FANUC Robot series

R-30*i*A/R-30*i*A Mate/R-30*i*B/R-30*i*B Mate/R-30*i*B Plus/
R-30*i*B Mate Plus/R-30*i*B Compact Plus/R-30*i*B Mini Plus CONTROLLER

EtherNet/IP

OPERATOR'S MANUAL

B-82854EN/04

Thank you very much for purchasing FANUC Robot.

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

In this manual, we endeavor to include all pertinent matters. There are, however, a very large number of operations that must not or cannot be performed, and if the manual contained them all, it would be enormous in volume. It is, therefore, requested to assume that any operations that are not explicitly described as being possible are "not possible".

SAFETY PRECAUTIONS

This chapter describes the precautions which must be followed to enable the safe use of the robot. Before using the robot, be sure to read this chapter thoroughly.

For detailed functions of the robot operation, read the relevant operator's manual to understand fully its specification.

For the safety of the operator and the system, follow all safety precautions when operating a robot and its peripheral equipment installed in a work cell.
For safe use of FANUC robots, you must read and follow the instructions in the "FANUC Robot series SAFETY HANDBOOK (**B-80687EN)**".

1   PERSONNEL

Personnel can be classified as follows.

Operator:
- Turns the robot controller power ON/OFF
- Starts the robot program from operator panel

Programmer or Teaching operator:
- Operates the robot
- Teaches the robot inside the safeguarded space

Maintenance technician:
- Operates the robot
- Teaches the robot inside the safeguarded space
- Performs maintenance (repair, adjustment, replacement)

- The operator is not allowed to work in the safeguarded space.
- The programmer or teaching operator and maintenance technician are allowed to work in the safeguarded space. Work carried out in the safeguarded space include transportation, installation, teaching, adjustment, and maintenance.
- To work inside the safeguarded space, the person must be trained on proper robot operation.

Table 1 (a) lists the work outside the safeguarded space. In this table, the symbol "O" means the work allowed to be carried out by the specified personnel.

Table 1 (a) List of work outside the Safeguarded Space

	Operator	Programmer or Teaching operator	Maintenance technician
Turn power ON/OFF to Robot controller	O	O	O
Select operating mode (AUTO/T1/T2)		O	O
Select remote/local mode		O	O
Select robot program with teach pendant		O	O
Select robot program with external device		O	O
Start robot program with operator's panel	O	O	O
Start robot program with teach pendant		O	O
Reset alarm with operator's panel		O	O
Reset alarm with teach pendant		O	O
Set data on teach pendant		O	O
Teaching with teach pendant		O	O
Emergency stop with operator's panel	O	O	O
Emergency stop with teach pendant	O	O	O
Operator's panel maintenance			O
Teach pendant maintenance			O

During robot operation, programming and maintenance, the operator, programmer, teaching operator and maintenance technician take care of their safety using at least the following safety protectors:

- Use clothes, uniform, overall adequate for the work
- Safety shoes
- Helmet

2   DEFINITION OF SAFETY NOTATIONS

To ensure the safety of users and prevent damage to the machine, this manual indicates each precaution on safety with "**WARNING**" or "**CAUTION**" according to its severity. Supplementary information is indicated by "**NOTE**". Read the contents of each **"WARNING"**, "**CAUTION**" and "**NOTE**" before using the robot.

Symbol	Definitions
⚠**WARNING**	Used if hazard resulting in the death or serious injury of the user will be expected to occur if he or she fails to follow the approved procedure.
⚠**CAUTION**	Used if a hazard resulting in the minor or moderate injury of the user, or equipment damage may be expected to occur if he or she fails to follow the approved procedure.
NOTE	Used if a supplementary explanation not related to any of WARNING and CAUTION is to be indicated.

TABLE OF CONTENTS

APPENDIX

1   INTRODUCTION

The EtherNet/IP interface supports an I/O exchange with other EtherNet/IP enabled devices over an Ethernet network. The EtherNet/IP specification is managed by the Open DeviceNet Vendors Association (www.odva.org).

From the EtherNet/IP Specification (Release 1.0) Overview :

> "EtherNet/IP (Ethernet/Industrial Protocol) is a communication system suitable for use in industrial environments. EtherNet/IP allows industrial devices to exchange time-critical application information. These devices include simple I/O devices such as sensors/actuators, as well as complex control devices such as robots, programmable logic controllers, welders, and process controllers.
>
> EtherNet/IP uses CIP (Control and Information Protocol), the common network, transport and application layers also shared by ControlNet and DeviceNet. EtherNet/IP then makes use of standard Ethernet and TCP/IP technology to transport CIP communications packets. The result is a common, open application layer on top of open and highly popular Ethernet and TCP/IP protocols.
>
> EtherNet/IP provides a producer/consumer model for the exchange of time-critical control data. The producer/consumer model allows the exchange of application information between a sending device (e.g., the producer) and many receiving devices (e.g., the consumers) without the need to send the data multiple times to multiple destinations. For EtherNet/IP, this is accomplished by making use of the CIP network and transport layers along with IP Multicast technology. Many EtherNet/IP devices can receive the same produced piece of application information from a single producing device.
>
> EtherNet/IP makes use of standard IEEE 802.3 technology; there are no non-standard additions that attempt to improve determinism. Rather, EtherNet/IP recommends the use of commercial switch technology, with 100 Mbps bandwidth and full-duplex operation, to provide for more deterministic performance. "

The terms adapter and scanner are used throughout this manual. Although EtherNet/IP is a producer/consumer network, these terms are still appropriate to describe a device which creates the I/O connection (the scanner), and a device which responds to connection requests (the adapter). The scanner can also be called the connection originator. The adapter can also be called the connection target.

The following steps are necessary to configure EtherNet/IP with the robot as the adapter:

1   **Design and install the network.** It is critical to follow good network design and installation practices for a reliable network. Refer to Section 8.1.
2   **Set the IP addresses.** All devices on the network require a valid IP address. Refer to Section 2.3 for additional information for the robot.
3   **Configure the adapter devices.** Adapter devices might require configuration such as setting I/O sizes. Refer to Section 3.2.1 to configure the robot as an adapter.
4   **Configure the scanner devices.** Scanners must be configured with a list of devices (adapters) to connect to along with parameters for each connection. Refer to Section 3.2.2 to configure an Allen Bradley ControlLogix PLC to connect to the robot.
5   **Map EtherNet/IP I/O to digital, group, or UOP I/O points within the robot.** Refer to Section 6.2 for more information. Scanner connections can also be mapped to analog. Refer to Section 4.2.5.
6   **Backup the configuration.** Refer to Section 6.3 for details on doing this for the robot.

> **NOTE**
> If you need to perform diagnostics or troubleshooting, refer to Chapter 9
> DIAGNOSTICS AND TROUBLESHOOTING.

> **NOTE**
> For EtherNet/IP Safety function that exchanges safety signals on EtherNet/IP,
> please read "Dual Check Safety Function (ISO 13849-1:2006 compliant)
> operator's manual (B-83104EN)" or "Dual Check Safety Function operator's
> manual (B-83184EN) in addition to this manual.

2 SYSTEM OVERVIEW

2.1 OVERVIEW

The robot supports multiple connections. Each connection can be configured as either a Scanner connection, or as an Adapter connection. Adapter connections are normally to a cell controller or PLC to exchange cell interface I/O data. The EtherNet/IP Adapter option must be loaded to support this functionality.

Each Scanner connection can be configured to exchange I/O with a remote device capable of acting as an adapter on an EtherNet/IP network. The EtherNet/IP Scanner option must be loaded to support this functionality (the EtherNet/IP Scanner option includes the adapter functionality as well).

The EtherNet/IP interface corresponds to Rack 89 in the robot for I/O mapping. The slot number reflects the connection number from the EtherNet/IP interface user interface screen. Any amount of I/O can be mapped within EtherNet/IP, up to the maximum supported on the robot. Analog I/O is supported on scanner connections.

Good network design is critical to having reliable communications. Excessive traffic and collisions must be avoided or managed. Refer to Section 8.1 for details.

2.2 SPECIFICATION OVERVIEW

Table 2.2 (a), Table 2.2 (b) and Table 2.2 (c) provide an overview of specifications for EtherNet/IP.

Table 2.2 (a) R-30*i*A Specification Overview

Item	Specification
Number Adapter Connections	0–32
Number Scanner Connections	32 minus (the number of adapter connections)
Minimum RPI	8 msec
Maximum Number of Digital or Analog Inputs per connection	64 Words (16 bits each) or 128 Bytes (1 byte, 8 bits) or 1028 Points
Maximum Number of Digital or Analog Outputs per connection	64 Words (16 bits each) or 128 Bytes (1 byte, 8 bits) or 1028 Points
Maximum Number of Input bytes per connection (combination of Digital and Analog)	64 Words (16 bits each) or 128 Bytes (1 byte, 8 bits) or 1028 Points
Maximum Number of Output bytes per connection (combination of Digital and Analog)	64 Words (16 bits each) or 128 Bytes (1 byte, 8 bits) or 1028 Points
Supported Signal Types	Digital, Group, UOP, Analog (for scanner connections only)

Table 2.2 (b) R-30*i*B Specification Overview

Item	Specification
Number Adapter Connections	0–32
Number Scanner Connections	32 (minus the number of adapter connections)
Minimum RPI	8 msec
Maximum Number of Digital or Analog Inputs per connection	7DC1(V8.10P), 7DC2(V8.20P) : 248 Words (16 bits each) or 496 Bytes (1 byte, 8 bits) or 3968 Points 7DC3(V8.30P) : 252 Words (16 bits each) or 504 Bytes (1 byte, 8 bits) or 4032 Points
Maximum Number of Digital or Analog Outputs per connection	7DC1(V8.10P), 7DC2(V8.20P) : 248 Words (16 bits each) or 496 Bytes (1 byte, 8 bits) or 3968 Points 7DC3(V8.30P) : 252 Words (16 bits each) or 504 Bytes (1 byte, 8 bits) or 4032 Points
Maximum Number of Input bytes per connection (combination of Digital and Analog)	7DC1(V8.10P), 7DC2(V8.20P) : 248 Words (16 bits each) or 496 Bytes (1 byte, 8 bits) or 3968 Points 7DC3(V8.30P) : 252 Words (16 bits each) or 504 Bytes (1 byte, 8 bits) or 4032 Points
Maximum Number of Output bytes per connection (combination of Digital and Analog)	7DC1(V8.10P), 7DC2(V8.20P) : 248 Words (16 bits each) or 496 Bytes (1 byte, 8 bits) or 3968 Points 7DC3(V8.30P) : 252 Words (16 bits each) or 504 Bytes (1 byte, 8 bits) or 4032 Points
Supported Signal Types	Digital, Group, UOP, Analog (for scanner connections only)

Table 2.2 (c) R-30*i*B Plus Specification Overview

Item	Specification
Number Adapter Connections	0–64
Number Scanner Connections	64 (minus the number of adapter connections)
Minimum RPI	8 msec
Maximum Number of Digital or Analog Inputs per connection	252 Words (16 bits each) or 504 Bytes (1 byte, 8 bits) or 4032 Points
Maximum Number of Digital or Analog Outputs per connection	252 Words (16 bits each) or 504 Bytes (1 byte, 8 bits) or 4032 Points
Maximum Number of Input bytes per connection (combination of Digital and Analog)	252 Words (16 bits each) or 504 Bytes (1 byte, 8 bits) or 4032 Points
Maximum Number of Output bytes per connection (combination of Digital and Analog)	252 Words (16 bits each) or 504 Bytes (1 byte, 8 bits) or 4032 Points
Supported Signal Types	Digital, Group, UOP, Analog (for scanner connections only)

NOTE
- The maximum number of Input/Output bytes per connection in Table 2.2 (a), Table 2.2 (b) and Table 2.2 (c) is the range of value that can be set as the data size of EtherNet/IP connection. On the other hand, the maximum number of I/O that can be used in a robot controller differs according to the application software and other option software configuration.
- In order for the scanner to work, the EtherNet/IP Scanner option must be loaded.
- For EtherNet/IP Safety function that exchanges safety signals on EtherNet/IP, please read "Dual Check Safety Function (ISO 13849-1:2006 compliant) operator's manual (B-83104EN)" or "Dual Check Safety Function operator's manual (B-83184EN) in addition to this manual.

NOTE
Large Forward Open is not supported.

2.3 ETHERNET CONNECTION AND IP ADDRESS ASSIGNMENT

The robot must have a valid IP (Internet protocol) address and subnet mask to operate as an EtherNet/IP node. Details on the Ethernet interface and TCP/IP configuration can be found in the "Ethernet Function OPERATOR'S MANUAL (B-82974EN)".

The Ethernet interface supports 10Mbps and 100Mbps baud rates, along with half and full duplex communication. By default both interfaces will auto-negotiate and should be connected to a switch which supports 100Mbps full duplex connections. The LEDs located near the RJ45 connectors on the main CPU board are useful in confirming link establishment (for details on the LEDs, refer to appendix "Diagnostic Information" in the "Ethernet Function OPERATOR'S MANUAL (B-82974EN)").

