FANUC Robot series

R-J3 ∕ R-J3*i*B ∕ R-30*i*A CONTROLLER

CIMPLICITY HMI for Robots

OPERATOR'S MANUAL

Before using the Robot, be sure to read the "FANUC Robot Safety Manual (B-80687EN)" and understand the content.

This manual can be used with controllers labeled R-30*i*A or R-J3*i*C. If you have a controller labeled R-J3*i*C, you should read R-30*i*A as R-J3*i*C throughout this manual.

- No part of this manual may be reproduced in any form.
- All specifications and designs are subject to change without notice.

The products in this manual are controlled based on Japan's "Foreign Exchange and Foreign Trade Law". The export from Japan may be subject to an export license by the government of Japan.
Further, re-export to another country may be subject to the license of the government of the country from where the product is re-exported. Furthermore, the product may also be controlled by re-export regulations of the United States government.
Should you wish to export or re-export these products, please contact FANUC for advice.

In this manual we have tried as much as possible to describe all the various matters.
However, we cannot describe all the matters which must not be done, or which cannot be done, because there are so many possibilities.
Therefore, matters which are not especially described as possible in this manual should be regarded as "impossible".

I. SAFETY

1   SAFETY PRECAUTIONS

For the safety of the operator and the system, follow all safety precautions when operating a robot and its peripheral devices installed in a work cell.

1.1   **OPERATOR SAFETY**

Operator safety is the primary safety consideration. Because it is very dangerous to enter the operating space of the robot during automatic operation, adequate safety precautions must be observed.

The following lists the general safety precautions. Careful consideration must be made to ensure operator safety.

(1) Have the robot system operators attend the training courses held by FANUC.

> FANUC provides various training courses. Contact our sales office for details.

(2) Even when the robot is stationary, it is possible that the robot is still ready to move state and is waiting for a signal. In this state, the robot is regarded as still in motion. To ensure operator safety, provide the system with an alarm to indicate visually or aurally that the robot is in motion.
(3) Install a safety fence with a gate so that no operator can enter the work area without passing through the gate. Equip the gate with an interlock that stops the robot when the gate is opened.

> The controller is designed to receive this interlock signal. When the gate is opened and this signal received, the controller stops the robot in an emergency. For connection, see Fig.1.1.

(4) Provide the peripheral devices with appropriate grounding (Class 1, Class 2, or Class 3).
(5) Try to install the peripheral devices outside the work area.
(6) Draw an outline on the floor, clearly indicating the range of the robot motion, including the tools such as a hand.
(7) Install a mat switch or photoelectric switch on the floor with an interlock to a visual or aural alarm that stops the robot when an operator enters the work area.
(8) If necessary, install a safety lock so that no one except the operator in charge can turn on the power of the robot.

> The circuit breaker installed in the controller is designed to disable anyone from turning it on when it is locked with a padlock.

(9) When adjusting each peripheral device independently, be sure to turn off the power of the robot.
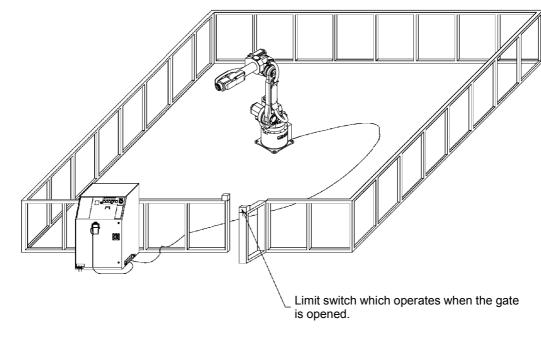

Limit switch which operates when the gate is opened.

Panel board

FENCE1
FENCE2

Note) Terminals FENCE1 and FENCE2 are on the PC board in the operator's panel.

Fig.1.1 (a)   Safety Fence and Safety gate (For R-J3*i*B CONTROLLER)

Limit switch which operates when the gate is opened.

Panel board
EAS1
EAS11
EAS2
EAS21

Note) Terminals EAS1, 11 and EAS2, 21 are on the PC board on the operator's panel. Refer to the R-30*i*A CONTROLLER MAINTENANCE MANUAL.
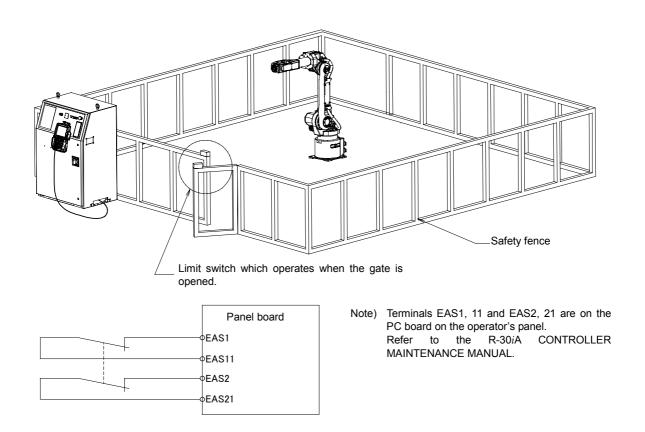
Fig.1.1(b)   Safety Fence and Safety (For R-30*i*A CONTROLLER)

1.1.1   **Operator Safety**

The operator is a person who operates the robot system.   In this sense, a worker who operates the teach pendant is also an operator. However, this section does not apply to teach pendant operators.

(1)   If it is not necessary for the robot to operate, turn off the power of the robot controller or press the EMERGENCY STOP button, and then proceed with necessary work

(2)   Operate the robot system at a location outside the work area.

(3)   Install a safety fence with a safety gate to prevent any worker other than the operator from entering the work area unexpectedly and also to prevent the worker from entering a dangerous area.

(4)   Install an EMERGENCY STOP button within the operator's reach.

> The robot controller is designed to be connected to an external EMERGENCY STOP button.   With this connection, the controller stops the robot operation when the external EMERGENCY STOP button is pressed. See the diagram below for connection.
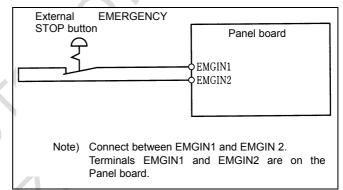

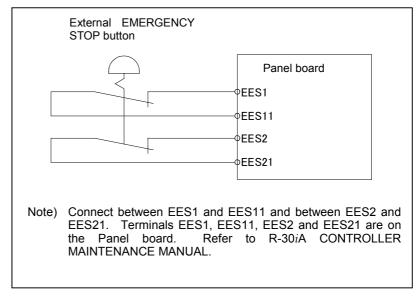**Fig.1.1.1(a)   Connection Diagram for External Emergency Stop Switch (For R-J3*i*B CONTROLLER)**

Fig.1.1.1(b)   Connection Diagram for External Emergency Stop Switch (For R-30*i*A CONTROLLER)

1.1.2   **Safety of the Teach Pendant Operator**

While teaching the robot, it is necessary for the operator to enter the work area of the robot. It is particularly necessary to ensure the safety of the teach pendant operator.

(1) Unless it is specifically necessary to enter the robot work area, carry out all tasks outside the area.
(2) Before teaching the robot, check that the robot and its peripheral devices are all in the normal operating condition.
(3) When entering the robot work area and teaching the robot, be sure to check the location and condition of the safety devices (such as the EMERGENCY STOP button and the deadman's switch on the teach pendant).

The teach pendant supplied by FANUC is provided with a teach pendant enable switch and a deadman's switch in addition to the EMERGENCY STOP button. The functions of each switch are as follows.	
EMERGENCY STOP button :	Pressing this button stops the robot in an emergency, irrespective to the condition of the teach pendant enable switch.
Deadman's switch :	The function depends on the state of the teach pendant enable switch.
When the enable switch is on -	Releasing the finger from the dead man's switch stops the robot in an emergency.
When the enable switch is off-The deadman's switch is ineffective	

NOTE
   The deadman's switch is provided so that the robot operation can be stopped simply by releasing finger from the teach pendant in case of emergency.

(4) The teach pendant operator should pay careful attention so that no other workers enter the robot work area.

NOTE

    In addition to the above, the teach pendant enable switch and the deadman's switch also have the following function.By pressing the deadman's switch while the enable switch is on, the emergency stop factor (normally the safety gate) connected to the controller is invalidated.   In this case, it is possible for an operator to enter the fence during teach operation without pressing the EMERGENCY STOP button.   In other words, the system understands that the combined operations of pressing the teach pendant enable switch and pressing the deadman's switch indicates the start of teaching.

    The teach pendant operator should be well aware that the safety gate is not functional under this condition and bear full responsibility to ensure that no one enters the fence during teaching.

---

(5) When entering the robot work area, the teach pendant operator should enable the teach pendant whenever he or she enters the robot work area.   In particular, while the teach pendant enable switch is off, make certain that no start command is sent to the robot from any operator's panel other than the teach pendant.

The teach pendant, operator panel, and peripheral device interface send each robot start signal. However the validity of each signal changes as follows depending on the ON/OFF switch on the Teach pendant and the three modes switch on the Operator's panel and Remote condition on the software.

Operator 's panel Three modes switch	Teach pendant ON/OFF switch	Software remote condition	Teach pendant	Operator's panel	Peripheral devices
T1/T2 AUTO (Except RIA)	On	Independent	Allowed to start	Not allowed	Not allowed
AUTO	Off	Remote OFF	Not allowed	Allowed to start	Not allowed
AUTO	Off	Remote ON	Not allowed	Not allowed	Allowed to start

NOTE)   When starting the system using the teach pendant in the RIA specification, the three modes switch should be T1/T2.

---

(6) To start the system using the operator's box, make certain that nobody is in the robot work area and that there are no abnormal conditions in the robot work area.

(7) When a program is completed, be sure to carry out a test run according to the procedure below.

   (a) Run the program for at least one operation cycle in the single step mode at low speed.
   (b) Run the program for at least one operation cycle in the continuous operation mode at low speed.
   (c) Run the program for one operation cycle in the continuous operation mode at the intermediate speed and check that no abnormalities occur due to a delay in timing.
   (d) Run the program for one operation cycle in the continuous operation mode at the normal operating speed and check that the system operates automatically without trouble.
   (e) After checking the completeness of the program through the test run above, execute it in the automatic operation mode.

(8) While operating the system in the automatic operation mode, the teach pendant operator should leave the robot work area.

1.1.3   **Safety During Maintenance**

For the safety of maintenance personnel, pay utmost attention to the following.

(1) Except when specifically necessary, turn off the power of the controller while carrying out maintenance. Lock the power switch, if necessary, so that no other person can turn it on.

(2) When disconnecting the pneumatic system, be sure to reduce the supply pressure.

(3) Before the start of teaching, check that the robot and its peripheral devices are all in the normal operating condition.

(4) If it is necessary to enter the robot work area for maintenance when the power is turned on, the worker should indicate that the machine is being serviced and make certain that no one starts the robot unexpectedly.

(5) Do not operate the robot in the automatic mode while anybody is in the robot work area.

(6) When it is necessary to maintain the robot alongside a wall or instrument, or when multiple workers are working nearby, make certain that their escape path is not obstructed.

(7) When a tool is mounted on the robot, or when any moving device other than the robot is installed, such as belt conveyor, pay careful attention to its motion.

(8) If necessary, have a worker who is familiar with the robot system stand beside the operator's panel and observe the work being performed.   If any danger arises, the worker should be ready to press the EMERGENCY STOP button at any time.

(9) When replacing or reinstalling components, take care to prevent foreign matter from entering the system.

(10) When handling each unit or printed circuit board in the controller during inspection, turn off the power of the controller and also turn off the circuit breaker to protect against electric shock.

(11) When replacing parts, be sure to use those specified by FANUC. In particular, never use fuses or other parts of non-specified ratings.   They may cause a fire or result in damage to the components in the controller.

1.2 SAFETY OF THE TOOLS AND PERIPHERAL DEVICES

1.2.1 Precautions in Programming

(1) Use a limit switch or other sensor to detect a dangerous condition and, if necessary, design the program to stop the robot when the sensor signal is received.

(2) Design the program to stop the robot when an abnormal condition occurs in any other robots or peripheral devices, even though the robot itself is normal.

(3) For a system in which the robot and its peripheral devices are in synchronous motion, particular care must be taken in programming so that they do not interfere with each other.

(4) Provide a suitable interface between the robot and its peripheral devices so that the robot can detect the states of all devices in the system and can be stopped according to the states.

1.2.2 Precautions for Mechanism

(1) Keep the component cells of the robot system clean, and operate the robot in an environment free of grease, water, and dust.

(2) Employ a limit switch or mechanical stopper to limit the robot motion so that the robot does not come into contact with its peripheral devices or tools.

1.3   SAFETY OF THE ROBOT MECHANISM

1.3.1   Precautions in Operation

(1) When operating the robot in the jog mode, set it at an appropriate speed so that the operator can manage the robot in any eventuality.
(2) Before pressing the jog key, be sure you know in advance what motion the robot will perform in the jog mode.

