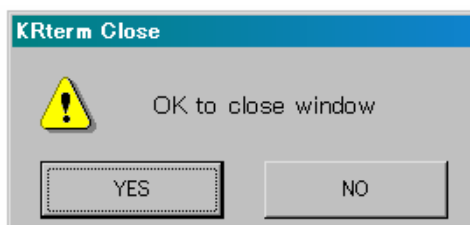
- (1) Turn the **HOLD/RUN** switch to HOLD.
- (2) Turn OFF the motor power by pressing the **EMERGENCY STOP** button.
- (3) Turn OFF the **CONTROL POWER**.

3. Shut down KRterm.

- (1) Choose from the menu bar [FILE(F)] → [END(X)].



- (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.

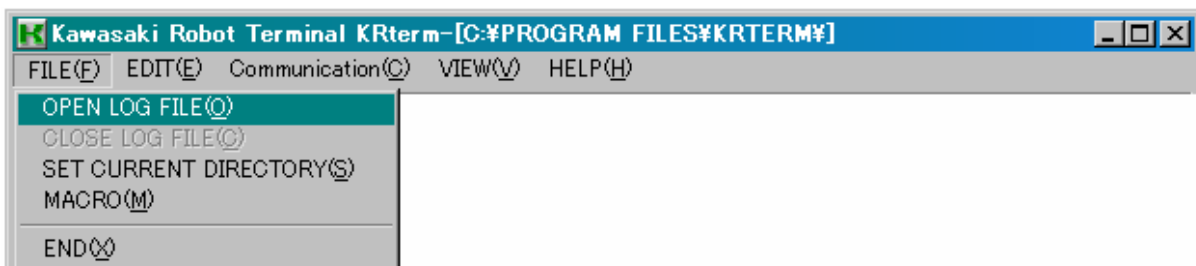
2.6.4 USEFUL FUNCTIONS OF KRTERM

2.6.4.1 CREATING LOGFILES

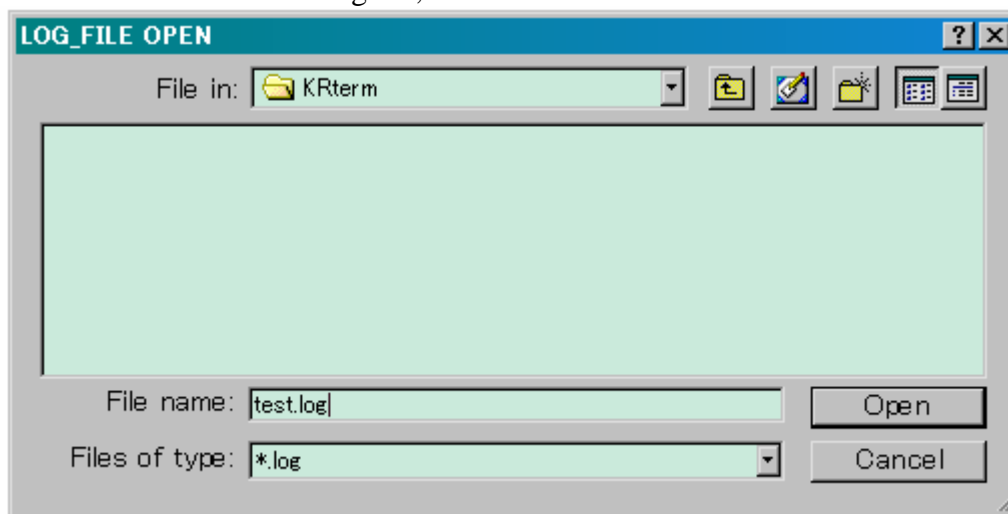
The contents displayed on the KRterm screen can be saved as a log file. This is useful when making printout of the robot operation procedures.

1. Start logging.

(1) Choose from the menu bar [FILE(F)] → [OPEN LOG FILE(O)].



(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.

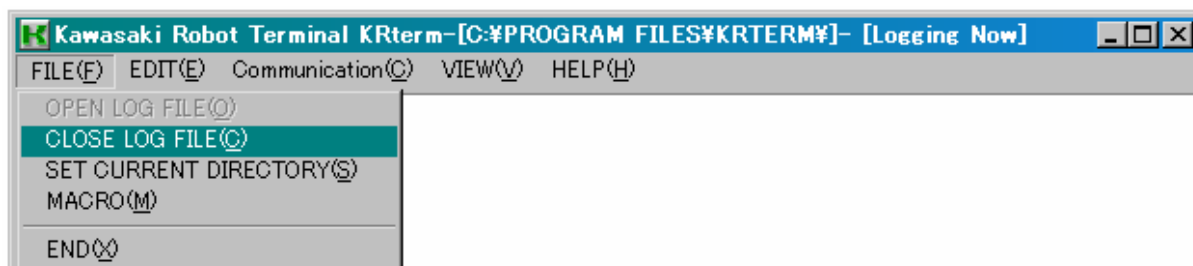


The contents on the display are recorded while this message is shown.

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(F)] → [CLOSE LOG FILE(C)].



2.6.4.2 MACRO FUNCTIONS

Macro functions are provided in the KRterm and KCwin32 system. 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 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 or KCwin32 for more details.



MEMO

3.0 INFORMATION EXPRESSIONS IN AS LANGUAGE

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

3.1 Notation and Conventions

3.2 Pose Information, Numeric Information and Character Information

3.3 Variables

3.4 Variable Names

3.5 Defining Pose Variables

3.6 Defining Real Variables

3.7 Defining String Variables

3.8 Numeric Expressions

3.9 String Expressions

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 RUN/HOLD, Backspace

3. Abbreviations

Keywords can be abbreviated. For example, EXECUTE command can be abbreviated as EX. See Appendix 2 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 necessary data to complete commands or other functions. For example, in the 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 arguments. 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

Defines and modifies the robot's posture at the specified pose (location), and describes the rotation amount of the robot joints. 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 define real values. The values for the variables can range from $-3.4\text{E}+38$ to $3.4\text{E}+38$ ($-3.4 \times 10^{38} \sim 3.4 \times 10^{38}$). 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 – 32	1 – 96
External input signal	1001 – 1032	1001 – 1096
Internal signal	2001 – 2256	2001 – 2256

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 posture of the robot in one word.

3.2.1 POSE INFORMATION

Pose information, also known as positional information, is used to specify the position and posture of the robot in the given work area. The robot’s position and posture refers to the position and posture of the tool center point (TCP) of the robot. The position and posture together is called the pose (location) 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 posture.

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. Using encoder values, rotational axes produce angular displacement described in degrees, and linear axes produce linear displacement described in millimeters.

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 value

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 on the base coordinates, the

posture by Euler's OAT angles*. Some of the commonly used transformation values are: the tool transformation value, describing the pose of the tool coordinates relative to the null tool coordinates, and work based transformation values, describing the pose of the tool coordinates relative to the work coordinates.

NOTE* See Appendix 5 Euler's O,A,T Angles.

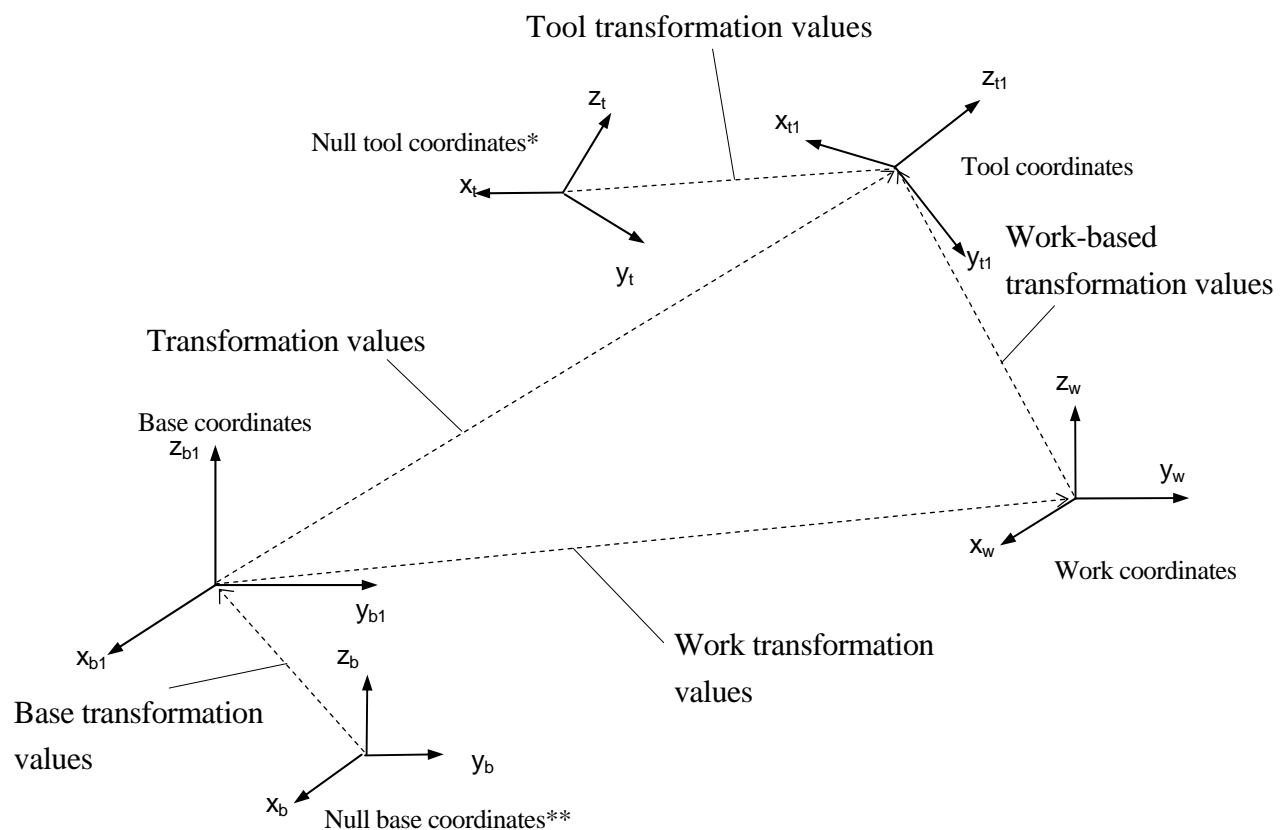
Example

	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	<ul style="list-style-type: none"> · Playback precision is achieved and there is no ambiguity about robot configuration at a pose 	<ul style="list-style-type: none"> · 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. work coordinates) · Convenient for processing as the data are shown in XYZOAT values
Disadvantage	<ul style="list-style-type: none"> · TCP changes when the tool is changed (null tool coordinates remain the same) · Cannot use relative coordinates (e.g. work coordinates, etc.) 	<ul style="list-style-type: none"> · 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	<ul style="list-style-type: none"> · 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 	<ul style="list-style-type: none"> · Describing relative coordinates such as work 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, degree, 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 the three types:

1. Real numbers

Real numbers can have both integers and fractions. It can be a positive or a negative value between -3.4×10^{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 (whole numbers) 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	<code>^B101</code>	(5 in decimal)
	<code>^HC1</code>	(193 in decimal)
	<code>¬^B1000</code>	(−8 in decimal)
	<code>¬^H1000</code>	(−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 names 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 defined under one name using the array variable.

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 redefined each time the program is executed, and are not saved in the memory. 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 is run. 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. Since local variables are not saved in the memory, the value of a local variable .pose cannot be displayed using the below command.

>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 VARIABLE NAMES

Variable names must start with an alphabetic 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 variable is limited to fifteen characters. Only the first fifteen characters will be valid with longer names. 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 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 values. Character string variables are preceded by “\$” to differentiate them from real variables.

pick	(transformation value)	#pick	(joint displacement value)
count	(real variable)	\$count	(string 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 values 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 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 having to teach 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. Calculated values can be used as 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 under 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 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 order as in (2) above.

Example 2 Using transformation values

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

3.5.2 DEFINING BY PROGRAM INSTRUCTIONS

1. HERE instruction stores the robot's current pose under the specified name.

```
HERE pose
```

2. POINT instruction substitutes a pose variable 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**]

When a pose variable name starting with a # is used, the variable describes joint displacement values (i.e. #pick, #start).

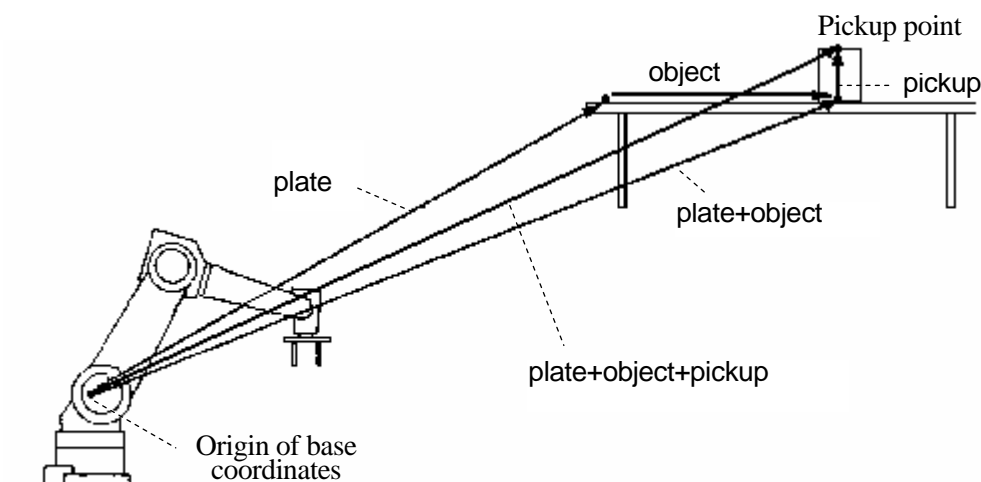
When a pose variable has no prefix and starts with any alphabetical character, the variable describes transformation values (i.e. pick, start).

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 for the transformation values relative to the base coordinates describing the coordinates at the plate. Then, if the pose of an object relative to the pose of 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 of the plate changes, only the transformation values for the plate will need revising and the rest can be used as is.



The compound transformation values can be defined using any command or instruction used to define pose variables. (It is easiest to use the HERE command.)

First, use the teach pendant to jog 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, jog 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 variable “b” is defined relatively to transformation 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”, “LMOVE pickup” unless you are certain of its purpose and its effect on the program.

When using compound transformations 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 100mm above “pickup”
JMOVE	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 100mm 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 when the POINT command is used. 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 at “p2”. The value of the rightmost variable on the right side of the expression is assigned to the variable on the left side of “=”.

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 value of JT7 for the variable “p”.

3.6 DEFINING REAL VARIABLES

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

$$\text{Real_variable_name} = \text{numeric_value}$$

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., x[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=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 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, this instruction replaces the current string with the new string on the right side.

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 a real number value. 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. When expressions containing parentheses are evaluated, 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 3.7 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, but the second statement is easier to understand.

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

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 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 string. If all the characters are the same, the two strings is 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.

4.0 AS PROGRAM

This chapter explains about AS programs. It explains how to create and execute 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

4.2 Creating and Editing Programs

4.3 Executing Programs

4.4 Program Flow

4.5 Robot Motion

4.1 TYPES OF AS PROGRAM

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 KCwin32 terminal software 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 this program.

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 although PC programs cannot be used in motion instruction. Instructions that cause robot motion, except for the BRAKE instruction. Also, BASE and TOOL functions are not available for PC programs.

4.1.3 AUTOSTART

A PC program can be set to start automatically when the control 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; e.g. MC CONTINUE, etc. (See 6.9 MC program instruction.)

This is a sample autostart program. In this example, the robot monitors **MOTOR POWER**, and executes program pg1 when the power 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 the MOTOR POWER ON  
2  WAIT SIG(27)          ;checks if the robot is at home position*  
3  MC EXECUTE pg1        ;Executes pg1(robot motion program)
```

NOTE * Set home position 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

2.

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.

3. Label


Labels are used in a program to branch the program. A label can be either an integer from 1 to 9999 or a string of up to 15 alphanumeric characters, period or underscore (starting with alphabetical character), followed by a colon (:). Labels are inserted at the beginning of a program line, right after the step number. Labels can be used as a branch destination from anywhere within the program.

4. 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.)

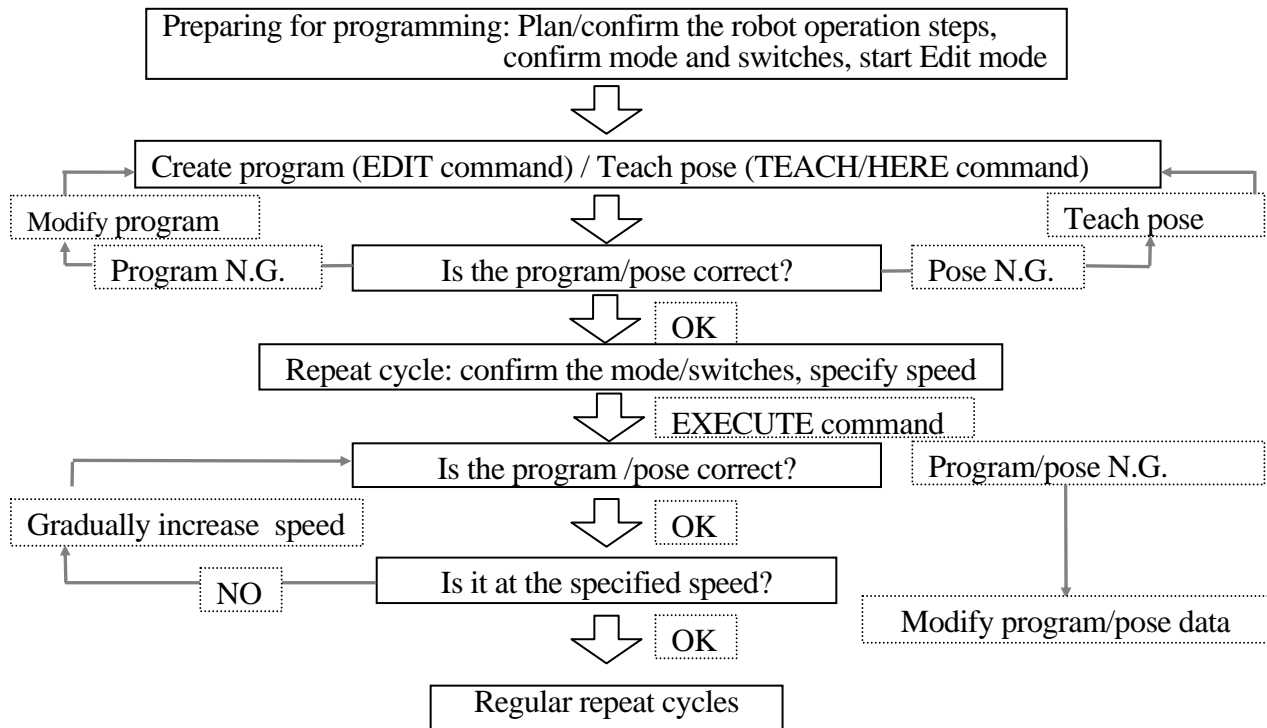
4.2.2 EDITOR COMMANDS

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 step count	Deletes program steps.
E	Exits editor mode, and returns to monitor mode.
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 step count	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 PROCEDURE

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

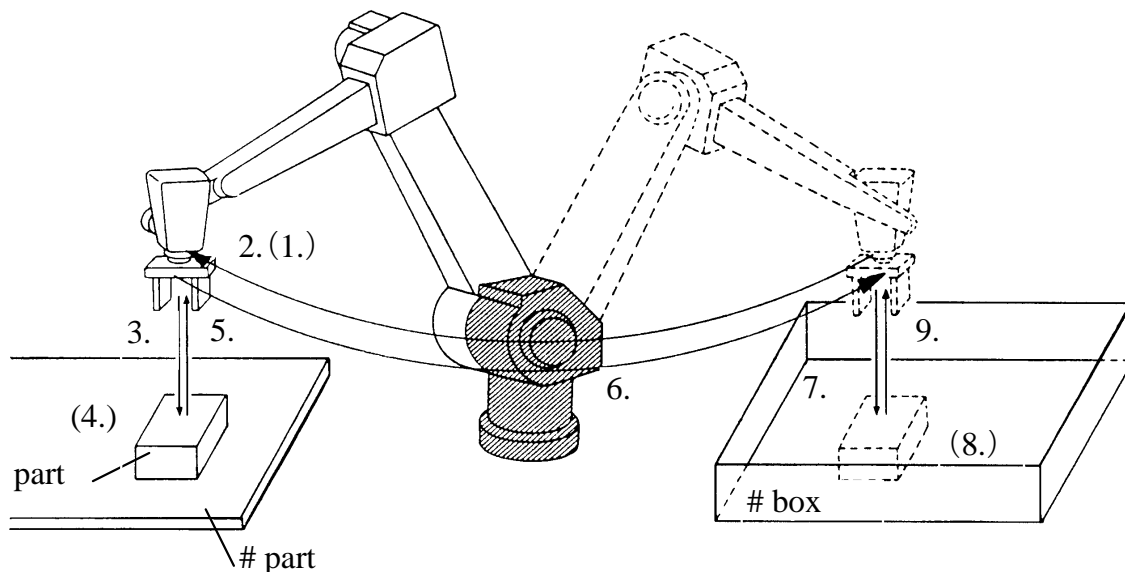
Here is a sample program. The robot will perform the task shown on the next page: pick up a part fed in by the supply shoot (conveyor), and place it in the box.

First define all the motions required to complete the task:

1. Check if the hand is open.
2. Move to a pose 50mm 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 150mm above the supply shoot.
6. Move to a position 200mm 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 180mm above the box.

The variables #part, #box which express position and posture are called pose (location) data in the AS system. Define the pose variable names as shown in Chapter 3 before executing the program.



AS Editor is used to create and edit programs. 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 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 also 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. 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 turn the **HOLD/RUN** switch to RUN position.

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 at 10%.

> 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, turn the **HOLD/RUN** switch to HOLD. The robot will decelerate and stop. In case of emergency, press the **EMERGENCY STOP** button on the controller 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, the **CYCLE START** button on the controller panel 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 **CYCLE START** on the operation panel begins execution. Execution can also be started using the CONTINUE command.

3. Running program via STEP command or **CHECK** 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** key* on the teach pendant.

Note * When using the **CHECK** 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. Turn the **HOLD/RUN** switch in the operation panel 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 the **EMERGENCY STOP** button, release the lock of the **EMERGENCY STOP**, and press the **MOTOR POWER** to turn ON the motor power. Robot starts moving when you press the **CYCLE START**.
2. When the **HOLD/RUN** switch was used to stop the robot, turn the switch to the RUN position to resume.
3. To resume after the ABORT or HOLD command or when program execution was suspended by an error, use the 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.

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

The STOP instruction may not stop the execution in some cases. If the specified execution cycles are remaining, 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 1,2 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 a parameter name to transformation values put a "&" in front of the parameter variable name in order to differentiate from the 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 “c” is the sum of transformation values “a” and “b”.

```
position()
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)

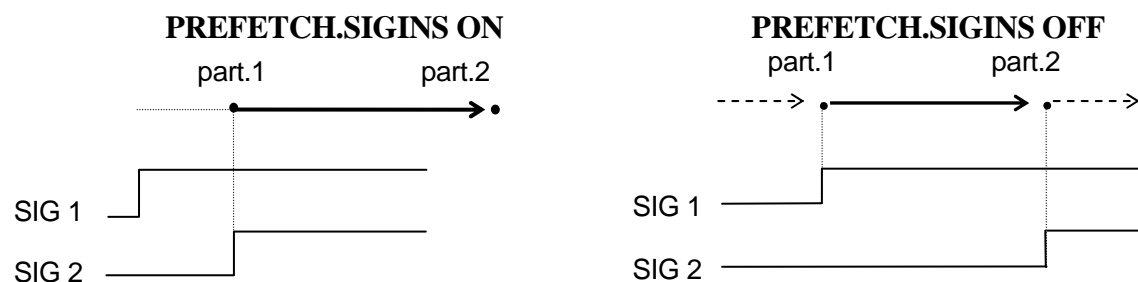
Under certain conditions, like when there is an error or when a specific external signal is input, program execution may be interrupted and another program will be executed. This occurs independently from the flow of execution of the main program and is called asynchronous processing (interruption). 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 (enable early processing of signal I/O commands) or OFF (disable 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.

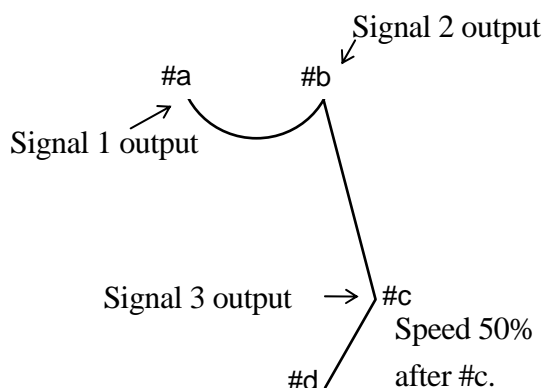
When PREFETCH.SIGINS is OFF, the signals are output after the robot reaches the destination of the motion instructions and the axis coincide.