The IP address(es) can be configured in the following ways :
- Manually configured on the robot teach pendant – Refer to the "Setting Up TCP/IP" chapter in the "Ethernet Function OPERATOR'S MANUAL (B-82974EN)".
- DHCP (Dynamic Host Configuration Protocol) – Refer to the "Dynamic Host Configuration Protocol" chapter in the "Ethernet Function OPERATOR'S MANUAL (B-82974EN)".

NOTE
DHCP is an optional software component. It is important to utilize static or infinite lease IP addresses when using EtherNet/IP.

Either one or both Ethernet ports can be configured for use with EtherNet/IP. Note that in order to use both ports at the same time they must be properly configured on separate subnets. Refer to the "Setting Up TCP/IP" chapter in the "Ethernet Function OPERATOR'S MANUAL (B-82974EN)". Also note that port 2 (CD38B) is optimized for Ethernet I/O protocols such as EtherNet/IP. The preferred setup is to connect port 1 (CD38A) to your building network to access the robot through HTTP, FTP, and so forth, and to connect port 2 (CD38B) to an isolated network for use by EtherNet/IP.

NOTE
Be sure that all EtherNet/ IP node IP addresses are configured properly before you perform the functions in this manual. The PING utility can be used to verify basic communications. Refer to Chapter 9 DIAGNOSTICS AND TROUBLESHOOTING for more information.

> **NOTE**
> Set the system variable $TCPIPCFG.$ARPSIZE to the number of connected devices or more, when connecting more than 17 devices. The maximum limit of value is 255. The default value of the system variable $TCPIPCFG.$ARPSIZE is changed from 16 to 32 in 7DF1(V9.10P)/41 or later, 7DF3(V9.30P)/24 or later, 7DF5(V9.40P)/29 or later.

2.4 ADAPTER MODE CONFIGURATION OUTLINE

Perform the following steps to configure the adapter connection on the robot:

- Configure the I/O size on the robot. Refer to Section 3.2.1.
- Map the physical EtherNet/IP I/O to logical I/O points (digital, group, or UOP) on the robot. Refer to Section 6.2.
- Configure the remote scanner (for example, ControlLogix PLC). Refer to Section 3.2.2.

Table 2.4(a) and Table 2.4(b) provides a summary of the adapter configuration. This information is used in the scanner device (for example, PLC) configured to communicate with the robot EtherNet/IP Adapter interface.

Table 2.4 (a) R-30*i*A/R-30*i*B Adapter Configuration Summary

ITEM	DESCRIPTION
Vendor ID	356 or 8 (Please refer Table 2.4(c))
Product Code	2 or 32 (Please refer Table 2.4(c))
Device Type	12 or 140 (Please refer Table 2.4(c))
Communication Format	Data － INT
Input Assembly Instance	101 － 132
Input Size	User Configurable, Set in 16-bit Words
Output Assembly Instance	151 － 182
Output Size	User Configurable, Set in 16-bit Words
Configuration Instance	100
Configuration Size	0

Table 2.4 (b) R-30*i*B Plus Adapter Configuration Summary

ITEM	DESCRIPTION
Vendor ID	356
Product Code	4 or 40 (Please refer Table 2.4(d))
Device Type	12 or 140 (Please refer Table 2.4(d))
Communication Format	Data － INT
Input Assembly Instance	101 － 132 for slots 1 － 32 and 1101 － 1132 for slots 33－ 64
Input Size	User Configurable, Set in 16-bit Words
Output Assembly Instance	151 － 182 for slots 1 － 32 and 1151 － 1182 for slots 33－ 64
Output Size	User Configurable, Set in 16-bit Words
Configuration Instance	100
Configuration Size	0

The default I/O size for the adapter connection is four words for both inputs and outputs. This corresponds to 64 I/O points based on a 16-bit word. This size must be configured on the robot teach pendant, as well as on the remote scanner (for example, PLC).

Refer to Chapter 3 ADAPTER CONFIGURATION for details.
Adapter identity information can be found in EDS files. For quick look, it is listed below in Table 2.4 (c), Table2.4 (d).

Table 2.4 (c) R-30*i*A/R-30*i*B Adapter Identity Information

ADAPTER TYPE	VENDOR ID	DEVICE TYPE	PRODUCT CODE	REVISION (MAJOR.MINOR)
Safety Disabled	356	12	2	(R-30*i*A) 2.2 (R-30*i*B) 2.4
Safety Enabled	8	140	32	2.2

Table 2.4(d) R-30*i*B Plus Adapter Identity Information

ADAPTER TYPE	VENDOR ID	DEVICE TYPE	PRODUCT CODE	REVISION (MAJOR.MINOR)
Safety Disabled	356	12	4	3.1
Safety Enabled	356	140	40	3.1

2.5 SCANNER MODE CONFIGURATION OUTLINE

The robot must be configured to initiate EtherNet/IP connections.
Perform the following steps to configure the scanner connection on the robot :
- Configure the robot scan list on the teach pendant. Refer to Section 4.2.3.
- Map the physical EtherNet/IP I/O to logical I/O points (for example, digital, group, analog, or UOP) on the robot. Refer to Section 6.2.

For each connection the following data must be provided on the robot teach pendant. (Refer to the manual that applies to the adapter device being configured for more information.)

- Name/IP address
- Vendor ID
- DeviceType
- Product Code
- Input Size (16-bit words or 8-bit byes)
- Output Size (16-bit words or 8-bit byes)
- RPI (ms)
- Input assembly instance
- Output assembly instance
- Configuration instance

> **NOTE**
> The robot currently cannot be configured for devices with a non-zero configuration size from the teach pendant. Refer to Appendix A for information on third party configuration tools.

3   ADAPTER CONFIGURATION

3.1   OVERVIEW

The robot adapter connections are normally to a cell controller or PLC to exchange cell interface I/O data. The EtherNet/IP Adapter Option must be loaded to support this functionality.

The following steps are required to configure the adapter connection on the robot:
- Configure I/O size on the robot. Refer to Section 3.2.1.
- Map the physical EtherNet/IP I/O to logical I/O points (digital, group, or UOP) on the robot. Refer to Section 6.2.
- Configure the remote scanner (for example, ControlLogix PLC). Refer to Section 3.2.2.

3.2   SETTING UP YOUR ROBOT

3.2.1   Configuring the Robot I/O Size

The Input size and Output size are set in 16-bit word sizes. This means if 32 bits of input and 32 bits of output are needed then the Input size and Output Size would be set to 2 words each. The default size of the adapter connection is 4 words (64 bits) of input and 4 words (64 bits) of output. Changes in I/O size require you to turn off and turn on the robot to take effect.

Refer to Procedure 3-1 to configure I/O size on the robot.

Table 3.2.1 (a) describes the items displayed on the EtherNet/IP Status screen.

Table 3.2.1 (b) describes the items on the EtherNet/IP Configuration screen.

Table 3.2.1 (a) EtherNet/IP Status Screen Descriptions

ITEM	DESCRIPTION
Description Default: ConnectionX where X is the slot number of the Adapter.	This item is the description of the adapter or scanner. This can be set as desired to coordinate with your equipment.
TYP Default: ADP	This item indicates whether the connection is configured as an Adapter, or as a Scanner.
Enable Default: TRUE (for slot 1), FALSE (except for slot 1)	This item indicates whether the adapter or scanner is enabled (TRUE) or disabled (FALSE).
Status	The **Status** field can have the following values : • OFFLINE– the connection is disabled. • ONLINE – the connection is enabled but is not active (for example, waiting for a connection). • RUNNING - the connection is enabled and active (I/O is being exchanged). • <RUNNING> - the connection is enabled and active (I/O is being exchanged), and auto-reconnect is enabled. See Table 4.2.4 for more information on the auto-reconnect setting. • PENDING – changes have taken place in configuration. You must turn off the robot, then turn it on again.
Slot	This item is the value used when mapping EtherNet/IP I/O to digital, group, or UOP I/O signals.

Table 3.2.1 (b) EtherNet/IP Configuration Screen Descriptions

ITEM	DESCRIPTION
Description	This item is the comment that shows up on the Status screen. It is set on the Status screen as well.
Input size (words) Default:	This item is the number of 16 bit words configured for input.
Output size (words) Default:	This item is the number of 16 bit words configured for output.
Alarm Severity Default: WARN	This item indicates the severity of alarm that will be posted by the adapter connection. The valid choices are STOP, WARN, and PAUSE.
Scanner IP	The IP address of the connected scanner.
API 0 => T	Actual Packet Interval at which the scanner/originator is producing.
API T => 0	Actual Packet Interval at which the adapter/target is producing.

Procedure 3-1 Configuring I/O Size on the Robot

Steps

1   Press the [MENU] key.
2   Select I/O.
3   Press F1, [TYPE], and select EtherNet/IP. You will see a screen similar to the following.

```
I/O EtherNet/IP            JOINT  10 %
EtherNet/IP List(Rack 89)       1/64
 Description  TYP  Enable  Status  Slot
 Connection1  ADP   TRUE   ONLINE   1
 Connection2  ADP  FALSE  OFFLINE   2
 Connection3  ADP  FALSE  OFFLINE   3
 Connection4  ADP  FALSE  OFFLINE   4
 Connection5  ADP  FALSE  OFFLINE   5
 Connection6  ADP  FALSE  OFFLINE   6
 Connection7  ADP  FALSE  OFFLINE   7
 Connection8  ADP  FALSE  OFFLINE   8
```

Refer to Table 3.2.1(a) for descriptions of these screen items.

4   Move the cursor to select a connection. If the connection is configured as a scanner, move the cursor to the TYP column and press F5, [ADP]. This configures the connection as an adapter.
5   Move the cursor to select the desired adapter. If you plan to make changes to the adapter configuration, you must first disable the connections. Otherwise, the configuration screen is read-only.

> **NOTE**
>   If the adapter connection is Enabled, the first line of the adapter configuration screen will display "Adapter config (Read-only)" and the items on the screen cannot be modified. To make changes to the adapter configuration screen, you must disable the adapter connection on the EtherNet/IP Status screen.

6   To change adapter status:
   a.   Move the cursor to highlight the field in the Enable column for the adapter.
   b.   To disable the adapter and change the status to OFFLINE, press F5, [FALSE].
         To enable the adapter and change the status to ONLINE, press F4, [TRUE].

7   Move the cursor to the Description column. Press F4, [CONFIG]. You will see a screen similar to the following.

```
Adapter configuration :
     Description :      Connection1
     Input size (words) : 4
     Output size(words) : 4
     Alarm Severity : WARN

     Scanner IP : ****************
     API O=>T : 0
     API T=>O : 0
```

Refer to Table 3.2.1 (b) for descriptions of these screen items.

8   To change the I/O size:
     a   Move the cursor to select "Input size (words)."
     b   Type the value you want and press the [Enter] key.
     c   Move the cursor to select "Output size (words)."
     d   Type the value you want and press the [Enter] key.
     e   Move the cursor to select the alarm severity.
     f   Pres F4, [CHOICE], and select the desired severity.
     g   To return to the previous screen, press F3, [PREV].

9   After modifying the adapter configuration, you must enable the connection on the EtherNet/IP status screen. If any changes were made, the status will show as "PENDING". This indicates that you must cycle power in order for the changes to take effect.

NOTE
   To map EtherNet/IP I/O to digital, group, or UP I/O, refer to Section 6.2.

3.2.2   Configuring the Remote Scanner

The EtherNet/IP Interface status screen should show that the adapter connection is ONLINE. This means it is available and waiting for a request from a scanner (for example, PLC) to exchange I/O. If the adapter status is not ONLINE, refer to Procedure 3-1, Step 6.

Table 3.2.2 (a) and Table 3.2.2 (b) provide a summary of the adapter configuration. This information is used to configure the remote scanner (for example, PLC).