1.3.2   Precautions in Programming

(1) When the work areas of robots overlap, make certain that the motions of the robots do not interfere with each other.
(2) Be sure to specify the predetermined work origin in a motion program for the robot and program the motion so that it starts from the origin and terminates at the origin. Make it possible for the operator to easily distinguish at a glance that the robot motion has terminated.

1.3.3   Precautions for Mechanisms

(1) Keep the work area of the robot clean, and operate the robot in an environment free of grease, water, and dust.

1.4   **SAFETY OF THE END EFFECTOR**

1.4.1   **Precautions in Programming**

(1) To control the pneumatic, hydraulic and electric actuators, carefully consider the necessary time delay after issuing each control command up to actual motion and ensure safe control.

(3) Provide the end effector with a limit switch, and control the robot system by monitoring the state of the end effector.

1.5   **SAFETY IN MAINTENANCE**

(1) Never enter the robot work area while the robot is operating. Turn off the power before entering the robot work area for inspection and maintenance.

(2) If it is necessary to enter the robot work area with the power turned on, first press the EMERGENCY STOP button on the operator's box.

(3) When replacing or reinstalling components, take care to prevent foreign matter from entering the system. When replacing the parts in the pneumatic system, be sure to reduce the pressure in the piping to zero by turning the pressure control on the air regulator.

(4) When handling each unit or printed circuit board in the controller during inspection, turn off the power of the controller and turn off the circuit breaker to protect against electric shock.

(5) When replacing parts, be sure to use those specified by FANUC. In particular, never use fuses or other parts of non-specified ratings. They may cause a fire or result in damage to the components in the controller.

(6) Before restarting the robot, be sure to check that no one is in the robot work area and that the robot and its peripheral devices are all in the normal operating state.

1.6   **WARNING LABEL**
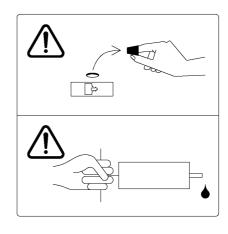
(1) Greasing and degreasing label

Fig. 1.6 (a) Greasing and Degreasing Label

Description

When greasing and degreasing, observe the instructions indicated on this label.

1) When greasing, be sure to keep the grease outlet open.
2) Use a manual pump to grease.
3) Be sure to use a specified grease.

(2)   Step-on prohibitive label

Fig. 1.6 (b) Step-on Prohibitive Label

Description

Do not step on or climb the robot or controller as it may adversely affect the robot or controller and you may get hurt if you lose your footing as well.

(3)   High-temperature warning label

Fig. 1.6 (c) High-temperature warning label

Description

Be cautious about a section where this label is affixed, as the section generates eat.   If you have to inevitably touch such a section when it is hot, use a rotective provision such as heat-resistant gloves.

TABLE OF CONTENTS

Ⅱ. OPERATION

1   PREFACE

CIMPLICITY HMI allows the user at a personal computer on which the CIMPLICITY HMI software is installed to remotely monitor, record, analyze, and control data of robot control units installed in a factory. To do this, CIMPLICITY uses an Ethernet communication feature that is provided as a standart feature of the robot control unit.

This document handles both CIMPLICITY version 4.01 with service pack 10 or later, and version 5.5.

And the pictures of CIMPLICITY screen in this document are captured on CIMPLICITY version 5.5.

The difference between CIMPLICITY version 4.01 and 5.5 is that, CIMPLICITY version 4.01 can connect to robot control unit with the device model type both GE Fanuc Series 90-30 and 90-70, but CIMPLICITY version 5.5 can connect with the device model type only GE Fanuc Series 90-30.

If you update CIMPLICITY version 4.01 to 5.5 and use the old project, check the model type of all devices registered in the project.

2 ENVIRONMENT

2.1 **REQUIRED SOFTWARE**

CIMPLICITY HMI does not require any special options.

R-J3 and R-J3*i*B do not require any special option software, either.

R-30*i*A(R-J3*i*C) requires "HMI Device (SNPX)" option software if "FRA Params" option software is installed.

The applicable series and edition of the R-J3 system software is 7D70, A1 or later. And R-J3*i*B system software 7D80, 45 or later, and 7D81, 09 or later, and R-30*i*A(R-J3*i*C) support the new functions such as Multiplex Communicating.

	Standard functions	New functions	Required software option
R-J3	7D70, A1 or later	Not supported	None
R-J3*i*B	All versions	7D80, 45 or later 7D81, 09 or later	None
R-30iA(R-J3iC) Standard	All versions	All versions	None
R-30iA(R-J3iC) FRA Params	All versions	All versions	HMI Device (SNPX)

2.2 **RESTRICTION ON USE WITH OTHER OPTIONAL FUNCTIONS**
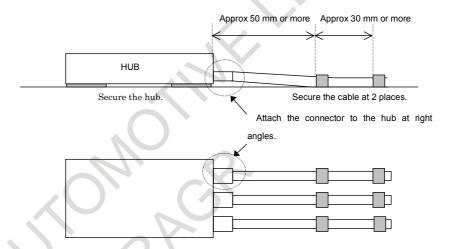
CIMPLICITY HMI cannot be used together with the robot link function.

2.3   **NETWORK**

- For the Ethernet Cable, use noise-resistant twisted-pair cable for 10BASE-T.
- For the hub (line concentrator) required for implementing a network, use a switching hub where possible. FANUC recommends a switching hub of which modular connector has no metal shield. This is prevent noise from entering the network from the backbone.
- Perform cabling so that the hub and Ethernet cables make good contact.

  For example, install the hub and cables as follows:

- The installation environment of the hub must conform the specification defined by the manufacturer of the hub. For example, the hub must be installed in a vibration-free, dust-proof place.
- Sufficient measures should be taken to prevent noise.

> ⚠ **CAUTION**
> The customer is responsible for purchasing cables and hubs. Make Provisions to protect yourself in the event of a hub failure, by purchasing spare parts.

3

CAUTIONS (BE SURE TO READ THE FOLLOWING:)

The following provides basic items the customer should bear in mind when using this function. The customer's system must at least satisfy these items. Accordingly, the customer may need to take additional safety measures according to the customer's safety standards for the system.

3.1   **CABLING AND CONNECTION**

- Run the Ethernet cable so that an operator can move safely and freely about with no tripping hazard.
- Run the Ethernet cable so as to avoid subjecting it to noise. Keep the cable clear of any noise source.
- Run the hub power cable clear of any traffic so as to avoid a tripping hazard and to avoid inadvertently disconnecting the power cable.
- Consider the location of cabling and installation of the switching hub so that the hub and the cable connectors make good contact.

The following provides very important items. Be sure to observe these items.
- After connecting one end of the Ethernet cable to the hub, and attaching the other end of the cable to the Ethernet cable jack on the main board of the robot controller, the LED on the printed circuit board located on the far side if the cable jack lights in orange. The hub and robot controller must both have been turned on. If the LED does not light, the main board may have a problem. Contact the FANUC Service Center.
- On the front of the hub, there is an LED indicator that provides the status of communications. If communication cannot be performed normally, the LED must be checked. For this reason, the hub must be installed so that the user can easily check the LED.

4

SETTINGS ON THE ROBOT SIDE

Before this function can be used, settings related to the Ethernet network must be made on the robot side.

4.1 **NETWORK-RELATED SETTINGS**

This function uses Ethernet. First, make settings for using Ethernet.
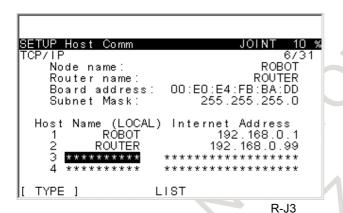
4.1.1 **Setting the Host Names, Internet (IP) Addresses, and Subnet Mask**
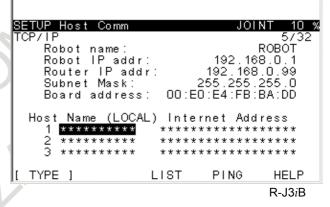
Display the protocol setup screen of the host communication setup screen.
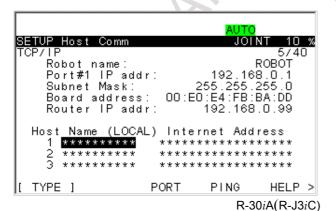
On this setup screen, set the following items:

- Node name of the controller
- Node name of the router (Even when there is no router, be sure to enter a dummy node name.)
- Subnet mask
- Node names and IP addresses of all controllers (including this controller) that involve robot link communication

For example, the following settings are made:

R-J3

R-J3*i*B

R-30*i*A(R-J3*i*C)

Set system-specific node names and Internet (IP) addresses. Any names may be used, but they should be intelligible.

> ⚠ **CAUTION**
> The settings on this screen are made by entering characters. If a space exists in any character string (be aware of a space at the beginning of a character string), correct communication is impossible. In such a case, delete the entire line, and reenter the correct character string.

4.1.2 Setting Full-Duplex Mode (on the Robot Side)

When using a switching hub, set the robot in full-duplex mode. When using an ordinary hub, there is no need to change the setting. Set system variable $ENETMODE.$FULL_DUPLEX to TRUE.

4.1.3 Setting Full-Duplex Mode (on the Hub Side)

When the hub is equipped with a DIP switch or other means to switch between full-duplex and half-duplex modes, set the full-duplex mode, set the autonegotiate mode. (Or set the full-duplex mode if the autonegotiate mode cannot be set.)

4.2 CHECKING NETWORK-RELATED SETTINGS

When you have completed making the settings as described earlier, check the following items to confirm that the settings are correct:

1. Is an Ethernet cable attached between the hub and robot controller?
   -> If not, attach it now.

2. When the power to the hub and robot controller is on, is the orange LED on? This LED is located on the printed circuit board on the far side of the Ethernet cable jack on the main board of the robot controller.
   -> If the LED is not on, reconnect the Ethernet cable and turn on the power to the hub and controller again. If the LED is still off, the main board may be faulty. Contact the FANUC Service Center.

5

SETTING UP CIMPLICITY HMI

Create a project that communicates with robots on CIMPLICITY HMI.
The Process is as follows:
1.   Create a new project.
2.   Register the port, devices, and points.
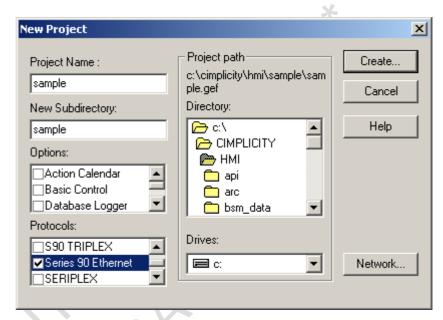
First, start the CIMPLICITY HMI Workbench program.

5.1   CREATING A NEW PROJECT

Before CIMPLICITY HMI can be connected to robot controller over an Ethernet, Series 90 Ethernet must be specified as the protocol when a new project is created.

1.   Open the File menu, and select New, then Project.
The New Project dialog box appears.

Set the following:
Project Name and New Subdirectory : Arbitray
Series 90 Ethernet in Protocols : Check this item.
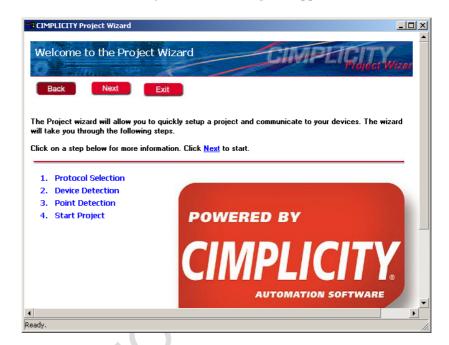
2.   Click Create.
Project Properties appears.

3.   Click OK.

- 13 -

The CIMPLICITY Project Wizard dialog box appears.
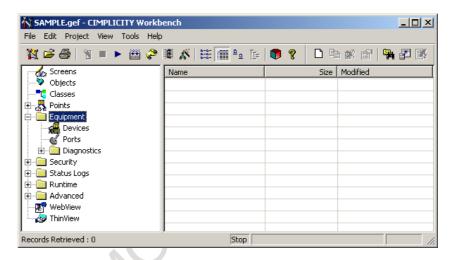
4.   Click Exit.

Here, settings will be made individually without using the wizard. You can now close the dialog box.

5.2   **REGISTERING A PORT**

Next, you will register a port.

1.   Double-click the Equipment folder on the tree.
The Equipment folder opens.

2.   Double-click Ports.
The New Port dialog box appears.
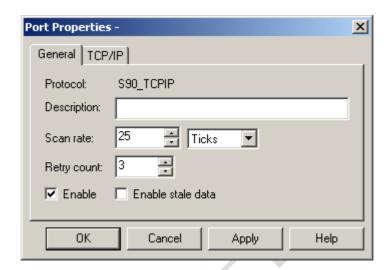
Set the following:
Protocol : S90_TCPIP
Port : Arbitray (TCPIP0 in this case)

3.   Click OK.
The Port Properties dialog box appears.

Set the following:
Scan Rate : 25 Ticks (250ms)
This sets the period of intervals that CIMPLICITY HMI acquires data from R-J3.
The period is set so that CIMPLICITY acquires data every 250ms.