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

```



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]

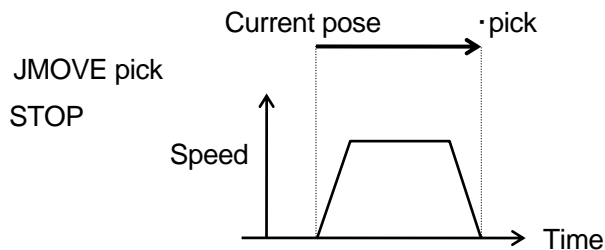
When PREFETCH.SIGINS is ON, the program processes the next step until it has to wait for the robot to reach the specified pose. However, the timing is affected by other settings and command/ instructions such as WAIT instruction or the CP switch. WAIT instruction suspends the processing of steps until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

As demonstrated here, it is important to note that the timing in which the AS system processes the program and 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.

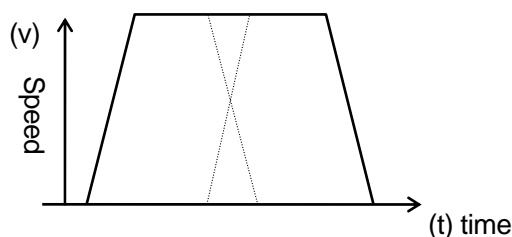


When executing a motion instruction like the one above, the robot accelerates smoothly up to the current speed setting as it moves towards the pose “pick”. As the robot approaches “pick”, it gradually decelerates until it stops at the pose. Series of motions such as this, carried out by one motion instruction,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

JMOVE pos.1
JMOVE pos.2
STOP

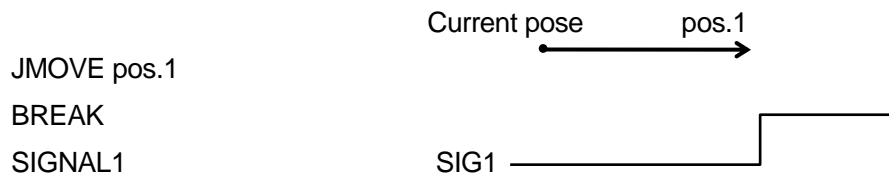


Motion like this, where the robot performs a series of motions making a smooth transition between the motion segments without stopping at each destination, is called CP (Continuous Path) motion. Turning OFF the CP system switch disables the CP function. If the CP switch is turned OFF, the robot will 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.



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

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.)

[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, it requires a certain amount of time to calculate the transition between the motion segments. Therefore, if the distance between the two positions is too short, the calculation may not be finished before reaching the second position, thus causing 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 define the poses unnecessarily close together.

·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.

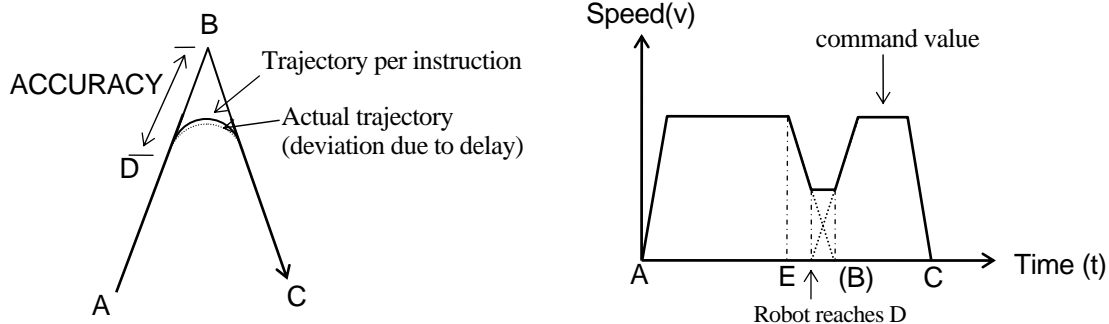
·CP Switch..... Enables or disables CP motion.

4.5.4.1 CP ON ...STANDARD MOTION TYPE

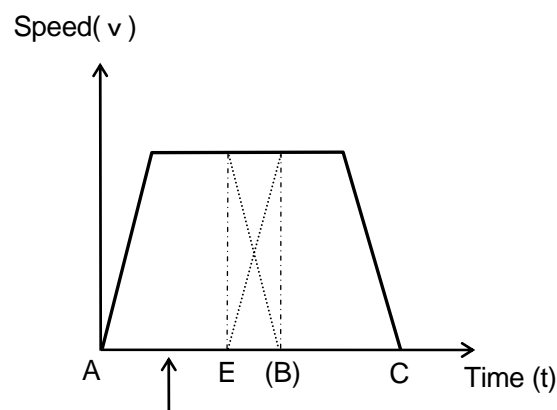
For example the robot takes the motions below with the CP switch ON: A →B→C.

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.

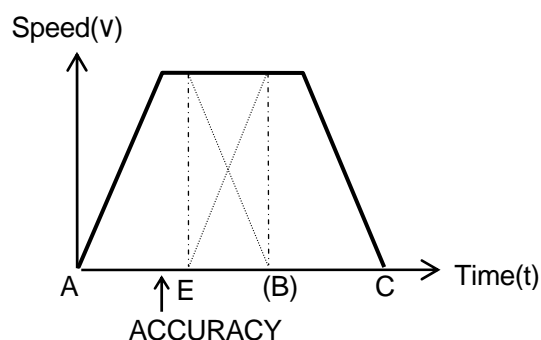


The greater the range specified by ACCURACY, the earlier the superposing will begin. However, acceleration on next path does not begin before the point where the robot starts to decelerate (point E), therefore it can be said that the ACCURACY effect is saturated at a certain value, i.e. there is no effect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

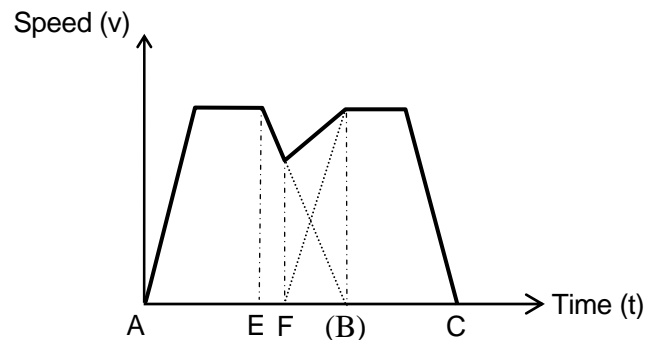


Even if command value reaches the accuracy point at this time, acceleration for next path will not start until deceleration begins at point E.

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).



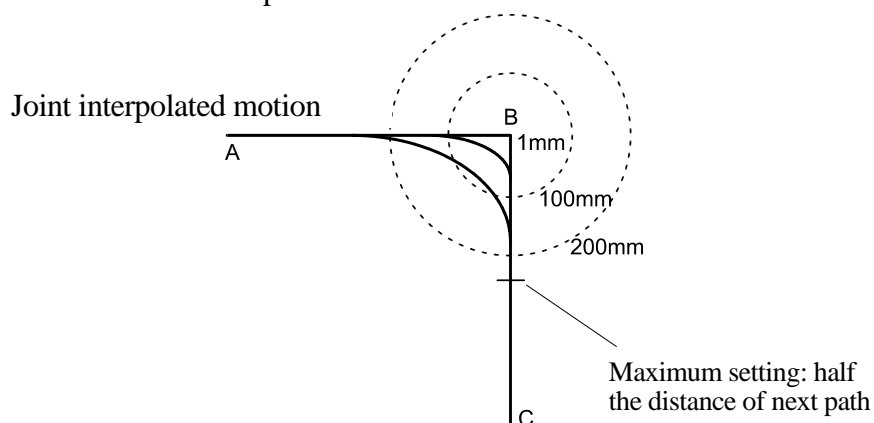
4.5.4.2 CP ON ... MOTION TYPE 2

In Motion type 2, the concept of accuracy and velocity in linear motion and circular motion is different from that of Standard motion type. 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

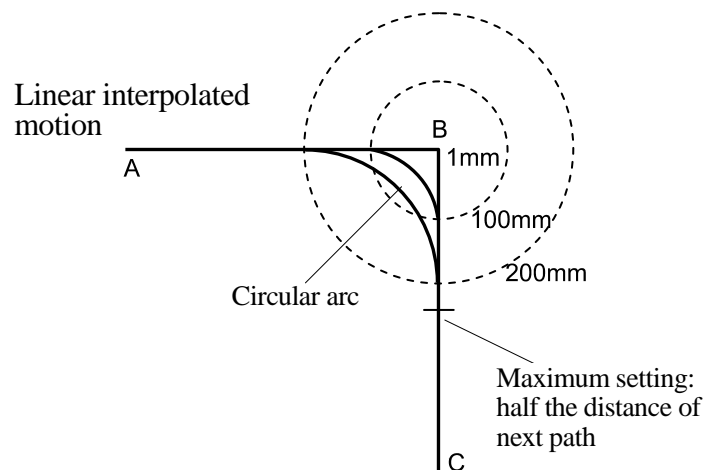
The motion path of the robot corresponding to the accuracy setting is shown in the figure below. In this example the accuracy values at point B are 1 mm, 100 mm, and 200 mm. In the same way as Standard motion, the robot starts to shortcut before reaching point B, but does not necessarily start turning 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 interpolated 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 work/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 \leftrightarrow motion based on the fixed tool coordinates)
- When the processing branches due to conditions set by instruction such as IF

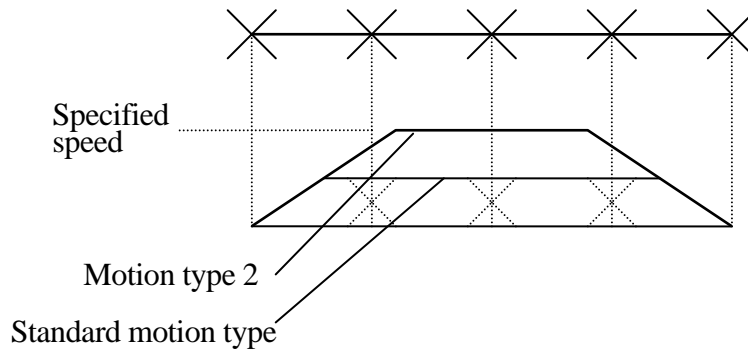
2. Speed setting

(1) Speed in joint interpolated motion

The 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 the accuracy in joint interpolated motion:

- When a waiting instruction (TWAIT, SWAIT, etc.) is executed at point B
- When a work/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 from ordinary mode (work is fixed and tool moves) to fixed tool dimensions
- When the processing branches due to conditions set by instruction such as IF

[NOTE]

When attempting to execute a program where the robot posture changes greatly within a short distance, the time it takes to change the posture 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

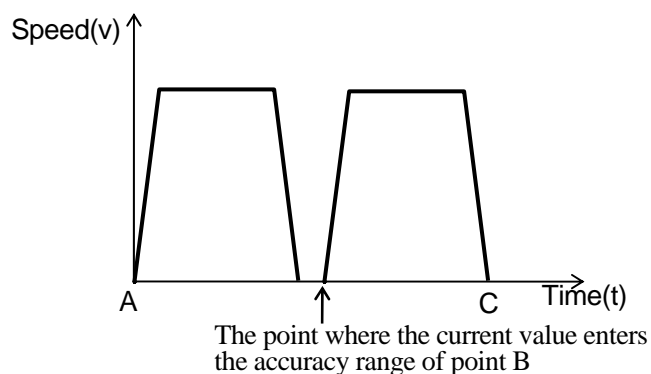
In Motion type 2 the maximum speed is automatically set according to the robot's capacity to carry out proper circular interpolated motion.

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.

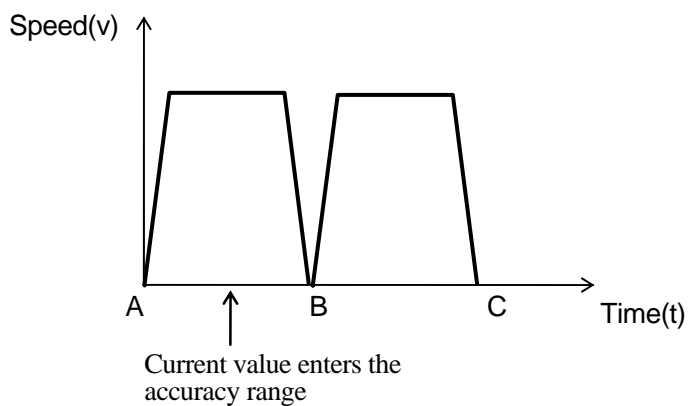
4.5.4.3 CP OFF

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 FS10, 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 possible 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
```

path[0] to path[10] is to be defined by manual teaching pose or by calculation.

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.

5.0 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.

5.1 Editor Commands

5.2 Program and Data Control Commands

5.3 Program and Data Storage Commands

5.4 Program Control Commands

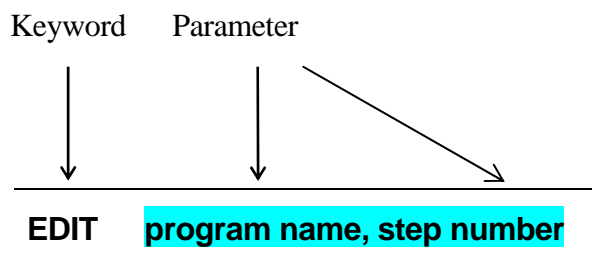
5.5 Pose Information Commands

5.6 System Control Commands

5.7 Binary Signal Commands

5.8 Message Display Commands

EXAMPLE



Parameters marked with 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

1. 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 entered program does not exist, a new program is created.

2. 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 is stopped.

C program name, step number

Function

Changes the program currently selected in editor mode.

Parameter

1. Program name

Selects the program to be edited.

2. Step number

Selects the step number to start editing. If no step is specified, the first step 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 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 **step count**

Function

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

Parameter

Step count

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 **Enter** key. All lines written before exiting the insert mode are inserted in the program.

Example

CLOSEI command 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 Enter	;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 **Enter** while in the insert mode.

D **step count**

Function

Deletes the specified number of steps including the current step.

Parameter

Step count

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 renumbered and automatically moved up 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

1. Existing characters

Specifies which characters are overwritten in the current step.

2. 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 Enter.

Once Enter key 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 “online”, not zero).

Example

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

```
3      JMOVE abc
3?O 
3      JMOVE abc  ; delete “abc” using 
3      JMOVE def  ; Enter “def”
3      JMOVE def
3?
```

[NOTE]

This command cannot be used with the teach pendant.

E

Function

Exits from the editor mode and returns to monitor mode.

XD **step count**

Function

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

Parameter

Step count

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 **step count**

Function

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

Parameter

Step count

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 **variable name**

Option

Function

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

Parameter

Variable name

Specifies pose variable name of destination to be taught, expressed in transformation values or joint displacement values. Read as an array variable if specified in the form of A[], it will be taken variable. 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 parameter

2 JAPPRO #a

3? T pos

3 JMOVE pos0

4 JMOVE pos1

5 LMOVE pos2

⋮

<Teach using TP. Press Cancel to return to AS>
(3 steps are taught here)

Without parameter

2 JAPPRO #a

3? T <Teach joint values using TP. Press 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 ON.

5.2 PROGRAM AND DATA CONTROL COMMANDS

CARD_FDIR*	Lists names of the programs and variables in PC card.
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.
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.
CARD_FDEL *	Deletes programs and variables in PC card.
CARD_VERIFY*	Turns ON/OFF the verifying function in PC card drive operation.
RENAME	Changes the name of a program.
XFER	Copies steps from one program to another.
COPY	Copies programs.
CARD_COPY*	Copies programs in PC card.
TRACE	Turns ON/OFF the TRACE Function. (Option)
SETTRACE	Reserves memory for logging. (Option)
RETRACE	Releases memory reserved with SETTRACE. (Option)
LSTRACE	Displays the logging data. (Option)

NOTE* These commands are used to control PC card memories, but the same commands work for floppy disk if CARD_ is changed to FD_. See the explanations for each command for further information.

CARD_FDIR

FD_FDIR

Function

Displays the name of programs or variables. CARD_FDIR refers to data on the PC card, and FD_FDIR refers to data on the floppy disk.

Explanation

By using the CARD_FDIR and FD_FDIR commands, all the subroutines and variables used in the programs are displayed.



Example

>FD_FDIR 

Displays the names all programs, subroutines and variables on the floppy disk.

>CARD_FDIR

Displays the names of all the programs, subroutines and variables on the PC.

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

[NOTE]

Program and variable names with * or ~ displayed at the beginning of the name means that the contents of that program or variable are not yet defined.

LIST	program name,
LIST/P	program name,
LIST/L	pose (location) variable,
LIST/R	real variable,
LIST/S	string variable,

Function

Displays the specified program and data.

Parameters

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

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 main program.

Example

>LIST Displays the contents of all programs, including the variables and their values.

>LIST/L Displays all the pose (location) 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 . To quit the display, press .

DELETE	program name,
DELETE/P	program name,
DELETE/L	pose variable,
DELETE/R	real variable [array elements] ,
DELETE/S	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)

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 are also deleted. (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/R and DELETE/S command, all the elements in that array variable are deleted. If the element(s) are specified, only the specified element(s) are deleted.

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”.

CARD_FDEL file name,

FD_FDEL file name,

Function

Deletes the specified data from the memory.

Parameters

File name

Specifies the name of the file to delete.

Explanation

CARD_FDEL and FD_FDEL commands delete the programs in the specified file completely; i.e. the main program itself AND, if used in the program, the following data are also deleted. (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.

CARD_VERIFY mode

FD_VERIFY mode

Function

Turns ON/OFF the verify function in the driver board operation when data is written on the PC card (CARD_VERIFY), or floppy disk (FD_VERIFY).

Parameters

Mode

- 0: Turns OFF the verify function.
- 1: Turns ON the verify function.
- 2: Returns if the current setting of the verify function is ON or OFF.

If omitted, 2 is assumed.

RENAME new program name=existing program name

CARD_RENAME new program name=existing program name

FD_RENAME new program name=existing program name

Function

Changes the name of a program currently held in memory. RENAME changes the name of programs in robot memory, CARD_RENAME those in PC card, and FD_RENAME, floppy disk.

Parameters

1. New program name

Sets new name for the program.


2. Existing program name

Specifies the current name of the program.

Explanation

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

Example

>FD_RENAME test=test.tmp 

Changes the name of the program in floppy disk, from “test.tmp” to “test”.

**XFER destination program name, step number1 =
source program name, step number2, step count**

Function

Copies and transfers steps from one program to another program.

Parameters

1. Destination program name

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

2. 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.

3. Source program name

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

4. 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.

5. Step count

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 +

CARD_COPY new program name =
source program name + source program name +

FD_COPY new program name =
source program name + source program name +

Function

Copies a complete program to a new program. **FD_COPY** copies the programs on the floppy disk, and **CARD_COPY** copies the programs on the PC card.

Parameters

1. New program name

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

2. 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.

TRACE **stepper number:** ON/OFF

Option

Function

Logs and traces the actual contents of robot and PC programs that were executed.

Parameters

1. Stepper number

Specifies the type of program to be traced by 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 type is specified, all the programs are logged.

2. ON/OFF

Starts/Ends tracing.

Explanation

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

SETTRACE **step count**

Function

Reserves the necessary memory to log the tracing data.

Parameters

1. Step count

Specifies the number of steps to log in the setting range 1 to 9999. If not specified, memory for 100 steps will be saved.

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 for logging 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 using the 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**1. Stepper number**

Specifies the type of program to be displayed by 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 type is specified, the log for robot program is displayed.

2. Log line 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 for logging” 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 $\left[\begin{array}{|c|} \hline \downarrow \\ \hline \end{array} \right]$ displays the next 9 lines.

L $\left[\begin{array}{|c|} \hline \downarrow \\ \hline \end{array} \right]$ displays the previous 9 lines.

S **number** $\left[\begin{array}{|c|} \hline \downarrow \\ \hline \end{array} \right]$ 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** $\left[\begin{array}{|c|} \hline \downarrow \\ \hline \end{array} \right]$ 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-----

5.3 PROGRAM AND DATA STORAGE COMMANDS

CARD_FORMAT	Formats PC card.
FD_FORMAT	Formats floppy disk.
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.
LOAD *	Loads programs and data to robot memory.

NOTE* These commands save data to personal computers. To save the data on PC card or floppy disk, add the prefix CARD_ or FD_ to the command. See the explanation for each command for further information.

CARD_FORMAT

FD_FORMAT **format type**

Function

CARD_FORMAT command initializes the SRAMPC card.

FD_FORMAT command initializes the floppy disk.

Parameters

Format type

Selects from below the type of the floppy disk to format. If omitted, 1 is assumed.

1: 1.44MB

2: 1.25MB

Explanation

This command overwrites the format data on the memory and creates a new file directory. This erases all the contents of the existing memory.

[NOTE]

A new SRAMPC card must be formatted before programs and data can be stored on it.

Flash ATA PC cards do not need to be formatted. Usually, they are sold formatted.

Example

```
>CARD_FORMAT   
Are you sure? (Yes:1, NO:0)? 1  
Now formatting card...  
>  
Formatting done.
```

SAVE/SEL	file name=program name,
-----------------	--------------------------------------

CARD_SAVE/SEL	file name=program name,
----------------------	--------------------------------------

FD_SAVE/SEL	file name=program name,
--------------------	--------------------------------------

Function

SAVE command stores the program and variable data on the computer hardware. (Use only when a PC is connected to the robot controller.)

CARD_SAVE command stores the programs and variable data on a PC card.

FD_SAVE command stores the programs and variable data on a floppy disk.

Parameters

1. File name

Saves the specified program under this file name. If the extension is not specified, the extension “.as” is automatically added to the file name.

2. 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.

The SAVE command (without /SEL) stores the specified program(s), including any variables and subroutines used by the program(s).


The 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 file name
SAVE/SYS file name
SAVE/ROB file name
SAVE/ELOG file name

```

```

CARD_SAVE/P/SEL file name=program name, .....
CARD_SAVE/L/SEL file name=program name, .....
CARD_SAVE/R/SEL file name=program name, .....
CARD_SAVE/S/SEL file name=program name, .....
CARD_SAVE/A file name
CARD_SAVE/SYS file name
CARD_SAVE/ROB file name
CARD_SAVE/ELOG file name

```

```

FD_SAVE/P/SEL file name=program name, .....
FD_SAVE/L/SEL file name=program name, .....
FD_SAVE/R/SEL file name=program name, .....
FD_SAVE/S/SEL file name=program name, .....
FD_SAVE/A file name
FD_SAVE/SYS file name
FD_SAVE/ROB file name
FD_SAVE/ELOG file name

```

Function

Stores in the disk file the program (/P), pose variable (/L), real variable (/R), string variable (/S), auxiliary information (/A), system data (/SYS), robot data (/ROB), and error log (/ELOG).

As with SAVE command, CARD_SAVE/ and FD_SAVE/ command are used to save files in PC card and floppy disk respectively (use SAVE/ command only when a PC is connected to the robot controller. See 2.6.2 Uploading and Downloading Data).

Parameters

1. File name

Saves the data under this file name. If the extension is not specified, the following extensions are automatically added to the file name according to the type of data in the file:

program	.PG	system data	.SY
pose information	.LC	robot data	.RB

real variables	.RV	error log	.EL
string variables	.ST		
auxiliary information	.AU		

2. 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 disk file.

Explanation

1. SAVE/P

Stores in the specified disk 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 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.

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.


6. SAVE/ELOG

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

7. 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 the NOTE box regarding 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

CARD_LOAD/Q file name

FD_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).

CARD_LOAD command loads the files on the PC card into the robot memory.

FD_LOAD command loads the files on the floppy disk into the robot memory.

Parameters

File name

Saves the specified program under this file name. If no extension is specified, the extension “.as” is automatically added to the file name.

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 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.
```

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 limit release option is set, then the value can be set 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 and the program speed 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 maximum speed is set at 30%.

>SPEED 50 The maximum speed is set at 50%.

>SPEED 100 The maximum speed is set at 100%.

>SPEED 200 The maximum speed is set at 100%.

PRIME program name, execution cycles, step number

Function

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

Parameter

1. Program name

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

2. 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).

3. 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 the **CYCLE START** switch.

[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

1. Program name

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

2. 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).

3. 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

1. Program name

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

2. Execution cycles

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

3. 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 turning the **HOLD/RUN** switch 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

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 unchanged. 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.

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	Defines a pose variable as the current pose.
POINT	Defines a pose variable.
POINT/X	Sets the X value of a pose variable.
POINT/Y	Sets the Y value of a pose variable.
POINT/Z	Sets the Z value of a pose variable.
POINT/OAT	Sets the OAT values of a pose variable.
POINT/O	Sets the O value of a pose variable.
POINT/A	Sets the A value of a pose variable.
POINT/T	Sets the T value of a pose variable.
POINT/7	Sets the seventh axis value for a pose variable.

HERE pose variable name

Function

Defines a pose (location) variable name as the current pose. The pose may be expressed in transformation values, joint displacement values or compound transformation values.

Parameter

Pose (location) variable name

Can be specified in transformation values, joint displacement values, or compound transformation values.

Explanation

The pose may be expressed in transformation values, joint displacement 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 **ENTER** 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 posture of the tool coordinates.

Example

>HERE #pick  Defines the robot's current pose as “#pick” (joint displacement values)

>HERE place  Defines the robot's current pose as “place” (transformation values)

HERE plate+object  Defines the robot's current pose as “object ” relative to the pose “plate”(compound transformation values). Error occurs if “plate ” is undefined.

POINT pose variable name = pose values, joint displacement values

Function

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

Parameter

1. Pose variable name

Specifies the name of pose information variable to be defined (may be defined in joint displacement values, transformation values, or compound transformation values)

2. Pose values

If not specified, the “=” sign is also omitted.

3. Joint displacement values

This parameter must be set if the pose variable name on the left is defined by joint displacement values and the pose information values on the right are transformation values (if the parameter on the left is not defined by joint displacement values, this parameter cannot be set). The joint displacement values specified here define 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 on the right to the pose variable. When the pose values are not specified, any values already defined for that pose are displayed on the terminal, and can be edited. If the pose variable 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 **ENTER** at the prompt.

If the variable is defined by joint displacement values, then joint values appear on the display. If the variable is defined 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 posture 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 the variable 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 the variable on the left. If the joint displacement values on the right are specified, the transformation value is transformed with the robot taking the configuration of the specified joint displacement values. If not specified, the transformation value is transformed with the robot in its current configuration.

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

Example

>POINT #park Displays the values of pose "#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" to transformation values of "pick1" and displays the values for correction.

>POINT pos0=#pos0 Transforms the joint displacement values "#pos0" into transformation values and assigns them to "pos0"

>POINT #pos1=pos1,#pos2 Transforms the transformation values "pos1" into joint displacement values using the robot configuration given by #pos2 and assigns the values to "#pos1".

POINT/ X	transformation variable name=transformation values
POINT/ Y	transformation variable name=transformation values
POINT/ Z	transformation variable name=transformation values
POINT/ OAT	transformation variable name=transformation values
POINT/ O	transformation variable name=transformation values
POINT / A	transformation variable name=transformation values
POINT / T	transformation variable name=transformation values
POINT/ 7	transformation variable name=transformation values

Function

Assigns the components of the transformation values specified on the right of “=” to the corresponding component of the transformation values on the left of “=”. The values will be displayed on the terminal for editing.

Parameter

1. Variable name

Selects the transformation variable name to be defined (transformation values or compound transformation values)

2. Transformation values

If not specified, the “=” sign is also omitted.

Explanation

Assigns only the specified components (X, Y, Z, O, A, T) 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 the **ENTER** 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.
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.
SETHOME	Sets the home position.
SET2HOME	Sets the home position 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.
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.
BATCHK	Enables/disables battery low voltage check.
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.
PLCAOUT	Sets real values to output data. (Option)
TPLIGHT	Turns on teach pendant backlight. (Option)
IPEAKLOG	Displays peak current values. (Option)
IPEAKCLR	Resets peak current values. (Option)
OPEINFO	Displays operation information. (Option)
OPEINFOCLR	Clears operation information. (Option)

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				
	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

The current monitor speed (in percentages)

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 **ENTER** is pressed again.

WHERE	Displays the current robot pose in transformation values in the base coordinates and the joint angles (JT1, JT2, ..., JT3).
WHERE 1	Displays the current pose by joint angles.
WHERE 2	Displays the current pose in XYZOAT in the base coordinates (mm, deg).
WHERE 3	Displays the current instructed values(deg).
WHERE 4	Displays deviations from the instructed 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 work 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.

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 ENTER 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.

Example

>FREE

Total memory 262144 bytes

Available memory size 262128 bytes (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?”

Explanation

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

Year	(00～99)
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 ENTER 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 09:47:33
```

```
Change? (If not , hit RETURN only)
```

```
> 
```

ULIMIT	Joint displacement values
LLIMIT	Joint displacement values

Function

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

Parameter

Joint displacement values

Sets the software limit (upper or lower) in joint displacement values. If this parameter is not specified, the current value is displayed.

Explanation

If the parameter is specified, the values entered 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 **ENTER** 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) ENTER
```

```
>ULIMIT #upper ENTER                     Sets the upper software limit to the pose defined as
                                           “#upper”.
```

:

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

:

BASE transformation values

Function

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

Parameter

Transformation values (or Compound transformation values)

Defines the new base coordinates. The transformation values here describe the pose of the base coordinates with respect to the null base coordinates, expressed in null base coordinates. If no values are entered, 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 **ENTER**. 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.