Table 3.2.2 (a) R-30*i*A/R-30*i*B Adapter Configuration Summary

ITEM	DESCRIPTION
Vendor ID	356 or 8 (Please refer Table 2.4(c))
Product Code	2 or 32 (Please refer Table 2.4(c))
Device Type	12 or 140 (Please refer Table 2.4(c))
Communication Format	Data – INT
Input Assembly Instance	101 –132
Input Size	User Configurable, Set in 16-bit Words
Output Assembly Instance	151 –182
Output Size	User Configurable, Set in 16-bit Words
Configuration Instance	100
Configuration Size	0

Table 3.2.2 (b) R-30*i*B Plus Adapter Configuration Summary

ITEM	DESCRIPTION
Vendor ID	356
Product Code	4 or 40 (Please refer Table 2.4(d))
Device Type	12 or 140 (Please refer Table 2.4(d))
Communication Format	Data – INT
Input Assembly Instance	101 –132 for slots 1 –32 and 1101 — 1132 for slots 33–64
Input Size	User Configurable, Set in 16-bit Words
Output Assembly Instance	151 –182 for slots 1 –32 and 1151 –1182 for slots 33–64
Output Size	User Configurable, Set in 16-bit Words
Configuration Instance	100
Configuration Size	0

Table 3.2.2 (c) Connection Points

Slot Number	Input Assembly Instance	Output Assembly Instance
1	101	151
2	102	152
3	103	153
4	104	154
5	105	155
6	106	156
7	107	157
8	108	158
9	109	159
10	110	160
11	111	161
12	112	162
13	113	163
14	114	164
15	115	165
16	116	166
17	117	167
18	118	168
19	119	169
20	120	170
21	121	171
22	122	172
23	123	173
24	124	174
25	125	175
26	126	176
27	127	177
28	128	178
29	129	179
30	130	180
31	131	181
32	132	182
33	1101	1151
34	1102	1152
35	1103	1153
36	1104	1154
37	1105	1155
38	1106	1156
39	1107	1157

Slot Number	Input Assembly Instance	Output Assembly Instance
40	1108	1158
41	1109	1159
42	1110	1160
43	1111	1161
44	1112	1162
45	1113	1163
46	1114	1164
47	1115	1165
48	1116	1166
49	1117	1167
50	1118	1168
51	1119	1169
52	1120	1170
53	1121	1171
54	1122	1172
55	1123	1173
56	1124	1174
57	1125	1175
58	1126	1176
59	1127	1177
60	1128	1178
61	1129	1179
62	1130	1180
63	1131	1181
64	1132	1182

> **NOTE**
> The slots after slot 33 can only be used with the R-30*i*B Plus Controller.

Use Procedure 3-2 or Procedure 3-3 to configure the Allen Bradley ControlLogix PLC. For other scanners, refer to their configuration software in conjunction with Table 3.2.2 (a).

Procedure 3-2 Configuring the Scanner Using Generic Ethernet Module in RS-Logix5000 Software

Steps

> **NOTE**
> The following screens show how to configure the scanner using RS-Logix5000 software, which is used with the Allen Bradley ControlLogix PLC. This example assumes that an EtherNet/IP Bridge module has been added to the configuration in the ControlLogix PLC.

1   To add the robot adapter connection to the configuration, right-click the EtherNet/IP Bridge module in the PLC, and select "New Module".
2   Select "Generic Ethernet Module," and click OK. You will see a screen similar to the following.
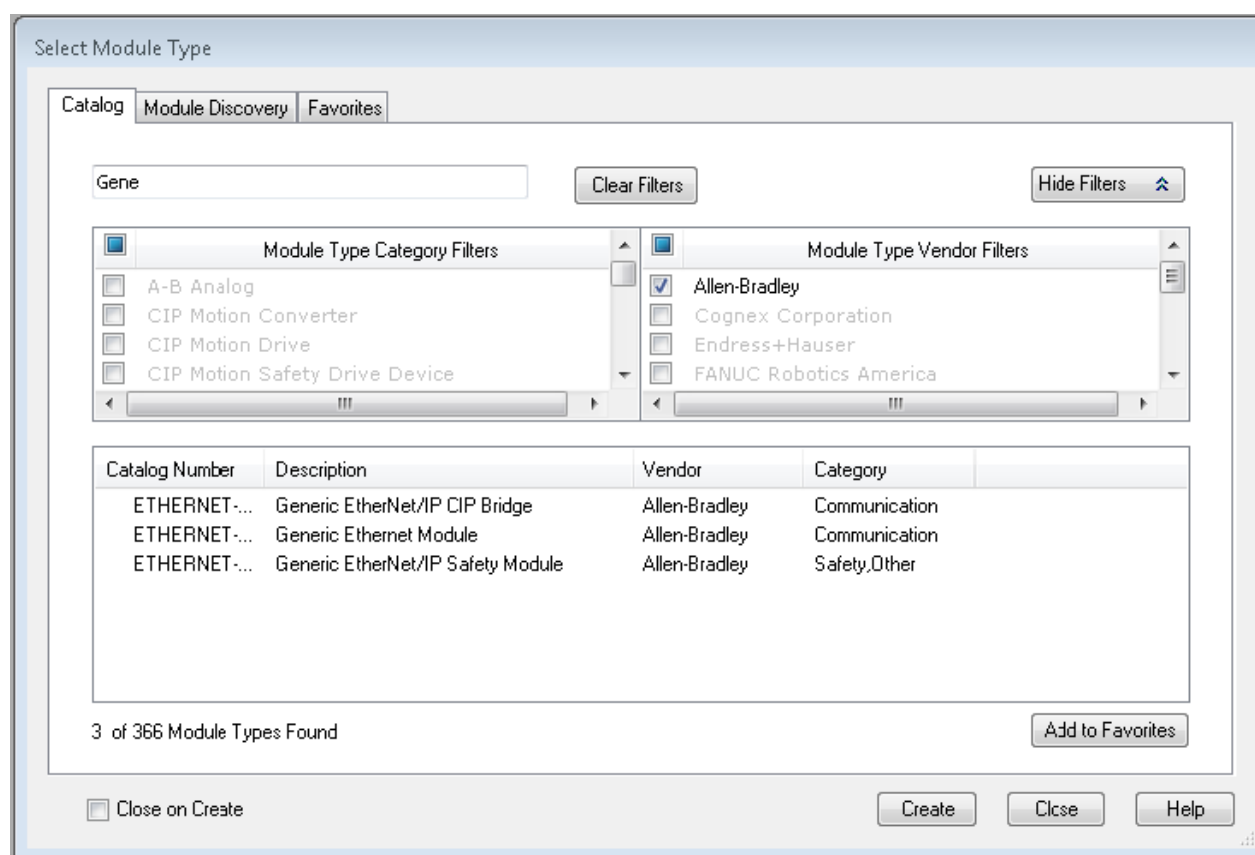
Fig. 3.2.2 (a) Create Ethernet/IP Scanner

3   In the next screen, RSLogix will ask for information regarding the communication to the robot. Type a name for the robot adapter connection.

In the example below we call the module "Example_Robot". This name will create a tag in RSLogix, which can be used to access the memory location in the PLCs memory where the data for the Example_Robot will be stored. A description can also be added if desired (optional).
4   Select the "Comm_Format," which tells RSLogix the format of the data.

In this example we select Data-INT, which will represent the data in the robot as a field of 16-bit words.

5   Set connection parameters as follows:

Each of the 64 Connections have different Connection parameters, which are based on the slot number. Refer to Table 3.2.2 (b) to determine the correct parameters. The size of the input connection and the output connection must correspond to the size that we have configured for the robot. In this example we configured the robot for 4 (16 bit) words of input data and output data. The configuration instance should be set to 100 and size of the configuration instance is set to 0.

In the following example, we will be setting up connection parameters corresponding to the adapter in slot 1.

6   Type the IP address that we have configured for the module.

This could be the Host Name if the DNS (Domain Name Service) is configured and available for the PLC to resolve names to IP addresses (if names are used be sure the DNS server is very reliable and always available to the PLC during operation). You will see a screen similar to the following.

7   The RPI (requested packet interval) is set on the next screen. This sets the rate for I/O updates.
In the following example the RPI is set to 10ms. This means the PLC will send its outputs to the robot every 10ms, and the robot will send its inputs to the PLC every 10ms. You will see a screen similar to the following.
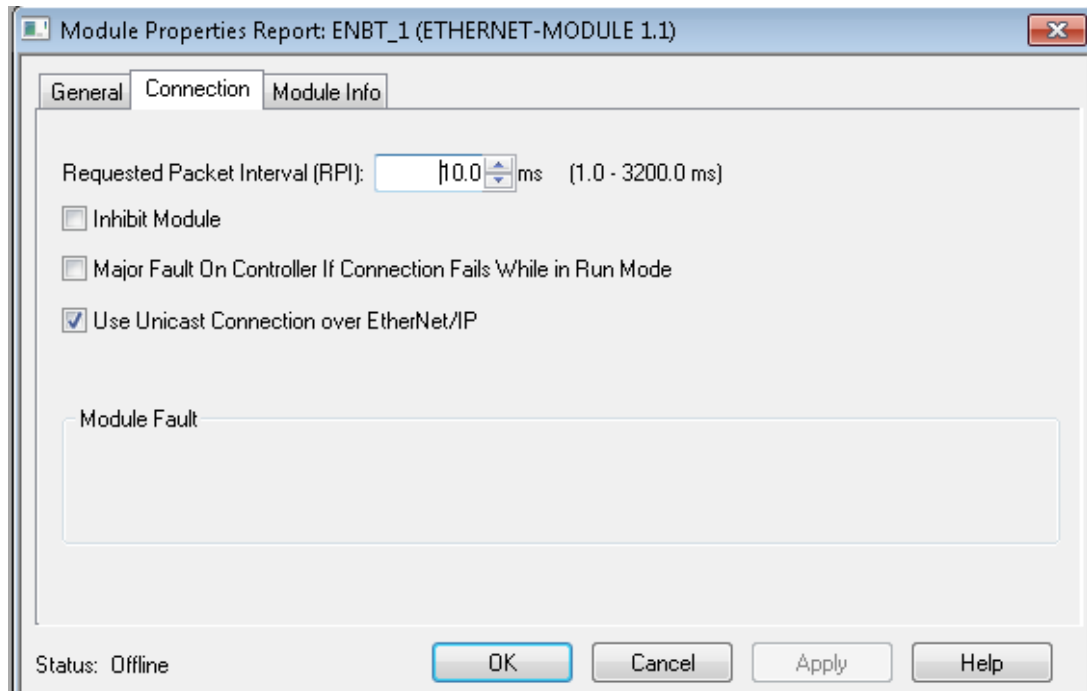
Fig. 3.2.2(b) Set RPI

8   Press Finish to complete this step.
9   To download the new configuration to the ControlLogix PLC, select Download from the
     Communications menu. You will see a screen similar to the following.
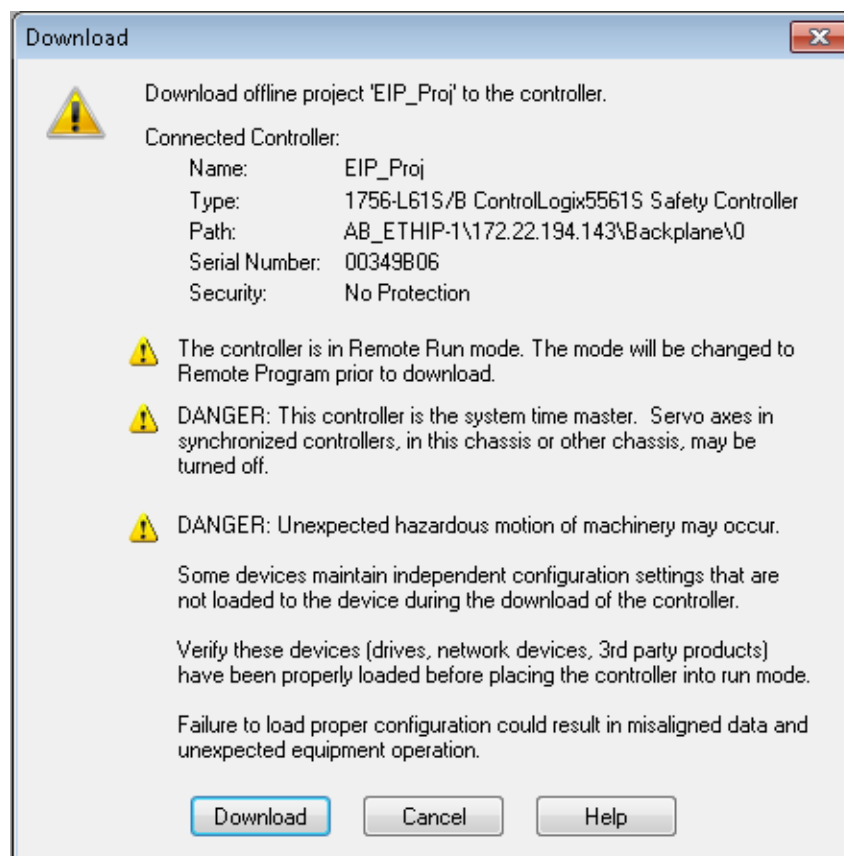
Fig. 3.2.2(c) Download Project

10   If there are any errors, a warning triangle will be present on the Example_Robot in the I/O
     configuration listing. Double click the module to view any error that is reported.