4.   Click OK.
A port for connecting robot controller is then registerd under the name MASTER_TCPIP0.

5.3   **REGISTERING DEVICES**

Next, you will register devices.

1.   Double-click Devices in the Equipment folder.
The New Device dialog box appears.
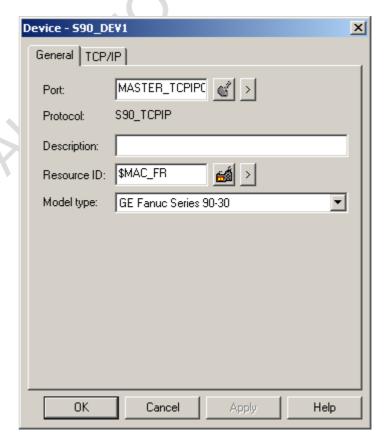
Set the following:
Device : Arbitrary (example : S90_DEV1)
Port : MASTER_TCPIP0 (the port registered in the previous step)

2.   Click OK.
The Device Properties dialog box appears.
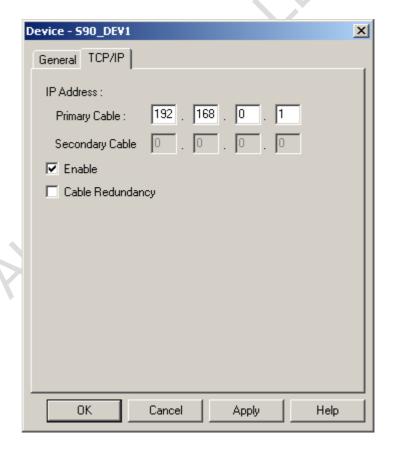
Set the following:
Model Type : GE Fanuc Series 90-30

⚠ **CAUTION**
  CIMPLICITY version 4.01 can connect to robot
  controller control unit with the model type both GE
  Fanuc Series 90-30 and 90-70, but CIMPLICITY
  version 5.5 can connect with the model type
  GE-FANUC Series 90-30 only.
  In the case that you update CIMPLICITY to version
  5.5 or later, your project that uses GE-FANUC
  Series 90-70 may not work. Please change the
  device type from GE-Fanuc Series 90-70 to 90-30.

Select the TCP/IP tab, and set the IP address of R-J3 set on the host communication setup screen.
Check Enable.
Clicking OK registers S90_DEV1 as Device.
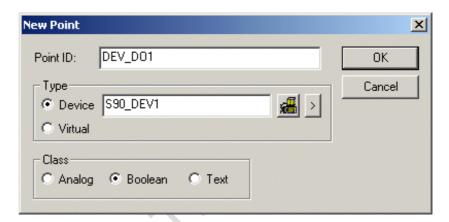
5.4   **REGISTERING POINTS**

Next, you will register points. (Here, you will register DO[1])

1.   Double-click Points of the tree.
The New Points dialog box appears.
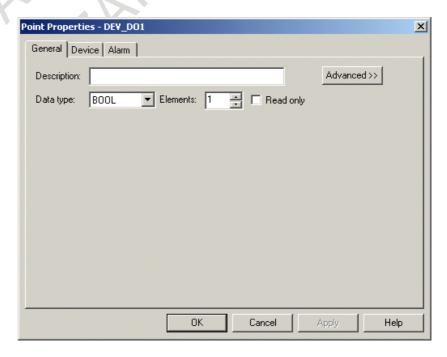
Set the following:
Point ID : Arbitrary (here, DEV_DO1 is set.)
Type : Select Device and enter S90_DEV1 (this is the device that you registered previously.)
Class : Select the class according to the data to be referenced. (here, select Boolean.)
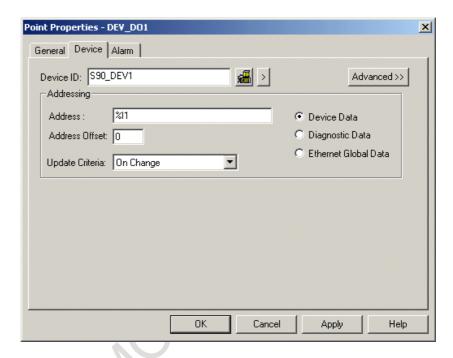
2.   Click OK.
The Point Properties dialog box appears.

3.   Click the Device tab.

In the Address field of Addressing, enter the address of the data you want to reference. (Here, enter %I1 for DO[1].)
For details on how to specify an address, see Chapter 6.

6

ADDRESS ASSIGNMENT TO POINTS

Many types of robot controller data are associated with addresses of the Series 90 PLC manufactured by GE Fanuc (called PLC addresses hereinafter). Therefore, CIMPLICITY can read and write robot controller data by reading and writing PLC address.

6.1   READING AND WRITING I/O SIGNALS (%I, %Q, %M, %AI, %AQ)

CIMPLICITY can read and write DI[1] of robot controller as PLC address %Q1. Robot controller I/O signals correspond to PLC addresses on a one-to-one basis as listed below.

Robot controller I/O signal		PLC address	Example
Digital Input	DI[*x*]	%Q*x*	DI[1] ⇔ %Q1
Digital Output	DO[*x*]	%I*x*	DO[1] ⇔ %I1
Robot Input	RI[*x*]	%Q(5000+x)	RI[1] ⇔ %Q5001
Robot Output	RO[*x*]	%I(5000+x)	RO[1] ⇔ %I5001
UOP Input	UI[*x*]	%Q(6000+x)	UI[1] ⇔ %Q6001
UOP Output	UO[*x*]	%I(6000+x)	UO[1] ⇔ %I6001
SOP Input	SI[*x*]	%Q(7000+x)	SI[0] ⇔ %Q7001
SOP Output	SO[*x*]	%I(7000+x)	SO[0] ⇔ %I7001
Weld Input	WI[*x*]	%Q(8000+x)	WI[0] ⇔ %Q8001
Weld Output	WO[*x*]	%I(8000+x)	WO[0] ⇔ %Q8001
Wire Stick Input	WSI[*x*]	%Q(8400+x)	WSI[0] ⇔ %Q8400
Wire Stick Output	WSO[*x*]	%I(8400+x)	WSO[0] ⇔ %Q8401
Group Input	GI[*x*]	%AQ*x*	GI[1] ⇔ %AQ1
Group Output	GO[*x*]	%AI*x*	GO[1] ⇔ %AI1
Analog Input	AI[*x*]	%AQ(1000+x)	AI[1] ⇔ %AQ1001
Analog Output	AO[*x*]	%AI(1000+x)	AO[1] ⇔ %AI1001

PMC keep relay
    DO[*x*]   (*x* : 10001 – 10144)
    K*a.b*
		%I*x*
		%I((*a**8)+*b*+10001)
			DO[10001] ⇔ %I10001
			K2.5 ⇔ %I10022

PMC internal relay
        DO[*x*]   (*x* : 11001 – 23000)
        R*a.b*
		%M(*x*-11000)
		%M((*a**8)+*b*+1)
			DO[11001] ⇔ %M1
			R2.5 ⇔ %M22

PMCdata table
        GO[*x*]   (*x* : 10001 – 12000)
        D(*a**2), D((*a**2)+1)
		%AI(*x*-6000)
		%AI(*a*+4001)
			GO[10001] ⇔ %AI4001
			D4, D5 ⇔ %AI4003

> ⚠ **CAUTION**
> In the PLC address, %I and %AI are input signals, and %Q and %AQ are output signals. When robot controller data is read and written, however, the robot controller input signals correspond to %Q and %AQ, and the robot controller output signals correspond to %I and %AI. Note that, therefore, the meanings of the robot controller signals are opposite to those of the PLC signals.

⚠ **CAUTION**
When CIMPLICITY writes data to an input signal
such as DI and AI, the value changes momentarily,
then it is soon restored to its actual input value.
The momentary change   in value, however, may
cause an unpredictable errors. Therefore, avoid
write into input signals such as DI and AI.

⚠ **CAUTION**
The reading from and writing to Weld Signal
Input/Output and Wire Stick Input/Output functions
are supported by R-J3iB system software 7D80, 45
or later, and 7D81, 09 or later, and R-30iA(R-J3iC).

6.2 **READING FROM AND WRITING TO REGISTERS (%R)**

The standard correspondence between robot controller data and PLC addresses is as follows:

Robot controller data	PLC address	Example
Register	%Rx	R[1] ⇔ %R1

⚠ **CAUTION**
USE the value in the register as a 16-bit signed integer. Any fractional part of the register value is rounded off. Unless the register value is within the range from –32768 to 32767, the register value cannot be read or written correctly.

The correspondence between registers and PLC addresses is defined with system variable $SNPX_ASG. The standard $SNPX_ASG settings are provided as listed below. As a result, registers correspond to PLC addresses as shown above. When the $SNPX_ASG settings have been modified, the above correspondence is not obtained. Therefore, before reading from or writing into registers through CIMPLICITY, check that $SNPX_ASG is set as follows:

System variable	Value
$SNPX_ASG[1].$ADDRESS	1
$SNPX_ASG[1].$SIZE	10000
$SNPX_ASG[1].$VAR_NAME	R[1]@1.1
$SNPX_ASG[1].$MULTIPLY	1

$SNPX_ASG consists of 80 arrays $SNPX_ASG[1] to [80], each of which has four elements including $ADDRESS, $SIZE, $VAR_NAME, and $MULTIPLY. By modifying these settings, many types of data can be assigned to %R. When CIMPLICITY reads from and writes into position registers and system variables, which will be described later, $SNPX_ASG is also used.

When assigning registers, set the elements of $SNPX_ASG as follows:

$SNPX_ASG element	Explanation
$ADDRESS	Meaning: Start address of %R to be assigned. Range: 1 to 16384
$SIZE	Meaning: Number of %R's to be assigned. Two %R's are required per register. Set this element according to the number of registers you want to read from or write to. （Addint @ to $VAR_NAME changes the number of %R's per register.） Range: 1 to 16384

$VAR_NAME	Meaning: Character string indicating the data to be assigned. When assigning a register, specify, for exammple, R[1]. The number in brackets is a register number. Consecutive registers such as R[2] to R[5] can be assigned at one time. In this case, the number of the registers is four, so set 8 in $SIZE and set R[2] in $VAR_NAME. Here, the index 2 indicates that registers are assigned sequentially from R[2]. The number of %R's per register can be set ti 1 by adding @1.1 to the end of the character string. In this case, 16-bit data is used. Example: R[1]@1.1 R[1] indicates that registers are assigned sequentially from R[1]. @1.1 indicates that data is used as 16-bit data.
$MULTIPLY	Meaning: Multiplier The register value is multiplied by the value set in $MULTIPLY, then the multiplication result is read or written. When $MULTIPLY is set to 0, it has a special meaning. %R can be read and written as 32-bit real type data. When $MULTIPLY is set to non-zero value, 32-bit signed integer data is used, with its fractional part rounded off. Range: 0.0001 to 10000, 0 Example: Suppose that the register value is 123.45: When $MULTIPLY is 1, 123 is read. When $MULTIPLY is 10, 1235 is read. When $MULTIPLY is 0.1, 12 is read. When $MULTIPLY is 0, 123.45(real number) is read.

An example of setting $SNPX_ASG is given below:

	$ADDRESS	$SIZE	$VAR_NAME	$MULTIPLY
$SNPX_ASG[1]	1	2	R[1]@1.1	1
$SNPX_ASG[2]	3	4	R[1]	100
$SNPX_ASG[3]	7	4	R[2]	0.1
$SNPX_ASG[4]	11	2	R[1]	0

Then, the %R-to-register correspondence is follows:

PLC address	Robot controller data that can be read and written
%R1	16-bit signed integer of R[1]
%R2	16-bit signed integer of R[2]
%R3-4	32-bit signed integer obtained by multiplying R[1] by 100
%R5-6	32-bit signed integer obtained by multiplying R[2] by 100
%R7-8	32-bit signed integer obtained by dividing R[2] by 10
%R9-10	32-bit signed integer obtained by dividing R[3] by 10
%R11-12	32-bit real number of R[1]

$SNPX_ASG[1] indicates that two %R's, %R1 and %R2, are assigned to registers sequentially from R[1] in the format of 16-bit signed integers multiplied by one. Since $R is 16-bit data, the number

of %R's per register is one. Therefore, %R1 corresponds to the 16-bit signed integer of R[1], and %R2 corresponds to the 16-bit signed integer of R[2].

$SNPX_ASG[2] indicates that four %R's, %R3 to %R6, are assigned to registers sequentially from R[1] in the format of 32-bit signed integers multiplied by 100. Each register uses two %R's. Therefore, %R3 and %R4 are regarded as 32-bit signed integers obtained by multiplying R[1] by 100, and %R5 and %R6 are regarded as 32-bit signed integers obtained by multiplying R[2] by 100.