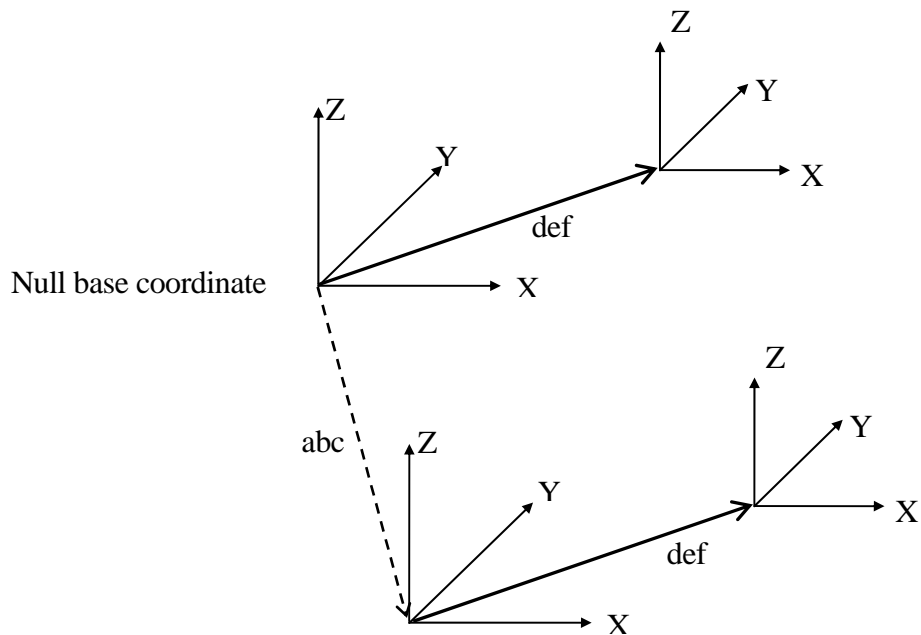
When a pose variable is used as the parameter and if that pose variable is redefined, note that the base transformation must also be redefined using the BASE command and the newly defined pose as the parameter. The change made in the pose variable will then be reflected to the base transformation.

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

[NOTE]

BASE abc <input type="button" value="↵"/>	BASE NULL <input type="button" value="↵"/>
DO JMOVE def <input type="button" value="↵"/>	DO JMOVE def <input type="button" value="↵"/>

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 pose variable “abc”.

TOOL transformation values

Function

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

Parameter

Transformation values (or Compound transformation values)

Defines the new tool coordinates. The transformation values here describe the pose of the tool coordinates with respect to the null tool coordinates, expressed in null tool coordinates. If no values are entered, the current tool transformation values are displayed.

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 **ENTER**. 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)

SETHOME	accuracy, HERE
SET2HOME	accuracy, HERE

Function

Sets and displays the HOME position.

Parameter

1. Accuracy

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

2. 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 **ENTER** key. If no change is made, press only **ENTER**.

Two HOME positions (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 Sets the accuracy at 2 mm, and changes the HOME position by entering the new values.

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) **ENTER**

>

>SETHOME 10,HERE **ENTER**

Sets the current pose as the HOME position. 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 position.

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

1. 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.


2. ON or OFF

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

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.PC	OFF	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 changes the total number of external and internal I/O signals.

Explanation

The current setting and the message “Change?” are displayed. This command changes only the software setting. Make sure the number of signals corresponds with the hardware setting.

Example

```
>ZSIGSPEC 
DO, DI, INT          (DO=Ext. output signal, DI= Ext. input signal, INT=Int. signal)
64  64  128
Change? (If not , hit RETURN only)
32,32,32
DO, DI, INT
32  32  32
Change? (If not , hit RETURN only) 
```

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) 3

[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.

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
EXT_IT	HOME1, HOME2
EXT. SLOW REPEAT MODE	POWER ON
	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

[**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 ~ number of signals installed

Dedicated input signals: 1001 ~ 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 to correspond to a robot's known mechanical position. Also, the current amount of offset between the robot mechanical position and the encoder origin (zeroing data) can be displayed using this command.

Parameter

Joint number

(1) To reset the encoder rotation count:

Enter the joint number plus 100. For example, to reset the encoder rotation count on joint two, enter:

ZZERO 102 

If "100" is entered as the joint number, all the joints are reset to 0°.

(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 set at 0° as zeroing data.

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

[NOTE]

Reset the encoder rotation count 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   268435456  268435456  268435456  268435456  268435456  268435456
data
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 for the specified joint position.

```
>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 so that the current pose is 0°.

```
>ZZERO 0 
          JT1      JT2      JT3      JT4      JT5      JT6
Set data 268427264 268427264 268427264 268427264 268427264 268427264
Current  268427264 268427264 268427264 268427264 268427264 268427264
Set current values of all joints as zeroing data? (Enter 1 to set.)1 
Setting complete. 
```

5. Sets the robot to recognize the encoder value describing the current pose of joint 2 as 0°

```
>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 alpha character command displays the monitor commands or program instructions starting with that alphabet. Entering HELP/F 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 current robot connected
Joint number:	number of joints of the robot
Serial number:	serial number of the robot
Software version:	version number of the AS software
Servo:	version number of the servo software
Num of signals:	total number of output, input, and internal signals available in this system
Clamp number:	total number of clamps available in this system
Motion type:	motion type of the robot
Servo type:	type of servo 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 : JS005-E001 Num of axes: 6 Serial No.1
Software version : version 000004-04...97/01/27 13:11
Servo : SAOA00-UX120-01
Number of signals: output=32 input=32 internal=256
Clamp number :2 MOTION TYPE: 1 SERVO TYPE:1

WEIGHT	load mass, center of gravity location X, center of gravity location Y, center of gravity location 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 work). The data is used to determine the optimum acceleration of the robot arm.

Parameter

1. Load mass

The mass of the tool and work (in kilograms). Range: 0.0 to the maximum load ability (kg).

2. Center of gravity location(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

3. Inertia moment about X axis, inertia moment about Y axis, inertia moment about Z axis

(Option)

Sets the inertia moment around each axes. Unit is $\text{kg}\cdot\text{m}^2$. 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 location. Incorrect data may weaken or shorten the longevity of parts or cause overload / deviation errors.

BATCHK

Function

Enables or disables the battery low voltage check.

Explanation

>batchk

BATTERY ERROR CHECK (0: Ineffect, 1: Effect)

(Enter only: No change ^C: Exit): Now 1 Change ?

If no change is made, press only **ENTER**. Enter 0 to disable battery check and 1 to enable it.

[NOTE]

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

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 **ENTER**. To change the speed, enter the new value and press **ENTER**.

[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]

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

ENV2_DATA

Function

Sets software environmental data.

Explanation

>ENV2 DATA

TEACH PENDANT (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]

+ (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]

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

PLCAOUT data number = real value

Option

Function

Sets the given real number value to the specified data number.

Parameter

1. Data number

Specifies the output data number in whole number. Acceptable number is from 1 to 32.


2. Real value

Specifies the real number value that is to be set to the output data, entered in decimal notation. It can also be set in variable names. Acceptable number is from 0 to 65535.

[**NOTE**]

This command is only valid when the “Built-in Sequencer Function” option is ON. If the option is OFF, the following error message appears (E1102) Cannot execute, no option set up. - Check option specs.

Example

>PLCAOUT 13=120  Sets “120 (decimal notation)” to data number 13.

TPLIGHT

Option

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 1KA board.

Example

```
>IPEAKCLR
```

```
Are you sure? (Yes:1,No;0) 1 
```

```
>
```

OPEINFO robot number: joint number

Option

Function

Displays the operation information.

Parameter

1. Robot number

Specifies the robot if more than one robot is controlled by one controller.

2. 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 500mm/s

OPEINFOCLR

Option

Function

Resets the operation information.

5.7 BINARY SIGNAL COMMANDS

The D series 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 a group of signals to be equal to the specified value.
SCNT	Outputs counter signal upon reading a specified value. (Option)
SCNTRESET	Clears the counter signal number. (Option)
SFLK	Turns ON/OFF the flicker signal. (Option)
SFLP	Turns ON/OFF signals with SET/RESET signals. (Option)
SOUT	Outputs signal when specified condition is met. (Option)
STIM	Turns ON timer signal when specified signal is ON (Option)
SETPICK	Sets the time to start clamp close control. (Option)
SETPLACE	Sets the time to start clamp open control. (Option)

RESET

Function

Turns OFF all the external output signals. Dedicated signals, clamp signals and antinomy signals 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 an external output signal or an internal signal.

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	2001–2256
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, no output signals are changed. An error occurs when selecting a signal number already set as a dedicated signal.

Example

>SIGNAL -1,4,2010 

External output signal 1 is 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

1. Signal number

Selects the number of the external output signal or internal signal (only positive values). An error occurs if the signal number is already used as a dedicated signal.

Acceptable Signal Numbers

External output signal	1 – actual number of signals or 64 (the smaller of the two)
Internal signal	2001–2256

2. 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

1. 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. An error occurs if the signal number is already used by a dedicated signal.

Acceptable Signal Numbers

External output signal	1 – actual number of signals or 64 (the smaller of the two)
Internal signal	2001–2256

2. 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 decimal value specified. If the value is not specified, the current signal states are displayed.

Parameter**1. Starting signal number**

Specifies the first signal to set the signal state.

2. Number of signals

Specifies the number of signals to be set ON/OFF. The maximum number allowed is 16.

3. 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.

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	2001–2256

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 1,8=100  External output signals 1–8 are set to output 01100100
(the binary notation of 100).

**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

1. Counter signal number

Specifies the signal number to output. Setting range for counter signal numbers: 3097 to 3128.

2. 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.

3. 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.

4. Counter clear signals

Specified by signal number or logical expressions. If this signal is turned ON, the internal counter is reset to 0.

5. Counter value

When the internal counter reaches this value, the specified 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 count down 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 (flicker) the specified signal in specified time cycle.

Parameter

1. Signal number

Specifies the number of signal to flicker. Setting range: 3065 to 3096.

2. 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**1. Output signal**

Specifies the signal number of 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).

2. Set signal expression

Specifies the signal number or logical expression to set the output signal.

3. 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

1. 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).

2. Signal expression

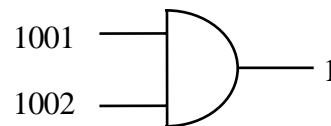
Specifies a signal number or a logical expression.

Explanation

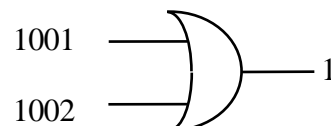
This command is for logical calculation of signals. Logical expressions such as AND and OR are used. The specified signal is output when that condition is set.

Example

SOUT 1 = 1001 AND 1002

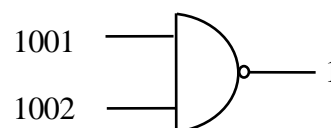


SOUT 1 = 1001 OR 1002

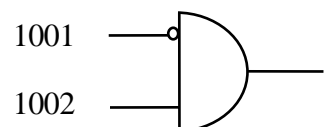


SOUT -1 = 1001 AND 1002

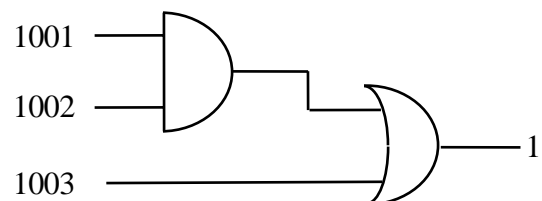
SOUT 1 = NOT(1001 AND 1002)



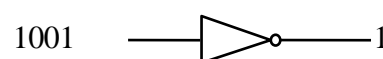
SOUT 1 = -1001 AND 1002



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**1. Timer signal**

Selects the signal to turn ON. Acceptable signal numbers are from 3001 to 3064.

2. Input signal number

Specifies in whole numbers 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.

3. 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 3128, 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.

5.8 MESSAGE DISPLAY COMMANDS

PRINT	Displays data.
TYPE	Displays data.
IFPWPRINT	Displays specified character string in a display window.

PRINT	device number: print data,
TYPE	device number: print data,

Function

Displays on the terminal the print data specified in the parameter.

Parameter

1. Device number

Select the device for displaying the data:

1: Personal computer

2: Teach pendant

If not specified, the data is displayed on the currently selected device.

2. 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. Press **NEXT PAGE** to return to the regular 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. -1.234 E+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 six or more, and smaller than 32.
- /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 Em.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-1 blank 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, 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”.

IFWPRINT	window,	row, column, background color, label color	=
		“character string”, “character string”,	

Option

Function

Displays the specified character string in the string window set by Auxiliary Function 0509 (Interface panel screen).

Parameter

1. Window

Selects the window in which to display the string. Select from 1 to 4 (standard) in Auxiliary Function 0509.

2. Row

Specifies the row in the window for displaying the string. Acceptable number is from 1 to 16, though it depends on the window size. If not specified, 1 is assumed.

3. Column

Specifies the column in the window for displaying the string. Acceptable number is from 1 to 43, though it depends on the window size. If not specified, 1 is assumed.

4. 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.

5. Label color

Selects the color of the characters displayed. Acceptable numbers are from 0 to 15. If not specified, the characters are displayed in black.

6. 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.

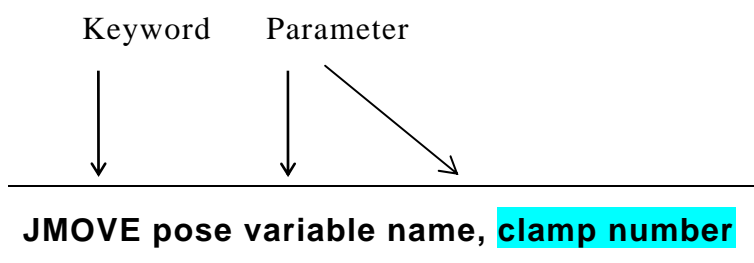
Explanation

IFWPRINT 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.

6.0 PROGRAM INSTRUCTIONS

- 6.1 Motion Instructions
- 6.2 Speed and Accuracy Control Instructions
- 6.3 Clamp Control Instructions
- 6.4 Configuration Instructions
- 6.5 Program Control Instructions
- 6.6 Program Structure Instructions
- 6.7 Binary Signal Instructions
- 6.8 Message Control Instructions
- 6.9 Pose Information Instructions
- 6.10 Program and Data Control Instructions

Example



Parameters marked with 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 position.
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 name,	clamp number
LMOVE	pose variable name,	clamp number

Function

Moves the robot to the specified pose.

JMOVE: Moves in joint interpolated motion.

LMOVE: Moves in linear interpolated motion.

Parameter

1. Pose variable name

Specifies the destination of the robot. (Can be in transformation values, compound transformation values, joint displacement values or pose information functions.)

2. 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 the time to stop the robot motion in seconds.

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, the execution of program steps continue through all the steps before the next motion instruction.

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 the time to stop the robot motion in seconds.

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 name, distance
LAPPRO	pose variable name, distance

Function

Moves to tool Z direction to a specified distance from the taught pose.

JAPPRO: Moves in joint interpolated motion.

LAPPRO: Moves in linear interpolated motion.

Parameter

1. Pose variable names

Specifies the end pose (in transformation values or joint displacement values)

2. 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 posture is set at the posture 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 in the direction of the Z axis of the tool coordinates from the pose “place”, 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 position number
-------------	-----------------------------

Function

Moves in joint interpolated motion to pose defined as HOME or HOME2.

Parameter

Home position number

Specifies the home position number (1 or 2). If omitted, HOME 1 is selected.

Explanation

Two home positions can be set (HOME 1 and HOME 2). This instruction moves the robot to one of the home positions in joint interpolated motion. The home position should be defined beforehand using the SETHOME or SET2HOME commands. If the home position is not defined, the null origin (all joints at 0°) is assumed as the home position.

Example

HOME Moves to the home position defined by SETHOME command in joint interpolated motion.

HOME 2 Moves to the home position defined by SET2HOME command in joint interpolated motion

DRIVE joint number, displacement, speed

Function

Moves a single joint of the robot.

Parameter**1. 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.)

2. 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).

3. 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 -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 axis 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 $\pm 180^\circ$. 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 $\pm 180^\circ$. 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 $\pm 180^\circ$. 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

If the reference motion direction is set along the tool Z direction in each application, DO ALIGN enables easy alignment in teaching operations, of the tool direction to the closest axis of the base coordinates.

HMOVE pose variable name, 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

1. Pose variable

Specifies the destination of the robot motion. (Can be in transformation values, compound transformation values, joint displacement values or pose information functions.)

2. 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 joint moves in joint interpolation. This instruction is used when the robot is to be moved in linear motion but the posture of the wrist joint changes greatly between the beginning and end of the motion.

XMOVE mode pose variable name 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**1. 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.

2. Pose variable name

Specifies the destination of the robot motion. (Can be in transformation values, compound transformation values, joint displacement values or pose information functions.)

3. 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 1064 (the smaller of the two).
Internal signal	2001 to 2256

[**NOTE**]

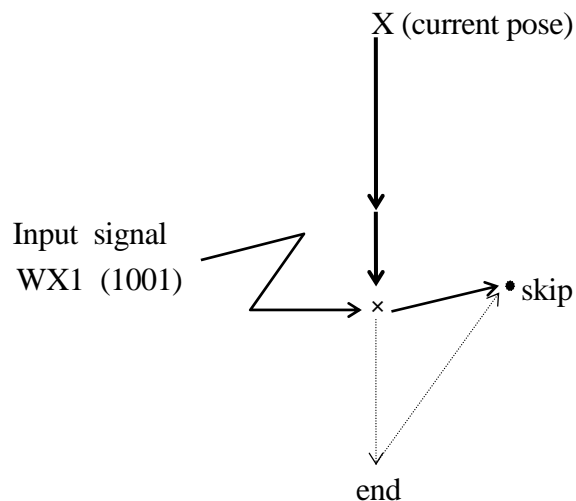
- . When monitoring the rising and trailing edge of the signal, the program branches only when there is a change in the signal status. Therefore, if the rising edge of signal is monitored and the signal is ON at the time XMOVE is executed, the program will not be interrupted until that signal turns OFF 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 name,	clamp number
C2MOVE	pose variable name,	clamp number

Option

Function

Moves the robot to the specified pose following a circular path.

Parameter

1. Pose variable name

Specifies the destination of the robot motion. (Can be in transformation values, compound transformation values, joint displacement values or pose information functions.)

2. 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

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**]

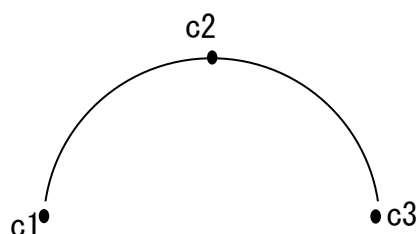
The following motion instructions are needed before the C1MOVE instruction:
ALIGN, C1MOVE, C2MOVE, DELAY, DRAW, TDRAW, DRIVE, HOME,
JMOVE, JAPPRO, JDEPART, LMOVE, LAPPRO, LDEPART, STABLE,
XMOVE

C1MOVE instruction must be followed by C1MOVE or C2MOVE instruction.

C1MOVE instruction must precede a C2MOVE instruction..

Example

```
JMOVE c1
C1MOVE c2
C2MOVE c3
```

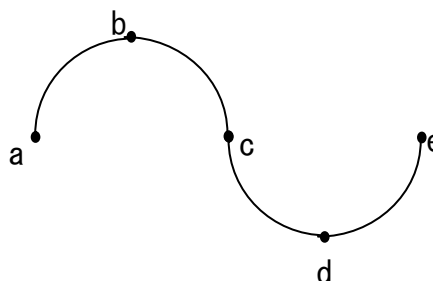


The robot moves in joint interpolated motion to c1 and then moves in a circular interpolated motion following the arc created by c1, c2, c3.

```
JMOVE #a
C1MOVE #b
C2MOVE #c
C1MOVE #d
C2MOVE #e
```

} arc a,b,c

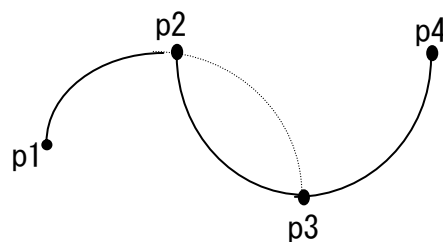
} arc c,d,e



```
LMOVE #p1
C1MOVE #p2
C1MOVE #p3
C2MOVE #p4
```

} arc p1,p2,p3

} arc p2,p3,p4



6.2 SPEED AND ACCURACY CONTROL INSTRUCTIONS

SPEED	Sets the motion speed (program 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)

SPEED speed, rotational speed, ALWAYS

Function

Specifies the robot motion speed.

Parameter**1. Speed**

Specifies the program speed. Usually it is specified in percentages between 0.01 to 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. If the unit is omitted, it is read as percent (%).

2. Rotational speed (Option)

Specifies the rotational speed of the tool posture in linear and circular interpolated motions.

Usually it is specified in percentages between 0.01 to 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 %.

3. 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 monitor speed and the motion speed set by this 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 (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

SPEED 50 Sets the speed of the next motion to 50 %.

SPEED 100 Sets the speed of the next motion to 100 %.

SPEED 200 Sets the speed of the next motion to 100 % (speed over 100 % is considered 100 %).

SPEED 20MM/S ALWAYS The speed of the origin of the tool coordinate (TCP) is set at 20 mm/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 the origin of the tool coordinates when the monitor speed is 100%).

SPEED 5 S Sets the speed of the next robot motion so that the destination is reached in 5 seconds. (The speed of the origin of the tool coordinates when the monitor speed is 100%).

SPEED 100 MM/S, 10 DEG/S Sets the speed of the next motion. Speed setting that requires longer time to reach the destination, takes priority.

ACCURACY distance ALWAYS

Function

Sets the accuracy when determining the robot pose.

Parameter

1. Distance

Specifies the distance of accuracy range in millimeters.

2. 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.

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.

Be careful not to set the accuracy range too small. It 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

1. 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.

2. 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 robots motion speed (block speed). The robot motion speed is calculated by
monitor speed \times program speed \times 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 \times program speed \times 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 $100\% \times 50\% \times \text{block speed}$. Therefore, 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 a program is executed using the EXECUTE command from the teach pendant, the block speed is set at the default value of 100 %. When the program 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 %.

6.3 CLAMP CONTROL INSTRUCTIONS

OPEN	Output clamp open signal when next motion instruction begins.
OPENI	Output clamp open signal when current motion instruction is completed.
CLOSE	Output clamp close signal when next motion instruction begins.
CLOSEI	Output clamp close signal when current motion instruction is completed.
RELAX	Turns OFF clamp signal when next motion instruction begins.
RELAXI	Turns OFF clamp signal when current motion instruction is completed.
OPENS	Output clamp open signal during execution of motion instruction. (Option)
CLOSES	Output clamp close signal during execution of motion instruction. (Option)
RELAXS	Turns OFF clamp signal during execution of motion instruction. (Option)
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 valve for both OPEN and CLOSE (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

Option

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.

These instructions are 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

1. Gun number

Specifies gun number 1 or 2.

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

1. Gun number

Specifies gun number 1 or 2.

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. Always use a variable for this parameter to change the timing outside the program (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

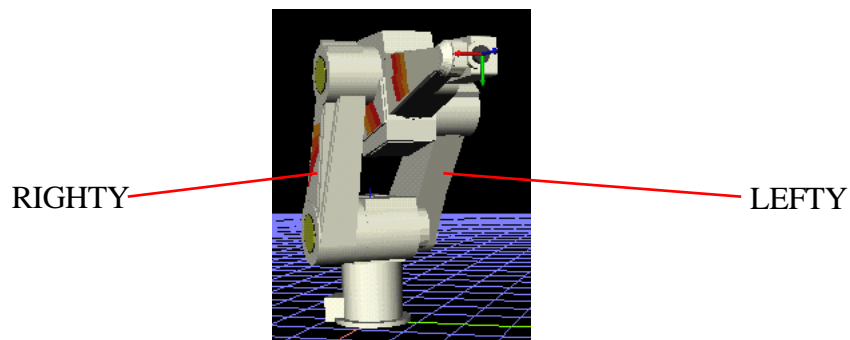
RIGHTY	Changes configuration so the robot arm resembles a person's right arm.
LEFTY	Changes configuration so the robot arm resembles a person's left arm.
ABOVE	Changes configuration so the elbow joint is in the above position.
BELOW	Changes configuration so the elbow joint is in the below position.
UWRIST	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

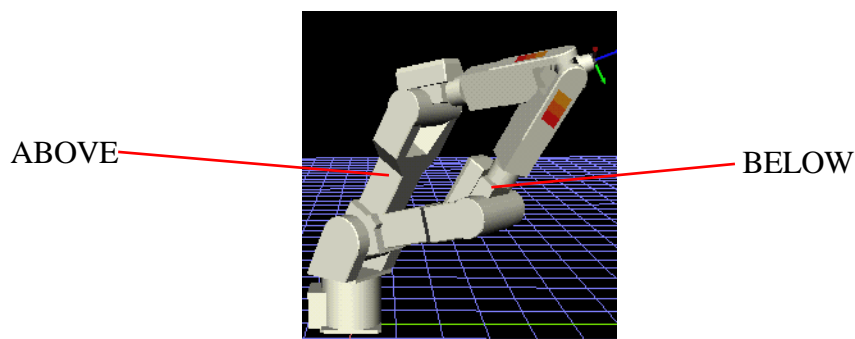


ABOVE
BELOW

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

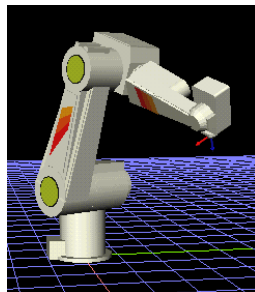


UWRIST **DWRIST**

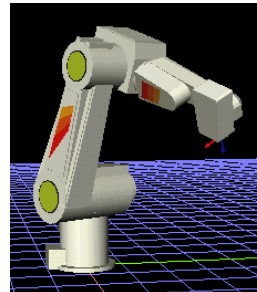
Function

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	Jumps to specified label when condition is set.
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. (Resumption not possible.)
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.
CALLAUX	Displays auxiliary function screen. (Option)

GOTO label IF condition

Function

Jumps to the program step with the specified label.

Parameter**1. Label**

Specifies label of the program step to jump to. The label can be any whole number between 0 and 32767.

2. 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**1. Condition**

Specifies the condition in expressions, e.g. $n = 0$, $n > 3$, $m + n < 0$.

2. 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 whole number 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 the whole 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 $\text{flag} \neq 0$ 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 name 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.

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, to hold the program execution.

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.

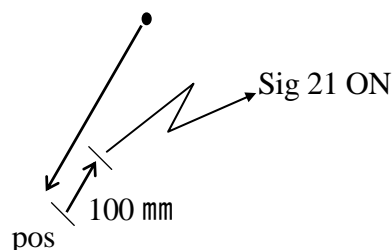
[NOTE]

MVWAIT instruction cannot be used in PC programs. Also, this instruction cannot be used with DO command.

Example

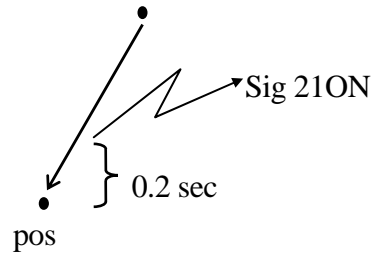
In the diagram below, the robot moves towards pose “pos”, and when coming within 100 mm to “pos”, the signal 21 is turned ON. This is true only when the signal read ahead function (PREFETCH.SIGINS) is ON and the robot is in the accuracy range.