- 15 -

Procedure 3-3 Configure the Scanner using AOP (Add-On-Profile) in RS-Logix5000 Software

Steps

> **NOTE**
> - Procedure 3-3 corresponds to R-30*i*B Plus only.
> - The following screens show how to configure the scanner using FANUC's Add-On-Profile (AOP) in RS-Logix5000 software, which is used with the Allen Bradley ControlLogix PLC. AOP version that supports R30*i*B and R30*i*B Plus both is v1.39.01.0. This section demonstrates setting up the scanner connection between a GuardLogix PLC (v26.1)) and a FANUC Robot R30*i*B Plus using the v1.39.01.0 AOP. The FANUC Robot AOP v1.39.01.0 is included in the official Rockwell Studio 5000 v31 release. AOP version can be found following direction in Fig. 3.2.2 (d).
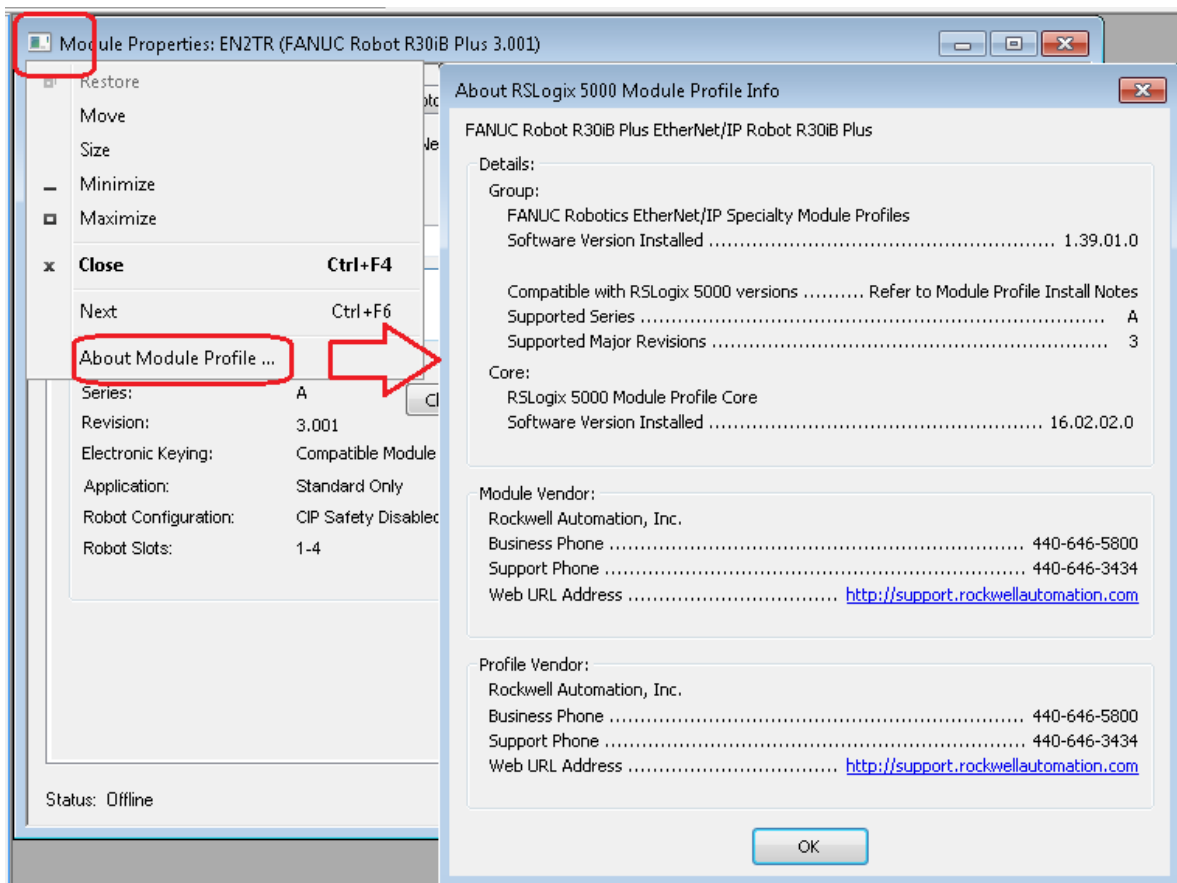
Fig. 3.2.2 (d) AOP Version

- Standard I/O Connection only (up to 500 bytes each way)
- Standard (up to 500 bytes each way) and Safety I/O (up to 8 bytes) Connections

1   Starting with an existing GLX Studio 5000 project choose "Add Module" for the appropriate Ethernet bridge module, select the FANUC Robot or FANUC Robot R30*i*B Plus variant for R30*i*B Plus AOP, and click create in Fig. 3.2.2(e) :
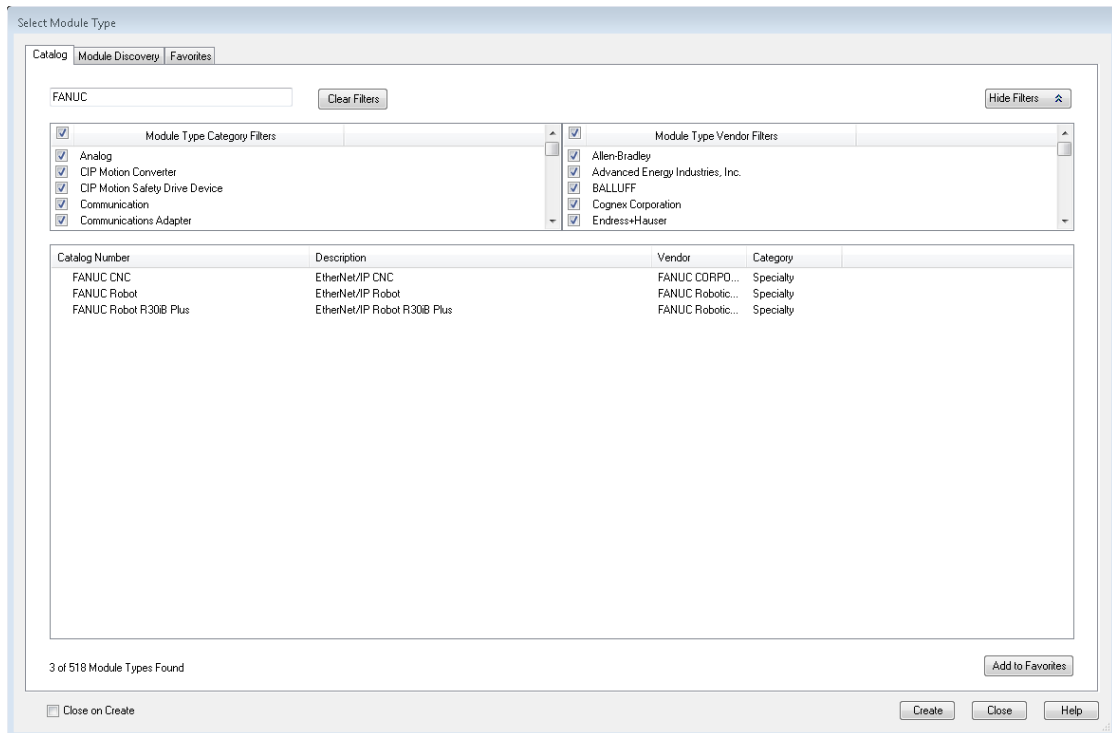
Fig. 3.2.2 (e) Create Ethernet/IP Scanner using AOP

2   Enter Name and IP Address for the target robot connection in Fig. 3.2.2 (f) :
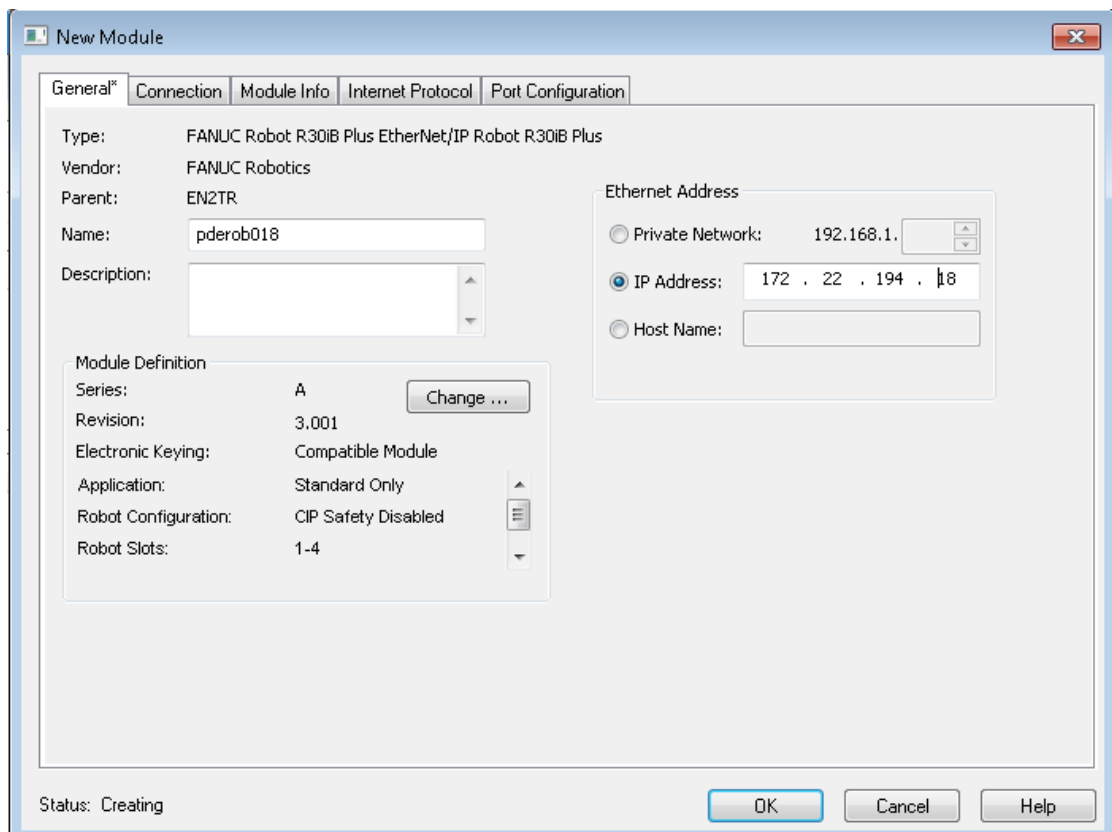
Fig. 3.2.2 (f) Set Name and IP Address of the Robot

3   Click Change… in Module Definition area and choose Compatible Module under Electronic Keying
     (Please refer Table 2.4(d) if Exact Match keying is desired) :
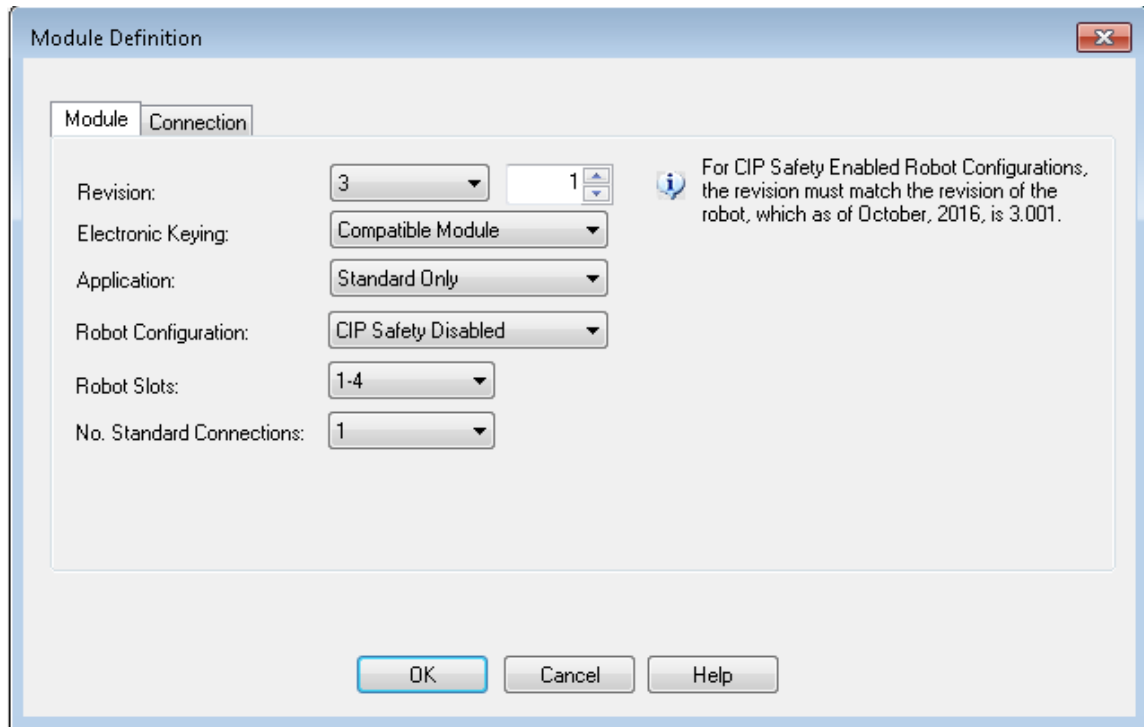
Fig. 3.2.2 (g) Set Electronic Keying