$SNPX_ASG[3] indicates that four %R's, %R7 to %R10, are assigned to registers sequentially from R[2] in the format of 32-bit signed integers divided by 10. Therefore, %R7 and %R8 are regarded as 32-bit signed integers obtained by dividing R[2] by 10, and %R9 and %R10 are regarded as 32-bit signed integers obtained by dividing R[3] by 10.

$SNPX_ASG[4] indicates that two %R's, %R11 and %R12, are assigned to registers sequentially from R[1] in real number format. Therefore, %R11 and %R12 are regarded as real numbers of R[1].

6.3   **READING FROM AND WRITING TO POSITION REGISTERS (%R)**

As with registers, position registers are assigned to %R using $SNPX_ASG to allow CIMPLICITY to read from and write to position registers. Unlike registers, position registers are not assigned as standard. So $SNPX_ASG must always be set.

When assigning position registers, set the elements of $SNPX_ASG as follows:

$SNPX_ASG element	Explanation
$ADDRESS	Meaning: Start address of %R to be assigned Range: 1 to 16384
$SIZE	Meaning: Number of %R's to be assigned Fifty %R's are required per position register. (The number of %R's used per position register can be changed by specifying @ in $VAR_NAME) Range: 1 to 16384
$VAR_NAME	Meaning: Character string indicating the data to be assigned When assigning a position register, specify, for example, PR[1]. The bumber in brackets is a position register number. Consecutive position registers such as PR[2] to PR[5] can be assigned at one time. In this case, the number of the position registers is four, so set 200 in $SIZE and set PR[2] in $VAR_NAME. Here, the index 2 indicates that registers are assigned sequentially from PR[2]. In a multi-group system, PR[1] indicates group 1 fo PR[1]. To specify data of group 2, specify the group before the index, such as PR[G2:1]. When @ is specified after the character string, it is possible to assign, for example, J1 through J6 only. This will be explained in detail later.

$MULTIPLY	Meaning: Multiplier
	Only the elements such as X, Y, Z, and J1 that have a real number are affected by $MULTIPLY. For the affected elements, see the table given below.
	The value of each element of the position register is multiplied by the value set in $MULTIPLY, then the multiplication result is read or written.
	When $MULTIPLY is set to 0, it has a special meaning. %R can be read and written as 32-bit real type data.
	When $MULTIPLY is set to a non-zero value, 32-bit signed integer data is read and written with its fractional part rounded off.
	Range: 0.0001 to 10000, 0
	Example: Suppose that a position register element value is 123.45:
	When $MULTIPLY is 1, 123 is read.
	When $MULTIPLY is 10, 1235 is read.
	When $MULTIPLY is 0.1, 12 is read.
	When $MULTIPLY is 0, 123.45 (real number) is read.

Each position registers uses 50 %R's. The contents of the 50 %R's are listed below:

%R address		Explanation	Influence by $MULTIPLY
Cartesian coordinate data			
1-2	X	32-bit signed integer or real number (mm)	Influenced
3-4	Y	32-bit signed integer or real number (mm)	Influenced
5-6	Z	32-bit signed integer or real number (mm)	Influenced
7-8	W	32-bit signed integer or real number (deg)	Influenced
9-10	P	32-bit signed integer or real number (deg)	Influenced
11-12	R	32-bit signed integer or real number (deg)	Influenced
13-14	E1	32-bit signed integer or real number (mm, deg)	Influenced
15-16	E2	32-bit signed integer or real number (mm, deg)	Influenced
17-18	E3	32-bit signed integer or real number (mm, deg)	Influenced
19	FLIP	16-bit signed integer (1:Flip, 0:Non flip)	Not influenced
20	LEFT	16-bit signed integer (1:Left, 0:Right)	Not influenced
21	UP	16-bit signed integer (1:Up, 0:Down)	Not influenced
22	FRONT	16-bit signed integer (1:Front, 0:Back)	Not influenced
23	TURN4	16-bit signed integer (-128~127)	Not influenced
24	TURN5	16-bit signed integer (-128~127)	Not influenced
25	TURN6	16-bit signed integer (-128~127)	Not influenced
26	VALIDC	16-bit signed integer (*1)	Not influenced
Joint data			
27-28	J1	32-bit signed integer or real number (mm, deg)	Influenced
29-30	J2	32-bit signed integer or real number (mm, deg)	Influenced
31-32	J3	32-bit signed integer or real number (mm, deg)	Influenced
33-34	J4	32-bit signed integer or real number (mm, deg)	Influenced
35-36	J5	32-bit signed integer or real number (mm, deg)	Influenced
37-38	J6	32-bit signed integer or real number (mm, deg)	Influenced
39-40	J7	32-bit signed integer or real number (mm, deg)	Influenced

41-42	J8	32-bit signed integer or real number (mm, deg)	Influenced
43-44	J9	32-bit signed integer or real number (mm, deg)	Influenced
45	VALIDJ	16-bit signed integer (*2)	Not influenced
Coordinate system number			
46	UF	16-bit signed integer (0 to 15)　(*3)	Not influenced
47	UT	16-bit signed integer (0 to 15)　(*4)	Not influenced
48-50	Reserve	Not used	Not influenced

*1　VALIDC indicates wheter the position register has valid Cartesian coordinates. VALIDC is set to 0 in one of the following case. In other cases, it is set to 1.
　　– There is data tha is not taught ("******" is indicated on the teach pendant).
　　– The position register is in the joint format, and cannot be converted to the Caretesian coordinate format.
　　Writing a value into VALIDC changes the format of the position register to the Cartesian coorinate format. You may write any value into VALIDC.

*2　VALIDJ indicates wheter the position register has valid values in joint format. VALIDJ is set to 0 in one of the following cases. In other cases, it is set to 1.
　　– There is data tha is not taught ("******" is indicated on the teach pendant).
　　– The position register is in the Cartesian coordinate format, and cannot be converted to the joint format.
　　Writing a value into VALIDJ changes the format of the position register to the joint format. You may write any value into VALIDJ.

*3　UF indicates the number of the user coordinate system used.
　　If UF is o, the world coordinate system is used.
　　If UF is 15, the user coordinate system currently selected is used.
　　　Normally, UF of the position register is set to 15 (F is indicated on the teach pendant).
　　You cannot change this value from the teach pendant. Note that once you have changed this value with CIMPLICITY, you cannot restore the original value without using CIMPLICITY.

*4　UT indicates the number of the tool coordinate system used.
　　If UT is 0, the mechanical interface coordinate system is used.
　　If UT is 15, the tool coorinate system currently selected is used.
　　Normally, UT of the position register is set to 15 (F is indicated on the teach pendant).
　　You cannot change this value from the teach pendant. Note that once you have changed this value with CIMPLICITY, you cannot restore the original value without using CIMPLICITY.

Position registers have two data formats: the Cartesian coordiante format and joint format. You can check position register contents on the position register screen on the teach pendant to find which data format is currently used for a certain position register. If X, Y, Z, W, P, and R are indicated, the Cartesian coordinate format is currently used; if J1 trough J6 are indicated, the joint format is used.

When reading from a position register through CIMPLICITY, you can read the value of a desired element at any time regardless of the current data format of the position register. Read operation does not change the data format of the position register. When the data format og the data read through CIMPLICITY differs from the current data format of the position register, note the following:

- When a value in the position register is not within the stroke range, or when there is an element indicated as "******", the position data format cannot be converted, so the elements not related to the current data format are all set to 0.
  For example, suppose that a position register is in the Cartesian coordinate format and that the value of X is 10000 which is beyond a stroke. In this case, reading J1 to J9 through CIMPLICITY all read 0. However, X and other elements in the Cartesian coordinate format, UF, and UT can be read correctly.

- When the data format of the data read through CIMPLICITY differs from the current data format of the position register, data format conversion is performed at the time of communication. This processing requires several milliseconds to several tens of milliseconds, which adversely affects the communication response time. When many position registers are read, data conversion is performed several times, so this may affect the communication response time significantly.

When an untaught position register is indicated on the position register screen on the teach pendant, each element is indicated as "******". When an element indicated in such a way is read through CIMPLICITY, 0 is read. To determine whether the element is actually set to 0 or is an untaught element, check the values of VALIDC and VALIDJ. If there is any untaught element, VALIDC or VALIDJ is set to 0.
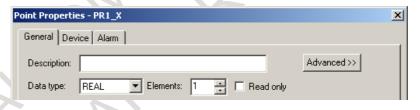
When a value is written into an element of a position register through CIMPLICITY, and that element is in the Cartesian coordinate format, the data format of the position register becomes the Cartesian coordinate format. When a value is written into an element that is in the joint format, the data format of the position register becomes the joint format. When a coordinate system number is written, the data format of the position register is left unchanged.

An example of setting $SNPX_ASG is given below:

	$ADDRESS	$SIZE	$VAR_NAME	$MULTIPLY
$SNPX_ASG[1]	1	50	PR[1]	1

Then, the points corresponded to the elements of position register are as follows:

Point Name	Address	Data Type	Point Name	Address	Data Type
PR1_X	%R1	REAL	PR1_J1	%R27	REAL
PR1_Y	%R3	REAL	PR1_J2	%R29	REAL
PR1_Z	%R5	REAL	PR1_J3	%R31	REAL
PR1_W	%R7	REAL	PR1_J4	%R33	REAL
PR1_P	%R9	REAL	PR1_J5	%R35	REAL
PR1_R	%R11	REAL	PR1_J6	%R37	REAL
PR1_E1	%R13	REAL	PR1_J7	%R39	REAL
PR1_E2	%R15	REAL	PR1_J8	%R41	REAL
PR1_E3	%R17	REAL	PR1_J9	%R43	REAL
PR1_FLIP	%R19	INT			
PR1_LEFT	%R20	INT	PR1_UF	%R46	INT
PR1_UP	%R21	INT	PR1_UT	%R47	INT
PR1_FRONT	%R22	INT			
PR1_TURN4	%R23	INT	PR1_VALIDC	%R26	INT
PR1_TURN5	%R24	INT	PR1_VALIDJ	%R45	INT
PR1_TURN6	%R25	INT			

Specifying @ in $VAR_NAME

Adding @ to $VAR_NAME was described in the section on registers. Here, @ will be explained in detail.

Suppose that you want to read or write just X, Y, and Z of PR[1] through PR[3]. Normally, $SNPX_ASG is set as follows:

	$ADDRESS	$SIZE	$VAR_NAME	$MULTIPLY
$SNPX_ASG[1]	1	150	PR[1]	1

When the above setting is made, the %R-to-position-register correspondence is as follows:

PLC address	Robot controller data that can be read and written
%R1-2	32-bit signed integer of X of PR[1]
%R3-4	32-bit signed integer of Y of PR[1]
%R5-6	32-bit signed integer of Z of PR[1]
%R51-52	32-bit signed integer of X of PR[2]
%R53-54	32-bit signed integer of Y of PR[2]
%R55-56	32-bit signed integer of Z of PR[2]
%R101-102	32-bit signed integer of X of PR[3]
%R103-104	32-bit signed integer of Y of PR[3]
%R105-106	32-bit signed integer of Z of PR[3]

Eighteen %R's are actually read from and written to. With the above setting, however, 150 %R's are occupied. Also, CIMPLICITY attempts to read partial data not actually used such as %R7 to %R50. This is because reading a large amount of data at one time makes communications more efficient than reading small amounts of data severak times. When the position register is in the Cartesian coordinate format, however, data format conversion must be performed for the part from %R27 to %R45 which is joint data. This deteriorates the communication response time.

For efficient communication, set $SNPX_ASG as follows:

	$ADDRESS	$SIZE	$VAR_NAME	$MULTIPLY
$SNPX_ASG[1]	1	18	PR[1]@1.6	1

When the above setting is made, the %R-to-position-register correspondence is as follows:

PLC address	Robot controller data that can be read and written
%R1-2	32-bit signed integer of X of PR[1]
%R3-4	32-bit signed integer of Y of PR[1]
%R5-6	32-bit signed integer of Z of PR[1]
%R7-8	32-bit signed integer of X of PR[2]
%R9-10	32-bit signed integer of Y of PR[2]
%R11-12	32-bit signed integer of Z of PR[2]
%R13-14	32-bit signed integer of X of PR[3]
%R15-16	32-bit signed integer of Y of PR[3]
%R17-18	32-bit signed integer of Z of PR[3]

The modification to the settings include addition of @1.6 after PR[1] of $VAR_NAME and change of the value in $SIZE from 150 to 18. As a result of these modifications, the number of %R's used by one position register, which is normally 50, has decreased to 6. That is, the value "6" at the end of "@1.6" specifies the number of %R's used by one position register. The value "1" in "@1.6" specifies the

location of these six %R's in the data structure of the position register. Here, these six %R's are located at addresses 1 through 6 in the data structure.

As explained above, by specifying @ in $VAR_NAME, a certain part can be taken from the data structure of the position register that uses 50 %R's and can be assigned repeatedly.