LMOVE lc
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 lc
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

1. String expression

Specifies the subroutine name in the form of a string expression.

2. 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 ($\neq 0$) 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)
```

```
$temp2=""
```

Converts into a string expression, the real value given to "num", and jumps to the subroutine named "sub12".

```
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.

Explanation

This instruction calls the specified program when an error occurs. PC programs can be called too.

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 instructions 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.

CALLAUX auxiliary function number

Option

Function

Displays the specified auxiliary function screen.

Parameter

Auxiliary function number

Specifies the auxiliary function number, between 1 to XXXX.

Explanation

1. Enables displaying and setting of the auxiliary function from a program.
2. If program execution is held while the CALLAUX instruction is being executed, the auxiliary function screen turns OFF. If the CALLAUX instruction is executed from several programs, the CALLAUX instruction waits until the first CALLAUX instruction is terminated or held.
3. If an undefined auxiliary function number is specified in this instruction, Error(P2030) Undefined function number occurs. Specify the number from 1 to XXXX. (Note that not every number is assigned a function).

6.6 PROGRAM STRUCTURE INSTRUCTIONS

IF.....THEN...ELSE.....END

WHILE.....DO.....END

DO.....UNTIL

FOR.....END

CASE.....OF.....VALUE.....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

1. Logical expression

Logical expression or real value expression. Tests if this value is TRUE (not 0) or FALSE(0).

2. Program instructions (1)

The program instructions entered here are executed if the above logical expression is TRUE.

2. 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 %.

```
21    IF n>5 THEN
22          sp=10
23    ELSE
24          sp=20
25    END
26    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.

```
71    IF m THEN
72          IF SIG(1001) THEN
73                PRINT"Input signal is TRUE"
74          ELSE
75                PRINT"Input signal is FALSE"
76          END
77    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

1. Condition

Logical expression or real value expression. Checks if this value is TRUE (not 0) or FALSE (0).

2. 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 empty), 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.

```
20      .  
21      .  
22      .  
23      WHILE SIG(1001,1002) DO  
24          CALL part1  
25          CALL part2  
26      END  
27      .  
28      .  
29      .  
30      .
```

DO
program instructions
UNTIL logical expression

Function

Creates a DO loop.

Parameter

1. Program instructions

These instructions are repeated as long as the logical expression is FALSE.

2. 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 procedure is as follows:

1. Executes the program instructions.
2. Checks the value of the logical expression and if the result is FALSE, step 1 is repeated. If the result is TRUE, it jumps to step 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.

```
10      .  
11      .  
12      .  
13      DO  
14          CALL get.part  
15          CALL put.part  
16      UNTIL SIG(buffer.full)  
17      .  
18      .  
19      .
```

```
FOR  loop variable = start value  TO  end value  STEP  step value  
  program instructions  
END
```

Function

Repeats program execution.

Parameter

1. 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.

2. Starting value

Real value or expression. Sets the first value of the loop variable.

3. 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.

4. Step value

Real value or expression that can be omitted. This value is added or subtracted to the loop variable after each loop. Always enter this command with the STEP statement. If step value is not specified, 1 is added to the loop variable. In this case, the STEP statement can be omitted.

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 procedure is 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 step 7.
 - b. If the step value is negative and the loop variable is smaller than the end value, jump to step 7.

In other cases, goes on to step 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 step 3.
7. Executes the program instructions after the END statement. (The value for the loop variable at the time of the comparison test at step 3 above does not change.)

[NOTE]

There must be an END statement for each FOR statement.

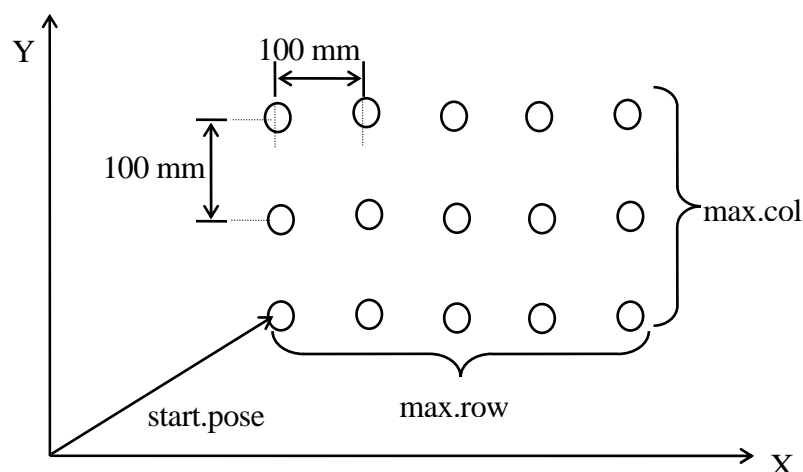
Beware that if the loop variable is greater than the end value (less if the step value is negative) the first time it is checked, none of the program instructions between FOR and END is executed.

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

1. Index variable

Real value variable or expression. Decides which CASE structure to execute according to the value of this variable.

2. 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 find 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
```

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 be equal to the specified value.
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.
SETPLACE	Sets the time to start clamp open control.
CLAMP	Controls open/close of clamp signals.

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	2001–2256

See 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

1. 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	2001–2256

2. 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

1. 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	2001–2256

2. Time

Specifies the time to delay the output of the signal in seconds.

See 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 commands, 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

1. 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	2001–2256

2. 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**1. 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	2001–2256

2. Number of signals

Specifies the number of signals to be set ON/OFF. The maximum number allowed is 16.

3. 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.

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	2001–2256

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

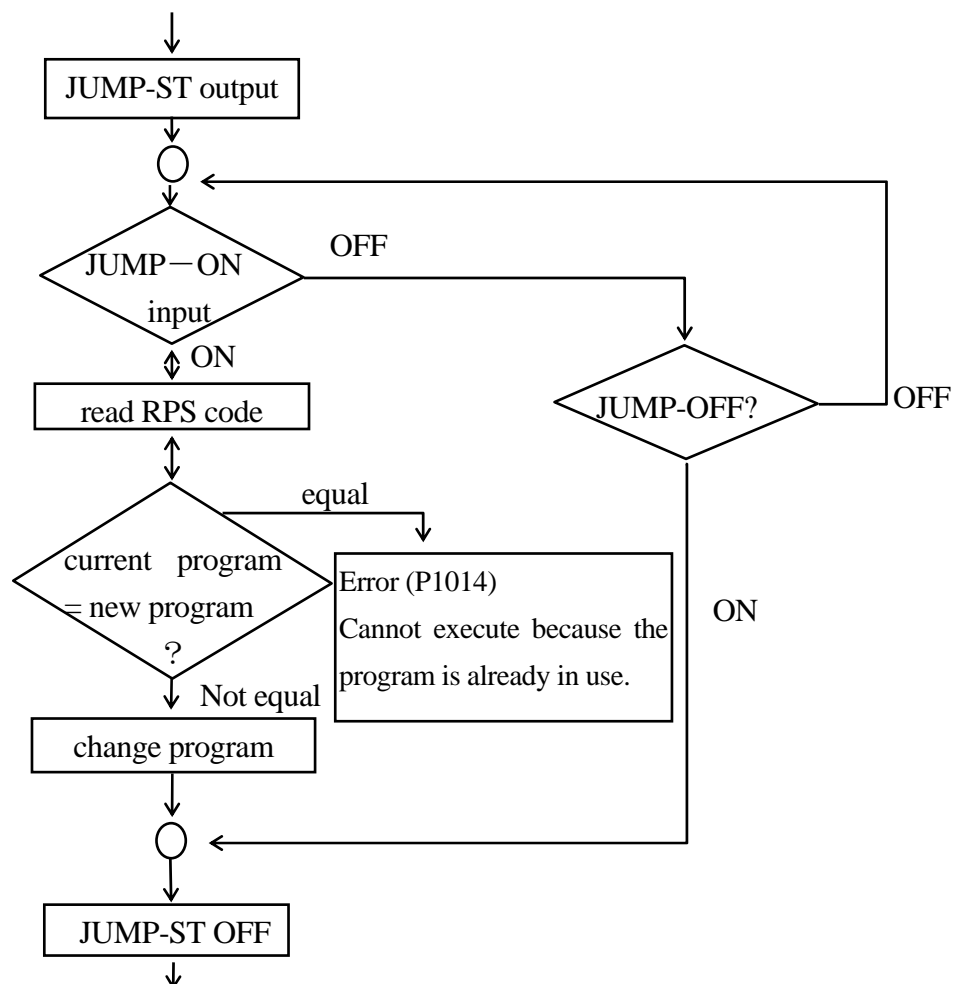
Function

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.



[**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 set 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

1. 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.

2. 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

3. 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.

4. Label

Specifies which label to jump to when the specified signal is input.

4. 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. 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.

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 ~ 2256

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

1. Counter signal number

Specifies the signal number to output. Setting range for counter signal numbers: 3097 to 3128.

2. 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.

3. 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.

4. Counter clear signals

Specified by signal number or logical expressions. If this signal is turned ON, the internal counter is reset to 0.

5. Counter value

When the internal counter reaches this value, the specified 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 count down 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 a counter signal number.

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

1. Signal number

Specifies the number of the signal to flicker. Setting range: 3065 to 3096.

2. 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 also 5.7 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**1. Output signal**

Specifies the number of the signal to output. A positive number turns ON the signal; a negative number turns it OFF. Only the signal numbers of output signals can be specified. (1 to actual number of signals).

2. Set signal expression

Specifies the signal number or logical expression to set the output signal.

3. 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

1. Signal number

Specifies the number of the signal to output. Only the signal numbers of output signals can be specified. (1 to actual number of signals).

2. Signal expressions

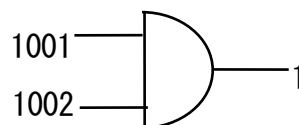
Specifies a signal number or a logical expression.

Explanation

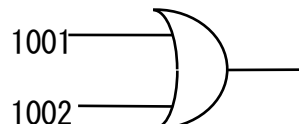
This command is for logical calculation of signals. Logical expressions such as AND and OR are used. The specified signal is output when that condition is set. (See also 5.7 SOUT monitor command.)

Example

SOUT 1 = 1001 AND 1002

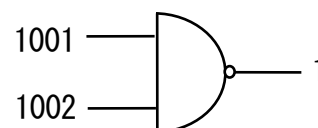


SOUT 1 = 1001 OR 1002

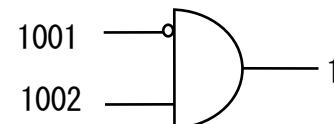


SOUT -1 = 1001 AND 1002

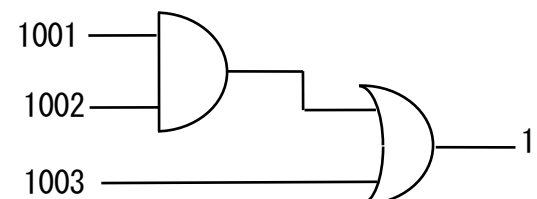
SOUT 1 = NOT(1001 AND 1002)



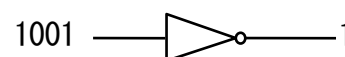
SOUT 1 = -1001 AND 1002



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**1. Timer signal**

Selects the signal number to turn ON. Setting range: 3001 to 3064.

2. Input signal number

Specifies in whole numbers 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.

3. 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

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,

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, 3seconds 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.

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.
IFWPRINT	Displays specified character string in a display window.

PRINT	device number: print data,
TYPE	device number: print data,

Function

Displays on the terminal the print data specified in the parameter.

Parameter

1. 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.

2. 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 also 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

1. 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.

2. Character string

Specifies the characters to display on the terminal.

3. 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 ENTER 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 **ENTER** key or **CTRL** + **C** is pressed, “0” is assigned to real variables, and in case of character string variable, a null string is assigned.

If “2” is entered for device number, the teach pendant screen changes automatically to keyboard screen. Press **NEXT PAGE** to return to the regular 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 ENTER 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, a variety of alphanumeric characters can be input without causing an error.

PROMPT "Enter the number of parts: ", \$input

**IFWPRINT window, 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

1. Window

Selects the window to display the string. Select a window in Auxiliary Function 0509, from 1 to 4 (standard).

2. Row

Specifies the row in the selected window to display the string. Enter from 1 to 16; available rows depend on the window size. If not specified, 1 is assumed.

3. Column

Specifies the column in the selected window to display the string. Enter from 1 to 43, available columns depends on the window size. If not specified, 1 is assumed.

4. Background color

Selects the background color of the selected window. Colors are numbered from 0 to 15. If not specified, the background is white.

5. Label color

Selects the color of the characters displayed. Colors are numbered from 0 to 15. If not specified, the characters are displayed in black.

6. 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

IFWPRINT 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.

6.9 POSE INFORMATION INSTRUCTIONS

HERE	Defines a pose variable as the current pose.
POINT	Defines a pose variable.
POINT/X	Sets the X value of a pose variable.
POINT/Y	Sets the Y value of a pose variable.
POINT/Z	Sets the Z value of a pose variable.
POINT/OAT	Sets the OAT values of a pose variable.
POINT/O	Sets the O value of a pose variable.
POINT/A	Sets the A value of a pose variable.
POINT/T	Sets the T value of a pose variable.
POINT/7	Sets the value of seventh axis of a pose variable.
DECOMPOSE	Assigns the components of pose information to elements of array variable.
TOOL	Defines the pose of TCP.
BASE	Changes the robots base coordinate system.
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.
NCHON	Turns ON notch filter. (Option)
NCHOFF	Turns OFF notch filter. (Option)
WEIGHT	Sets the weight load data.
MC	Executes monitor commands from PC programs.
PLCAOUT	Sets real values to output data. (Option)
TPLIGHT	Turns on teach pendant backlight.

HERE pose variable name

Function

Defines a pose (location) variable name as the current pose. The pose may be expressed in transformation values, joint displacement values or compound transformation values.

Parameter

Pose (location) variable name

Can be specified in transformation values, joint displacement values, or compound transformation values

Explanation

The pose may be expressed in transformation values, joint displacement 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/ X	transformation variable name1=transformation variable name 2
POINT/ Y	transformation variable name1=transformation variable name 2
POINT/ Z	transformation variable name1=transformation variable name 2
POINT/ OAT	transformation variable name1=transformation variable name 2
POINT/ O	transformation variable name1=transformation variable name 2
POINT/ A	transformation variable name1=transformation variable name 2
POINT/ T	transformation variable name1=transformation variable name 2
POINT/ 7	transformation variable name1=transformation variable name 2

Function

Assigns the components of the transformation values specified on the right of “=” to the corresponding component of the transformation values on the left of “=”. The values will be displayed on the terminal for correction.

Parameter

1. Transformation variable name 1

Selects the transformation variable name to be defined (a single transformation variable or compound transformation values with transformation variable as the rightmost value)

2. Transformation variable name 2

Specifies the transformation variable to be used to define the component of the variable name on the left side of “=”. The following may be specified: a previously defined transformation value variable or compound transformation variables, or a transformation value function.

Explanation

Error occurs if any pose information on the right side of the “=” is not defined.

If compound transformation values are specified on the left side of the “=”, this instruction defines only the rightmost value in that compound value. However, error occurs if any variable other than the rightmost variable in the compound transformation value is undefined.

Example

POINT/X temp=tempx	Assigns the x component of the transformation values “tempx” to the x component of transformation values “temp”.
--------------------	--

DECOMPOSE array variable name[element number]=pose variable name

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

1. Array variable name

Specifies the name of the array variable into which the values of each component will be stored.

2. Element number

Specifies the first element in which to store the components.

3. Pose variable name

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 joints in the robot arm.

Example

DECOMPOSE X[0]=part Assigns the components of transformation values “part” to the first six elements in the array variable “x”.

DECOMPOSE angles[4]=#pick Assigns the components of joint displacement values “#pick” to element number 4 and the following elements in array variable “angles”.

For example, in the above instruction, if the values of #pick are (10,20,30,40,50,60) then,

angle[4]=10	angle[7]=40
angle[5]=20	angle[8]=50
angle[6]=30	angle[9]=60

BASE transformation values

Function

Defines the base transformation values.

Parameter

Transformation values or compound transformation values

Defines the new base coordinates. The transformation values here describe 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 values

Function

Defines the tool transformation values.

Parameter

Transformation values (Compound transformation values)

Defines the new tool coordinates. The transformation values here describe the pose of the tool coordinates with respect to the null tool coordinates, expressed in null tool coordinates.

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.

ULIMIT	joint displacement values
LLIMIT	joint displacement values

Function

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

Parameter

Joint displacement values

Sets the software limit (upper or lower) in joint displacement values.

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

1. Timer number

Specifies the number of the timer to set the time. Acceptable numbers are 1 to 10.

2. 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

1. @Timer variable

Specified the variable or array variable name to use as the timer name. Enter @ in front of a whole number variable.

2. 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

Turns ON (OFF) the switch specified here. More than one switch name can be entered separating each switch name by commas.

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

See also 5.6 ON/OFF monitor commands.

NCHON
NCHOFF

Option

Function

Turns ON (or OFF) the notch filter in the motion steps following the specified step. NCHON enables the notch filter, NCHOFF disables it.

Example

10	SPEED 100 ALWAYS	
20	HOME	
30	DRIVE5,90	
40	NCHOFF	} This linear movement is done with the notch filter OFF.
50	SPEED 10	
60	DRAW 1000	
70	NCHON	
80	DRAW,-500	

[**NOTE**]

Normally use with NCHON.

NCHOFF is used when small vibration can be seen in low speed operations, or when an external force is being applied to the robot.

The filter is reset with the execution of EXECUTE command. It is also reset when PRIME, STEP, MSTEP commands are first executed. The default setting is ON (notch filter enabled).

WEIGHT	load mass, center of gravity location X, center of gravity location Y, center of gravity location 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 work). The data is used to determine the optimal acceleration/deceleration of the robot arm.

Parameter

1. Load mass

The mass of the tool and work (in kilograms). Range: 0.0 to the maximum load capacity (kg).

2. Center of gravity location (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

3. Inertia moment about X axis, inertia moment about Y axis, inertia moment about Z axis

(Option)

Sets the inertia moment around each axes. Unit is $\text{kg}\cdot\text{m}^2$. 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

When using WEIGHT as a program instruction, if the parameters are not specified, the setting defaults to the maximum load capacity for that robot model.



DANGER

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.

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
```

PLCAOUT data number = real value

Option

Function

Sets the given real number value to the specified data number.

Parameter

1. Data number

Specifies the output data number in whole numbers. Acceptable number is from 1 to 32.

2. Real value

Specifies the value that is to be set to the output data, entered in decimal notation. It can also be set in variable names. Acceptable number is from 0 to 65535.

[NOTE]

This function is only valid when the “Built-in Sequencer Function” option is ON. If the option is OFF, the following error message appears (E1102) Cannot execute, no option set up. - Check option specs.

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.

6.10 PROGRAM AND DATA CONTROL INSTRUCTIONS

DELETE	Deletes programs and variables in robot memory. (Option)
DELETE/P	Deletes programs in robot memory. (Option)
DELETE/L	Deletes pose variables in robot memory. (Option)
DELETE/R	Deletes real variables in robot memory. (Option)
DELETE/S	Deletes string variables in robot memory. (Option)
TRACE	Turns ON/OFF the TRACE function.

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

Option

Function

Deletes the specified data from the memory.

Parameters

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

The name of the data to delete.

See also 5.2 DELETE monitor commands.

TRACE **stepper number**:ON/OFF

Function

Logs and traces the actual data from the program execution.

Parameters

1. Stepper number

Type of program specified using the following whole number values:

1: Robot program

1001: PC program 1

1004: PC program 4

1002: PC program 2

1005: PC program5

1003: PC program 3

If the type is not specified, all the programs are logged.

2. ON/OFF

Starts/Ends tracing.

Explanation

If the necessary memory is not saved using the SETTRACE command before TRACE ON, the error (P2034) “Memory undefined for logging” occurs.

See also 5.2 TRACE/SETTRACE monitor commands.

7.0 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 command/ program instruction.

CP	Enables or disables continuous path (CP) function.
CHECK.HOLD	Enables or disables input of commands from the keyboard when <u>HOLD/RUN</u> is in HOLD position.
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 control power ON. (Option)
ERRSTART.PC	Determines if the selected PC program is executed when an error occurs.
TRIGGER	Displays the ON/OFF status of <u>TRIGGER</u> switch.
CS	Displays the ON/ OFF status of <u>CYCLE START</u> .
POWER	Displays the ON/ OFF status of <u>MOTOR POWER</u> .
RGSO	Displays the ON/OFF status of servoing.
TEACH_LOCK	Displays the ON/OFF status of <u>TEACH LOCK</u> .
ERROR	Displays if the program is in error status or not.
REPEAT	Displays the status of <u>TEACH/REPEAT</u> switch.
RUN	Displays the status of <u>HOLD/RUN</u> switch.
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. (Option)
SLOW_START	Enables or disables the slow start function. (Option)
AFTER.WAIT.TMR	Sets how timers begin in block step programs. (Option)

CP

Function

Enables or disables continuous path (CP) function.

Explanation

This switch is used to turn ON (enable) or OFF (disable) the CP function. If 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 switch is in the HOLD position.

Explanation

If this switch is ON, the following commands entered from the keyboard are accepted only when the HOLD/RUN switch is in HOLD position. (The commands are accepted, but the motion does not start unless the HOLD/RUN switch is turned to RUN.)

EXECUTE, DO, STEP, MSTEP, CONTINUE

If this switch is OFF, the commands above are accepted regardless of the position of the HOLD/RUN switch. Operations not done by keyboard are not affected by this command. (If the HOLD/RUN switch is in RUN position, 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 execution cycle after an HOLD is applied.

Explanation

If this switch is OFF, the robot stops when the external HOLD signal is input, but the **CYCLE START** remains ON. Therefore, when the external HOLD signal is turned OFF, the robot resumes program execution.

If this switch is ON, the robot stops when the external HOLD signal is input, and the **CYCLE START** is turned OFF. Therefore, even if the external HOLD is turned OFF, the robot does not resume program execution. To resume program execution, follow regular program execution procedures (i.e. press **CYCLE START** button).

This stop has effect only on external HOLD and not the **HOLD/RUN** switch on the operation panel. If the **HOLD/RUN** switch on the operation panel is turned to HOLD, robot stops, but **CYCLE START** remains ON, regardless of the condition of this system switch. The robot resumes program execution when the **HOLD/RUN** switch is turned to RUN.

Default setting is OFF.

MESSAGES

Function

Enables or disables the output of message on the terminal.

Explanation

If this switch is ON, the messages are displayed on the terminal when the PRINT or TYPE command is used. If 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.

If 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. SIGNS

Function

Enables or disables early processing of signal input and output commands. Functions identical to OX.PREOUT but is effective only on AS programs.

Explanation

If 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

If this switch is ON, the commands above is executed as soon as the movement towards a taught point starts and the program is processed up to the step before the next motion instruction.

Default setting is OFF.

QTOOL

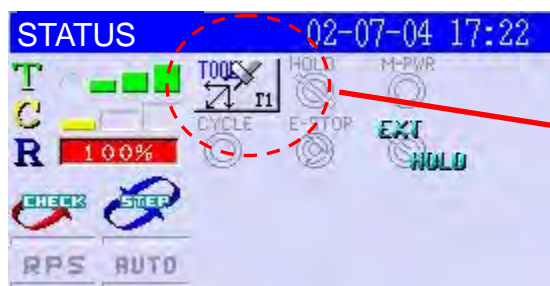
Function

Enables or disables tool transformation during block teaching.

Explanation

If this switch is ON, the following functions effect during teaching:

1. Automatic selection of the tool coordinates.
 - (1) If auxiliary data for a tool is taught at the current step, the tool transformation values registered as tool number 1 to 9.
 - (2) If an AS language command is taught at the current step, the tool coordinates remain unchanged.
 - (3) If nothing is taught yet in the current step, and the teach pendant is displaying the auxiliary data screen, the robot is jogged according to the tool coordinates recorded in the auxiliary data of the tool number (1 - 9) currently shown on screen.
 - (4) If nothing is taught yet in the current step and the teach pendant is not at the auxiliary data screen, the tool coordinates remain unchanged.
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”.

If this switch is turned OFF, and the tool taught via the AS language is used, the tool number is displayed as “T 0”. If teaching is always done via AS language, keep this switch OFF.

If this switch is OFF, the tool coordinates change when the TOOL command is executed, or when the TOOL instruction or a block instruction is executed within a program.

Default setting is ON.

RPS

Function

Enables or disables the random selection of programs.

Explanation

If this switch is OFF, the EXTCALL instruction and auxiliary data JUMP/ END 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

If this switch is ON, the program runs one time.

If 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

If this switch is ON, the program steps are run one step at a time.

If 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

Option

Function

Determines if the selected PC program starts automatically when the control power is turned ON.

Explanation

If this switch is ON, the PC program named AUTOSTART.PC starts automatically when the control power is turned ON. Five different PC programs can be selected to start automatically, each named AUTOSTART.PC and AUTOSTART2.PC to AUTOSTART5.PC. Turn ON each corresponding switch to start the desired program.

Default setting is OFF.

[NOTE]

If 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 when an error occurs.

Explanation

If this switch is ON, the PC program named ERRSTART.PC is automatically executed when an error occurs.

Once the ERRSTART.PC program is executed, this switch is turned OFF automatically.

Beware that ERRSTART.PC program is executed as the fifth PC program, so if five PC program are already running, ERRSTART.PC cannot be executed.

To reset automatic execution of the ERRSTART.PC program for the next error, be sure to set this switch to ON again.

Default setting is OFF.

[NOTE]

If 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.

Used with the SWITCH function, -1 is returned if the **TRIGGER** switch is ON, and 0 if it is OFF.

SWITCH (CS)

Function

Displays if **CYCLE START** button is ON or OFF.

This function does not turn ON/OFF the switch.

Used with the SWITCH function, -1 is returned if the **CYCLE START** is ON, and 0 if it is OFF.

SWITCH (POWER)

Function

Displays if **MOTOR POWER** switch is ON or OFF.

This function does not turn ON/OFF the switch.

Used with the SWITCH function, -1 is returned if **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.

Used with the 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.

Used with the 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.

Used with the SWITCH function, -1 is returned if error is occurring, and 0 if not occurring.

SWITCH (REPEAT)

Function

Displays if the TEACH /REPEAT switch on the control panel is in TEACH position or in REPEAT position.

This function does not turn ON/OFF the switch.

Used with the 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 the RUN /HOLD switch on the control panel is in RUN position or in HOLD position.

This function does not turn ON/OFF the switch.

Used with the SWITCH function, -1 is returned if the switch is in RUN position, and 0 if it is in HOLD position.