4   Click Connection Tab in Module Definition window and set Input and Output sizes. The default for
     the first connection in the robot when Ethernet/IP Adapter option is loaded is 4 words (8 bytes) each
     way. Please note that size in AOP is in bytes (8‑ bit) as shown in Fig. 3.2.2 (h). Setting the robot to
     a different IO size is detailed in Procedure 3-1.
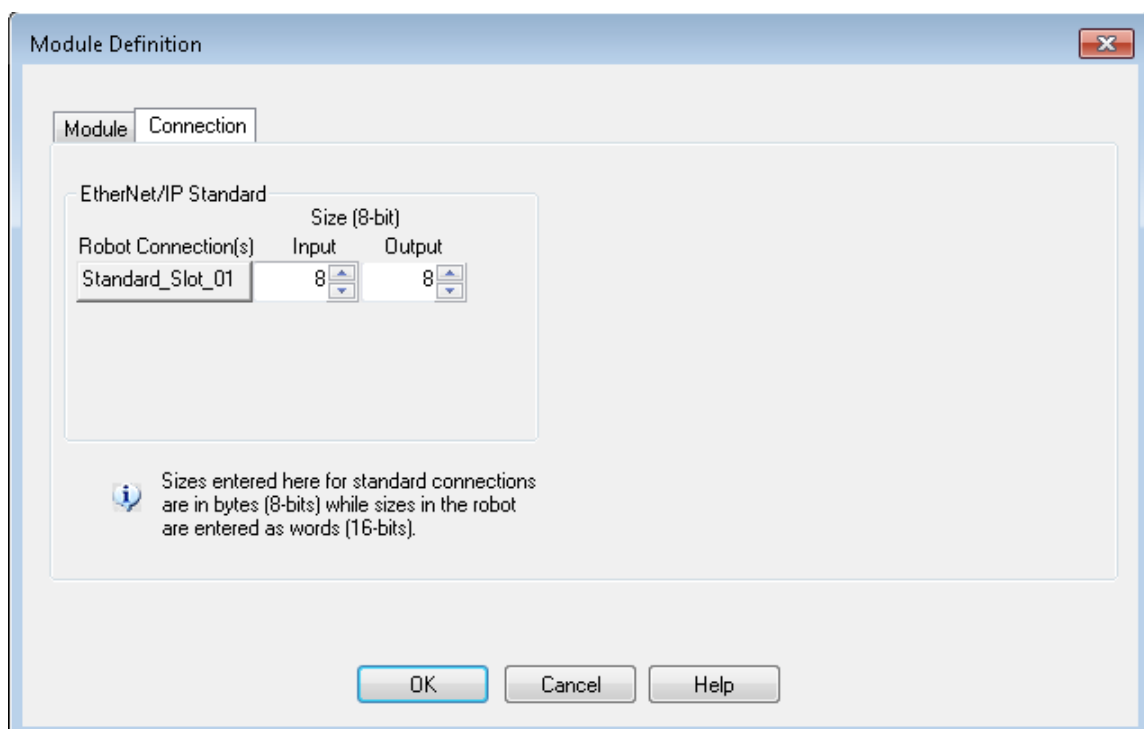
Fig. 3.2.2 (h) Set I/O Sizes

5   Click OK and accept the warning that these changes will cause module data types and properties to change. It comes back to main screen as shown in Fig. 3.2.2 (i).
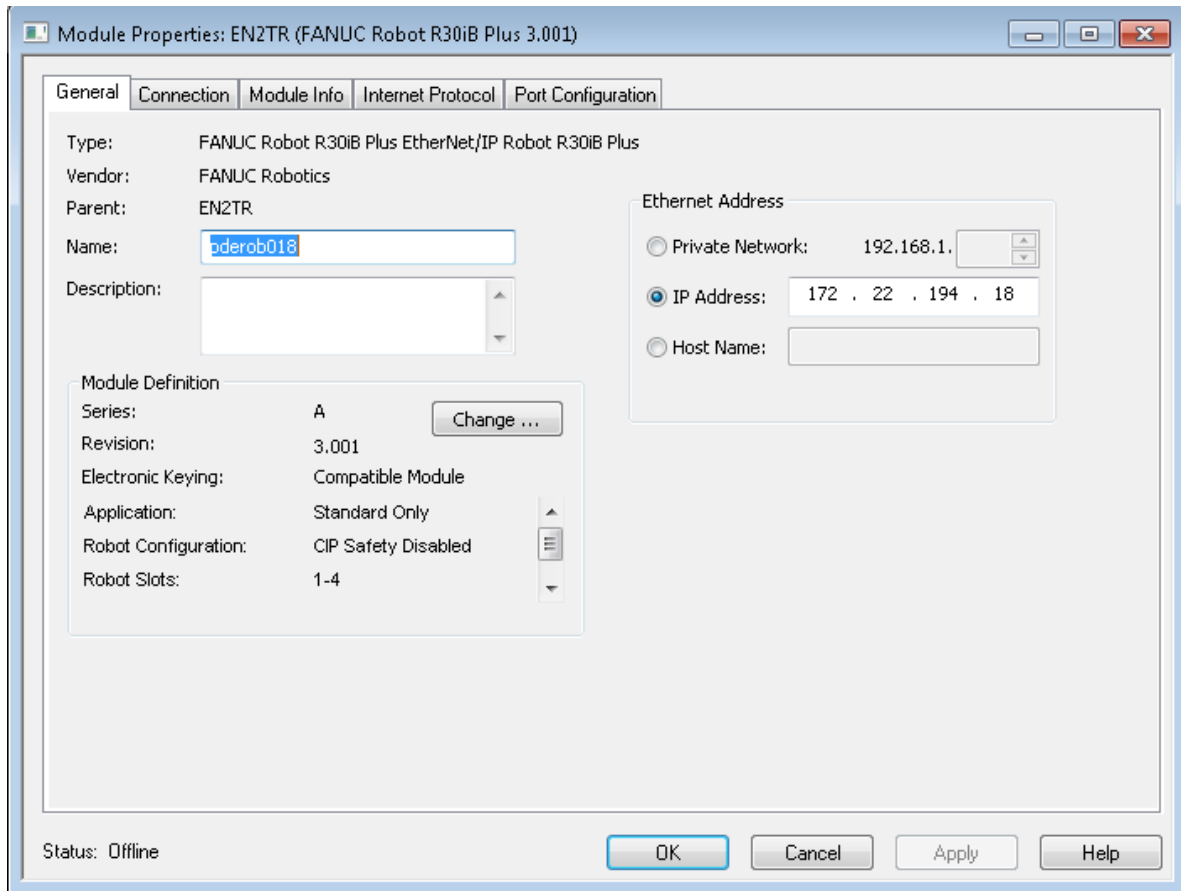
Fig. 3.2.2 (i) Completed Setup

6   Click OK to complete the configuration of this connection. Save and Download the project to the PLC as shown in step 9 of Procedure 3-2. The connection should begin running as shown in Fig. 3.2.2 (j). If the connection shows faulted look under the module fault area under the connection tab for that connection to begin debugging it :
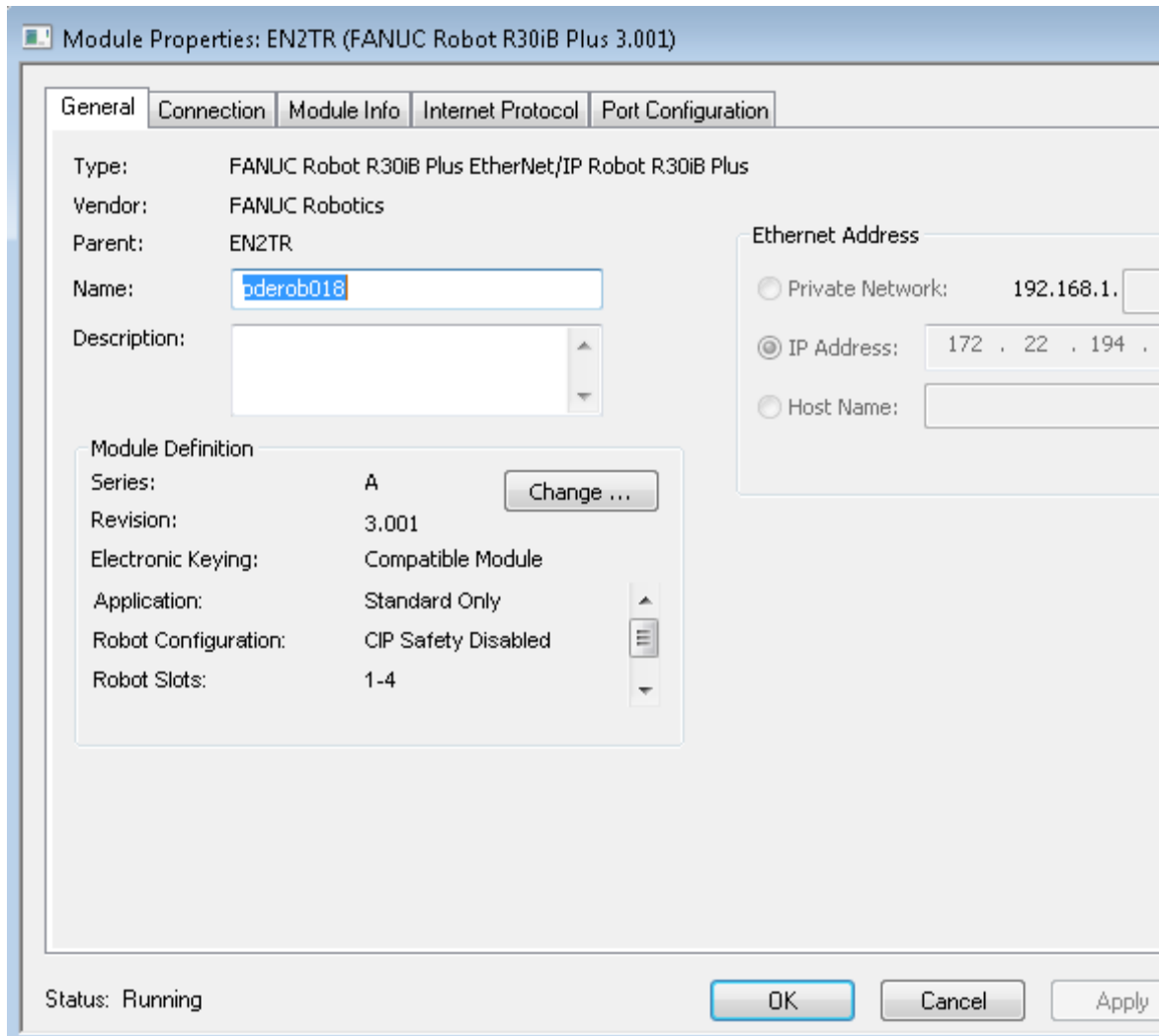
Fig. 3.2.2 (j) Running Status

3.2.3   Common Errors

The robot will post an alarm indicating that the adapter connection is idle if it is enabled but no scanner has connected to it. This is an informational alarm (warning level) by default. This message is reposted whenever the robot RESET button is pressed and the adapter connection is idle. If desired, the error severity level of EtherNet/IP adapter alarms can be increased. See Step 8 of Procedure 3-1 for more details.

If the adapter connection is lost the values of any mapped inputs will be zeroed out (by default). The last state behavior can be changed by setting the following system variable : $EIP_CFG.$KEEP_IO_ADP. The values are :

- FALSE : The last state values of the adapter inputs will be zero (default)
- TRUE : The last state values of the adapter inputs will be their last value

> **NOTE**
> If some of the EtherNet/IP adapter I/O signals are configured as UOP HOLD/IMSTP signals and communication is interrupted, then the default behavior will cause the robot to stop. This is due to the UOP inputs going to zero (last state behavior) and causing UOP HOLD and IMSTP alarms to be posted. It is typical for the adapter connection to be configured so that alarms are of "WARNING" severity and the "Last State" is set to FALSE (default behavior), which allows UOP in/out signals to stop the robot operation.

4   SCANNER CONFIGURATION

4.1   OVERVIEW

The robot scanner connections can be configured to exchange I/O with a remote device capable of acting as an adapter on an EtherNet/IP network. The EtherNet/IP Scanner option must be loaded to support this functionality. The EtherNet/IP Scanner option includes the adapter functionality as well. An example of an EtherNet/IP adapter device that the robot would connect to might be an I/O block.

The robot must be configured to initiate EtherNet/IP connections. Perform the following steps to configure the scanner connection on the robot :

- Configure the adapter device if required. Refer to Section 4.2.2.
- Configure the robot scan list on the teach pendant. Refer to Section 4.2.3 or configure the robot scan list from RSNetWorx for EtherNet/IP (Refer to Appendix A).
- Map the physical EtherNet/IP I/O to logical I/O points (digital, group, analog, or UOP) on the robot. Refer to Section 6.2.

> **NOTE**
> All scanlist configurations must either be done entirely from the *i*Pendant, or be done entirely from a third-party configuration tool such as RSNetWorx for EtherNet/IP.

4.2   SETTING UP YOUR ROBOT

4.2.1   Overview

For each connection, the following data must be provided on the robot teach pendant (see documentation for the adapter device being configured for more information) :

- Name/IP address
- Vendor ID
- Device Type
- Product Code
- Input Size (16-bit words or 8-bit byes)
- Output Size (16-bit words or 8-bit byes)
- RPI (ms)
- Input assembly instance
- Output assembly instance
- Configuration instance

> **NOTE**
> The robot currently cannot be configured for devices with a non-zero configuration size from the teach pendant. Refer to Appendix A for information on third party configuration tools.