@1.6

Number of %R's to be taken(the number of %R's used by one position register)

Start address of the part to be taken in the data structure

Specification of @ can be used not only for the position register but also for all data that uses $SNPX_ASG. In the standard setting, @1.1 used for register assignment functions in a similar manner.

Another example is given. To assign J1 through J6 of PR[3] to PR[10], set the following:

	$ADDRESS	$SIZE	$VAR_NAME	$MULTIPLY
$SNPX_ASG[1]	1	96	PR[3]@27.12	1

6.4   **READING AND WRITING THE CURRENT POSITION (%R)**

The current position of the robot can be read through CIMPLICITY. In the same manner of the position register, set $SNPX_ASG for %R assignment.

For the assignment of the current position, set the elements of $SNPX_ASG as follows:

$SNPX_ASG element	Explanation
$ADDRESS	Meaning: Start address of %R to be assigned Range: 1 to 16384
$SIZE	Meaning: Number of %R's to be assigned For the current position, 50 %R's are used. (The number of %R's used can be changed by specifying @ in $VAR_NAME) Range: 1 to 16384
$VAR_NAME	Meaning: Character string indicating the data to be assigned When assigning the current position, specify, for example, POS[0]. The number in brackets is the user coordinate system number. When 0 is specified as the user coordinate system number, the current position in the world coordinate system can be read. This is equivalent to selecting WORLD on the current position screen on the teach pendant. When 15 is specified as the user coordinate system number, the current position in the currently selected user coordinate system can be read. This is equivalent to selecting USER on the current position screen on the teach pendant. When a number from 1 to 9 is specified as the user coordinate system number, the current position in the specified user coordinate system can be read regardless of the user coordinate system currently selected. The data structure is the same as that of the position register. The current joint position is not affected by the user coordinates. Therefore, the current joint position can be read even when any user coordinates are specifies. In a multi-group system, POS[0] indicates the current position of the robot of group 1. To specify the robot of group 2, specify the group before the user coordinate system number, such as POS[G2:0]. When @ is specified after the character string, only J1 through J6 are assigned. This will be explained in detail later.
$MULTIPLY	Meaning: Multiplier Only the elements such as X, Y, Z, and J1 that have a real number are affected by $MULTIPLY. For the affected elements, see the table given below: The value of each element of the current position is multiplied by the value set in $MULTIPLY, then the multiplication result is read or written. When $MULTIPLY is set to 0, it has a special meaning. %R can be read and written as 32-bit real type data.

		When $MULTIPLY is set to a non-zero value, 32-bit signed integer data is read and written with the fractional part rounded off.
	Range:	0.0001 to 10000, 0
	Example:	Suppose that a position register element value is 123.45:
		When $MULTIPLY is 1, 123 is read.
		When $MULTIPLY is 10, 1235 is read.
		When $MULTIPLY is 0.1, 12 is read.
		When $MULTIPLY is 0, 123.45 (real number) is read.

> **NOTE**
> The current position is read only. Even when a write operation is performed, nothing occurs.

The data format is the same as that of the position register. Fifty %R's are used.

When UT is read, 15 is always read.

When UF is read, the user coordinate system number specified in $VAR_NAME is read.

Unlike the position register, consecutive assignment is not permitted for the current position. Even when settings are made as shown below, for example, the current position with user coordinate system number 1 cannot be assigned to %R51 to %R100. For POS[0] and POS[1], set $SNPX_ASG respectively.

	$ADDRESS	$SIZE	$VAR_NAME	$MULTIPLY
$SNPX_ASG[1]	1	100	POS[0]	1

6.5   **READING ALARM HISTORY (%R)**

Robot controller alarm history can be read through CIMPLICITY. Set $SNPX_ASG to perform %R assignment in the same manner as for registers.

When assigning alarm history, set the elements of $SNPX_ASG as follows:

$SNPX_ASG element	Explanation
$ADDRESS	Meaning: Start address of %R to be assigned Range: 1 to 16384
$SIZE	Meaning: Number of %R's to be assigned For each alarm, 100 %R's are used. (The number of %R's used for each alarm can be changed by specifying @ in $VAR_NAME) Range: 1 to 16384
$VAR_NAME	Meaning: Character string indicating the data to be assigned When assigning alarm history, specify, for example, ALM[1]. The number in brackets corresponds to the line number on the alarm screen. The line number of the latest alarm is set to 1. When ALM[1] is specified, the alarm displayed on the alarm issuance screen on the teach pendant is read. When E is added before the alarm number such as ALM[E1], the alarm displayed on the alarm history screen on the teach pendant is read. When the password option is ordered, specifying P before the alarm number such as ALM[P1] allows reading of the operation history displayed on the password history screen on the teach pendant.
$MULTIPLY	Meaning: Multiplier There is no data in alarm histroy that required the muliplier. $MULTIPLY for alarm history specifies the data format of character string data. Whenever communicating with CIMPLICITY, set this element to 1. Range: 1

> **NOTE**
> The alarm history is read only. Even when a write operation is performed, nothing occurs.

Each alarm use 100 %R's. The contents of the 100 %R's are listed below:

%R address	Explanation
1	Alarm ID 16-bit signed integer For SRVO-001, the alarm ID indicating the servo, 11, is read. The alarm ID number are listed in the alarm code table in the robot controller Operator's Manual.
2	Alarm number 16-bit signed integer For SRVO-001, 1 in 001 is read.

3	Cause code: alarm ID 16-bit signed integer	
	Normally, when an alarm is issued, the alarm is indicated in the first line of the teach pendant. Some alarms are indicated in the upper two lines of the teach pendant. In this case, the indication on the second line is the cause code. At this address, the alarm ID of the cause code can be read. For an alarm with no cause code, 0 is read.	
4	Cause code: alarm number 16-bit signed integer	
	The alarm number of the cause code can be read. For an alarm with no cause code, 0 is read.	
5	Alarm severity 16-bit signed integer	
	A numeric value indicating the alarm severity can be read.	
	NONE 128 SERVO 54 WARN 0 ABORT.L 11 PAUSE.L 2 ABORT.G 45 PAUSE.G 34 SERVO2 58 STOP.L 6 SYSTEM 122 STOP.G 38	
6	Date of occurrence (calendar year) 16-bit signed integer	
7	Date of occurrence (month) 16-bit signed integer	
8	Date of occurrence (day) 16-bit signed integer	
9	Time of occurrence (hours in 24-hour clock) 16-bit signed integer	
10	Time of occurrence (minutes) 16-bit signed integer	
11	Time of occurrence (seconds) 16-bit signed integer	
12-51	Alarm message Up to 80 characters	
	An alarm message can be read. The characters displayed on the first line on the teach pendant including the characters of the alarm code such as "SRVO-001" can be read directly.	
52-91	Cause code message Up to 80 characters	
	The alarm message of the cause code can be read.	
92-100	Alarm severity characters Up to 18 characters	
	A character string indicating alarm severity such as WARN can be read.	

For RESET, 0 is read for both the alarm ID and alarm number. For the alarm message, RESET is read.

When there are only two alarm lines on the alarm occurrence screen, the elements of ALM[3] and later are all set to 0.

In elements containing a character string such as an alarm message, the remaining part after the character string are set to 0. Katakana characters are indicated in shifted JIS character code.

Alarm history has such a large data structure that each alarm uses 100 %R's, and most of the data structure is for character strings such as an alarm message. If you don't need these strings, @ can be specified for efficient communication.

For example, set $SNPX_ASG as follow:

	$ADDRESS	$SIZE	$VAR_NAME	$MULTIPLY
$SNPX_ASG[1]	1	12	ALM[E1]@1.4	1

Then, the %R-to-alarm-history correspondence is as follows:

PLC address	Robot controller data that can be read and written
%R1	Alarm ID of alarm 1
%R2	Alarm number of alarm 1
%R3	Alarm ID of cause code of alarm 1
%R4	Alarm number of cause code of alarm 1
%R5	Alarm ID of alarm 2
%R6	Alarm number of alarm 2
%R7	Alarm ID of cause code of alarm 2
%R8	Alarm number of cause code of alarm 2
%R9	Alarm ID of alarm 2
%R10	Alarm number of alarm 3
%R11	Alarm ID of cause code of alarm 3
%R12	Alarm number of cause code of alarm 3

6.6   READING THE PROGRAM EXECUTION STATUS (%R)

The robot controller program execution status can be read through CIMPLICITY. In the same manner as registers, set $SNPX_ASG for %R assignment.

For assignment of the program execution status, set the elements of $SNPX_ASG as follows:

$SNPX_ASG element	Explanation
$ADDRESS	Meaning: Start address of %R to be assigned Range: 1 to 16384
$SIZE	Meaning: Number of %R's to be assigned For one task, 18 %R's are used. (The number of R's used by one task can be changed by specifying @ in $VAR_NAME) Range: 1 to 16384
$VAR_NAME	Meaning: Character string indicating the data to be assigned When assigning the program execution status, specify, for example, PRG[1]. The number in brackets is the task number. Unless multitasking is performed, PRG[1] is always set. When two tasks are executed simultaneously in a multitasking system, PRG[1] and PRG[2] are used to read the execution statuses of these tasks. Between the two tasks, the task assigned to PRG[1] depends on the program activating timing and communication timing. Once the execution status of a task has been read as PRG[1], it can be read as PRG[1] continuously until program execution terminates.
$MULTIPLY	Meaning: Multiplier The program execution status has no data that requires the multiplier. $MULTIPLY specifies the data format of character string data. Whenever communicating with CIMPLICITY, set this element to 1. Range: 1

> **NOTE**
> The program execution status is read only. Even when a write operation is performed, nothing occurs.

Each task uses 18 %R's. The contents of the 18 %R's are listed below:

%R address	Explanation
1-8	Program name Up to 16 characters Name of a program being executed When a subprogram is called, the name of the subprogram being executed
9	Line number 16-bit signed integer Line number of a program being executed When a subprogram is called, the line number of the subprogram being executed

10	Execution status	16-bit signed integer
	End	0
	Pause	1
	Running	2
11-18	Calling program name	Up to 16 characters
	Name of a program started first	
	When no subprogram is called, the same name as the program name is set.	

When the program has already terminated, all elements are set to 0.

You can read just a part of elements by specifying @.
For example, set $SNPX_ASG as follows:

	$ADDRESS	$SIZE	$VAR_NAME	$MULTIPLY
$SNPX_ASG[1]	1	4	PRG[1]@9.2	1

Then, the %R-to-program-execution-status correspondence is as follows:

PLC address	Robot controller data that can be read and written
%R1	Line number of task 1
%R2	Execution status of task 1
%R3	Line number of task 2
%R4	Execution status of task 2

6.7 **READING FROM AND WRITING INTO SYSTEM VARIABLES (%R)**

Robot controller system variables can be read from and written into through CIMPLICITY. Set $SNPX_ASG to perform %R assignment in the same manner as for registers.

When assigning system variables, set the elements of $SNPX_ASG as follows:

$SNPX_ASG element	Explanation
$ADDRESS	Meaning: Start address of %R to be assigned Range: 1 to 16384
$SIZE	Meaning: Number of %R's to be assigned The number of %R's required for a system variable depends on the data type of the system variable. For details, see the table given below. (The number of %R's used for one system variable can be changed by specifying @ in $VAR_NAME.) Range: 1 to 16384
$VAR_NAME	Meaning: Character string indicating the data to be assigned When assigning a system variable, set a system variable name such as $WAITTMOUT. When a system variable with an array such as UALARM_SEV[1] is specified, array elements are assigned successively in the same manner as for registers. When assigning a KAREL variable, set character string indicating a KAREL program name and a KAREL variable name such as $[KAREL-program-name]KAREL-variable-name.
$MULTIPLY	Meaning: Multiplier The meaning of this element varies depending on the data type of the specified system variable. For details, see the table given below.

> ⚠ **CAUTION**
> The reading from and writing to KAREL variables functions are supported by R-J3iB system software 7D80, 45 or later, and 7D81, 09 or later, and R-30iA(R-J3iC).

When a system varialbe is read from or wriiten to, the data type used with CIMPLICITY reads from or writes into the system variable and the meaning of $MULTIPLY vary depending on the data type of the system variable. In addition, CIMPLICITY cannot sometimes read from or write into a system variable depending on the data type of the system variable.

For the data types of sysmte variables that can be read from and written into through CIMPLICITY, the number of %R's used for each system variable, the meaning of $MULTIPLY, and the data type used for read write by CIMPLICITY, see the table given below.