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   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
>
```

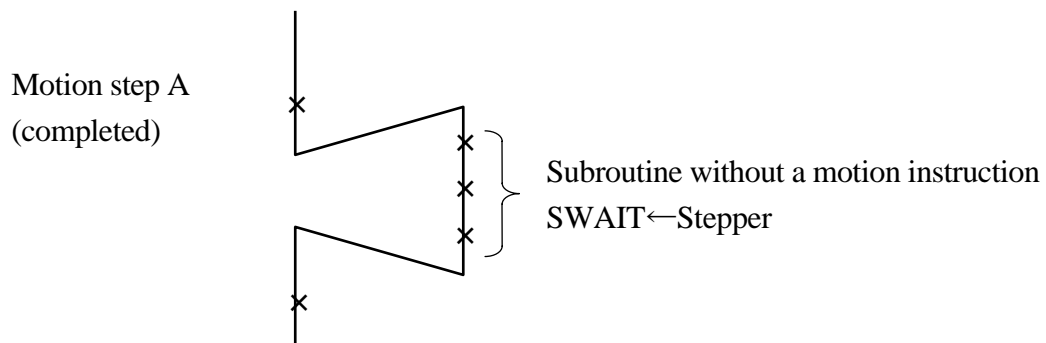
HOLD. STEP

Function

Selects the step to display when the program execution is held.

Explanation

If 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

If this switch is ON, the timing of the WS signal output is changed from the moment memory change occurs to the moment when input of the welding complete signal is received.

If this switch is OFF, the WS signal is output at each memory change.

Default setting is OFF.

FLOWRATE

Option

Function

Switches between flow rate control mode to speed output mode.

Explanation

If this switch is ON, the flow rate control mode is selected.

If 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

If this switch is ON, the welding is done when WS=0, as well as when WS≠0. (Pressurizing and welding)

If 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, effective for the entire program and taking precedence over the monitor speed setting.

Explanation

If this switch is ON, the robot moves at the absolute speed specified for the program when the below condition is true.

$$\text{Maximum speed} \times \text{Monitor Speed} > \text{Program Speed}$$

Example

If Max speed 2400 mm/s, Monitor Speed 10%, Program Speed 100 mm/s, then

$$2400 \times 0.1 > 100$$

Therefore the robot moves at the program speed 100 mm/s.

If Max speed 100%, Monitor Speed 10%, Program Speed 5%, then

$$100 \times 0.1 > 5$$

Therefore the robot moves at the program speed 5%.

If Max speed 2400 mm/s, Monitor Speed 2%, Program Speed 100 mm/s, then

$$2400 \times 0.02 < 100$$

Therefore the robot moves at the monitor speed 48 mm/s.

SLOW_START

Function

Enables (ON) or disables (OFF) the low start function. If the slow start function is enabled, the first motion instruction is executed in slow repeat speed.

Explanation

If this switch is turned ON, the slow start function is enabled.

If a program is stopped and then restarted, the first motion step to be executed will start at the slow repeat speed.

AFTER.WAIT. TIMR

Function

Sets the timing for starting timers in block instructions.

Explanation

If this switch is OFF, the timer starts when the axes coincide. If it is ON, the timer starts when the axis coincides and all the conditions set (e.g. WX, WAIT, RPS ON) are fulfilled.

Default setting is OFF.



MEMO

8.0 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

8.2 Relational Operators

8.3 Logical Operators

8.4 Binary Operators

8.5 Transformation Value Operators

8.6 String Operators

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	$i = i \text{ MOD } 2$
^	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 $i + 1$.

$i = i \text{ MOD } 2$ When i is 5, operator calculates $5 \div 2$ and assigns to i the remainder of 1.

$i = i ^ 3$ 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 $i = i / 0$
 $i = i \text{ MOD } 0$

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	$i \leq j$
=<	Same as above	$i \leq j$
>=	TRUE (-1) when left side value is greater than or equal to right side	$i \geq j$
=>	Same as above	$i \geq j$
==	TRUE (-1) when the two sides are equal	$i == j$
<>	TRUE (-1) when the two sides are not equal	$i \neq j$

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 $t == 5$

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.

F $i + j > 100$ 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 $\$a == \text{"abc"}$ GOTO 20

When $\$a$ is "abc" (i.e. $\$a == \text{"abc"}$ is true), the program jumps to the step labeled 20. If not, the program proceeds to the next step.

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 $0 \times 1=0$, $1 \times 1=1$, $0 \times 0=0$ (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). However, 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.

$$\begin{array}{r}
 i=5 \quad 0101 \\
 j=9 \quad 1001 \\
 \hline
 1101 \Rightarrow 13
 \end{array}$$

i BAND j If i=5, j=9, then the result is 1.

$$\begin{array}{r}
 i=5 \quad 0101 \\
 j=9 \quad 1001 \\
 \hline
 0001 \Rightarrow 1
 \end{array}$$

i BXOR j If i=5, j=9, then the result is 12.

$$\begin{array}{r}
 i=5 \quad 0101 \\
 j=9 \quad 1001 \\
 \hline
 1100 \Rightarrow 12
 \end{array}$$

COM i If i=5 then the result is -6.

$$i=5 \text{ 0... } \underline{0101}$$

8.5 TRANSFORMATION VALUE OPERATORS

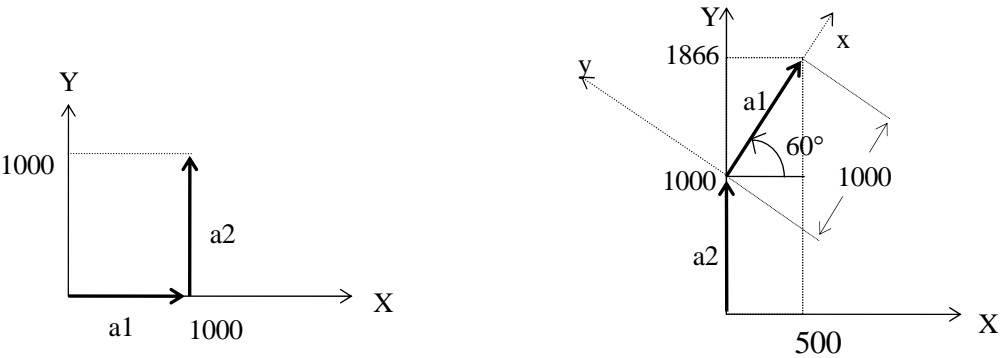
In the AS system, operators + and − are used to determine the compound transformation values (the XYZOAT values). However note that unlike the usual addition or subtraction, the commutative law does not hold true with the transformation operation. Arithmetic expression “a + b” and “b + a” will result the same, but “pose a + pose b” will not necessarily equal “pose b + pose a”. This is because in transformation operations, the postures of the axes are taken into consideration. An example of this is shown below:

$$a1 = (1000, \quad 0, \quad 0, \quad 0, \quad 0, \quad 0)$$

$$a2 = (\quad 0, 1000, \quad 0, \quad 60, \quad 0, \quad 0)$$

$$a1+a2 = (1000, 1000, \quad 0, 60, \quad 0, \quad 0)$$

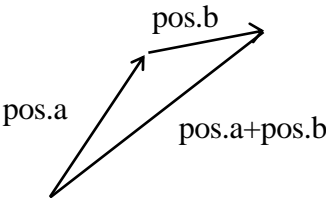
$$a2+a1 = (500, 1866, \quad 0, 60, \quad 0, \quad 0)$$



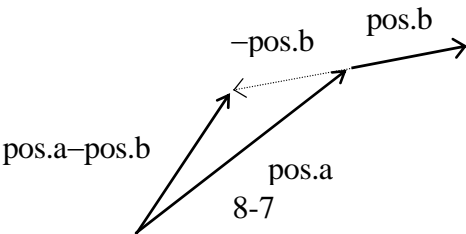
Operator	Function	Example
+	Addition of two transformation values	pos.a+pos.b
−	Subtraction of two transformation values	pos.a−pos.b

Example

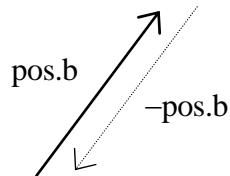
pos.a+pos.b



pos.a−pos.b



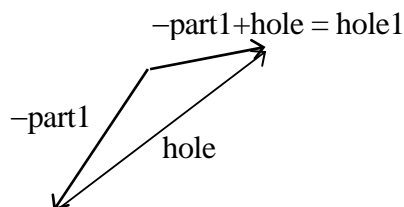
Transformation operator “-” used with a single value (e.g. -pos.b) signifies the inverse value of pos.b. For example, when the transformation value pos.b defines the pose of object B relative to object A, then -pos.b defines the pose of object A relative to object B.



In the example below, “hole1” is to be defined relative to “part1” (defined in advance). This can be done using the compound transformation value part1+hole.

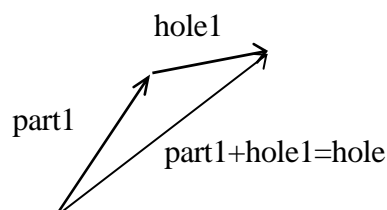
Jog the robot to the pose to be defined “hole1” and teach that pose as “hole” using the HERE command. Using this pose (“hole”), “hole1” can be defined.

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.

When \$b="abc", \$c="123" then \$a is "abc123".



MEMO

9.0 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.

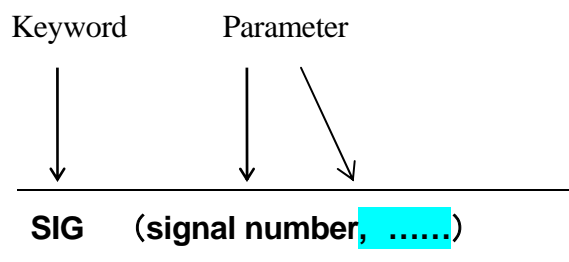
9.1 Real Value Functions

9.2 Pose Value Functions

9.3 Mathematical Functions

9.4 String Functions

EXAMPLE



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.
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. (Option)
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.
SWITCH	Returns the status of system switch.
TASK	Returns the status of program execution.
ERROR	Returns the error number.
PLCAIN	Returns the data number. (Option)
PRIORITY	Returns the priority of robot program. (Option)
UTIMER	Returns the current timer value.
MSPEED	Returns monitor speed of Robot 1. (Option)
MSPEED2	Returns monitor speed of Robot 2. (Option)
INRANGE	Checks if pose is in motion range. (Option)
SYSDATA	Returns AS parameters. (Option)

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	2001–2256

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. 2ms 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

1. Starting signal number

Specifies the first signal to read.

2. Number of signals

Specifies the number of signals to read. The maximum number accepted is 16.

Explanation

This function returns the decimal value corresponding to the bit pattern of the signals specified. 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	2001–2256

[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. 2ms 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 bit pattern 0101 or 5 in decimal notation.

Signals:	1008	1007	<u>1006</u>	<u>1005</u>	<u>1004</u>	<u>1003</u>	1002	1001
State:	1	1	0	1	0	1	0	0

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 to 10.

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, transformation value)

Function

Calculates the distance between two poses that are expressed in transformation values.

Parameter

Transformation value

Specifies names of the two transformation values that define the distance 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)

DY (transformation value)

DZ (transformation value)

Function

Returns the transformation values (X, Y, Z) of the specified pose.

Parameter

Transformation value

Specifies the name of the pose whose X, Y, or Z component is required.

Explanation

These three functions return 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 value 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 name, element number)

Option

Function

Returns the specified element of the specified pose.

Parameter

1. Pose variable name

Specifies the pose in joint displacement values or transformation values.

2. 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” is 0, 0, 0, -160, 0, 0, 300, then inputting this function as:

```
type  DEXT(aa, 7)
```

returns 300 as the value of JT7 of the variable “aa”.

ASC (string, character number)

Function

Returns the ASCII value of the specified character in a string expression.

Parameter

1. 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.

2. 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 result of functions TRUE and ON is 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 result of functions FALSE and OFF is 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

1. 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.

2. String 1

Expressed in character string, character string variable, or string expression, specifies the string where string 2 is searched.

3. 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 (1 if not specified) is returned.

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

Explanation

Returns the integer on left side of the decimal point (with values 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.

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.

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.

Example

TYPE \$ERROR(ERROR) TYPE instruction displays the error message of the error code returned by the function ERROR.

PLCAIN (data number)

Option

Function

Returns the value of the specified data number in whole numbers.

Parameter

1. Data number

Specifies the number of input data in whole number. Acceptable number is from 1 to 32.

[NOTE]

This function is valid only when the “Built-in Sequencer Function” option is ON. If the option is OFF, the following error message appears (E1102) Cannot execute, no option set up. - Check option specs.

Example

aa=PLCAIN(12) Returns the value of the 12th input data in whole numbers.

PRIORITY

Option

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 name)

Function

Returns the current value of the @timer variable set by UTIMER instruction.

Parameter

@timer variable name

Specifies the timer variable name set by UTIMER instruction. An @ sign is added to the beginning of the variable name so that a whole number variable name can be specified.

M SPEED M SPEED2

Option

Function

Returns the current monitor speed (0 to 100 %).

MSPEED is for Robot 1 and MSPEED2, for Robot 2.

INRANGE (pose variable, joint displacement values)

Option

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

1. Pose variable

Specifies which pose to check. (Joint displacement values, transformation values, or compound transformation values).

2. Joint displacement values

This parameter is entered only when the specified pose is in transformation values or compound transformation values. The robot configuration is calculated by the given 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 pos1 is 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)

Option

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.

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 position. (Option)
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 RUN/HOLD switch on the operation panel is turned to HOLD, the robot stops immediately, but the pose returned by DEST/ #DEST functions describe the pose the robot was heading for at that moment.

Example

DEST/ #DEST functions are convenient when resuming the motion interrupted by ONI...CALL instruction. Include the following instructions as shown in the subroutine below so that the robot can return to the path motion that it left.

POINT save=HERE	Stores the current pose as “save”.
POINT old=DEST	Stores the destination pose as “old”.
JDEPART 50.0	Backs 50 mm. (Joint move)
:	
:	
JAPPRO save, 50.0	Approaches the pose “save” by 50 mm above the pose. (Joint move)
LMOVE save	Move to pose “save”. (Linear move)
LMOVE old	Move to pose “old”. (Linear move)

**FRAME(transformation values 1, transformation values 2,
transformation values 3, transformation values 4)**

Function

Returns the transformation values of the frame (relative) coordinates with respect to the base coordinates. Note that only the translational components are used as positional information to determine the frame coordinates.

Parameter

1. Transformation values1, transformation values 2

Specifies 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 transformation values 1 to transformation values2.

2. Transformation values 3

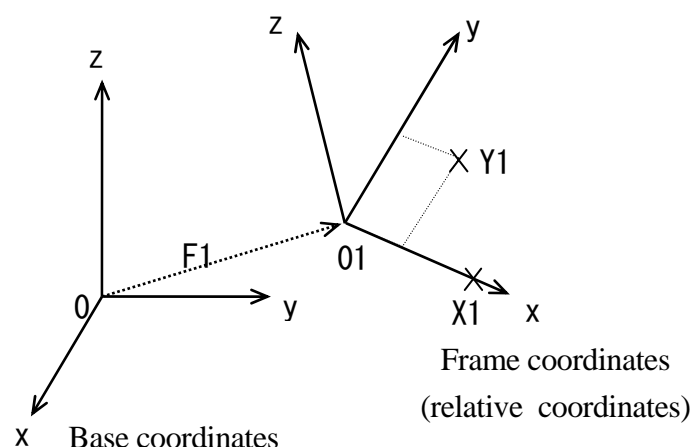
Specifies the direction of the Y axis. The Y axis of the frame coordinates is set so that the three points, transformation values1, transformation values 2, and transformation values 3, are on the XY plane and transformation values 3 on the positive Y axis.

3. Transformation values 4

Specifies the origin of the frame coordinates, which equals the values returned by this function.

Explanation

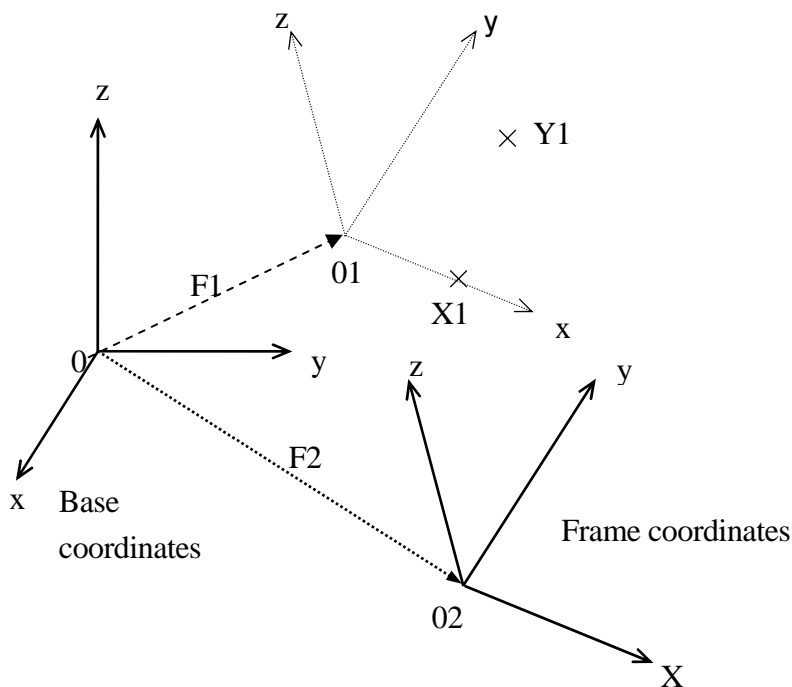
POINT F1=FRAME(O1, X1, Y1, O1) Sets frame coordinates as in the diagram below.



If the poses in the frame coordinates are taught as F1+A, then only F1 needs reteaching 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)
set.

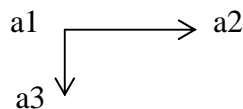
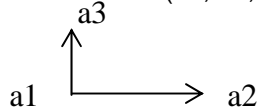
Sets frame coordinates as in the diagram below is



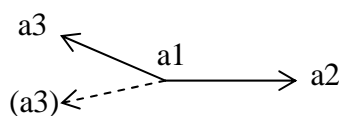
[NOTE]

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 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

1. X component, Y component, Z component

Specifies the translation components X, Y, Z. If not specified 0 is assumed.

2. 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 value 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	}	Specifies the value of each angle (in degrees for rotational joints)
jt2		
jt3		
:		
jt6		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 $-a/2^\circ$.

SHIFT(transformation values 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 pose described by the parameter “transformation values”.

Parameter

1. Transformation values

Specifies the pose to shift in transformation values.

2. X shift, Y shift, Z shift

Specifies the values to add to each of the components of the transformation specified above. 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 given transformation values. The result is returned in transformation values.

Example

If the transformation values of “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 variable “y”.

AVE_TRANS(transformation values 1, transformation values 2)

Function

Returns the average values of the two transformation values.

Parameter

Transformation values 1, transformation values 2

Explanation

This function calculates the transformation values of the pose which defines the midpoint for each of the components of transformation values 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 values)

Option

Function

Returns the sum of the traverse axis value and the X component of the transformation values.

Parameter

Transformation values

Specifies the transformation values to add to the traverse axis.

TRSUB(transformation values)

Option

Function

Returns the value gained by subtracting traverse axis value from the X component of the transformation values.

Parameter

Transformation values

Specifies the transformation values that are subtracted by the traverse axis.

#HOME (home position number)

Option

Function

Returns the currently set home position.

Parameter

Home position number

Specifies the home position number.

1: Specifies home position 1 set by SETHOME command.

2: Specifies home position 2 set by SET2HOME command.

If not specified, 1 is selected.

Explanation

Returns the pose of the currently set home position 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 position (in the direction of the Z axis only), and then it moves to the home position.