4.2.2   Configure the Adapter Device

See the documentation for the adapter device being configured. Configuring the adapter is typically a matter of connecting the device to the network and setting the IP address. The device should successfully respond to a PING request before proceeding. Refer to Chapter 9 DIAGNOSTICS AND TROUBLESHOOTING for details on using PING.

> **NOTE**
> Refer to section 2.2 for information of the number of scanner connections.

4.2.3   Configure the robot scan list

Use Procedure 4-1 to configure the robot scan list from the teach pendant.

Table 4.2.3 (a) describes the items displayed on the EtherNet/IP Status screen. Table 4.2.3 (b) describes the items displayed on the EtherNet/IP scanner configuration screen.

> **NOTE**
> When the scanner connection type is other than Exclusive-Owner, set O=>T formant and T=>O format by selecting the connection type in Advanced EtherNet/IP Scanner Configuration screen. Refer to 4.2.4 for more detail.

Table 4.2.3 (a) EtherNet/IP Status Screen Item Descriptions

ITEM	DESCRIPTION
Description Default: ConnectionX where X is the slot number of the Adapter.	This item is the description of the adapter or scanner. This can be set as desired to coordinate with your equipment.
TYP Default: ADP	This item indicates whether the connection is configured as an Adapter, or as a Scanner.
Enable Default: TRUE (for slot 1), FALSE (except for slot 1)	This item indicates whether the adapter or scanner is enabled (TRUE) or disabled (FALSE).
Status	The Status field can have the following values : • OFFLINE– the connection is disabled. • ONLINE – the connection is enabled but is not active (for example, waiting for a connection). • RUNNING - the connection is enabled and active (I/O is being exchanged). • \<RUNNING\> - the connection is enabled and active (I/O is being exchanged), and auto-reconnect is enabled. See Table 4.2.4 for more information on the auto-reconnect setting. • PENDING – changes have taken place in configuration. You must turn off the robot, then turn it on again.
Slot	This item is the value used when mapping EtherNet/IP I/O to digital, group, or UOP I/O signals.

Table 4.2.3 (b) Scanner Configuration Screen Item Descriptions

ITEM	DESCRIPTION
Description	This item is the comment that shows up on the Status screen.
Name/IP address	This item is the hostname or IP address of the device to which you are connecting. If a hostname is used, it must be in the local host table or available through DNS.
Vendor ID	This item is the vendor ID of the device to which you are connecting. Refer to the adapter target) device's documentation of EDS files for assigned value. The vendor ID, Device Type, and Product Code can be entered if electronic keying is needed this information must match the device in order to make a successful connection). If the fields are left at 0 then the keying is ignored.
Device Type	This item is the Device Type of the device to which you are connecting. Refer to the adapter target) device's documentation or EDS file for assigned value. The vendor ID, Device Type, and Product Code can be entered if electronic keying is needed this information must match the device in order to make a successful connection). If the fields are left at 0 then the keying is ignored.
Product code	This item is the product code of the device to which you are connecting. Refer to the adapter target) device's documentation or EDS file for assigned value. The vendor ID, Device Type, and Product Code can be entered if electronic keying is needed this information must match the device in order to make a successful connection). If the fields are left at 0 then the keying is ignored.
Input size Range: 0 – 64 (R-30iA), 0 – 248 (R-30iB), 0 – 252 (R-30iB Plus) Default: 0	This item is the number of words or bytes configured for input. The default data type is 16-bit words, but can be configured as 8-bit bytes. To change the data type, modify the I/O Data Type field in the EtherNet/IP Advanced Scanner Configuration Screen (See Table 4.2.4). The Input size and Output size need to match the adapter device to which the robot will connect.
Output size Range: 0 – 64 (R-30iA), 0 – 248 (R-30iB), 0 – 252 (R-30iB Plus) Default: 0	This item is the number of words or bytes configured for output.The default data type is 16-bit words, but can be configured as 8-bit bytes. To change the data type, modify the I/O Data Type field in the EtherNet/IP Advanced Scanner Configuration Screen (See Table 4.2.4). The Input size and Output size need to match the adapter device to which the robot will connect.
RPI (ms) Minimum : 8ms Max : 5000 Default: 32	This item is the requested packet interval. This defines how often I/O updates are done. The minimum value allowed is 8 ms, however this value should be set based on application requirements. Be aware that fast I/O updates cause excessive network traffic. The R-30iA controller can support a maximum of 1250 packet per second, and the R-30iB or later controller can support a maximum of 1500 packet per second. Both Originator-to-Target and Target-to-Originator packets must be factored into this calculation. The RPI must be calculated to sum of Ethernet port 1 and port 2.
Assembly instance (input)	The Input, Output, and Configuration instance values need to be set based on the adapter device to which the robot will connect.
Assembly instance (output)	The Input, Output, and Configuration instance values need to be set based on the adapter device to which the robot will connect.
Configuration instance	The Input, Output, and Configuration instance values need to be set based on the adapter device to which the robot will connect.

> **NOTE**
> Please note that up to 64 connections can be configured for R-30*i*B Plus.
> Maximum 48 connections are supported simultaneously. Approaching 48
> enabled connections may result in limited network services overall. The
> minimum value of RPI will become as follows on standard software load with
> Ethernet/IP option. If any other network applications are running, the minimum
> value of RPI may become larger. I/O sizes were distributed evenly on number of
> connections given maximum 1024 bytes of IN and 1024 bytes OUT. As noted I/O
> can't exceed 1024 bytes for each side (IN or OUT) totalling all connections.

> **NOTE**
> Connections after connection 33 are for R-30*i*B Plus only.

Procedure 4-1 Configuring the Robot Scan List

Steps

1   Press the [MENU] key.
2   Select I/O.
3   Press F1, [TYPE], and select EtherNet/IP. You will see a screen similar to the following.

```
I/O EtherNet/IP            JOINT 10 %
EtherNet/IP List(Rack 89)       1/64
 Description  TYP Enable  Status  Slot
 Connection1  ADP  TRUE   ONLINE    1
 Connection2  ADP  FALSE  OFFLINE   2
 Connection3  ADP  FALSE  OFFLINE   3
 Connection4  SCN  FALSE  OFFLINE   4
 Connection5  SCN  FALSE  OFFLINE   5
 Connection6  SCN  FALSE  OFFLINE   6
 Connection7  SCN  FALSE  OFFLINE   7
 Connection8  SCN  FALSE  OFFLINE   8
```

4   Move the cursor to the connection you want to set. If the connection is configured as an adaptor,
    move the cursor to the TYP column, and press F4. This configures the connection as a scanner.

> **NOTE**
> If the scanner connection is enabled, the first line of the scanner configuration
> screen will display "Scanner config (Read-only)" and the items on the screen
> cannot be modified. To make changes to the read-only scanner configuration
> screen, you must disable the scanner connection on the EtherNet/IP status
> screen.

5   To change scanner status:
    a   Move the cursor to highlight the field in the Enable column for the scanner you want to modify.
    b   To disable the scanner and change the status to OFFLINE, press F5, [FALSE].
        To enable the scanner and change the status to RUNNING, press F4, [TRUE].

> **NOTE**
> The status will not change until the connection has been established and I/O is
> being exchanged.

6   Press F4, [CONFIG]. You will see a screen similar to the following.

```
I/O EtherNet/IP            JOINT 10 %
Scanner configuration :         1/10
    Description :    Connection1
    Name/IP address : 192.168.0.12
    Vendor Id : 0
    Device Type : 0
    Product code : 0
    Input size (words): 1
    Output size (words): 1
    RPI (ms) : 32
    Assembly instance(input) : 1
    Assembly instance(output) : 2
    Configuration instance : 4
```

7   Move the cursor to select each item and set the appropriate value.

NOTE
> If you make changes to I/O size, you must turn off then turn on the controller in order for the changes to take effect. Other changes in the configuration do not require you to turn off then on the controller to take effect.

8   Press the [PREV] key to return to the EtherNet/IP Status screen. You can enable the connection. If the status is PENDING then you must turn off then turn on the controller in order for the changes to take effect.

NOTE
> Any enabled scanner connections which are not RUNNING or PENDING will be retried each time the robot is RESET.

NOTE
> To map EtherNet/IP I/P to digital, group, analog, or UOP I/O, refer to Section 6.2.

4.2.4   Advanced EtherNet/IP Scanner Configuration

An advanced EtherNet/IP configuration screen is provided to allow you to access advanced scanner configuration options. Table 4.2.4 describes the items displayed on the Advanced Scanner Configuration Screen. The advance screen can be accessed and configured by using Procedure 4-2.

NOTE
> Some items such as Quick Connect feature can be used in 7DC1 (V8.10P) or later.

Table 4.2.4 EtherNet/IP Advanced Scanner Configuration Screen Item Descriptions

ITEM	DESCRIPTION
I/O Data Type Default: 16-bit words	This item indicates allows changing the data type to 16-bit words or 8-bit bytes.
Timeout Multiplier Default: DEFAULT	This item indicates allows changing the timeout multiplier. When set to DEFAULT, the controller will intelligently choose an appropriate multiplier based on the RPI value.

ITEM	DESCRIPTION
Reconnect Default: FALSE	When recconet is TRUE, the scanner will attempt to re-establish the connection when the connection is enabled and in an OFFLINE state. The reconnect parameter was designed for tool changing applications, and enabling reconnect can have side effects like all EtherNet/IP alarms relating to connection establishment and connection time-outs for the corresponding connection will not be posted. As a result the user cannot notice when I/O is not updated correctly, which may cause injury or property damage. Therefore, the user must carry out risk assessments when setting reconnect to TRUE. For tool changing applications, the user must use the corresponding MACROS and follow the programming guidelines as described in Appendix B.
Major Revision Default: 0	This item indicates the major revision number of the device being scanned. Is sometimes required by third-party configuration devices.
Minor Revision Default: 0	The minor revision number of the device being scanned. Is sometimes required by third-party configuration devices.
Alarm Severity	This item indicates the severity of alarm that will be posted by the scanner connection. The valid choices are STOP, WARN, and PAUSE.
Quick Connect Default: FALSE	If this item is set to TRUE, the scanner will attempt to establish the connection in quick connect mode if connection is started using KAREL macro. See Appendix A for details. The quick connect parameter was designed for tool changing applications. Enabling connect using KAREL macro forces scanner to wait for gratuitous ARP from adapter device before initiating the connection. As such, it is recommended that non-tool changing applications do not enable this parameter in a production environment.
Originator To Target RPI(ms) Default: 32	This item indicates the Requested Packet Interval for the scanner to produce at in milliseconds. This field allows for the scanner to have different RPIs for producing and consuming data.
Transport Type Default: UNICAST	This item allows the scanner to request that the adapter send data using a point-to-point/unicast connection, or to multicast data. If multicasting is not required, we strongly recommend setting this value to UNICAST. However, a small number of adapter devices only support the MULTICAST setting.
Target To Originator RPI(ms) Default: 32	This item indicates the Requested Packet Interval for the scanner to consume at in milliseconds. This field allows for the scanner to have different RPIs for producing and consuming data.
Connection Type Default: (blank)	This item allows the user to set up a scanner connection of type Exclusive-Owner, Input-Only, or Listen-Only. When a connection type is selected, the O=>T Format and T=>O Format fields will automatically be modified to correspond with the selected Connection Type. This field will be blank after each power-cycle, as this field is only an aid in selecting the proper O=>T and T=>O formats. Exclusive-Owner is the most common connection type.
O=>T Format Default: Run/Idle Header	The format of the producer's data packet. By default this is set to Run/Idle Header, consistent with an Exclusive-Owner Connection Type.
T=>O Format Default: Modeless	The format of the consumer's data packet. By default this is set to Modeless, consistent with an Exclusive-Owner Connection Type.
Configuration String Status Size(bytes)	Some EtherNet/IP adapters accept or require a non-zero length configuration string. This configuration data can only be configured on the robot using a third party configuration tool such as RSNetWorx for EtherNet/IP (Refer to Appendix A in the manual). This status item displays how much configuration data is currently configured for the connection. If no third party configuration tool is used, this item will always be 0.

Procedure 4-2 Configuring Advanced Scanner Options

1   Press the [MENU] key.
2   Select I/O.
3   Press F1, [TYPE], and select EtherNet/IP.
4   Move the cursor to a Scanner connection.
5   Press F4, [CONFIG].
6   Press F2, [ADV]. You will see a screen similar to the following:

```
I/O EtherNet/IP              JOINT 100 %
Advanced configuration :          1/12
General
   I/O Data Type :     16-BIT WORDS
   Timeout Multiplier :4
   Reconnect :         FALSE
   Major Revision :    0
   Minor Revision :    0
   Alarm Severity :    STOP
   Quick Connect :     FALSE
Originator To Target
   RPI :                32
Target To Originator
   Transport Type :    UNICAST
   RPI :                32
Connection Type
   Type :               Exclusive-Owner
    O=>T Format :    Run/Idle Header
    T=>O Format :    Modeless
Configuration String Status
   Size(bytes) :        0
```

7   Move the cursor to select each item and set the appropriate value.
8   Press the [PREV] key to return to the EtherNet/IP Scanner configuration screen.
9   Press the [PREV] key to return to the EtherNet/IP Status screen. You can enable the connection.
     If the status is PENDING then you must turn off then turn on the controller in order for the changes
     to take effect.