- 41 -

Data type of system variable	Number of %R's required per system variable	Meaning of $MULTIPLY and data type used for read/write
INTEGER 32-bit signed integer	2	The value in the system variable, multiplied by the value in $MULTIPLY, is read/written. The fractional part is rounded off. When $MULTIPLY is 0, it has the same meaning as when it is 1. Read/write data as 32-bit signed integer.
SHORT 16-bit signed integer	2	
BYTE 8-bit signed integer	2	
REAL 32-bit real number	2	The value in the system variable, multiplied by the value in $MULTIPLY, is read/written. When $MULTIPLY is 0, it has a special meaning, and data can be read/written a 32-bit real type data. When $MULTIPLY is a non-zero value, data is treated as 32-bit signed integer data with its fractional part rounded off.
BOOLEAN TRUE／FALSE	2	Read/write data as a 32-bit signed integer. If TRUE is set, 1 is read. If FALSE is set, 0 is read. Writing 0 sets FALSE. Writing a non-zero value sets TRUE. $MULTTIPLY is not used.
POSITION Position data	50	This data type has the same data structure as the position register. Each element can be read/written. $MULTIPLY has also the same meaning as for the position register.
STRING Character string	40	The data type of a character string is specified. Whenever communicating with CIMPLICITY, set 1.

For INTEGER, SHORT, and BYTE, data can be read and written through CIMPLICITY similarly, so there is no need to distinguish these data types. When an integer value is indicated for a system variable on the system variable screen, the system variable is of the INTEGER, SHORT, or BYTE type.

When decimal positions are indicated for a system variable on the system variable screen, the system variable is of the REAL type.

When TRUE or FALSE is indicated for a system variable on the system variable screen, the system variable is of the BOOLEAN type.

When a character string is indicated for a system variable on the system variable screen, the system variable is of the STRING type.

When POSITION is indicated for a system variable on the system variable screen, the system variable is of the POSITION type.

> ⚠ **CAUTION**
> Even when a system variable is indicated as a
> read only variable on the system variable screen,
> CIMPLICITY can write into this system variable. If
> an inappropriate value is written into a system
> variable, the system may be affected significantly.
> Therefore, before writing a value into a system
> variable, carefully consider the meaning of the
> system variable.

You can specify @ to read only specific elements.
For example, to assign X, Y, and Z of user coordinate systems 1 and 2
of group 1, set $SNPX_ASG as follows:

	$ADDRESS	$SIZE	$VAR_NAME	$MULTIPLY
$SNPX_ASG[1]	1	12	$MNUFRAME[1,1]@1.6	1

Then, the %R-to-system-variable correspondence is as follows:

PLC address	Robot controller data that can be read and written
%R1-2	X of user coordinate system 1 of group 1 (X of $MNUFRAME[1,1])
%R3-4	Y of user coordinate system 1 of group 1 (Y of $MNUFRAME[1,1])
%R5-6	Z of user coordinate system 1 of group 1 (Z of $MNUFRAME[1,1])
%R7-8	X of user coordinate system 2 of group 1 (X of $MNUFRAME[1,2])
%R9-10	Y of user coordinate system 2 of group 1 (Y of $MNUFRAME[1,2])
%R11-12	Z of user coordinate system 2 of group 1 (Z of $MNUFRAME[1,2])

6.8 **READING AND WRITING THE COMMENT OF REGISTERS, POSITION REGISTERS, AND I/O (%R)**

The comment of robot controller registers, position registers, and I/O can be read from and written to through CIMPLICITY. In the same manner as registers, set $SNPX_ASG for %R assignment.

For assignment of the comment, set the elements of $SNPX_ASG as follows:

$SNPX_ASG element	Explanation
$ADDRESS	Meaning: Start address of %R to be assigned Range: 1 to 16384
$SIZE	Meaning: Number of %R's to be assigned For one comment, 40 %R's are used. (The number of %R's used by one comment can be changed by specifying @ in $VAR_NAME) Range: 1 to 16384
$VAR_NAME	Meaning: Character string indicating the data to be assigned When assigning the comment of register, specify, for example, R[C1]. The number in brackets is the register number. When assigning the comment of others, specify as listed below: Register R[C1] Position Register PR[C1] DI DI[C1] DO DO[C1] RI RI[C1] RO RO[C1] UI UI[C1] UO UO[C1] SI SI[C1] SO SO[C1] WI WI[C1] WO WO[C1] WSI WSI[C1] WSO WSO[C1] GI GI[C1] GO GO[C1] AI AI[C1] AO AO[C1] Comments of consecutive registers such as R[2] to R[5] can be assigned at one time. In this case, the number of the register is four and the number of %R's required per register is fourty, so set 160 in $SIZE and set R[C2] in $VAR_NAME. Here, the index 2 indicates that registers are assigned sequentially from R[2]. The number of %R's per comment can be set to 5 by adding @1.5 to the end of the character string. In this case, the top 10 characters in the comment are read/written. (One %R includes 2 characters.) Example: R[C1]@1.5 R[C1] indicates that registers that comment is read/written are assigned sequentially from R[1].

	@1.1 indicates that the top 10 characters in the comment are read/written.
$MULTIPLY	Meaning: $MULTIPLY specifies the data format of character string data.
	Whenever communicating with CIMPLICITY, set this element to 1.

⚠ **CAUTION**
This function is supported by R-J3iB system software 7D80, 45 or later, and 7D81, 09 or later, and R-30iA(R-J3iC).

You can specify @ to read only specific elements.
For example, set $SNPX_ASG as follows:

	$ADDRESS	$SIZE	$VAR_NAME	$MULTIPLY
$SNPX_ASG[1]	1	50	R[C1]@1.10	1

Then, the %R-to-comment-of-register correspondence is as follows:

PLC address	Robot controller data that can be read and written
%R1-10	Top 20 characters of comment of R[1]
%R11-20	Top 20 characters of comment of R[2]
%R21-30	Top 20 characters of comment of R[3]
%R31-40	Top 20 characters of comment of R[4]
%R41-50	Top 20 characters of comment of R[5]

6.9   **READING AND WRITING THE VALUE AND SIM STATUS OF I/O (%R)**

I/O values can be read and written assigning to %I, %Q, %AI, %AQ. And I/O values can be also assigned to %R like registers. And I/O SIM status can be read and written through CIMPLICITY.
Set $SNPX_ASG to perform %R assignment in the same manner as for registers.

When assigning the value or SIM status of I/O, set the elements of $SNPX_ASG as follows:

$SNPX_ASG element	Explanation
$ADDRESS	Meaning: Start address of %R to be assigned Range: 1 to 16384
$SIZE	Meaning: Number of %R's to be assigned The number of %R's required for I/O depends on the value of $MULTIPLY. When $MULTIPLY is 1, one %R is required per one I/O. When $MULTIPLY is 0, one %R is required per sixteen I/O. (In the case GI/O and AI/O, one %R is always required per one I/O undepending on the value of $MULTIPLY.) Range: 1 to 16384
$VAR_NAME	Meaning: Character string indicating the data to be assigned When assigning I/O, specify, for example, DI[1]. The number in brackets is an I/O number. Value SIM status DI DI[1] DI[S1] DO DO[1] DO[S1] RI RI[1] RI[S1] RO RO[1] RO[S1] UI UI[1] UO UO[1] SI SI[1] SO SO[1] WI WI[1] WI[S1] WO WO[1] WO[S1] WSI WSI[1] WSI[S1] WSO WSO[1] WSO[S1] GI GI[1] GI[S1] GO GO[1] GO[S1] AI AI[1] AI[S1] AO AO[1] AO[S1] Example: Assigning the values of 32 I/O's, DI[11] to DI[42] When $MULTIPLY is 1, set 32 to $SIZE, and set DI[11] to $VAR_NAME. Here, the index 11 indicates that I/O's are assigned sequentially from DI[11]. When $MULTIPLY is 0, set 2 to $SIZE because one %R is assigned to 16 I/O's, and set DI[11] to $VAR_NAME.
$MULTIPLY	Meaning: $MULTIPLY specifies if I/O is assigned as a bit value or not. One I/O is assigned to one %R 1 Sixteen I/O's are assigned to one %R 0

⚠ **CAUTION**
This function is supported by R-J3iB system
software 7D80, 45 or later, and 7D81, 09 or later,
and R-30iA(R-J3iC).

When $MULTIPLY is 1, the value or SIM status of one I/O is
assigned to one %R.
When the value is ON, the value of the corresponded %R is 1. When
OFF, the value is 0.

When $MULTIPLY is 0, the values or SIM statuses of sixteen I/O's
are assigned one %R as bit values.
When the value is ON, the value of the corresponded bit of %R is 1.
When OFF, the value is 0.
The I/O with less index number is assigned to the less significant bit
of %R.

Example: DI[1] to DI[16] are assigned to %R
When DI[1] is ON and others are OFF, the value of %R1 is 1.
When DI[16] is ON and others are OFF, the value of %R1 is 32768.
(In the case 16-bit signed integer)

6.10   SETTING $SNPX_ASG FROM CIMPLICITY (%G)

When you want to read or write data other than I/O signal data, you must set $SNPX_ASG. Usually, this is set on the system variable screen. $SNPX_ASG can also be set through communication from CIMPLICITY.

When a character string is written to PLC address %G, robot controller performs processing regarding the written character string as a command. Robot controller performs command processing similarly at any address number if it is within the %G area. The following commands can be executed:

CLRASG   **Erasing $SNPX_ASG**

Format: CLRASG
Function: Initializes all $SNPX_ASG settings.
Execute this command once before executing a SETASG command sequence.

SETASG   **Setting $SNPX_ASG**

Format: SETASG   ($ADDRESS)   ($SIZE)   ($VAR_NAME) [($MULTIPLY)]
Function: Sets a specifiled value in $SNPX_ASG.
Robot controller automatically selects an unused $SNPX_ASG area for setting. So there is no need to specify the number of $SNPX_ASG.
$MULTIPLY may be omitted. By default, $MULTIPLY is set to 1.
Before executing the SETASG command, execute CLRASG once.

SETVAR   **Setting a system variable**

Format: SETVAR   (system-variable-name)   (value)
Function: Set a value in a specified system variable.
The data types of system variables that can be set are INTEGER, SHORT, BYTE, REAL, BOOLEAN, and STRING.
For the BOOLEAN type, specify 1 for TRUE, or 0 for FALSE.
When specifying a character string that includes blanks, enclose the character string with double quotation marks ("").

CLRALM   **Erasing alarm histories**

Format: CLRALM
Function: Initializes all alarm histories.

Example: When setting $SNPX_ASG as follows:

	$ADDRESS	$SIZE	$VAR_NAME	$MULTIPLY
$SNPX_ASG[1]	1	2	R[1]@1.1	1
$SNPX_ASG[2]	3	4	R[1]	100
$SNPX_ASG[3]	7	4	R[2]	0.1
$SNPX_ASG[4]	11	2	R[1]	0

Write the following character strings in %G sequentially:

CLRASG
SETASG 1 2 R[1]@1.1 1
SETASG 3 4 R[1] 100
SETASG 7 4 R[2] 0.1
SETASG 11 2 R[1] 0

Some commands can be written at once marking off them by CR(0x0A) or LF(0x0D).
In this case, adding CR or LF at the end of the last command makes the execution a little faster.

⚠ **CAUTION**
CLRALM command and the function writing some commands at once are supported by R-J3iB system software 7D80, 45 or later, and 7D81, 09 or later, and R-30iA(R-J3iC).

6.11   **NOTES AND TIPS ON USAGE**

6.11.1   **$SNPX_ASG Is Set, but Data That Should Have Been Assigned Cannot Be Read/Written.**

When a %R area not assigned with $SNPX_ASG is read from, 0 is always read. When a non-zero value is set at the %R address in question, the %R address can be considered to have been assigned to other data.

When $SNPX_ASG is set as follows, for example, %R101 through %R150 are assigned both by $SNPX_ASG[1] and $SNPX_ASG[2].

	$ADDRESS	$SIZE	$VAR_NAME	$MULTIPLY
$SNPX_ASG[1]	1	1000	R[1]@1.1	1
$SNPX_ASG[2]	101	50	PR[1]	100

In this case, $SNPX_ASG with the smaller number takes preceedence.

In the above example, $R101 to %R150 are assigned to R[101] to R[150]. So, PR[1] cannot be read from nor written to.

Next, suppose that 0 is set at the %R address in question. Also in this case, it is considered that other data is assigned to the %R area as explained above, and that the data is 0 by accident. Check for a duplicate assignment.

If a duplicate assignment is not made, the assignment setting itself may have a problem. Check the $SNPX_ASG setting. The following are error-prone:

- The correct formats for $VAR_NAME are listed below. If the setting does not follow any of these formats, assignment is not performed.