```
POINT homepos = #HOME(1)
IF DZ(homepos) > DZ(HERE) THEN
  HERE tmp
  POINT/Z tmp = homepos
  LMOVE tmp
END
HOME
```

**CCENTER(transformation values 1, transformation values 2,
transformation values 3, configuration transformation values)**

Option

Function

Returns the center of the arc created by the three specified pose.

Parameter

1. Transformation values1, transformation values2, transformation values3

Specifies the three points on the arc.

2. Configuration transformation values

Specifies the transformation values to determine the configuration of the robot.

**CSHIFT(transformation values 1, transformation values 2,
transformation values 3, object transformation values BY shift amount)**

Option

Function

Returns the pose that was shifted the specified amount from the object pose. The robot shifts towards the center of the circle created by the three poses specified.

Parameter

1. Transformation values1, transformation values2, transformation values3

Specifies the three points on the arc.

2. Object transformation values

Specifies the object pose in transformation values.

3. 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 $\pi(3.1415\cdots)$.	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 $2\pi r$ 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 $\tan^{-1}(\text{rise/run})$ 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.

\$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.

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. 0 or 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

1. String

Character string, string variable, or string expression.

2. 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.

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

1. String

Character string, string variable, or string expression.

2. 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.

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

1. String

Character string, string variable, or string expression.

2. Real values (or numeric expression)

Specifies the starting position of the string is to be taken.

3. 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**1. String variable**

Specifies the string from where the characters are taken. Characters extracted as a result of this function are removed from this string.

2. Separator character

Specifies the character to read as separator. (Any character in the string can be specified as separator).

3. 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 exists 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 "" (or 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

1. 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. -1.234 E+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 six or more, and smaller than 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 is greater than 0.01 and can be expressed in Fm.n format within “m” digits, 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 July 10, 2002, then the value returned is 07/10/2002).

\$DATE(2) dd/mmm/yyyy

(If the date is July 10, 2002, then the value returned is 10/JUL/2002). 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 July 10,2002, then the value returned is 2002/07/10)

\$TIME

Function

Returns the system time in following string format:

hh:mm:ss

Example

18: 27: 50

The hour is expressed in 24 hours.

10.0 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)
PCEND	Stops execution of specified PC program. (M, P)
PCCONTINUE	Resumes execution of PC program. (M)
PCKILL	Initializes the PC program execution stack. (M)
PCSTEP	Executes a single step of a PC program. (M)
PCSCAN	Specifies PC program processing time. (Option) (P)

Example

Keyword	Parameter
↓	↓
<hr/>	
PCSTATUS	program number
<hr/>	

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 status:	Program is not running
	Execution cycles:	
(2)	Completed cycles:	11
(3)	Remaining cycles:	Infinite
(4)	Program name	Prio 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

(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

1. 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.

2. 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.

3. 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.

4. 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

1. PC program number

Selects the number of the PC program to end. Acceptable range: 1 to 5. If not specified, 1 is assumed.

2. 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, RETURN instruction (or similar instruction) is executed, regardless of remaining cycles. The remaining cycles can be executed using the 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 a robot control program.

PCCONTINUE	PC program number	NEXT
-------------------	--------------------------	-------------

Function

Resumes execution of a suspended PC program. Or, skips the WAIT instruction in the PC program. (M)

Parameter

1. PC program number

Selects the number of the PC program to resume execution. Acceptable range: 1 to 5. If not specified, 1 is assumed.

2. 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. This command with the parameter NEXT 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

1. 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.

2. 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.

3. Execution cycle

Specifies how many times the program step is to be executed. If not specified, 1 is assumed.

4. 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
---------------	-------------

Option

Function

Sets the cycle time for executing the PC program. (P)

Parameter

Time

Sets the 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.

APPENDIX 1 ERROR CODES

Code	Former Code	Error Message
P0100	-350	Illegal input data.
P0101	-351	Too many arguments.
P0102	-353	Input data is too big.
P0103	-361	Illegal PC number.
P0104	-365	Illegal Robot number.
P0105	-366	Illegal program.
P0106	-367	Illegal priority.
P0107	-368	Invalid coordinate value.
P0108	-400	Syntax error.
P0109	-401	Invalid statement.
P0110	-402	Ambiguous statement.
P0111	-403	Cannot use this command or instruction here.
P0112	-404	Cannot execute with DO command.
P0113	-406	Not a program instruction.
P0114	-410	Illegal expression.
P0115	-411	Illegal function.
P0116	-412	Illegal argument of function.
P0117	-413	Invalid variable (or program) name.
P0118	-414	Illegal variable type.
P0119	-415	Illegal array index.
P0120	-416	Missing parenthesis.
P0121	-417	Expected to be a binary operator.
P0122	-418	Illegal constant.
P0123	-419	Illegal qualifier.
P0124	-420	Invalid label.
P0125	-422	Missing expected character.
P0126	-423	Illegal switch name.
P0127	-424	Ambiguous switch name.
P0128	-425	Illegal format specifier.
P0129	-426	Duplicate statement label.
P0130	-430	Cannot define as array.
P0131	-431	Dimension exceeds 3.
P0132	-433	Array variable exist.
P0133	-434	None array variable exist.
P0134	-435	Array variable expected.
P0135	-440	Local variable expected.
P0136	-441	Unexpected suffix.
P0137	-442	Mismatch of arguments at subroutine call.
P0138	-443	Mismatch of argument type at subroutine call.
P0139	-450	Illegal control structure.

Code	Former Code	Error Message
P0140	-451	Step:XX Wrong END statement.
P0141	-452	Step:XX Extra END statement.
P0142	-453	Step:XX Cannot terminate DO with END.
P0143	-454	Step:XX No VALUE statement after CASE.
P0144	-455	Step:XX Preceding IF missing.
P0145	-456	Step:XX Preceding CASE missing.
P0146	-457	Step:XX Preceding DO missing.
P0147	-458	Step:XX Cannot find END of XXs.
P0148	-459	Step:XX Too many control structures.
P0149	-460	Variable (or program) already exists.
P0150	-461	Variable of different type already exists.
P0151	-464	Internal buffer over for complicated expression.
P0152	-465	Undefined variable (or program).
P0153	-466	Illegal clock value.
P0154	-470	Expect '='.
P0155	-471	Expect ')'.
P0156	-472	Expect ']'.
P0157	-473	Expect "TO".
P0158	-474	Expect "BY".
P0159	-475	Expect ':'.
P0160	-476	Expect "ON/OFF".
P0161	-477	Expect Robot Number.
P0162		Cannot modify position in this instruction.
P1000	-200	Cannot execute a program because motor power is OFF.
P1001	-201	Cannot execute a program in TEACH mode.
P1002	-202	Cannot execute a program because teach lock is ON.
P1003	-213	Cannot execute a program because of EXT-IT.
P1004	-217	Cannot execute a program because of program reset.
P1005	-218	Cannot execute program because of EXT. START ENABLE.
P1006	-219	Cannot execute program because of EXT. START DISABLE.
P1007	-220	Start signal was inputted. Not RPS_END step.
P1008	-221	Cannot execute a program because of HOLD sw.
P1009	-300	Program is already running.
P1010	-301	Robot control program is already running.
P1011	-302	Cannot continue this program. Use EXEC.
P1012	-303	Robot is moving now.
P1013	-304	Cannot execute because in error now. Reset error.
P1014	-314	Cannot execute because the program is already used.
P1015	-326	Cannot delete because used by another command.
P1016	-327	Used in programs.
P1017	-328	Used in editor.
P1018	-329	KILL or PCKILL to delete program.

Code	Former Code	Error Message
P1019	-308	PC program is running.
P1020	-320	Cannot operate because teach pendant on operation.
P1021	-306	Cannot execute with DO command.
P1022	-324	Cannot execute with MC instruction.
P1023	-325	Cannot execute in Robot program.
P1024	-405	Statement cannot be executed.
P1025	-1231	The function cannot be executed because of the unsetting.
P1026	-835	Cannot KILL because the program is running.
P1027	-211	Cannot edit a program because teach lock is ON.
P1028	-330	Cannot paste.
P1029	-490	Program name not specified.
P1030	-494	Program is interlocked by another procedure.
P1031	-700	No free memory.
P1032	-801	No program step.
P1033	-834	Program name already exists.
P1034	-1905	This program is not editable.
P1035	-545	Record inhibited. Set "record accept" and operate again.
P1036	-548	Program-change inhibited. Set "ACCEPT" and operate again.
P1037	-1907	A program including "zn_8_n_4_" in name is illegal. Change the name of these programs.
P1038	-800	Program does not exist.
P1039	-208	Teach pendant is not connected.
P1040		Cannot execute this command in I/F panel.
P2000	-101	Turn off motor power.
P2001	-207	Turn to HOLD at HOLD/RUN sw.
P2002	-103	There is no external axis.
P2003	-104	Illegal positioner type.
P2004	-105	Cannot change data because user data exist.
P2005	-106	Graphic area error.
P2006	-107	Option is OFF.
P2007	-212	Cannot execute because executed by other device.
P2008	-514	Device is not ready.
P2009	-523	Illegal file name.
P2010	-531	Disk is not ready.
P2011	-533	Invalid disk format.
P2012	-535	Disk is protected.
P2013	-536	Disk full.
P2014	-537	Too many files.
P2015	-538	Can't write on read-only file.
P2016	-551	Cannot open the file.
P2017	-552	Cannot close the file.
P2018	-559	Storage data logging now.

Code	Former Code	Error Message
P2019	-563	ADC function is in use by other process.
P2020	-591	Illegal device number.
P2021	-598	Cannot execute on this terminal.
P2022	-654	Cannot use DOUBLE OX.
P2023	-670	In cooperative mode.
P2024	-678	Invalid coordinate value X.
P2025	-679	Invalid coordinate value Y.
P2026	-680	Invalid coordinate value Z.
P2027	-690	Cannot use because signal is used in I/F panel.
P2028	-1223	Arm ID board is busy.
P2029	-1247	Axis setting data is incorrect.
P2030	-444	Undefined function number.
P2031	-446	Deleted step was destination step of Jump or Call instruction.
P2032	-360	Illegal WHERE parameter.
P2033	-390	Logging is running.
P2034	-391	Undefined memory.
P2035	-392	Non data.
P2036	-393	Memory verify error.
P2037	-463	Real time path modulation is already running.
P2038	-100	Matrix calculation error.
P2039	-447	Cannot execute from the FN instruction.
P2040	-4210	Card is not ready.
P2041	-4211	Wrong card loaded.
P2042	-4212	Card is write-protected.
P2043	-4213	Card battery is low voltage.
P2044	-4214	Card is not formatted.
P2045	-4215	Can not format this card.
P2046	-4216	Card initialization error.
P2047	-4217	File is already open.
P2048	-4218	File does not exist in card.
P2049	-4219	Attempted to open too many files.
P2050	-4220	Unexpected error during card access.
P2051	-4221	Illegal sequence numbers for file I/O data.
P2052	-4252	[LSEQ]Program includes unavailable instruction.
P2053	-4253	[LSEQ]Too many steps exist.
P2054	-4254	[LSEQ]Invalid type of signal variable.
P2055	-4255	[LSEQ]Program is already running..
P2056	-4256	[LSEQ]Signal number is over limit.
P2057	-4263	[SerialFlash]Cannot open file.
P2058	-4264	[SerialFlash]Data read error.
P2059	-4265	[SerialFlash]Data write error.
P2060	-4266	[SerialFlash]File or directory doesn't exist.

Code	Former Code	Error Message
P2061	-4269	File does not exist in floppy.
P2062	-4270	[FDD/PCCARD]Failure in writing data by verify function.
P2063	-4271	[FDD/PCCARD]Illegal response for setting of verify function.
P2064	-4272	[FDD]Few available capacity exist.
P2065	-4273	[Multi Disks] Invalid disk was loaded.
P2066	-4278	Boot flash state is write-disenable.
P2067	-4279	[Serial Flash] File directory error.
P2068	-4280	Can not execute the program because it is edited now.
P2069		[FDD/PCCARD]Device is in use in other progress.
P4000	-3052	DEVNET)Already in that mode.
P4001	-3064	PROFIBUS)Interface not enable
P4002	-3065	FIELD-BUS) No I/F assigned
P5000	-316	Waiting weld completion.
P5001	-318	Waiting retract or extend pos. input signal.
P5002	-319	Spot sequence is running.
P5003	-371	External-axis type and Gun type data mismatch.
P6000	-2101	Shifted location of STEP-XX is out of range.
P6001	-2102	STEP-XX4d in source Program is out of motion range.
P6002	-2106	Specified PAINTING DATA BANK is undefined.
P6003	-2144	Cannot execute program because of suspend playback.
P6500	-818	Cannot generate working line direction.
P6501	-819	Illegal tool posture.
P6502	-934	In multi pass welding. Database does not exist.
P6503	-935	In multi pass welding, Expanded point is out of limit.
P6504	-944	Step:XX Preceding L.START missing.
P7000		Cannot program reset, because of not home1 position.
P7001		Only NOP Interp, because of the Pressure adjustment mode.
W1000	-50	Cannot move along straight line JT-XX in this configuration.
W1001	-57	Over maximum joint speed in check. Set low speed.
W1002	-58	Operation information was cleared.
W1003	-63	Calibration was failed. Retry after changing configuration.
W1004	-64	Out of range of jt-XX-M motion limit during measure. Check motion limit range.
W1005	-65	Illegal center of gravity, default parameter is set.
W1006	-66	Illegal load moment. default parameter is set.
W1007	-102	Application setting is changed. Turn OFF & ON the control power.
W1008	-108	Parameter is changed. Turn OFF & ON the control power.
W1009	-317	Position envelope error of jt-XX-M at last E-stop.
W1010	-1022	RAM battery low voltage.
W1011	-1217	PLC alarm. (XX)
W1012	-1249	Servo parameter is changed. Turn OFF & ON the control power.
W1013	-1511	Encoder battery low voltage. [Servo(XX)]
W1014	-1248	Number of axis is changed. Initialize again.

Code	Former Code	Error Message
W1015	-61	Needs inspection. possibility of failure.
W1016	-60	Torque of motor is over limit. JT-XX
W1017		Encoder battery low voltage.[External axis(XX)]
W5000	-609	Release the wait cond. because the pressure measurement mode.
W5001	-605	PLC communication error.
W5002	-606	Weld controller # XX not connected.
W5003	-607	Weld controller # XX no response.
W5004	-608	Weld controller #-XX response error.
W5005	-621	(Spot welding) No response from RWC XX.
W5006	-622	(Spot welding) RWC response error XX.
W5007	-623	(Spot welding) Weld fault XX.
W5008	-625	(Spot welding) Cable disconnecting error XX.
W5009	-626	(Spot welding) Internal leak XX.
W5010	-627	(Spot welding) Main cable exchange alarm XX.
W5011	-629	(Spot welding) No connection with RWC XX.
W6000	-2109	It is time to grease reduction gears and motor bearings.
W6001	-2110	It is time to change the robot main cable.
W6002	-2111	It is time to change cooling fans in the controller.
W6003	-2112	It is time to change the DC power supply in the controller.
W6004	-2113	It is time to change the servo power unit.
W6005	-2114	It is time to change the power amplifier for the robot arm.
W6006	-2115	It is time to change the power amplifier for the robot wrist.
W6007	-2116	It is time to change the power amplifier for the traveler.
E0001	-1	Unknown error.
E0100	-445	Abnormal comment statement exists.
E0101	-802	Nonexistent label.
E0102	-803	Variable is not defined.
E0103	-804	Location data is not defined.
E0104	-805	String variable is not defined.
E0105	-807	Program or label is not defined.
E0106	-808	Value is out of range.
E0107	-809	No array suffix.
E0108	-810	Divided by zero.
E0109	-811	Floating point overflow.
E0110	-812	Too long string.
E0111	-813	Illegal exponential operation.
E0112	-814	Too complicated expression.
E0113	-815	No expressions to evaluate.
E0114	-817	SQRT parameter is negative.
E0115	-820	Illegal array suffix value.
E0116	-821	Illegal argument value.
E0117	-822	Illegal joint number.

Code	Former Code	Error Message
E0118	-840	Too many subroutine calls.
E0119	-842	Nonexistent subroutine.
E0900	-427	Block step instruction check sum error.
E0901	-499	Step data is broken.
E0902	-816	Expression data is broken.
E0903	-1019	Check sum error of system data.
E1000	-561	ADC channel error.
E1001	-562	ADC input range error.
E1002	-571	PLC interface error.
E1003	-573	Built-in PLC is not installed.
E1004	-578	INTER-bus board is not ready.
E1005	-983	Spin axis encoder difference error.
E1006	-1029	Touch panel switch is short-circuited.
E1007	-1201	Power sequence board is not installed.
E1008	-1202	Second Power sequence board is not installed.
E1009	-1203	No XX-M I/O board is not installed.
E1010	-1205	Power sequence detects error.
E1011	-1206	Built-in sequence board is not installed.
E1012	-1208	RI/O board or C-NET board is not installed.
E1013	-1210	INTER-BUS board is not installed.
E1014	-1211	Dual port memory for communication is not installed.
E1015	-1212	Amp Interface board is not installed.(code=XX)
E1016	-1213	No XX-M CC-LINK board is not installed.
E1017	-1214	PLC error. Error code is Hex.XX.
E1018	-1215	INTER-BUS status error.
E1019	-1216	Power sequence board for safety unit is not installed.
E1020	-1218	External equipment is abnormal.
E1021	-1221	Arm ID board error. (code XX)
E1022	-1222	Power sequence board error. (code XX)
E1023	-1228	Communication error in robot network.
E1024	-1262	EXT.AXIS cut sequence error.(code XX)
E1025	-1263	EXT.AXIS connect sequence error.(code XX)
E1026	-1264	Main CPU ID mismatch.
E1027	-1336	Safety circuit was cut off.
E1028	-1500	Jt-XX-M of motor is overload.
E1029	-1513	Encoder rotation data is abnormal.(jt -XX-M)
E1030	-1516	Encoder data is abnormal.(jt -XX-M)
E1031	-1518	Miscount of encoder data.(jt -XX-M)
E1032	-1521	Mismatch ABS and INC encoder data(jt -XX-M).
E1033	-1524	Encoder line error of (jt -XX-M) .
E1034	-1550	Encoder initialize error (jt -XX-M).
E1035	-1553	Encoder response error(jt -XX-M).

Code	Former Code	Error Message
E1036	-1554	Encoder communication error.(jt -XX-M)
E1037	-1555	Encoder data conversion error.(jt -XX-M)
E1038	-1556	Encoder ABS-track error.(jt -XX-M)
E1039	-1557	Encoder INC-pulse error.(jt -XX-M)
E1040	-1558	Encoder MR-sensor error. (jt -XX-M)
E1041	-1601	Limit switch (JT -XX) is ON.
E1042	-1602	Limit switch signal line is disconnected.
E1043	-1605	Teach Plug Is abnormal.
E1044	-3000	FIELD-BUS interface board is not installed.
E1045	-628	(Spot welding) Gun-clamp mismatch.
E1046	-659	Too short distance between start point and end point.
E1047	-2015	Cannot execute in conveyor follow mode.
E1048	-109	Offset data of zeroing is illegal value.
E1049	-897	Current position is out of area.
E1050	-634	Encoder and brake power off signal is not dedicated.
E1051	-653	Illegal antinomy OX output.
E1052	-656	Work sensing signal is not dedicated.
E1053	-657	Work sensing signal is already inputted.
E1054	-661	Cannot execute motion instruction.
E1055	-662	Start point position error for circle.
E1056	-663	MASTER robot has already existed.
E1057	-664	This robot is not MASTER.
E1058	-665	SLAVE robot has already existed.
E1059	-699	Not an instruction for cooperative motion.
E1060	-671	Cannot execute in check back mode.
E1061	-672	Cannot execute in ONE program.
E1062	-673	JT2 and JT3 are interfered at start location.
E1063	-674	JT2 and JT3 are interfered at end location.
E1064	-681	Illegal pallet number.
E1065	-682	Illegal work number.
E1066	-683	Illegal pattern number.
E1067	-684	Illegal pattern type.
E1068	-685	Illegal work data.
E1069	-686	Illegal pallet data.
E1070	-693	ON/ONI signal is already inputted.
E1071	-694	XMOVE signal is already inputted.
E1072	-697	Home position data is not defined.
E1073	-824	Illegal timer number.
E1074	-825	Over maximum signal number.
E1075	-826	Illegal clamp number.
E1076	-827	Illegal time value.
E1077	-828	No value set.

Code	Former Code	Error Message
E1078	-829	Illegal signal number.
E1079	-837	Cannot use dedicated signal.
E1080	-838	Not RPS mode.
E1081	-839	Cannot use negative value.
E1082	-850	Out of absolute lower limit.
E1083	-851	Out of absolute upper limit.
E1084	-852	Out of user lower limit.
E1085	-853	Out of user upper limit.
E1086	-855	Start location of jt-XX is out of motion range.
E1087	-856	End location of jt-XX is out of motion range.
E1088	-857	Destination is out of motion range.
E1089	-858	Illegal configuration for linear motion.
E1090	-863	External modulation data is not inputted.
E1091	-864	External modulation data is abnormal.
E1092	-865	Modulation data is over limit.
E1093	-866	Illegal motion command in execute modulate motion.
E1094	-871	Illegal joint number.
E1095	-872	Cannot execute motion instruction in PC program.
E1096	-873	Illegal auxiliary data number.
E1097	-874	Not next C1MOVE or C2MOVE instruction.
E1098	-875	No C1MOVE(CIR1) ins.
E1099	-876	Check three points (back and forth).
E1100	-877	Cannot execute in sealing specification.
E1101	-879	Cannot execute except in sealing specification.
E1102	-896	Because option is not set up, can't execute.
E1103	-898	Over conveyer position.
E1104	-978	Too many SPINMOVE instructions.
E1105	-1032	Destination is in CUBE.
E1106	-1051	Cannot execute in this robot.
E1107	-1219	Cannot use SEPARATE CONTROL.
E1108	-1227	Robot network ID is duplicate.
E1109	-1267	Conveyor I/F board is not installed.
E1110	-1735	GROUP is not primed.
E1111	-2018	JT-XX cannot move for regulate of motion.
E1113	-658	Work sensing signal is not detected.
E1114	-666	Interruption in cooperative control.
E1115	-669	Forced termination of cooperative control.
E1116	-979	Spin axis is not stopped on every 360 degrees.
E1117	-1011	Process time over.
E1118	-1012	Command position of jt-XX suddenly changed.
E1119	-1014	Out of motion limit in commanded pos. of JT-XX
E1120	-1017	Current command between Jt2 and Jt3 is interfered.

Code	Former Code	Error Message
E1121	-1031	Other robot is already in the interference area.
E1122	-1308	Unexpected motor power off.
E1123	-1503	Speed error jt -XX-M.
E1124	-1504	Position envelope error jt -XX-M.
E1125	-1505	Velocity envelope error jt -XX-M.
E1126	-1506	Command speed error of jt -XX-M.
E1127	-1507	Command acceleration error of jt -XX-M.
E1128	-1600	Uncoincidence error between destination and current position of jt -XX.
E1129	-1737	Jt-XX moved during regulation of external axis.
E1130	-1902	Jt-XX-M collision is detected.
E1131	-1904	Jt-XX-M unexpected shock is detected.
E1132	-1906	Motor power off while tool calibration. Quit tool calibration.
E1133	-2011	Work has reached conveyor max value.
E1134	-2012	Abnormal work pitch of conveyor.
E1135	-600	Motor power OFF.
E1136	-675	Standard terminal is not connected.
E1137	-676	Cannot input/output to multi function panel.
E1138	-698	Aux. terminal is not connected.
E1139	-869	DA board is not installed.
E1140	-2009	No conveyor axis.
E1141	-2010	Conveyor position is over limit.
E1142	-2013	No following axis.
E1143	-2014	Conveyor axis number is not set.
E1144	-4245	No arm control board.
E1145	-4249	Cannot use specified channel because using now.
E1146	-4251	[LSEQ]Aborted by process time over.
E1147	-4257	Cannot open setting file, so cannot set to shipment state.
E1148	-4258	Cannot read setting file, so cannot set to shipment state.
E1149	-4259	Cannot open setting data, so cannot set to shipment state.
E1150	-4260	Cannot read setting data, so cannot set to shipment state.
E1151	-4261	Too many data for setting to the shipment state.
E1152	-4262	Name of setting data for shipment state is too long..
E1153	-4268	Power sequence board detects error.(Code=XX)
E1154	-1204	Option SIO port not installed.
E1155	-560	A/D converter is not installed.