4.2.4.1   Quick connect feature

> **NOTE**
> This feature can be used in 7DC1 (V8.10P) or later.

To make using EtherNet/IP feasible for changes in robot end-of-arm tooling, especially when frequent changes are involved, a methodology for EtherNet/IP "Quick Connect" must be established.

This means that the robot scanner should establish an EtherNet/IP implicit connection and begin exchanging I/O with a new end-of-arm adapter device "within 150 ms after receiving gratuitous ARP from the adapter device". As per spec EIP Vol 2 V1.11 adapter device shall power-up within 300 ms. Altogether power-up and first IO data exchange shall not exceed 500ms.

> **NOTE**
> Please note that Quick Connect feature is disabled by factory default setting.

Enable Quick Connect Feature in FANUC's Scanner

Quick Connect feature can be turned on by setting 'Quick Connect' parameter to TRUE. Quick Connect feature is only effective when it is started using KAREL macros. See Appendix B for more details.

> **NOTE**
> Please note that FANUC's adapter does not support quick connect feature.

Enable Quick Connect Feature in Adapter (not FANUC's)

Follow : Menu >> I/O >> EthernetI/P >> NEXT >>F2 (EXP_MSG) >> Input Mode >> F4 (Q-Conn).
You will see a screen similar to Fig. 4.2.4.1 (a).


```
I/O EtherNet/IP
                                                    2/3
        Explicit Message Query
              Input Mode:         Q-Conn
              IP Addr:            ███████████
              Class:              245
              Instance:           1
              Attribute:          12
              Service:            Get Att
              Value Size:         Byte(1)
              Value:              0




  [ TYPE ]              EXEC              ?  HELP
```

Fig. 4.2.4.1 (a) Enabling Quick Connect using Explicit Messaging

Fill-in IP address of target device and cursor down to Service. You will see a screen similar to Fig. 4.2.4.1 (b). Now you can choose the service you want to perform. To check current status of quick connect feature on target device leave default i.e. 'Get Att' or press F2 to select. Now cursor up or down to see 'EXEC' on F3 and press F3 to execute selected service.

```
I/O EtherNet/IP
                                                 3/3
     Explicit Message Query
          Input Mode:          Q-Conn
          IP Addr:              172.22.200.167
          Class:               245
          Instance:            1
          Attribute:           12
          Service:             Get Att
          Value Size:          Byte(1)
          Value:               0




      [ TYPE ]    Get Att     QC_ON      QC_OFF    ?  HELP
```

Fig. 4.2.4.1(b) Fill-in IP Address and Select Service

```
I/O EtherNet/IP
                                                 2/3
     Explicit Message Query
          Input Mode:          Q-Conn
          IP Addr:              172.22.200.167
          Class:               245
          Instance:            1
          Attribute:           12
          Service:             QC_ON
          Value Size:          Byte(1)
          Value:               1




      [ TYPE ]               EXEC              ?  HELP
```

Fig. 4.2.4.1(c) Select Quick Connect Service

To turn on quick connect feature press F3 (QC_ON) or press F4 (QC_OFF) to turn off quick connect feature on target device. Now cursor up or down (Fig. 4.2.4.1(c)) to see 'EXEC' on F3 and press F3 to execute selected service.

Quick Connection Time

The scanner connection is estimated as follows after device powers up and sends gratuitous ARP indicating that it is ready to communicate. This time may vary due to network load along with switch and adapter device behavior. Fig. 4.2.4.1 (d) shows general quick connect setup.

```
Connection Time = 60 + 10 x (no of devices −  3) miliseconds
```

Table 4.2.4.1. Connection Time

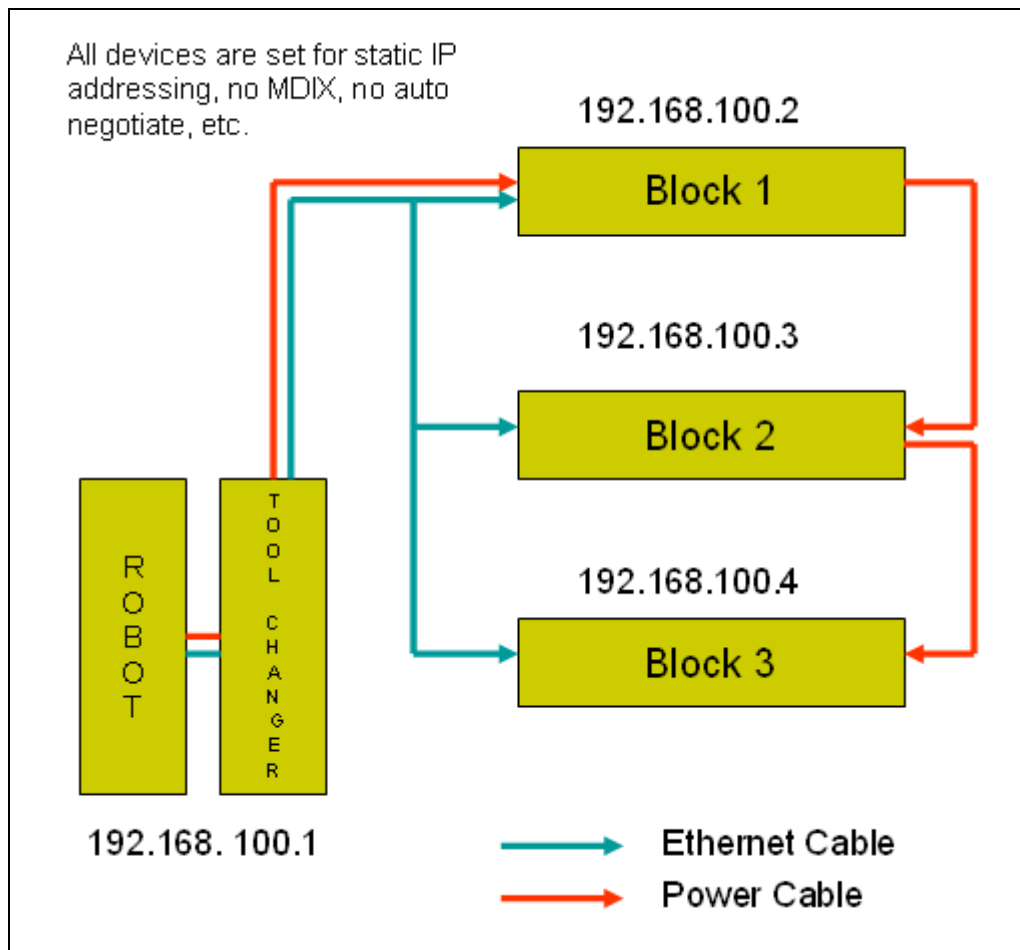| No of Devices	Connection Time
3	60 + 10 x (3–3) = 60 ms
7	60 + 10 x (7–3) = 100 ms

Fig. 4.2.4.1 (d) Quick Connection Setup

4.2.5   Analog I/O

4.2.5.1   Overview

I/O for EtherNet/IP Scanner connections can be mapped to analog. Analog and digital I/O can be intermixed—for example a device may produce sixteen points of Digital Inputs and two words of Analog Inputs on the same connection to the robot controller.

Table 4.2.5.1 describes the items displayed on the Scanner Analog Configuration Screen. The analog screen can be accessed and configured by using Procedure 4-3.

Table 4.2.5.1 Scanner Analog Configuration Screen Setup Items

ITEM	DESCRIPTION
RANGE Default: 1 – maximum allocated I/O	The Range of I/O points to be mapped to an analog channel, or a digital start point.
TYPE Default: Digital	The type of I/O being mapped: Analog or Digital.
START PT/CHNL Default: 1	The analog channel or the digital start point.

Procedure 4-3 Configuring Scanner Analog I/O

1   Press the [MENU] key.
2   Select I/O.
3   Press F1, [TYPE], and select EtherNet/IP.
4   Move the cursor to a Scanner connection.
5   Press F4, [CONFIG].
6   Type the correct input and output sizes.
7   Press F4, [ANALOG]. You will see a screen similar to the following.

```
I/O EtherNet/IP              JOINT 100 %
Map Inputs:                     1/1
  #       RANGE     TYPE    START PT/CHNL
  1 [    1- 128]  Digital        1
```

8   Move the cursor to the RANGE column and select the range of the first collection of Inputs. If you do not want to intermix Analog and Digital Inputs, do not modify this column.
9   Select the type of Inputs, Analog or Digital, in the TYPE column.
10   Select the channel for Analog Input, or the point for Digital Input in the START PT/CHNL column.
11   Repeat as necessary as additional rows are automatically created.
12   Press F2, [IN/OUT]. You will see a screen similar to the following.

```
I/O EtherNet/IP              JOINT 100 %
Map Outputs:                    1/1
  #       RANGE     TYPE    START PT/CHNL
  1 [    1- 128]  Digital        1
```

13   Repeat Step 8 through Step 11 as necessary for Outputs.
14   Press the [PREV] key to return to the EtherNet/IP Scanner configuration screen.
15   Press the [PREV] key to return to the EtherNet/IP Status screen. You can enable the connection. If the status is PENDING then you must turn off then turn on the controller in order for the changes to take effect.

> **NOTE**
> The 16-bits of each analog I/O channel can be byte-swapped (toggled from big-endian to little-endian) on a per connection basis by toggling the system variable $EIP_SC[].$ANALOGFMT.
> When the system variable is set to 0, the data will be produced and consumed in big-endian format.
> When set to 1, little-endian format will be used.

4.2.5.2   Examples

Suppose a device produces sixteen consecutive points of digital input followed by two words of analog input. A properly configured EtherNet/IP Analog In screen would look like the following:

```
I/O EtherNet/IP            JOINT 100 %
Map Inputs:                     1/2
  #      RANGE   TYPE     START PT/CHNL
  1 [   1- 16]  Digital        1
  2 [  17- 48]  Analog         1
```

Suppose a device produces sixteen consecutive points of digital input followed by two words of analog input followed by eight more consecutive points of digital input. A properly configured EtherNet/IP Analog In screen would look like the following. Note that the 49th connection input point will be mapped as the 17th digital input point.
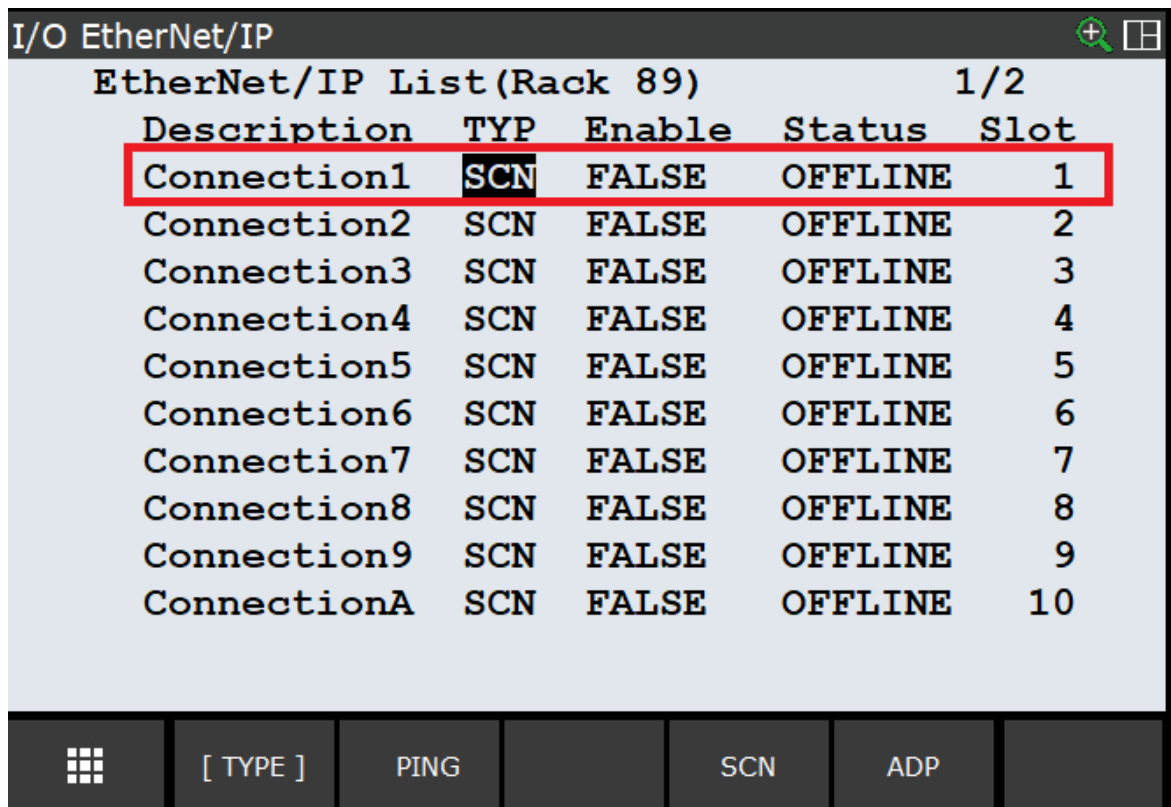
```
I/O EtherNet/IP            JOINT 100 %
Map Inputs:                     1/3
  #      RANGE   TYPE     START PT/CHNL
  1 [   1- 16]  Digital        1
  2 [  17- 48]  Analog         1
  3 [  49- 56]  Digital       17
```

4.2.6   Ethernet/IP Scanner Lite

This option is created to support customers who require one or two scanner connections. This has two separate options for single (Ethernet/IP Scanner Lt1) or double (EThernet/IP Scanner Lt2) scanners. Order number for Ethernet/IP Scanner Lt1 is **R889** and **R890** for Ethernet/IP Scanner Lt2. R889 supports only one scanner on slot 1 and R890 supports second connection on slot 2. Please note that Lite1 is required option to have Lite2. Unlike standard Ethernet/IP Scanner, Lite version doesn't include adaptor by default. It is low cost option so if adapter (R784) is needed, it needs to be ordered separately. For scanner configuration and functions, all standard scanner (R785) configuration and functions apply.