"R[number]"	
"PR[number]" "PR[G number:number]"	When specifying a group, specify a colon (:) after a group number.
"POS[number]" "POS[G number:number]"	When specifying a group, specify a colon (:) after a group number.
"ALM[number]" "ALM[E number]" "ALM[P number]"	Specify a number immediately after E and P. A colon (:) is not added.
"PRG[number]"	
System-variable-name	A system variable name begins with a dollar sign ($).
$[KAREL-program-name]KAREL-variable-name	
"DI[number]" "DI[S number]" "DI[C number]" "DO[number]" "DO[S number]" "DO[C number]" "RI[number]" "RI[S number]" "RI[C number]" "RO[number]" "RO[S number]" "RO[C number]"	

"UI[number]" "UI[S number]" "UI[C number]" "UO[number]" "UO[S number]" "UO[C number]" "SI[number]" "SI[S number]" "SI[C number]" "SO[number]" "SO[S number]" "SO[C number]" "WI[number]" "WI[S number]" "WI[C number]" "WO[number]" "WO[S number]" "WO[C number]" "WSI[number]" "WSI[S number]" "WSI[C number]" "WSO[number]" "WSO[S number]" "WSO[C number]" "GI[number]" "GI[S number]" "GI[C number]" "GO[number]" "GO[S number]" "GO[C number]" "AI[number]" "AI[S number]" "AI[C number]" "AO[number]" "AO[S number]" "AO[C number]"	

- When a blank is included in $VAR_NAME, assignment is not performed. Also, when using @, be sure to specify @ immediately after a variable name without inserting a blank.
- When successive assignment is performed, the number of %R's used for one element varies depending on the type of assigned data. Confirm the explanation of each data.
- When specifying @, follow the format "@number.number". A period (.) is used between the numbers. Be sure to specify @ immediately after the variable name.

6.11.2   Improving Communication Efficiency

- Read/Write of I/O signals is faster than read/write of %R's assigned with $SNPX_ASG. The speed of read/write of %R may be slower during program execution. On the other hand, read/write of I/O signals is not affected by program execution. When emphasis is placed on the communication speed, it is recommended that the use of %R be avoided if possible.
- Data conversion of a position register between the Cartesian coordinate and joint formats requires a significant amount of time. In position register assignment, @ can be used to assign only those elements that are actually required. This can improve communication efficiency. In addition, when data is read through CIMPLICITY, using the same data format as that of the actual position register eliminates the necessity of data conversion, thus improving communication efficiency.
- CIMPLICITY sometimes automatically reads from an address that actually is not used. In such a case, to prevent the slowing of communication speed, do not assign unnecessary data whenever possible. In addition, assign %R's so that they are as close as possible to compact the %R area for read/write. This can improve communication efficiency.

6.11.3   Version Of Communication Function With CIMPLICITY

The value of $SNPX_PARAM.$VERSION specifies the version of communication function with CIMPLICITY. Whe $SNPX_PARAM.$VERSION does not exist, the version is 1.

The functions supported in version 2 or later are listed below:

- Reading and writing the comment of registers, position registers, and I/O's
- Reading and writing the WDI/O and WSI/O
- Reading and writing the i/O SIM status
- Writing the plural commands at once
- CLRALM command
- Assigning I/O value to %R
- Reading and writing the KAREL variables
- Multi Connection and multiplex $SNPX_ASG

6.11.4   **Multi Connection And Multiplex $SNPX_ASG**

When the version is 2 or later, one robot controller can accept some TCP/IP connections with CIMPLICITY at once. (Only one connection can be accepted communicating by RS-232-C.)

Normally, the multi connection is disenabled. To enable the multi connection, set the number of CIMPLICITY to $SNPX_PARAM.$NUM_CIMP. (Setting too large value may make the robot controller stop because of the lack of memory area.)

When the version is 2 or later, the multiplex $SNPX_ASG is supported to accept that some CIMPLICITY set the different values to $SNPX_ASG.

When the version is 1, CLRASG command erases the setting of $SNPX_ASG. When the version is 2 or later, CLRASG command creates the local $SNPX_ASG for one CIMPLICITY that writes this command. After that, this local $SNPX_ASG is used for the communication with this CIMPLICITY instead of the system variable $SNPX_ASG. This local $SNPX_ASG is deleted when this connection is disconnected.

After CLRASG command is executed, the results of SETASG command cannot be confirmed on the system varialbe screen. When CLRASG command is not executed, the system varialbe $SNPX_ASG is used for SETASG command or reading/writing %R's.

To disenable the multiplex $SNPX_ASG, set 0 to $SNPX_PARAM.$NUM_CIMP. (Default value is 0.)

When the number of connection is over this value, the connection that has not communicated for the longest time is disconnected.

7

APPENDIX

7.1   SAMPLE PROJECT OF "HMI FOR ROBOT"

7.1.1   Sample Project Overview

CIMPLICITY HMI for Robot Package includes the sample project that communicates with 3 robots.

In the sample project, many points for robot controller data are registered, and there are some screens to monitor the robot information, such as, for example, program execution status, robot motion, the values of position registers, and alarm history.

You can easily create your own project modifying this sample project.

This sample project is consists of 6 screens.
- START.CIM
  Start page. Open this first, make Full Screen, and click the button to go TOP.CIM.
- TOP.CIM
  You can see the motion of 3 robots as 3D animation with ROBOGUIDE.
- UOP1.CIM, UOP2.CIM, UOP3.CIM
  You can see the statuses such as robot signals, robot program execution status, and current alarm.
- POS1.CIM, POS2.CIM, POS3.CIM
  You can see the current position of the robot as 3D animation.
- ALARM1.CIM, ALARM2.CIM, ALRAM3.CIM
  You can see the alarm history.
  And you can also get detail infomations about alarm in history as HTML document.
- REG1.CIM, REG2.CIM, REG3.CIM
  You can see and change the values and comment of position registers.

7.1.2   Registered Devices

Three devices named SNPE1, SNPE2, and SNPE3 for each robot.
Here, the Model type of the device must be "GE FANUC Series 90-30".

7.1.3   Registered Points

In this sample project, $SNPX_ASG in each robot controller must be as follow:
(These parameters are automatically set in the script. This will be explained in detail later.)

	$ADDRESS	$SIZE	$VAR_NAME	$MULTIPLY
$SNPX_ASG[1]	1	400	R[1]	1
$SNPX_ASG[2]	401	5000	PR[1]	0
$SNPX_ASG[3]	5401	1000	ALM[1]	1
$SNPX_ASG[4]	6401	1000	ALM[E1]	1
$SNPX_ASG[5]	7401	50	POS[0]	0
$SNPX_ASG[6]	7451	18	PRG[1]	1

$SNPX_ASG[7]	7501	100	R[C1]@1.10	1
$SNPX_ASG[8]	7601	100	PR[C1]@1.10	1
$SNPX_ASG[9]	7701	2	$mcr.$genoverride	1

And many poinst for robot controller data are registered for SNPE1 as follow:
("_2" and "_3" are added to the end of name of points for SNPE2 and SNPE3, such as DI1_2.)

- I/O Signal
- DI1 to DI20 (Degital Input DI[1] to DI[20]) : As BOOL
- DO1 to DO20 (Digital Output DO[1] to DO[20]) : As BOOL
- UI1 to UI18 (UOP Input UI[1] to UI[18]) : As BOOL
- UO1 to UO20 (UOP Output UO[1] to UO[20]) : As BOOL
- SI0 to SI15 (SOP Input SI[0] to SI[15]) : As BOOL
- SO0 to SO15 (SOP Output SO[0] to SO[15]) : As BOOL
- GI1 to GI5 (Group Input GI[1] to GI[5]) : As UINT
- GO1 to GO5 (Group Output GO[1] to GO[5]) : As UINT

- Register
- R1 to R10 (R[1] to R[10]) : As DINT
- RCOMMENT1 to RCOMMENT10 (Comments of R[1] to R[10]) : As STRING_20

- Position Register
- PR1_X, Y, Z, W, P, R to PR10_X, Y, Z, W, P, R (X, Y, Z, W, P, R of PR[1] to PR[10]) : As REAL
- PR1_FLIP to PR10_FLIP (FLIP of PR[1] to PR[10]) : As INT
- PR1_LEFT to PR10_LEFT (LEFT of PR[1] to PR[10]) : As INT
- PR1_UP to PR10_UP (UP of PR[1] to PR[10]) : As INT
- PR1_FRONT to PR10_FRONT (FRONT of PR[1] to PR[10]) : As INT
- PR1_T1, T2, T3 to PR10_T1, T2, T3 (TRUN4, 5, 6 of PR[1] to PR[10]) : As INT
- PR1_J1, J2, J3, J4, J5, J6 to PR10_J1, J2, J3, J4, J5, J6 (J1 to J6 of PR[1] to PR[10]) : As REAL
- PRCOMMENT1 to PRCOMMENT10 (Comments of PR[1] to PR[10]) : As STRING_20

- Current Position
- CUR_POS_X, Y, Z, W, P, R (X, Y, Z, W, P, R of Current Position) : As REAL
- CUR_POS_FLIP (FLIP of Current Position) : As INT
- CUR_POS_LEFT (LEFT of Current Position) : As INT
- CUR_POS_UP (UP of Current Position) : As INT
- CUR_POS_FRONT (FRONT of Current Position) : As INT
- CUR_POS_T1, T2, T3 (TRUN4, 5, 6 of Current Position) : As INT
- CUR_POS_J1, J2, J3, J4, J5, J6 (J1 to J6 of Current Position) : As REAL
- CUR_POS_VALIDC (VALIDC of Current Position) : As INT

- CUR_POS_VALIDJ (VALIDJ of Current Position) : As INT

- Current Alarm
- ALM1_ID (Alarm ID of Current Alarm) : As INT
- ALM1_NO (Alarm Number of Current Alarm) : As INT
- ALM1_MSG (Alarm Message of Current Alarm) : As STRING_80

- Alarm History
- ALM_ID_REAL1 to ALM_ID_REAL10 (Alarm ID of Alarms) : As INT
- ALME_NO_REAL1 to ALME_NO_REAL10 (Alarm Number of Alarms) : As INT
- ALME_MSG_REAL1 to ALME_MSG_REAL10 (Alarm Message of Alarms) : As STRING_80
- ALM_SEV_REAL1 to ALM_SEV_REAL10 (Alarm Severity of Alarms) : As INT
- ALME_DATE_REAL1[0] to ALME_DATE_REAL10[0] (Year of Occurence of Alarms) : As INT
- ALME_DATE_REAL1[1] to ALME_DATE_REAL10[1] (Month of Occurence of Alarms) : As INT
- ALME_DATE_REAL1[2] to ALME_DATE_REAL10[2] (Day of Occurence of Alarms) : As INT
- ALME_DATE_REAL1[3] to ALME_DATE_REAL10[3] (Hour of Occurence of Alarms) : As INT
- ALME_DATE_REAL1[4] to ALME_DATE_REAL10[4] (Minute of Occurence of Alarms) : As INT
- ALME_DATE_REAL1[5] to ALME_DATE_REAL10[5] (Second of Occurence of Alarms) : As INT

- Program Execution Status
- PRG_NAME (Program Name) : As STRING_20
- PRG_LINE (Line Number) : As UINT
- PRG_STATUS (Execution Status) : As UINT
- PRG_ROOT (Calling Program Name) : As STRING_20
- OVERRIDE (Override) : As DINT

- Command String
- COMMAND_1 (Command String for Setting $SNPX_ASG) : As STRING_80

> ⚠ **CAUTION**
> %R8001 to %R8100 and %R8101 to %R8200 are assigned to the comments of R[1] to R[10] and PR[1] to PR[10], and the points, RCOMMENT1 to RCOMMENT10 and PRCOMMENT1 to PRCOMMENT10, refer to these %R's.
> These settings are supported by R-J3iB system software 7D80, 45 or later, and 7D81, 09 or later, and R-30iA(R-J3iC).

7.1.4 $SNPX_ASG Settings (em_init.bcl)

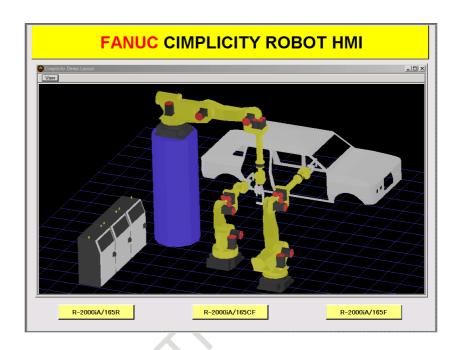In the VB script em_init.bcl, the values of $SNPX_ASG are set for assignment to %R's as follows:

```
intRobotSum = 3
For i = 1 To intRobotSum
    PointSet "COMMAND_" & CStr(i),"CLRASG"
    PointSet "COMMAND_" & CStr(i),"SETASG      1    400
R[1]   1.0"
    PointSet "COMMAND_" & CStr(i),"SETASG     401    5000
PR[1]   0"
    PointSet "COMMAND_" & CStr(i),"SETASG    5401    1000
ALM[1]   1.0"
    PointSet "COMMAND_" & CStr(i),"SETASG    6401    1000
ALM[E1]   1.0"
    PointSet "COMMAND_" & CStr(i),"SETASG    7401     50
POS[0]   0"
    PointSet "COMMAND_" & CStr(i),"SETASG    7451     18
PRG[1]   1.0"
    PointSet "COMMAND_" & CStr(i),"SETASG    7501    100
R[C1]@1.10   1.0"
    PointSet "COMMAND_" & CStr(i),"SETASG    7601    100
PR[C1]@1.10   1.0"
    PointSet "COMMAND_" & CStr(i),"SETASG    7701      2
$mcr.$genoverride   1.0"
Next i
```

Here, three points named COMMAND_1, COMMAND_2, and COMMAND_3 are assigned to %G1 of each robot. Setting command strings to these points (executing PointSet in VB Scrip), these commands are written to PLC address %G, and are executed in R-J3.