E1156		[ARM CONTROL BOARD] Process time over.
E1157	-1225	Arm ID I/F board error. (code XX)
E4000	-507	Data communication error.
E4001	-543	Data read error.
E4002	-544	Data write error.
E4003	-546	Unexpected error in file access.
E4004	-580	Communication retry error.

Code	Former Code	Error Message
E4005	-581	Communication process was stopped.
E4006	-583	Receive no data after request.
E4007	-584	Too long receive data(MAX=255 characters).
E4008	-585	Abnormal data (EOT) received in communication.
E4009	-586	Communication time out error.
E4010	-596	Cannot attach terminal.
E4011	-597	Cannot attach communication port.
E4012	-599	Waiting input data for PROPT. Connect input device.
E4013	-2200	TELNET) SEND error. CODE=XX
E4014	-2201	TELNET) RECV error. CODE=XX
E4015	-2202	TELNET) IAC receive error. CODE=XX
E4016	-2203	TELNET) Close failure. CODE=XX
E4017	-2204	TELNET) Main socket close failure. CODE=XX
E4018	-2205	TELNET) System error. CODE=XX
E4019	-2206	TCPIP) Socket open failure. CODE=XX Dst.IP=XX
E4020	-2207	TCPIP) Socket close failure. CODE=XX Dst.IP=XX
E4021	-2208	TCPIP)Communication Error. CODE=XX Dst.IP=XX.
E4022	-2209	TCPIP)Too Long Message.
E4023	-2210	TCPIP)Cannot reach the Host.
E4024	-2211	TCPIP)Communication Time Out. Dst.IP=XX.
E4025	-2212	TCPIP)Connection aborted.
E4026	-2213	TCPIP)No Buffer Space.
E4027	-2214	TCPIP)Bad Socket.
E4028	-2216	FTP)Data receive error.(code=XX)
E4029	-2217	FTP)Data send error.(code=XX)
E4030	-2218	FTP)Unrecognized command.(code=XX)
E4031	-2219	FTP)Failed logout with FTP server.(code=XX)
E4032	-2220	FTP)Not registered OS detected.
E4033	-2221	FTP)Failed connecting with server.(code=XX)
E4034	-2222	FTP)Failed to receive HOST OS information.(code=XX)
E4035	-2224	FTP)TCP/IP not initialized.
E4036	-2225	FTP)FTP service busy now.
E4037	-2226	FTP)Failed AUTO-SAVING.
E4038	-3001	FIELD-BUS-INIT) Error Reply. XX
E4039	-3002	FIELD-BUS-INIT) Reply timeout. XX
E4040	-3003	ANYBUS)IN-AREA request timeout. XX
E4041	-3004	FIELD-BUS) Slave port OFFLINE.
E4042	-3005	ANYBUS)OUT/FB.CTRL request timeout. XX
E4043	-3006	ANYBUS)OUT/FB.CTRL release timeout. XX
E4044	-3007	FIELD-BUS) Master port OFFLINE.
E4045	-3008	ABM-DN)mailbox error.
E4046	-3050	DN)master status. XX

Code	Former Code	Error Message
E4047	-3051	DN)node status. XX
E4048	-3053	DEVNET)Node XX not in the scan list.
E4049	-3054	DEVNET)not supported.
E4050	-4206	No response from the FDD/PC_CARD driver board.
E4051		No communication with FDD/PCCARD driver board.
E4052		[FDD/PCCARD]Failure in setting verify function. Please set again.
E4053	-3055	ABMA-PDP) I/F module error XXs XX
E4054	-3056	ABMA-PDP) I/O Data Communication error XX
E4055	-3057	ABMA-PDP) Timeout of sending I/O data XX
E4056	-3058	ABMA-PDP) Timeout of receiving I/O data XX
E4057	-3059	ABMA-PDP) Timeout of sending message XX
E4058	-3060	ABMA-PDP) Timeout of receiving message XX
E4059	-3061	ABMA-PDP)Check configuration data XX
E4060	-3062	PROFIBUS)Slave Diag-error response detected XX
E4061	-3063	PROFIBUS)Statistic counter-error response detected XX
E4062	-3066	ABMA-PDP)Status STOP XX
E4063	-3067	ABMA-PDP)Status OFFLINE XX
E5000	-603	Connected permission signal has not been turned on.
E5001	-648	RWC type Is not process control type.
E5002	-649	1GS board Is not process control type.
E5003	-611	Illegal extend (retract) output signal.
E5004	-615	Weld completion signal is already inputted.
E5005	-620	(Spot welding) Welding schedule setting data is abnormal.
E5006	-635	CLAMP SPEC is not set as PULSE.
E5007	-641	Servo welding gun is disconnected or other gun is connected.
E5008	-643	Measure of chip abrasion (STAGE1) was not executed.
E5009	-644	Work sensing signal(gun_chip touch signal) is not set.
E5010	-646	Servo welding gun mechanical parameter is not set.
E5011	-647	Servo welding gun clamp number is duplicate.
E5012	-689	Cannot change the gun because offset data Is abnormal.
E5013	-691	Cannot change plural guns at the same step.
E5014	-692	Gun is connected to another joint.
E5015	-695	Gun status disagrees with clamp status.
E5016	-696	Data of SRVPRESS is wrong.
E5017	-1912	Abrasion base data is not registered.
E5018	-610	Weld completion signal has not been detected.
E5019	-612	Weld fault signal is detected.
E5020	-613	Retract pos. monitor error.
E5021	-614	Extend pos. monitor error.
E5022	-616	Current gun retract position differs from a destination.
E5023	-617	Cannot measure abrasion because of abnormal abrasion.
E5024	-618	Work stroke close signal has not been detected.

Code	Former Code	Error Message
E5025	-619	Work stroke open signal has not been detected.
E5026	-624	(Spot welding) RWC error. XX
E5027	-630	Robot stopped in welding.
E5028	-631	Cannot achieve desired pressure.
E5029	-632	Gun Chip stuck.
E5030	-633	Copper plate abrasion over the limit. step=XX
E5031	-638	Weld completion signal is not turned off.
E5032	-642	Calibration does not end normally.
E5033	-645	Cannot weld because of abnormal thickness.
E5034	-660	Gun chip abrasion over the limit.
E5500	-1750	Vision board is not installed.
E5501	-2751	(Vision) Camera not connected.
E5502	-2710	(Vision) Illegal parameter.
E5503	-2711	(Vision) Illegal Symbol.
E5504	-2712	(Vision) Illegal name.
E5505	-2713	(Vision) Illegal image memory.
E5506	-2715	(Vision) Illegal histogram data.
E5507	-2716	(Vision) Illegal mode.
E5508	-2717	(Vision) Illegal density(/color).
E5509	-2718	(Vision) Illegal camera input assignment.
E5510	-2719	(Vision) Illegal camera ch. number.
E5511	-2720	(Vision) Illegal Window No.
E5512	-2722	(Vision) Illegal coordinates data.
E5513	-2723	(Vision) Illegal number.
E5514	-2726	(Vision) Illegal image code(binary/multi).
E5515	-2728	(Vision) Illegal threshold.
E5516	-2729	(Vision)PROTO(/TEMPLATE) not registered or already exists.
E5517	-2745	(Vision) Cal. data not registered.
E5518	-2746	(Vision) Graphic cursor is not initialized.
E5519	-2747	(Vision) Too many sample of PROTO object.
E5520	-2748	(Vision) Too many detection targets.
E5521	-2754	(Vision)Vision command not initiated.
E5522	-2768	(Vision) System data are abnormal now.
E5523	-2770	(Vision) Error on processing image(s).
E5524	-2771	(Vision) Sound port assigned another function.
E5525	-2749	(Vision) Lack of data storage area.
E5526	-2752	(Vision) Illegal synch. mode.
E5527	-2753	(Vision)Vision processing now.
E5528	-2766	(Vision) Image capture error.
E5529	-2767	(Vision) Time out or Buffer overflow.
E5530	-2769	(Vision) Failed to write on flash memory.
E5531	-2772	(Vision) Proto data abnormal, so initialized.

Code	Former Code	Error Message
E5532	-2774	(Vision) Work detection failure.
E5533	-2797	(Vision) Initialization error. Code = XX
E5534	-2799	(Vision) Vision system error.
E5535	-2744	(Vision) Stand-alone mode now.
E5536	-2732	(Vision) Inappropriate camera/projector parameter.
E5537	-2721	(Vision) Illegal camera switch assignment.
E5538	-2727	(Vision) This plane is assigned to another camera.
E5539	-2755	(Vision) Edge is not found.
E5540	-2761	(Vision) Inappropriate HSI data.
E5541	-2762	(Vision) H data range width is over 128.
E5542	-2731	(Vision) No distance image input unit.
E5543	-2758	(Vision) Inappropriate edge points data for calculation.
E5544	-2760	(Vision) Inappropriate color conversion table type setting on system configuration.
E5545	-2725	(Vision) Illegal area size.
E5546	-2756	(Vision) Slit image does not exist.
E5547	-2724	(Vision) Illegal no. of correlation vectors.
E5548	-2759	(Vision) Inappropriate vector data.
E5549	-2775	(Vision) X-Fit environment is not equipment
E5550	-2776	(Vision) Mouse is not initialized
E5551	-2777	(Vision) Camera switcher board is not installed.
E6000	-2100	Explosion proof TP is not connected.
E6001	-2140	The next step of XD(2)START must be LMOVE or HMOVE.
E6002	-2141	Signal condition is already inputted.
E6003	-2143	Door detect signal is not dedicated.
E6004	-2145	Location data has not detected.
E6005	-2173	Illegal setting of barrier unit.
E6006	-2142	Signal not detected.
E6007	-2103	Wrist can't be straightened any more (Singular point 1).
E6008	-2104	Wrist can't be bent any more (Singular point 2).
E6009	-2171	Lacks of air flux.
E6010	-2105	Out of MOVING AREA XYZ LIMIT.
E6011	-2172	Internal pressure low.
E6012	-2146	Relative distance between guns is too near.
E6013	-2147	Cannot set program no. in program queue.
E6014	-2148	Cannot set program no. in delay queue.
E6500	-999	No welding Interface board.
E6501	-998	No No.2 welding Interface board.
E6502	-900	Arc failure.
E6503	-901	Wire stuck.
E6504	-904	Arc start failure.
E6505	-948	Arc weld insulation defect.
E6506	-1610	Torch is interfered.

Code	Former Code	Error Message
E6507	-984	Illegal interpolation data.
E6508	-949	No E/N ratio D/A board.
E6509	-910	Work is not detected.
E6510	-911	Sensing direction is not detected.
E6511	-912	Insufficient sensing points.
E6512	-913	Mother or daughter work not exists.
E6513	-914	Too many sensing points.
E6514	-915	Illegal work designation.
E6515	-916	Illegal sensing points designation.
E6516	-917	Wire check failure.
E6517	-920	Illegal welding condition number.
E6518	-921	Weld data not set.
E6519	-922	Weld data is out of range.
E6520	-923	Out of Laser sensor tracking value.
E6521	-924	Out of Laser sensor tracking capacity.
E6522	-925	Laser sensor cannot detect joint.
E6523	-926	Calibration data between torch and camera is not ready.
E6524	-927	Error of calculated data using Laser sensor.
E6525	-929	Cannot detect joint, because of tracking by Laser sensor.
E6526	-936	Laser sensor controller no response.
E6527	-937	Laser sensor communication error. Code is XX.
E6528	-938	Start end is not found by Laser sensor.
E6529	-939	Finish end is not found by Laser sensor.
E6530	-941	Cannot use circular interpolation with Laser sensor function.
E6531	-942	Cannot turn Laser on because motor power is OFF.
E6532	-943	No communication board to Laser sensor.
E6533	-951	No RTPM board.
E6534	-960	Teaching point of RTPM is out of range.
E6535	-961	RTPM arc sensor error.
E6536	-964	RTPM current deviation error.
E6537	-962	RTPM tracking value is out of range.
E6538	-963	Out of RTPM tracking capacity.
E6539	-969	Out of AVC tracking value.
E6540	-971	Out of AVC tracking capacity.
E6541	-967	No AVC board.
E6542	-972	AVC voltage deviation error.
E6543	-968	Too many taught points for AVC.
E6544	-990	Hyper Arc tracking value is out of range.
E6545	-991	Out of Hyper Arc tracking capacity.
E6546	-993	Bead end is not found.
E6547	-994	Finish end is not found.
E6548	-995	Hyper Arc torch envelope error.

Code	Former Code	Error Message
E6549	-996	Hyper Arc torch calibration error.
E6550	-997	Hyper Arc Z phase index error.
E6551	-985	No Hyper Arc board.
E6552	-986	Hyper Arc board error. code is XX.
E6553	-987	Hyper Arc current sensor error.
E6554	-988	Hyper Arc voltage sensor error.
E6555	-989	Hyper Arc current deviation error.
E6556	-992	Hyper Arc amplifier error. code is XX.
E6557	-945	No Wire feeding Control board.
E6558	-946	Wire feeding Control error, code is XX.
E6559	-947	Wire feeding speed deviation error.
E6560	-965	Cannot calibrate again while welding.
E6561	-966	Cannot weld because re-calibration is executing again.
E6562	-902	Electric pole stuck.
E7000		Servo weld gun is disconnect.
E7001		Location data have is abnormal data.
E7002		Destination is far from target point.
D0001	-1100	CPU error.(code=XX)
D0002	-1101	Main CPU BUS error.(code=XX)
D0003	-1102	VME BUS error.(code=XX)
D0004	-4238	[ARM CONTROL BOARD] CPU error.(Code=XX)
D0005	-4239	[ARM CONTROL BOARD] CPU BUS error.(Code=XX)
D0900	-1003	Teach data is broken.
D0901	-1025	AS Flash memory sum check error.
D0902	-1026	Servo Flash memory sum check error.
D0903	-1028	IP board memory error.(XX)
D0904	-1805	Memory is locked due to AC_FAIL.
D1000	-4201	Read error of servo control software.
D1001	-4202	Download error of servo control software.
D1002	-4204	Init. error of servo software.
D1003	-4205	Init. error of servo control software.
D1004	-1300	[ARM CTRL BOARD] Watch dog error of Servo control software.
D1005	-1306	Servo board command error. (XX)
D1006	-1407	Servo system error.
D1007	-1567	Regenerative time over. [XX]
D1008	-1568	P-N low voltage. [XX]
D1009	-1569	P-N high voltage. [XX]
D1010	-1570	Regenerative resistor overheat. [XX]
D1011	-1903	Robot type mismatch between AS and servo software.
D1012	-1909	Servo type mismatch. Check the settings.
D1013	-4236	P-N capacitor is not discharged.
D1014	-4267	Servo system error.(Code=XX)

Code	Former Code	Error Message
D1015	-4274	The servo data file does not exist.
D1016	-4275	The servo data file does not include available servo data.
D1017	-4276	Download error of servo data.
D1500	-1517	Read error of encoder Jt-XX-M.
D1501	-1564	Changer defective connection or encoder communication error.
D1502	-1401	Amp overcurrent Jt-XX-M.
D1503	-1420	Current detector type (XX) mismatch!
D1504	-1424	Abnormal current feedback of Jt-XX-M. (Amp. failure or Power harness disconnect)
D1505	-1501	Motor harness disconnected or over heat.(XX)
D1506	-1559	Power module error Jt-XX-M.
D1507	-1800	AC primary power off.
D1508	-1801	24VDC power source is too low.
D1509	-1802	Primary power source is too high.
D1510	-1803	Primary power source is too low.
D1511	-1804	+12VDC or -12VDC is abnormal.
D1512	-4208	Brake line error for jt-XX-M
D1513	-4209	Brake power is abnormal.(XX)
D1514	-4222	I/O 24V fuse is open.
D1515	-4223	Mismatch of the safety circuit line number setting.
D1516	-4224	Mismatch of the HOLD-BACKUP-TIME safety circuit setting.
D1517	-4225	Safety circuit emergency line fuse is open.
D1518	-4226	Mismatch of the safety circuit emergency condition.
D1519	-4227	Mismatch of the safety circuit LS condition.
D1520	-4228	Mismatch of the safety circuit TEACH/REPEAT condition.
D1521	-4229	Mismatch of the safety circuit safety-fence condition.
D1522	-4230	Mismatch of the safety circuit enabling device condition.
D1523	-4231	Mismatch of the safety circuit ext. enabling device condition.
D1524	-4232	Incorrect operation of the safety relay.
D1525	-4233	Incorrect operation of MC(K1).
D1526	-4234	Incorrect operation of MC(K2).
D1527	-4235	Incorrect operation of MC(K3).
D1528	-4237	Controller temperature is out of range.
D1529	-4244	Signal harness is disconnected or Encoder power error.
D1530		Abnormal current limit of Jt-XX-M.
D2000	-928	Communication board for Laser sensor no response.
D2001	-1209	RI/O or C-NET board initialize error.
D2002	-1220	No response from the arm ID board.
D2003	-1229	No data in the arm ID board.
D2004	-1230	Mismatch data in the arm ID board.
D2005	-1266	CC-LINK software version mismatch.
D2006	-2174	No response from communication board for Explosion proof TP.

Code	Former Code	Error Message
D2007	-1207	No response from the built-in sequence board.
D2008	-1224	Group XX MC P-N voltage is stuck.
D2009	-2170	Internal pressure sensor error.
D2010	-4200	Sync. Error between User I/F and Arm control board.
D2011	-4203	Parameter download error between User I/F and Arm control board.
D2012	-677	SOFT ABSORBER error. Turn OFF & ON the control power.
D2013	-687	CHANGE GAIN error. Turn OFF & ON the control power.
D2014	-1226	Robot network initialize error.
D2016	-4207	No response from the arm control board.
D2017	-4240	[USER I/F BOARD] no response.
D2018	-4241	[ARM CTRL BOARD] no response.
D2019	-4242	[ARM CTRL BOARD] servo software no response.
D2020	-4243	[ARM CTRL BOARD] servo control software no response.
D2021	-4246	Arm data file is not found.
D2022	-4247	Arm data is not found.
D2023	-4248	Failed to load arm data.
D2024	-4250	[ARM CTRL BOARD] Robot type setting failed.
D2025		Robot code is mismatch between software and arm control board.
D2026		Code is mismatch between software and current sensor I/F board.
D2027		Code is mismatch between software and power block.

APPENDIX 2 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)	Page
ABORT	AB	Stops execution	M	ABORT	5-41
ABOVE	AB	Changes elbow joint to above position	P	ABOVE	6-31
ABS	ABS	Returns absolute value	F	ABS (real value)	9-37
ABS.SPEED	ABS.SPEED	Enables use of absolute speed	S	...ABS.SPEED...	7-18
ACCEL	ACCE	Sets acceleration	P	ACCEL speed ALWAYS	6-19
ACCURACY	ACCU	Sets accuracy range	P	ACCURACY distance ALWAYS	6-18
AFTER.WAIT.TMR	AF	Sets how timers begin in block step programs	S	...AFTER.WAIT.TMR...	7-19
ALIGN	AL	Aligns tool Z axis with base coordinate axis	P	ALIGN	6-10
AND	AND	Logical AND	O	... AND...	8-4
ASC	ASC	Returns ASCII value	F	ASC (string, character number)	9-9
ATAN2	ATAN2	Returns the arctangent value	F	ATAN2 (real value1, real value2)	9-37
AUTOSTART.PC	AUTOSTART.	Starts PC program automatically	S	...AUTOSTART.PC...	7-9
AUTOSTART2.PC	AUTOSTART2.	Starts PC program automatically	S	...AUTOSTART2.PC...	7-9
AUTOSTART3.PC	AUTOSTART3.	Starts PC program automatically	S	...AUTOSTART3.PC...	7-9
AUTOSTART4.PC	AUTOSTART4.	Starts PC program automatically	S	...AUTOSTART4.PC...	7-9
AUTOSTART5.PC	AUTOSTART5.	Starts PC program automatically	S	...AUTOSTART5.PC...	7-9
AVE_TRANS	AVE_TRANS	Returns average value	F	AVE_TRANS (transformation values1, transformation values2)	9-31
BAND	BAND	Binary AND	O	... BAND ...	8-6
BASE	BA	Defines base transformation values	M	BASE transformation values	5-59
BASE	BA	Defines base transformation values	P	BASE transformation values	6-85
BASE	BASE	Returns base transformation values	F	BASE	9-32
BATCHK	BAT	Enables/disables battery low voltage check	M	BATCHK	5-78
BELOW	BE	Changes elbow joint to below position	P	BELOW	6-31
BITS	BI	Sets output signals	M	BITS start number, number of signals = value	5-91
BITS	BI	Sets output signals	P	BITS start number, number of signals=decimal values	6-61
BITS	BITS	Returns bit patterns of signals	F	BITS (starting signal number, number of signals)	9-4

Name	Abbreviation	Function		Format (Parameter)	Page
BOR	BOR	Binary OR	O	... BOR ...	8-6
BRAKE	BRA	Stops robot motion immediately	P	BRAKE	6-20
BREAK	BRE	Causes a break in CP motion	P	BREAK	6-20
BSPEED	BSP	Sets block speed	P	BSPEED speed	6-21
BXOR	BX	Binary XOR	O	... BXOR ...	8-6
BY	BY		K	SHIFT (trans BY X shift, Y shift, Z shift)	9-30
C	C	Changes program to edit	E	C program name, step number	5-4
C1MOVE	C1	Circular interpolated motion	P	C1MOVE pose variable name, clamp number	6-13
C2MOVE	C2	Circular interpolated motion	P	C2MOVE pose variable name, clamp number	6-13
CALL	CA	Calls subroutine	P	CALL program name	6-36
CALLAUX	CALLAUX	Displays auxiliary function screen	P	CALLAUX auxiliary function number	6-45
CARD_COPY	CARD_COPY	Copies programs in PC card	M	CARD_COPY new program name = source program name + ...	5-22
CARD_FDEL	CARD_FDEL	Deletes data in PC card	M	CARD_FDEL file name...	5-19
CARD_FDIR	CARD_FDIR	Displays names of program/variable in PC card	M	CARD_FDIR	5-16
CARD_FORMAT	CARD_FORMAT	Formats PC card	M	CARD_FORMAT	5-28
CARD_LOAD	CARD_LOAD	Loads contents of PC card into robot memory	M	CARD_LOAD/Q filename	5-34
CARD_RENAME	CARD_REN	Changes program name	M	CARD_RENAME new program name = existing program name	5-20
CARD_SAVE	CARD_SA	Stores program/variable into a PC card	M	CARD_SAVE/SEL filename -program name...	5-29
CARD_SAVE/ELOG	CARD_SA/ELOG	Stores error log into a PC card	M	CARD_SAVE/ELOG file name	5-31
CARD_SAVE/P,L,R,S,A	CARD_SA/P,L,R,S,A	Stores data into a PC card	M	CARD_SAVE (/P) (/L) (/R) (/S) (/A) /SEL file name -program name...	5-31
CARD_SAVE/ROB	CARD_SA/ROB	Stores robot data into a PC card	M	CARD_SAVE/ROB file name	5-31
CARD_SAVE/SYS	CARD_SA/SYS	Stores system data into a PC card	M	CARD_SAVE/SYS file name	5-31
CARD_VERIFY	CARD_VERIFY	Turns ON/OFF verifying function	M	CARD_VERIFY mode	5-19
CASE	CASE	CASE structure	P	CASE number OF ... VALUE ... END	6-52
CCENTER	CCENTER	Returns the center of arc	F	CCENTER (pose1, pose2, pose3)	9-35
CHECK.HOLD	CH	Enables or disables input of commands from the keyboard when HOLD/RUN is in HOLD	S CHECK.HOLD....	7-3
\$CHR	\$CHR	Returns ASCII characters	F	\$CHR (real value)	9-40
CHSUM	CH	Enables/disables resetting of abnormal check sum error	M	CHSUM	5-82
CLAMP	CLAMP	Controls open/close clamp signals	P	CLAMP clamp number 1, ..., clamp number 8	6-74
CLOSE	CLOSE	Closes clamp hand	P	CLOSE clamp number	6-25
CLOSEI	CLOSEI	Closes clamp hand	P	CLOSEI clamp number	6-25

Name	Abbreviation	Function		Format (Parameter)	Page
CLOSES	CLOSES	Turns ON/OFF close clamp signal	P	CLOSES clamp number	6-27
COM	COM	Binary complement	O	... COM ...	8-6
CONTINUE	CON	Resumes execution	M	CONTINUE NEXT	5-42
COPY	COP	Copies programs in robot memory	M	COPY new program name = source program name + ...	5-22
COS	COS	Returns the cosine value	F	COS (real value)	9-37
CP	CP	Continuous path (CP) function	S	...CP.....	7-3
CS	CS	CYCLE START switch ON/OFF status	S	Switch (CS)	7-11
CSHIFT	CSHIFT	Returns the shifted pose	F	CSHIFT (pose1, pose2, pose3, object pose BY shift amount)	9-35
CYCLE.STOP	CY	Stops cycle with External HOLD	S	... CYCLE.STOP....	7-4
D	D	Deletes program steps	E	D step count	5-7
\$DATE	\$DATE	Returns system date	F	\$DATE (date form)	9-48
DECEL	DECE	Sets deceleration	P	DECEL speed ALWAYS	6-19
\$DECODE	\$DECODE	Extracts character ON,	F	\$DECODE (string variable, separator character, mode)	9-43
DECOMPOSE	DECO	Extracts components of pose variable	P	DECOMPOSE array variable name [element number]= pose variable name	6-84
DEFSIG	DEF	Displays and changes software dedicated signals	M	DEFSIG OUTPUT/INPUT	5-68
DELAY	DEL	Stops the robot for a given time	P	DELAY time	6-4
DELETE	DEL	Deletes data in memory	M	DELETE (/P) (/L) (/R) (/S) data, ...	5-18
DELETE	DEL	Deletes data in memory	P	DELETE (/P) (/L) (/R) (/S) data, ...	6-93
DEST	DEST	Returns destination in transformation values	F	DEST	9-21
#DEST	#DEST	Returns destination in joint displacement values	F	#DEST	9-21
DEXT	DEXT	Returns specified element of given pose	F	DEXT (pose variable name, element number)	9-8
DISPIO_01	DIS	Changes the display mode of IO command	S	... DISPIO_01....	7-15
DISTANCE	DISTANCE	Returns distance	F	DISTANCE (transformation value, transformation value)	9-6
DLYSIG	DL	Outputs signal after delay	M	DLYSIGNAL signal number time	5-86
DLYSIG	DL	Outputs signal after delay	P	DLYSIGNAL signal number time	6-59
DO	DO	Executes a single program instruction	M	DO program instruction	5-44
DO	DO	DO structure	P	DO ... UNTIL logical expression	6-49
DRAW	DRA	Moves the robot by a given amount	P	DRAW X translation, Y translation, Z translation, X rotation, Y rotation, Z rotation, speed	6-9
DRIVE	DRI	Moves a single joint	P	DRIVE joint number, displacement, speed	6-8
DWRIST	DW	Changes wrist configuration	P	DWRIST	6-34
DX	DX	Returns X component	F	DX (transformation value)	9-7
DY	DY	Returns Y component	F	DY (transformation value)	9-7

Name	Abbreviation	Function		Format (Parameter)	Page
DZ	DZ	Returns Z component	F	DZ (transformation value)	9-7
E	E	Exits from edit mode	E	E	5-10
EDIT	ED	Enters edit mode	M	EDIT program number, step number	5-3
ELSE	EL	IF structure	P	IF ... ELSE ... END	6-47
ENCCHK_EMG	ENCCHK_E	Sets deviation range of robot pose at E-stop	M	ENCCHK EMG	5-79
ENCCHK_PON	ENCCHK_P	Sets acceptable encoder value of robot pose at E-stop	M	ENCCHK PON	5-79
\$ENCODE	\$ENCODE	Returns string created by print data	F	\$ENCODE (print data, print data)	9-45
END	END	FOR structure, etc.	P	FOR ... END, CASE ... END	6-53
ENV_DATA	ENV_	Sets hardware environmental data	M	ENV DATA	5-81
ENV2_DATA	ENV2	Sets software environmental data	M	ENV2 DATA	5-81
ERESET	ERE	Resets error condition	M	ERESET	5-74
ERRLOG	ERRLOG	Displays error log	M	ERRLOG	5-63
ERROR	ERROR	Program error status	S	Switch (ERROR)	7-13
ERROR	ERROR	Returns error code	F	ERROR	9-15
\$ERROR	\$ERROR	Returns error messages	F	\$ERROR (error code)	9-47
\$ERRORS	\$ERRORS	Returns error messages	F	\$ERRORS (error number)	9-47
ERRSTART.PC	ERRS	Executes PC program when an error occurs	S	...ERRSTART.PC...	7-10
EXECUTE	EX	Executes a robot program	M	EXECUTE program name, execution cycles, step number	5-39
EXTCALL	EX	Calls program selected by signals	P	EXTCALL	6-63
F	F	Searches for a string	E	F character string	5-7
FD_COPY	FD_COPY	Copies programs in FD	M	FD_COPY new program name = source program name + ...	5-22
FD_FDEL	FD_FDEL	Deletes data in FD	M	FD_FDEL file name, ...	5-19
FD_FDIR	FD_FDIR	Displays names of program/variable in FD	M	FD_FDIR	5-16
FD_FORMAT	FD_FORMAT	Formats floppy disk	M	FD_FORMAT format type	5-28
FD_LOAD	FD_LOAD	Loads contents of FD into robot memory	M	FD_LOAD/Q filename	5-34
FD_RENAME	FD_RENAME	Changes program name	M	FD_RENAME new program name = existing program name	5-20
FD_SAVE	FD_SA	Stores program/variable into a floppy disk	M	FD_SAVE/SEL filename =program name...	5-29
FD_SAVE/ELOG	FD_SA/ELOG	Stores error log into floppy disk	M	FD_SAVE/ELOG file name	5-31
FD_SAVE/P,L,R,S,A	FD_SA/P...	Stores data into floppy disk	M	FD_SAVE (/P)/(L)/(R)/(S)/(A) /SEL file name=program name...	5-31