7.1.5 Running ROBOGUIDE To Monitor The Robot Motion (TOP.CIM)

On the screen TOP.CIM, 3D animation of three robots is displayed by ROBOGUIDE.
ROBOGUIDE is called, and is given the robot motion infomations in the VB Script defined in TOP.CIM.

In OnScreenOpen( ), ROBOGUIDE is called, is displayed, and open the robot system layout.

If robgraph   Is Nothing Then
     Set robgraph = CreateObject("RoboGuidecim.Core")
End If

First, ROBOGUIDE Object "robgraph" is created.

robgraph.show "cimplicity_demo"

ROBOGUIDE   opens   the   environment
C:/ROBOGUIDE/cimplicity_demo.

   robgraph.caption="Cimplicity Demo Layout"

The title of ROBOGUIDE windows is set Cimplicity Demo Layout.

   robgraph.move pleft, ptop, pwidth, pheigh

The size of ROBOGUIDE window is changet to (pwidth, pheight).
And the ROBOGUIDE window is moved to (pleft, ptop).
(The values of pwidth, pheight, pleft, and ptop are already given in other lines.)

Set objLayout = robgraph.layouts.Item("layout1")
objLayout.Selected = True

ROBOGUIDE opens the robot system layout Layout1.

   Set objRobot1 = objLayout.PlacedObjects.Item("R-2000i_165F")
   Set objRobot2 = objLayout.PlacedObjects.Item("R-2000i_165CF")
   Set objRobot3 = objLayout.PlacedObjects.Item("R-2000i_165R")

The robot objects objRobot1, objRobot2, and objRobot3 is assigned to the three robots, R-2000I_165F, R-2000I_165CF, and R-2000_165R, placed in Layout1.

> Set objView = robgraph.screens.currentview
> objView.X = -26
> objView.Y = -17
> objView.Z = 77
> objView.RX = 1220
> objView.RY = 86
> objView.RZ = -3250
> objView.ScaleV = 18
> objView.VGDistance = 250

The view point of ROBOGUIDE is changed to the specified parameters.

To get the parameters for the property X, Y, Z, RX, RY, RZ, ScaleV, and VGDistance, click the View button on ROBOGUIDE window. The text file C:/ROBOGUIDE/view.txt is created. In this text file shows the view point parameters.

> **robgraph.topmost = True**

ROBOGUIDE window is always displayed over CIMPLICITY screen.

In OnTimer1( ), ROBOGUIDE is given the current robot positions.
Here, OnTImer1( ) is automatically called every 250ms, so the robots displayed in the ROBOGUIDE window move like the real robot motions.

> **For i = 0 To 5**
>   **joint1(i) = PointGet ("CUR_POS_J" & Cstr(i+1))**
>   **joint2(i) = PointGet ("CUR_POS_J" & Cstr(i+1) & "_2")**
>   **joint3(i) = PointGet ("CUR_POS_J" & Cstr(i+1) & "_3")**
> **Next i**

Joint1( ), joint2( ), and joint3( ) is set the current position of three robots.

Here, the points CUR_POS_J1 to CUR_POS_J6 are assigned to the current position of R-2000i/165F in the joint format. CUR_POS_J1_2 to CUR_POS_J6_2 and CUR_POS_J1_3 to CUR_POS_J6_3 are also assigned to the current positions of R-2000i/165CF and R-2000i/165R in the joint format.

> If Not(objRobot1 Is Nothing) Then
>   objRobot1.SetJoint joint1()
> Else
>   Set objRobot1=
> objLayout.PlacedObjects.Item("R-2000i_165F")
> End If

ROBOGUIDE is given the values in joint1( ).

- 59 -
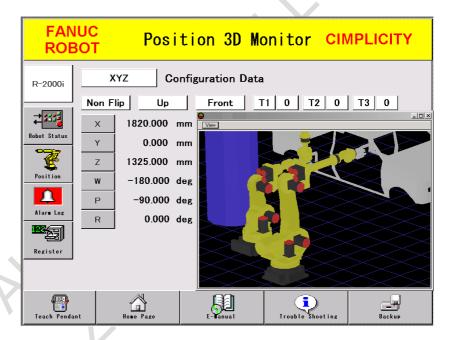
As a result, the robot R-2000I_165F in ROBOGUIDE is moved to the given position.
Other two robots are also moved in the same manner with R-2000I_165F.

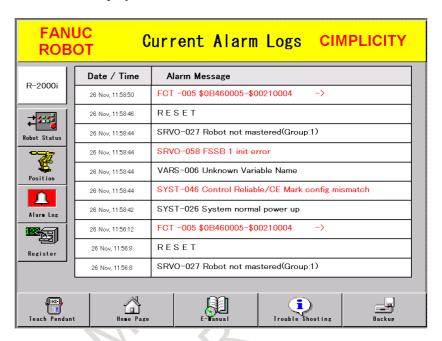When TOP.CIM is closed, OnScreenClose1( ) is called.
  robgraph.hide
ROBOGUIDE window is hidden in OnScreenClose1( ).

On the screen POS1.CIM, POS2.CIM and POS3.CIM, ROBOGUIDE is used in the same manner of TOP.CIM.

7.1.6 Getting The Alarm Information (ALARM1.CIM)

On the screen ALARM1.CIM, the history of alarms occurred in robot controller is displayed.

Here, CIMPLICITY can get the information of date and time of occurrence as the values of year, month, day, hour, minute, and second. And these infomations are displayed as character string in the format such as "Nov 26, 11:58:50".

The raw data CIMPLICITY can get, in this case, year, month, day, hour, minute, and second, are converted into the character string in OnTimer( ).

```
For i=1 To 10
    alm_month = Val(PointGet ("ALME_DATE_REAL" &
Cstr(i) & "[1]"))
    if alm_month = 1   Then alm_date1 = "Jan"
    if alm_month = 2   Then alm_date1 = "Feb"
    if alm_month = 3   Then alm_date1 = "Mar"
    if alm_month = 4   Then alm_date1 = "Apr"
    if alm_month = 5   Then alm_date1 = "May"
    if alm_month = 6   Then alm_date1 = "Jun"
    if alm_month = 7   Then alm_date1 = "Jul"
    if alm_month = 8   Then alm_date1 = "Aug"
    if alm_month = 9   Then alm_date1 = "Sep"
    if alm_month = 10 Then alm_date1 = "Oct"
    if alm_month = 11 Then alm_date1 = "Nov"
    if alm_month = 12 Then alm_date1 = "Dec"
    alm_date2 = PointGet ("ALME_DATE_REAL" &
Cstr(i) & "[2]")
    alm_date3 = PointGet ("ALME_DATE_REAL" &
Cstr(i) & "[3]")
```

      alm_date4 = PointGet ("ALME_DATE_REAL" & Cstr(i) & "[4]")
      alm_date5 = PointGet ("ALME_DATE_REAL" & Cstr(i) & "[5]")
      alm_date = alm_date1 + " " + alm_date2 + ", " + alm_date3 + ":" + alm_date4 + ":" + alm_date5
      PointSet ("ALME_DATE[" & Cstr(i-1) & "]"),alm_dateNext i

Finaly, created date/time strings are set to the point
ALME_DATE[1 to 10], and these points are displayed on the screen.

And, cliking the alarm message, the detail infomations about its alarm
is displayed as HTML document.
The diagnostic contents are installed into the local hard drive, and
alarm infomatios are included.
The file path instead of, for example, SRVO-001, is
"C:/Diagnostics/alarms/SRVO.htm#SRVO-001".

When the alarm message is clicked, this file path instead of the alarm
is created in and given to Web Browser in DispHelp( ).

    err_type_no = PointGet("ALME_ID[" & Cstr(alm_no) & "]")
    err_type = ""

    if err_type_no =   1 then err_type = "SRIO"
    if err_type_no =   2 then err_type = "FIlE"
    if err_type_no =   3 then err_type = "PROG"
    if err_type_no =   4 then err_type = "COND"
    if err_type_no =   5 then err_type = "ELOG"
    if err_type_no =   6 then err_type = "MCTL"
    if err_type_no =   7 then err_type = "MEMO"
    if err_type_no =   8 then err_type = "GUID"
    if err_type_no =   9 then err_type = "TPIF"
    if err_type_no = 10 then err_type = "FLPY"
    if err_type_no = 11 then err_type = "SRVO"
    if err_type_no = 12 then err_type = "INTP"
    if err_type_no = 13 then err_type = "PRIO"

if err_type_no = 15 then err_type = "MOTN"
if err_type_no = 16 then err_type = "VARS"
if err_type_no = 17 then err_type = "ROUT"
if err_type_no = 18 then err_type = "WNDW"
if err_type_no = 19 then err_type = "JOG"
if err_type_no = 20 then err_type = "APPL"
if err_type_no = 21 then err_type = "LANG"
if err_type_no = 23 then err_type = "SPOT"
if err_type_no = 24 then err_type = "SYST"
if err_type_no = 25 then err_type = "SCIO"
if err_type_no = 26 then err_type = "PALT"
if err_type_no = 28 then err_type = "PMON"
if err_type_no = 29 then err_type = "TOOL"
if err_type_no = 31 then err_type = "PWD"
if err_type_no = 32 then err_type = "VISN"
if err_type_no = 33 then err_type = "DICT"
if err_type_no = 34 then err_type = "KCIL"
if err_type_no = 36 then err_type = "TKSP"
if err_type_no = 37 then err_type = "COPT"
if err_type_no = 38 then err_type = "APSH"
if err_type_no = 42 then err_type = "CMND"
if err_type_no = 43 then err_type = "RPM"
if err_type_no = 44 then err_type = "LNTK"
if err_type_no = 45 then err_type = "WEAV"
if err_type_no = 46 then err_type = "TCPP"
if err_type_no = 47 then err_type = "TAST"
if err_type_no = 48 then err_type = "MUPS"
if err_type_no = 49 then err_type = "MIGE"
if err_type_no = 50 then err_type = "LSR"
if err_type_no = 51 then err_type = "SEAL"
if err_type_no = 53 then err_type = "ARC"
if err_type_no = 54 then err_type = "TRAK"
if err_type_no = 55 then err_type = "CMCC"
if err_type_no = 56 then err_type = "SP"
if err_type_no = 57 then err_type = "MACR"
if err_type_no = 58 then err_type = "SENS"
if err_type_no = 59 then err_type = "COMP"
if err_type_no = 60 then err_type = "THSR"
if err_type_no = 61 then err_type = "QMGR"
if err_type_no = 64 then err_type = "DJOG"
if err_type_no = 66 then err_type = "HRTL"
if err_type_no = 67 then err_type = "HOST"
if err_type_no = 68 then err_type = "MENT"
if err_type_no = 69 then err_type = "SSPC"
if err_type_no = 70 then err_type = "FIG"
if err_type_no = 71 then err_type = "HIGH"
if err_type_no = 72 then err_type = "DX"
if err_type_no = 73 then err_type = "CNTR"
if err_type_no = 75 then err_type = "PFMS"
if err_type_no = 76 then err_type = "DNET"
if err_type_no = 82 then err_type = "CD"

```
if err_type_no = 84 then err_type = "DMDR"
if err_type_no = 85 then err_type = "FRSY"
if err_type_no = 87 then err_type = "FLEX"
if err_type_no = 88 then err_type = "IB-S"
if err_type_no = 89 then err_type = "RTCP"
if err_type_no = 90 then err_type = "TG"
if err_type_no = 92 then err_type = "PROF"
if err_type_no = 93 then err_type = "RPC"
if err_type_no = 95 then err_type = "TSDT"
if err_type_no = 99 then err_type = "ELSE"

err_no = PointGet("ALME_NO[" & Cstr(alm_no) & "]")
err_no_str = PointGet("ALME_MSG[" & Cstr(alm_no) &
"]")
err_num = right$("000" & Cstr(err_no), 3)

If (err_no_str <> "") And (err_no_str <> "R E S E T") And
(err_type <> "") Then
    fname = "C:/Diagnostics/alarms/" & err_type &
".htm#" & err_type & "-" & err_num
    Set IE = CreateObject("InternetExplorer.Application")
    IE.Navigate fname
    IE.Visible = True
    Set IE = Nothing
End If
```

When the alarm is not RESET, the variable fname is set the file path created from Alarm ID and Alarm Number, and is given to the Internet Explorer object.

If you hope to put the HTML contents on the Web Server, create the URL string from Alarm ID and Alarm Number such as, for example, "http://192.168.0.1/Diagnostics/alarms/SRVO.htm#SRVO-001"

INDEX

Revision Record

FANUC Robot series (R-J3/R-J3iB/R-30iA CONTROLLER) CIMPLICITY HMI for Robots
OPERATOR'S MANUAL (B-82604EN)

Edition	Date	Contents	Edition	Date	Contents
01	Feb., 2007				

B-82604EN/01