FD_SAVE/ROB	FD_SA/ROB	Stores robot data into floppy disk	M	FD_SAVE/ROB file name	5-31
FD_SAVE/SYS	FD_SA/SYS	Stores system data into floppy disk	M	FD_SAVE/SYS file name	5-31
FD_VERIFY	FD_VERIFY	Turns ON/OFF verifying function	M	FD_VERIFY mode	5-19
FLOWRATE	FLOWRATE	Changes flow rate control mode	S	... FLOWRATE ...	7-17
FOR	FOR	FOR structure	P	FOR loop=start TO end STEP value	6-53

Name	Abbreviation	Function		Format (Parameter)	Page
FRAME	FRAME	Returns the transformation values for frame coordinates	F	FRAME (x1, x2, y, origin)	9-22
FREE	FR	Displays size of free memory	M	FREE	5-56
GOTO	G	Jumps to label	P	GOTO label IF condition	6-32
GUNOFF	GUNOFF	Turns OFF gun signals	P	GUNOFF gun number, distance	6-28
GUNOFFTIMER	GUNOFFTIMER	Controls gun output OFF timing	P	GUNOFFTIMER gun number, time	6-29
GUNON	GUNON	Turns ON gun signals	P	GUNON gun number, distance	6-28
GUNONTIMER	GUNONTIMER	Controls gun output ON timing	P	GUNONTIMER gun number, time	6-29
HALT	HA	Stops execution	P	HALT	6-42
HELP	HEL	Displays a list of AS commands and instructions	M	HELP	5-75
HELP/DO	HEL/DO	Displays a list of functions	M	HELP/DO	5-75
HELP/F	HEL/F	Displays a list of monitor commands	M	HELP/F	5-75
HELP/M	HEL/M	Displays a list of commands usable with MC instructions	M	HELP/M	5-75
HELP/MC	HEL/MC	Displays a list of instructions usable with DO command	M	HELP/MC	5-75
HELP/P	HEL/P	Displays a list of program instructions	M	HELP/P	5-75
HELP/PPC	HEL/PPC	Displays a list of instructions usable in PC programs	M	HELP/PPC	5-75
HELP/SW	HEL/SW	Displays a list of system switches	M	HELP/SW	5-75
HERE	HE	Records current pose	M	HERE pose variable name	5-46
HERE	HE	Records current pose	P	HERE pose variable name	6-81
HERE	HERE	Returns transformation value for current pose	F	HERE	9-26
#HERE	#HERE	Returns joint displacement value for current pose	F	#HERE	9-26
HMOVE	HM	Moves in hybrid motion	P	HMOVE pose variable name, clamp number	6-10
HOLD	HO	Stops execution	M	HOLD	5-40
HOLD.STEP	HOLD.STEP	Enables display of the step in execution when the program is held	S HOLD.STEP....	7-16
HOME	HO	Moves to home position	P	HOME position number	6-7
#HOME	#HOME	Returns the home position	F	#HOME (home position number)	9-34
HSETCLAMP	HSETCLAMP	Assigns signal number to operate clamps	M	HSETCLAMP	5-66
I	I	Inserts new steps	E	I	5-6
ID	ID	Displays version information	M	ID	5-76
IF	IF	Jumps to label when condition is set	P	IF condition GOTO label	6-35
IF	IF	IF structure	P	IF ... ELSE ... END	6-47
IFWPRINT	IFWPRINT	Displays string in string window set by aux. function	M	IFWPRINT window, row, column, background color, label color = "character string", ...	5-102

Name	Abbreviation	Function		Format (Parameter)	Page
IFWPRINT	IFWPRINT	Displays string in string window set by aux. function	P	IFWPRINT window, row, column, background color, label color = "character string", ...	6-79
IGNORE	IG	Cancel ON or ONI instruction	P	IGNORE signal number	6-68
INPUT	I	Software dedicated input signals	K	DEFSIG INPUT	5-68
INRANGE	INRANGE	Returns the result of motion range check	F	INRANGE (pose variable, joint displacement values)	9-18
INSTR	INSTR	Returns the starting point of the specified string	F	INSTR (starting point, string 1, string 2)	9-12
INT	INT	Returns the integer value so numeric expression	F	INT (numeric expression)	9-14
IO	IO	Displays signal states	M	IO/E signal number	5-55
IPEAKCLR	IPEAKCLR	Displays peak current value for each joint	M	IPEAKCLR	5-84
IPEAKLOG	IPEAKLOG	Displays peak current value log	M	IPEAKLOG	5-84
JAPPRO	JA	Approaches pose in joint interpolated motion	P	JAPPRO pose variable name, distance	6-5
JDEPART	JD	Withdraws from current pose in joint interpolated motion	P	JDEPART distance	6-6
JMOVE	JM	Starts joint interpolated motion	P	JMOVE pose variable name, clamp number	6-3
KILL	KI	Initialize program stack	M	KILL	5-43
L	L	Selects the previous step.	E	L	5-5
LAPPRO	LA	Approaches pose in linear interpolated motion	P	LAPPRO pose variable name, distance	6-5
LDEPART	LD	Withdraws from current pose in linear interpolated motion	P	LDEPART distance	6-6
\$LEFT	\$LEFT	Returns the leftmost characters	F	\$LEFT (string, number of characters)	9-41
LEFTY	LE	Changes to left-hand configuration	P	LEFTY	6-31
LEN	LEN	Returns number of characters	F	LEN (string)	9-9
LIST	LI	Displays data or program listing	M	LIST (/P)/(L)/(R)/(S) prog/data...	5-17
LLIMIT	LL	Sets lower limit of robot motion	M	LLIMIT joint displacement values	5-58
LLIMIT	LL	Sets lower limit of robot motion	P	LLIMIT joint displacement values	6-86
LMOVE	LM	Starts linear interpolated motion	P	LMOVE pose variable name, clamp number	6-3
LOAD	LOAD	Loads contents of PC into robot memory	M	LOAD/Q filename	5-34
LOCK	LO	Changes priority	P	LOCK priority	6-41
LSTRACE	LSTRACE	Displays the logging data	M	LSTRACE stepper number: logging number	5-25
M	M	Modifies characters	E	M/existing characters/new characters	5-8
MAXVAL	MAXVAL	Returns the largest value	F	MAXVAL (real value1, real value 2, ...)	9-13
MC	MC	Executes monitor commands from PC programs	P	MC monitor commands	6-90

Name	Abbreviation	Function		Format (Parameter)	Page
MESSAGES	ME	Enables or disables terminal output	S	...MESSAGES...	7-4
\$MID	\$MID	Returns characters	F	\$MID (string, real value, number of characters)	9-42
MINVAL	MINVAL	Returns the smallest value	F	MINVAL (real value1, real value 2, ...)	9-13
MM/MIN	MM/M	Millimeters per minute	K	... MM/M	6-16
MM/S	MM/S	Millimeters per second	K	... MM/S	6-16
MOD	MOD	Remainder	O	... MOD ...	8-2
MSPEED	MSPEED	Returns the current monitor speed	F	MSPEED	9-17
MSPEED2	MSPEED2	Returns the current monitor speed	F	MSPEED2	9-17
MSTEP	MS	Executes a single robot motion	M	MSTEP program name, execution cycles, step number	5-40
MVWAIT	MVWAIT	Waits until given time or distance is reached	P	MVWAIT value	6-39
NCHOFF	NCHOFF	Turns OFF notch filter	P	NCHOFF	6-88
NCHON	NCHON	Turns ON notch filter	P	NCHON	6-88
NEXT	N	Skips to next step	K	CONTINUE NEXT	5-42
NOT	NOT	Logical NOT	O	... NOT ...	8-4
NULL	NULL	Returns null transformation values	F	NULL	9-23
O	O	Places the cursor in current line	E	O	5-10
OFF	OF	Turns OFF system switches	M	switch name ... OFF	5-65
OFF	OF	Turns OFF system switches	P	switch name ... OFF	6-87
OFF	OFF	Returns FALSE value	F	... OFF ...	9-10
ON	ON	Turns ON system switches	M	switch name ON	5-65
ON	ON	Turns ON system switches	P	switch name ON	6-87
ON	ON	Sets interruption condition	P	ON mode signal number CALL program name, priority	6-65
ON	ON	Sets interruption condition	P	ON mode signal number GOTO label, priority	6-65
ON	ON	Returns TRUE value	F	... ON ...	9-10
ONE	ONE	Calls specified program when error occurs	P	ONE program name	6-44
ONI	ONI	Sets interruption condition	P	ONI mode signal number CALL program name, priority	6-65
ONI	ONI	Sets interruption condition	P	ONI mode signal number GOTO label, priority	6-65
OPEINFO	OPEINFO	Displays operation information	M	OPEINFO robot number, joint number	5-85
OPEINFOCLR	OPEINFOCLR	Resets operation information	M	OPEINFOCLR	5-85
OPEN	OPEN	Opens clamps	P	OPEN clamp number	6-24
OPENI	OPENI	Opens clamps	P	OPENI clamp number	6-24
OPENS	OPENS	Turns ON/OFF open clamp signal	P	OPENS clamp number	6-27
OPLOG	OP	Displays history of operations	M	OPLOG	5-63
OR	OR	Logical OR	O	... OR ...	8-4
OUTPUT	O	Software dedicated output signals	K	DEFSIG OUTPUT	5-68
OX.PREOUT	OX	Sets the timing of OUTPUT signal generation	S OX.PREOUT....	7-5

Name	Abbreviation	Function		Format (Parameter)	Page
P	P	Displays program steps	E	P step count	5-5
PAUSE	PA	Stops execution temporarily	P	PAUSE	6-41
PCABORT	PCA	Stops execution of PC program	M	PCABORT PC program number:	10-4
PCABORT	PCA	Stops execution of PC program	P	PCABORT PC program number:	10-4
PCCONTINUE	PCC	Resumes execution of PC program	M	PCCONTINUE PC program number NEXT	10-6
PCEND	PCEN	Stop execution of PC program	M	PCEND PC program number: task number	10-5
PCEND	PCEN	Stop execution of PC program	P	PCEND PC program number: task number	10-5
PCEXECUTE	PCEX	Executes PC program	M	PCEXECUTE PC program number, program name, execution cycle, step number	10-3
PCEXECUTE	PCEX	Executes PC program	P	PCEXECUTE PC program number, program name, execution cycle, step number	10-3
PCKILL	PCK	Initializes PC program stack	M	PCKILL PC program number:	10-4
PCSCAN	PCSC	Sets cycle time for PC program execution	P	PCSCAN time	10-8
PCSTATUS	PCSTA	Displays status of PC program	M	PCSTATUS PC program number:	10-2
PCSTEP	PCSTE	Executes single step of a PC program	M	PCSTEP PC program number: program name, execution cycles, step number	10-7
PI	PI	Returns the constant π	F	PI	9-37
PLCAIN	PLCAIN	Returns value of input data in whole number	F	PLCAIN (data number)	9-16
PLCAOUT	PLCAOUT	Sets given real number value to data number	M	PLCAOUT data number = real value	5-83
PLCAOUT	PLCAOUT	Sets real number values to data number	P	PLCAOUT data number =real value	6-90
POINT	PO	Defines pose variable	M	POINT pose variable name= pose values, joint displacement values	5-47
POINT	PO	Defines pose variable	P	POINT pose variable name1= pose variable name2, joint displacement values	6-82
POINT/7	PO/7	Assigns the value of joint 7	M	POINT/7 transformation variable name = transformation values	5-49
POINT/7	PO/7	Assigns the value of joint 7	P	POINT/7 transformation variable name = transformation variable name2	6-83
POINT/A	PO/A	Assigns A component value	M	POINT/A transformation variable name = transformation values	5-49
POINT/A	PO/A	Assigns A component value	P	POINT/A transformation variable name 1= transformation variable name2	6-83
POINT/O	PO/O	Assigns O component value	M	POINT/O transformation variable name = transformation values	5-49

Name	Abbreviation	Function		Format (Parameter)	Page
POINT/O	PO/O	Assigns O component value	P	POINT/O transformation variable name 1= transformation variable name2	6-83
POINT/OAT	PO/OAT	Assigns O, A and T component values	M	POINT/OAT transformation variable name = transformation values	5-49
POINT/OAT	PO/OAT	Assigns O, A and T component values	P	POINT/OAT transformation variable name 1 = transformation variable name2	6-83
POINT/T	PO/T	Assigns T component value	M	POINT/T transformation variable name = transformation values	5-49
POINT/T	PO/T	Assigns T component value	P	POINT/T transformation variable name1 = transformation variable name2	6-83
POINT/X	PO/X	Assigns X component value	M	POINT/X transformation variable name = transformation values	5-49
POINT/X	PO/X	Assigns X component value	P	POINT/X transformation variable name1 = transformation variable name2	6-83
POINT/Y	PO/Y	Assigns Y component value	M	POINT/Y transformation variable name = transformation values	5-49
POINT/Y	PO/Y	Assigns Y component value	P	POINT/Y transformation variable name1 = transformation variable name2	6-83
POINT/Z	PO/Z	Assigns Z component value	M	POINT/Z transformation variable name = transformation values	5-49
POINT/Z	PO/Z	Assigns Z component value	P	POINT/Z transformation variable name1 = transformation variable name2	6-83
POWER	POWER	MOTOR POWER switch ON/OFF status	S	switch (POWER)	7-11
#PPOINT	#PPOINT	Returns joint displacement values	F	#PPOINT (jt1,jt2,jt3,jt4,jt5,jt6)	9-29
PREFETCH.SIGINS	PR	Enables or disables early processing of I/O signals	S PREFETCH.SIGINS....	7-5
PRIME	PRIM	Sets up for program execution	M	PRIME program, execution cycles, step number	5-38
PRINT	PRIN	Displays data on the terminal	M	PRINT device number: print data, ...	5-99
PRINT	PRIN	Displays data on the terminal	P	PRINT device number: print data, ...	6-76
PRIORITY	PRIORITY	Returns priority number	F	PRIORITY	9-16
PROMPT	PROMPT	Displays message on terminal and waits for input	P	PROMPT device number: character string, variables	6-77
PULSE	PU	Turns ON signal for a given period of time	M	PULSE signal number, time	5-86
PULSE	PU	Turns ON signal for a given period of time	P	PULSE signal number, time	6-59
QTOOL	Q	Tool transformation during block teaching	S QTOOL....	7-6
R	R	Replaces characters	E	R character string	5-9

Name	Abbreviation	Function		Format (Parameter)	Page
RANDOM	RANDOM	Returns random number from 0 to 1	F	RANDOM	9-37
REC_ACCEPT	REC	Enables/disables RECORD/PROGRAM CHANGE function	M	REC_ACCEPT	5-80
RELAX	RELAX	Turns OFF clamp signals (close and open)	P	RELAX clamp number	6-26
RELAXI	RELAXI	Turns OFF clamp signals (close and open)	P	RELAXI clamp number	6-26
RELAXS	RELAXS	Turns ON/OFF clamp signal (close and open)	P	RELAXS clamp number	6-27
RENAME	REN	Changes program name	M	RENAME new program name = existing program name	5-20
REP_ONCE	REP	Sets if repeat cycle is done once or continuously	S	...REP_ONCE...	7-8
REPEAT	REPEAT	TEACH/REPEAT switch ON/OFF status	S	switch (REPEAT)	7-13
RESET	RES	Turns OFF all external output signals	M	RESET	5-87
RESET	RES	Turns OFF all external output signals	P	RESET	6-58
RETRACE	RETRACE	Releases memory set aside by SETTRACE	M	RETRACE	5-24
RETURN	RET	Returns to the caller program	P	RETURN	6-36
RETURNE	RETURNE	Returns to step after the error	P	RETURNE	6-44
RGSO	RGSO	Servoing switch ON/OFF status	S	switch (RGSO)	7-12
\$RIGHT	\$RIGHT	Returns the rightmost characters	F	\$RIGHT (string, number of characters)	9-41
RIGHTY	RI	Changes to right-hand configuration	P	RIGHTY	6-31
RPS	RP	Calls program selected by signals	S RPS....	7-7
RUN	RUN	HOLD/RUN switch status	S	switch (RUN)	7-14
RUNMASK	RU	Masks signals	P	RUNMASK start number, number of signals	6-60
RX	RX	Rotation about X Axis	F	RX (angle)	9-28
RY	RY	Rotation about Y Axis	F	RY (angle)	9-28
RZ	RZ	Rotation about Z Axis	F	RZ (angle)	9-28
S	S	Selects program step	E	S step number	5-4
SAVE	SA	Stores program/variable into a PC	M	SAVE/SEL filename =program name...	5-29
SAVE/ELOG	SA/ELOG	Stores error log into PC	M	SAVE/ELOG file name	5-31
SAVE/P,L,R,S,A	SA/P...	Stores data into PC	M	SAVE (/P)/(L)/(R)/(S)/(A)/SEL file name=program name...	5-31
SAVE/ROB	SA/ROB	Stores robot data into PC	M	SAVE/ROB file name	5-31
SAVE/SYS	SA/SYS	Stores system data into PC	M	SAVE/SYS file name	5-31
SCALL	SCA	Jumps to subroutine	P	SCALL string expression, variable	6-43
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	5-92

Name	Abbreviation	Function		Format (Parameter)	Page
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	6-69
SCNTRESET	SCNTR	Resets internal counter value	M	SCNTRESET counter signal number	5-93
SCNTRESET	SCNTR	Resets internal counter value	P	SCNTRESET counter signal number	6-70
SCREEN	SC	Control terminal display	S	... SCREEN...	7-7
SETHOME	SETH	Defines home position 1	M	SETHOME accuracy, HERE	5-62
SET2HOME	SET2	Defines home position 2	M	SET2HOME accuracy, HERE	5-62
SETPICK	SETPICK	Sets time to start clamp close control	M	SETPICK time1,..., time8	5-97
SETPICK	SETPICK	Sets time to start clamp close control	P	SETPICK time1,..., time8	6-73
SETPLACE	SETPLACE	Sets time to start clamp open control	M	SETPLACE time1,..., time8	5-98
SETPLACE	SETPLACE	Sets time to start clamp open control	P	SETPLACE time1,..., time8	6-73
SETTRACE	SETTRACE	Reserves necessary memory to log data	M	SETTRACE step count	5-24
SFLK	SFLK	Turns ON/OFF signal in given cycle time	M	SFLK signal number = time	5-93
SFLK	SFLK	Turns ON/OFF signal in given cycle time	P	SFLK signal number = time	6-70
SFLP	SFLP	Turns ON/OFF signal using set/reset signal	M	SFLP output signal = set signal expression, reset signal expression	5-94
SFLP	SFLP	Turns ON/OFF signal using set/reset signal	P	SFLP output signal = set signal expression, reset signal expression	6-71
SHIFT	SHIFT	Returns shifted transformation values	F	SHIFT (trans BY X shift, Y shift, Z shift)	9-30
SIG	SIG	Returns logical AND of signal states	F	SIG (signal number, ...)	9-3
SIGNAL	SIG	Turns signals ON/OFF	M	SIGNAL signal number....	5-88
SIGNAL	SIG	Turns signals ON/OFF	P	SIGNAL signal number....	6-58
SIN	SIN	Returns the sine value	F	SIN (real values)	9-37
SLOW_REPEAT	SL	Sets the repeat speed in slow repeat mode	M	SLOW REPEAT	5-80
SLOW_START	SLOW_START	Enables or disables the slow start function	S	...SLOW_START...	7-19
SOUT	SO	Outputs signal when condition is set	M	SOUT signal number = signal expression	5-96
SOUT	SO	Outputs signal when condition is set	P	SOUT signal number = signal expression	6-72
\$SPACE	\$SPACE	Returns blanks	F	\$SPACE (number of blanks)	9-40
SPEED	SP	Sets monitor speed	M	SPEED monitor speed	5-37
SPEED	SP	Sets program speed	P	SPEED speed, rotational speed, ALWAYS	6-16
SQRT	SQRT	Returns the square root	F	SQRT (real values)	9-37

Name	Abbreviation	Function		Format (Parameter)	Page
STABLE	STA	Holds robot motion for a given time	P	STABLE time	6-4
STATUS	STA	Displays system status	M	STATUS	5-52
STEP	STE	Executes a single step of a program	M	STEP program name, execution cycles, step number	5-40
STIM	STI	Turns ON timer signal	M	STIM timer signal = input signal number, time	5-96
STIM	STI	Turns ON timer signal	P	STIM timer signal = input signal number, time	6-73
STOP	STO	Terminates execution cycle	P	STOP	6-42
STP_ONCE	ST	Sets the execution as one step at a time or continuous	S	...STP_ONCE...	7-8
STPNEXT	STPNEXT	Executes the next step	M	STPNEXT	5-43
SWAIT	SW	Waits for desired signal state	P	SWAIT signal number,...	6-62
SWITCH	SW	Sets system switches	M	SWITCH switch name...=ON (=OFF)	5-64
SWITCH	SWITCH	Returns condition of system switch	F	SWITCH (switch name)	9-14
SYSDATA	SYSDATA	Returns parameters	F	SYSDATA (keyword, opt1)	9-19
SYSINIT	SY	Initialize system	M	SYSINIT	5-74
T	T	Enables teaching by TP in editor mode	E	T variable name	5-13
TASK	TASK	Returns execution status of program	F	TASK (task number)	9-15
TDRAW	TDRA	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	6-9
TEACH	TE	Teaches pose	M	TEACH pose variable name	5-50
TEACH_LOCK	TEACH_LOCK	TEACHLOCK switch ON/OFF status	S	switch (TEACH_LOCK)	7-12
THEN	THEN	IF structure	K	IF logical expression THEN ... ELSE ... END	6-47
TIME	TI	Sets and displays date and time	M	TIME yy-mm-dd hh:mm:ss	5-57
\$TIME	\$TIME	Returns system time	F	\$TIME	9-48
TIMER	TI	Sets timer	P	TIMER number = time	6-86
TIMER	TIMER	Returns timer value	F	TIMER (timer number)	9-5
TO	TO	FOR structure	K	FOR ... TO ... END	6-53
TOOL	TOOL	Defines tool transformation values	M	TOOL transformation values	5-61
TOOL	TOOL	Defines tool transformation values	P	TOOL transformation values	6-85
TOOL	TOOL	Returns tool transformation values	F	TOOL	9-32
TPLIGHT	TPLIGHT	Turns on the TP backlight	M	TPLIGHT	5-83
TPLIGHT	TPLIGHT	Turns on teach pendant backlight	P	TPLIGHT	6-91
TRACE	TRACE	Logs and traces programs	M	TRACE stepper number: ON/OFF	5-23
TRACE	TRACE	Logs and traces programs	P	TRACE stepper number: ON/OFF	6-94
TRADD	TRADD	Returns the sum of traverse axis and transformation values	F	TRADD (transformation values)	9-33
TRANS	TRANS	Returns transformation values	F	TRANS (X, Y, Z, O, A, T)	9-27
TRIGGER	TRIGGER	TRIGGER switch ON/OFF status	S	switch (TRIGGER)	7-10

Name	Abbreviation	Function		Format (Parameter)	Page
TRSUB	TRSUB	Returns the difference of traverse axis and transformation values	F	TRSUB (transformation values)	9-33
TWAIT	TW	Wait for a given period of time	P	TWAIT time	6-38
TYPE	TY	Displays data on the terminal	M	TYPE device number: print data, ...	5-99
TYPE	TY	Displays data on the terminal	P	TYPE device number: print data, ...	6-76
ULIMIT	UL	Sets lower limit of robot motion	M	ULIMIT joint displacement values	5-58
ULIMIT	UL	Sets lower limit of robot motion	P	ULIMIT joint displacement values	6-86
UNTIL	U	DO structure	P	DO ... UNTIL logical expression	6-49
UTIMER	UTIMER	Sets user timer	P	UTIMER @timer variable = timer value	6-87
UTIMER	UTIMER	Returns value of timer variable	F	UTIMER (@timer)	9-17
UWRIST	UW	Changes wrist configuration	P	UWRIST	6-32
VAL	VAL	Returns real value	F	VAL (string, code)	9-11
WAIT	WA	Waits for specified condition	P	WAIT condition	6-37
WEIGHT	WE	Sets load mass data	M	WEIGHT load mass, x, y, z	5-77
WEIGHT	WE	Sets load mass data	P	WEIGHT load mass, x, y, z	6-89
WHERE	W	Displays current robot pose	M	WHERE display mode	5-54
WHILE	WH	DO structure	P	WHILE ... DO ... END	6-49
WS.ZERO	WS.ZERO	Changes the weld processing	S	...WS.ZERO...	7-17
WS_COMPOFF	WS_COMPOFF	Changes the output timing of WS signal	S	...WS_COMPOFF...	7-16
XD	XD	Cuts step and pastes on the buffer	E	XD step count	5-11
XFER	XF	Copies and transfers steps	M	XFER destination program name, step number 1= source program name, step number 2, step count	5-21
XMOVE	XM	Moves until the signal changes	P	XMOVE mode pose name TILL signal number	6-11
XOR	XOR	Exclusive logical OR	O	XOR	8-4
XP	XP	Inserts contents of paste buffer	E	XP	5-12
XQ	XQ	Inserts contents of paste buffer in reverse order	E	XQ	5-12
XS	XS	Displays contents of paste buffer	E	XS	5-13
XY	XY	Copies step and pastes on the buffer	E	XY step count	5-11
ZSIGSPEC	ZSIG	Sets number of installed signals	M	ZSIGSPEC	5-66
ZZERO	ZZE	Sets zeroing data	M	ZZERO	5-71
FALSE	FALSE	Returns FALSE value	F	... FALSE ...	9-10
TRUE	TRUE	Returns TRUE value	F	... TRUE ...	9-10
-	-	Subtraction	O	... - ...	8-2
*	*	Multiplication	O	... * ...	8-2
/	/	Division	O	... / ...	8-2
^	^	Power	O	... ^ ...	8-2
+	+	Addition	O	... + ...	8-2
+	+	String concatenation	O	... + ...	8-9
<	<	less than	O	... < ...	8-3
<=	<=	less than or equal to	O	... <= ...	8-3
<>	<>	not equal to	O	... <> ...	8-3
=<	=<	less than or equal to	O	... =< ...	8-3

Name	Abbreviation	Function		Format (Parameter)	Page
=	=	equal to	O	... = ...	8-3
=>	=>	greater than or equal to	O	... => ...	8-3
>	>	greater than	O	... > ...	8-3
>=	>=	greater than or equal to	O	... >= ...	8-3

APPENDIX 3 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 4 LIMITATION OF SIGNAL NUMBERS

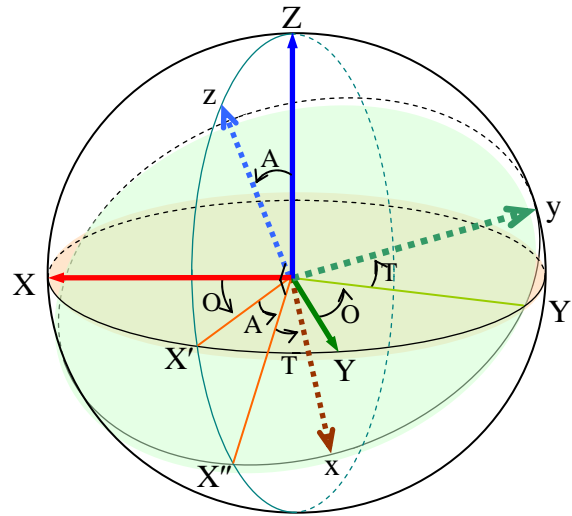
No	M, P, F*		Output Signals	Input Signals	Internal Signals
1	BITS	M P	1 to maxsig**	-----	2001 to 2256
2	BITS	F	1 to maxsig**	1001 to maxsig**	2001 to 2256
3	DEFSIG	M	1 to maxsig**	1001 to maxsig**	-----
4	DLYSIG	M P	1 to 64***	-----	2001 to 2256
5	ON, ONI	P	-----	1001 to 1064***	2001 to 2256
6	PULSE	M P	1 to 64***	-----	2001 to 2256
7	RUNMASK	P	1 to 64***	-----	2001 to 2256
8	SIGNAL	M P	1 to maxsig**	-----	2001 to 2256
9	SIG	F	1 to maxsig**	1001 to maxsig**	2001 to 2256
10	SWAIT	P	1 to maxsig**	1001 to maxsig**	2001 to 2256
11	XMOVE	P	-----	1001 to 1064***	2001 to 2256
12					
13					

NOTE * M= monitor command, P= program instruction, F= function

NOTE ** maxsig: number of external signals installed
32 (standard), 64 (option), 96 (option), 128 (option)

NOTE*** 64(1064): the maximum number that can be specified is 32(1032) if the standard I/O module is installed.

APPENDIX 5 EULER'S O,A,T ANGLES



The posture of a coordinate system $\Sigma(x,y,z)$ with respect to the base coordinate system $\Sigma(X,Y,Z)$ 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 $\Sigma(x,y,z)$ and $\Sigma(X,Y,Z)$ are located at 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 $\Sigma(X,Y,Z)$ to coincide with the coordinate system $\Sigma(x,y,z)$. The order of rotation must not be changed, or else will result differently.

1. O rotation of the coordinate system $\Sigma(X,Y,Z)$ around Z axis. (This moves $\Sigma(X,Y,Z)$ to $\Sigma(X',Y',Z)$.)
2. A rotation of the coordinate system $\Sigma(X',Y',Z)$ around Y' axis. (This moves $\Sigma(X',Y',Z)$ to $\Sigma(X'',Y',z)$.)
3. T rotation of the coordinate system $\Sigma(X'',Y',z)$ around z axis. (This moves $\Sigma(X'',Y',z)$ to $\Sigma(x,y,z)$.)

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.



MEMO

Kawasaki Robot Controller D Series
AS LANGUAGE REFERENCE MANUAL

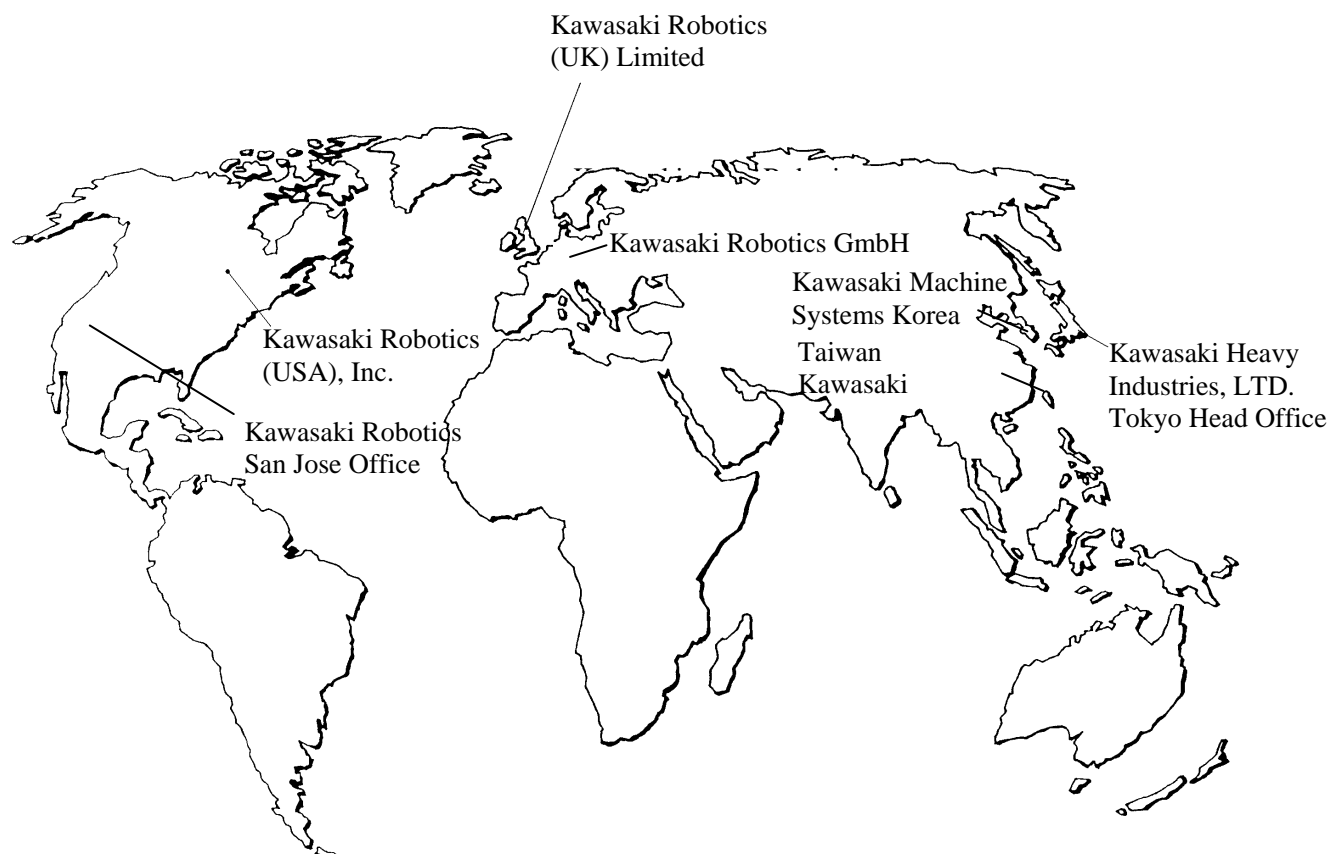
August 2002 : 1st Edition
December 2002 : 2nd Edition

Publication : KAWASAKI HEAVY INDUSTRIES, LTD.

90209-1017DEB

All rights reserved. Copyright © 2002 by KAWASAKI HEAVY INDUSTRIES, LTD.

Kawasaki Robot



KAWASAKI HEAVY INDUSTRIES, LTD. ROBOT DIVISION

Kawasaki Heavy Industries, Ltd.	41, 2-Chome Hamamatsu-cho Minato-ku Tokyo, 105-6116 JAPAN(World Trade Center building)	TEL 81-(0)3-3435-2501 FAX 81-(0)3-3437-9880
Kawasaki Robotics (USA), Inc.	28059 Center Oaks Court Wixom, Michigan, 48393 U.S.A	TEL 1-248-305-7610 FAX 1-248-305-7618
Kawasaki Robotics San Jose Office.	1020 Commercial Street, Suites 104-106 San Jose, CA 95112 U.S.A	TEL 1-408-392-0290 FAX 1-408-392-0294
Kawasaki Robotics GmbH	29 Sperberweg, 41468 Neuss, GERMANY	TEL 49-(0)2131-3426-0 FAX 49-(0)2131-3426-22
Kawasaki Robotics (UK) Limited	Units 6&7, Easter Court, Europa Boulevard Westbrook, Warrington, WA5 5ZB, England	TEL 44 -1925-71-3000 FAX 44 -1925-71-3001
Kawasaki Machine Systems Korea, Ltd	3Fl(307), Industrial Complex Support Bldg., 637 Kojan-Don, Namdong-Gu, Inchon, 405-310 Korea	TEL 82-32-821-6941/5 FAX 82-32-821-6947
Taiwan Kawasaki Robot Center	1F 122 Ching Hsan 7 Street, Hsinchu, Taiwan R.O.C.	TEL 886-3-666-1558 FAX 886-3-666-1559
Kawasaki Heavy Industries, Akashi	1-1, Kawasaki-cho Akashi, 673-8666 JAPAN	TEL 81-(0)78-921-1560 FAX 81-(0)78-923-6548

* All descriptions in this booklet are subject to change for improvement without prior notice.