4.2.6.1   Ethrenet/IP scanner with LITE1 (R889)

If LITE1 option is loaded, it enables scanner at slot 1. Remaining 63 slots are inactive if ADP (R784) is not loaded. Below in Fig. 4.2.6.1 (a) only Connection1 can be configured.
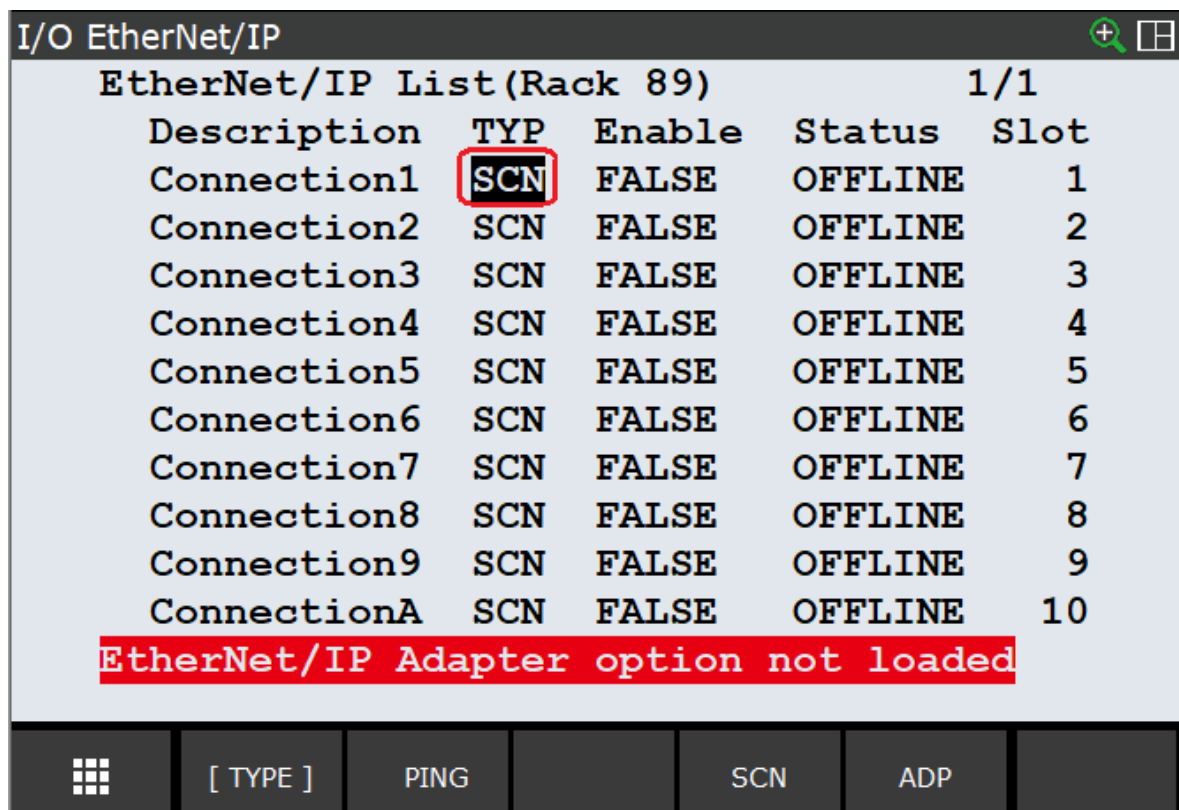
Fig. 4.2.6.1 (a) Ethrenet/IP Scanner with LITE1 (R889)

Since only R889 is loaded, only first SCN can be used and it can't not be changed to ADP as shown in Fig. 4.2.6.1 (b). If you try to change the TYP from SCN to ADP using F4, [ADP], it displays error in TP prompt line as shown below in Fig. 4.2.6.1 (b) i.e. EtherNet/IP Adapter option not loaded.

Fig. 4.2.6.1 (b) Ethernet/IP LITE SCN to ADP

4.2.6.2   Ethernet/IP scanner with LITE2 (R890)

If two SCNs are required, R890 need to be loaded on top of R889. If R889 and R890 both loaded, first two slots become available as scanners as shown in Fig. 4.2.6.2. Slot 2 can't be changed to ADP since adapter is not loaded.

Fig. 4.2.6.2 Ethernet/IP Scanner LITE2

4.2.7   Common Errors

If the connection is lost, the values of any mapped inputs will be zeroed out. The last state behavior can be changed by setting the following system variable : $EIP_CFG.$KEEP_IO_SCN. The values are :

- FALSE : The last state values of the adapter inputs will be zero (default)
- TRUE : The last state values of the adapter inputs will be their last value

This setting applies to all the scanner connections.

Any enabled scanner connections which are not RUNNING or PENDING will be retried each time the robot is RESET.

5   ETHERNET/IP TO DEVICENET ROUTING

5.1   OVERVIEW

EtherNet/IP and DeviceNet are both based on the Common and Industrial Protocol (CIP) which was initially defined by Rockwell with specifications managed by ODVA (www.odva.org). The robot can have connections to both Ethernet and DeviceNet networks and this feature provides the capability to route messages between two networks for configuration and diagnostics purposes.

This feature allows you to configure and manage the local robot DeviceNet network from personal computers (PCs) connected to their plant's Ethernet network. This eliminates someone having to connect a laptop PC to the physical robot in the DeviceNet network for certain third party device configuration and diagnostics functions. The PC software used is typically a tool such as RS-Networx for DeviceNet, which supports the following features:

- CIP routing
- Functions such as remotely configuring a device attached to the local DeviceNet network
- Network "who" of the local robot in the DeviceNet network from the PC connected to Ethernet.
- Explicit Messaging Connections to the devices in DeviceNet via "Class Instance Editor"

For more detailed technical information on EtherNet/IP to DeviceNet routing, refer to "Chapter 10 Bridging and Routing" in *The CIP Common specification (EtherNet/IP specification volume 1)*".

NOTE
The following option hardware is required:
- DeviceNet Interface board with SST DN3 or DN4 daugther board.

The following option software is required:
- EtherNet/IP Router (R-30*i*A/R-30*i*A Mate)   (R539)
- EtherNet/IP DN Router (R-30*i*B)   (R804)
- DeviceNet Interface (Master, Slave)   (J753)

5.2   GUIDELINES

Review the following guidelines before you use routing:
- Any errors returned from devices in DeviceNet are posted in the third party application software. (e.g. RSNetworx for DeviceNet)
- The G3_ONLY feature is supported on SST DN3 cards only
- Routing is limited to explicit messages directed to the connection manager object using the unconnected send service. Routing of I/O is not supported.
- Do not change the status of the devices while Routing is performed. This disrupts the connection between the master (robot in the DeviceNet network) and slave (device in DeviceNet network).

5.3 SETTING UP ETHERNET/IP TO DEVICENET ROUTING

EtherNet/IP to DeviceNet Routing is installed with the EthernetIP I/O RTR option. The default method for using Routing is to have it configured to start when the controller is turned on. Even though the EtherNet/IP to DeviceNet Routing interface screen is available (Refer to 13.3), some features can only be configured by setting system variables.

NOTE
    $EIP_RTR.$G3_ONLY is added to protect I/O performance. Group 2 devices need to set up Predefined Master/Slave connections to exchange Explicit Messaging packets and I/O packets. The priorities of Group 2 messages are predefined by ODVA. For example, Explicit Message request and response have higher priority than the Master's I/O poll request. Thus, there is a chance that an Explicit Message request and response will win over a Master's I/O poll request. This could affect I/O performance. When $EIP_RTR.$G3_ONLY is enabled, routing to Group 2 devices are not allowed.

$EIP_RTR.$DIN_NUM is added to enable/disable routing dynamically based on a digital input. For example, this feature can be useful for I/O sensitive processes such as dispensing around a windshield. When the corresponding DIN is ON, the routing is disabled. Refer to Table 5.3 for information on the system variables used in routing.

Table 5.3 EtherNet/IP to DeviceNet Routing System Variables

System Variable	Default Value	Description
$EIP_RTR.$ENABLE	TRUE	Enable EIP Router*
$EIP_RTR.$BOARD_1	FALSE	When this is enabled, DeviceNet Board 1 routes packets from EtherNet to DeviceNet.
$EIP_RTR.$BOARD_2	TRUE	When this is enabled, DeviceNet Board 2 routes packets from EtherNet to DeviceNet.
$EIP_RTR.$BOARD_3	FALSE	When this is enabled, DeviceNet Board 3 routes packets from EtherNet to DeviceNet.
$EIP_RTR.$BOARD_4	FALSE	When this is enabled, DeviceNet Board 4 routes packets from EtherNet to DeviceNet.
$EIP_RTR.$G3_ONLY	FALSE	When this is enabled, CIP packets are only routed to Group 3 only (UCMM capable) devices.
$EIP_RTR.$DIN_NUM	0	When DIN input port is specified and input port is ON, no packets are routed to DeviceNet network.*

* $ENABLE and $DIN_NUM can be set via user interface screen.

5.4 USING ETHERNET/IP TO DEVICENET ROUTING

After system variables are configured, enabling and disabling Router and Setting DIN[] can be done using the EtherNet/IP Configuration screen. Refer to Procedure 5-1 to set up EtherNet/IP to DeviceNet Routing. Refer to Procedure 5-2 for information on Routing using RSNetworx for DeviceNet.

Procedure 5-1 Setting Up EtherNet/IP to DeviceNet Routing

Conditions
- The controller is turned on.

Steps
1   Press the [MENU] key.
2   Select I/O.
3   Press F1, [TYPE].
4   Select EtherNet/IP. You will see a screen similar to the following.

```
I/O EtherNet/IP              JOINT 10 %
EtherNet/IP List(Rack 89)         1/8
 Description TYP Enable Status  Slot
1      Conn1 ADP  TRUE   ONLINE   1
2      Conn2 ADP  FALSE  OFFLINE  2
3      Conn3 ADP  FALSE  OFFLINE  3
4      Conn4 ADP  FALSE  OFFLINE  4
5      Conn5 ADP  FALSE  OFFLINE  5
6      Conn6 ADP  FALSE  OFFLINE  6
7      Conn7 ADP  FALSE  OFFLINE  7
8      Conn8 ADP  FALSE  OFFLINE  8
```

5   Press the [NEXT] key and then press F3, [RTR]. You will see a screen similar to the following.

```
I/O EtherNet/IP              JOINT 10 %
Router configuration :          1/2


    Enable : TRUE
    Disable when DIN[  0] is ON
```

6   Make sure Enable is set TRUE. If it is not, move the cursor to Enable, and press F4, [TRUE].
7   If you want to disable routing when a particular DIN is ON, move the cursor to DIN[ ], and type the port number. Note that DIN[ ] port 0 does not exist in the robot I/O. Thus, DIN[0] would not disable Routing unless the port number is changed.

> **NOTE**
> This feature is optional and can be useful for I/O sensitive processes such as dispensing around a windshield.

Procedure 5-2 Routing using RSNetworx for DeviceNet

Conditions

- $EIP_RTR.$BOARD_2 is set TRUE.
- Board 2 in DeviceNet is ONLINE.
- You are using a personal computer (PC) with Microsoft NT, and have installed RSNetworx for DeviceNet.

Steps

1   Launch RSNetworx for DeviceNet. For example on your PC, select Start, Programs, Rockwell software, RSNetworx, and then RSNetworx for DeviceNet. You will see a screen similar to the following.
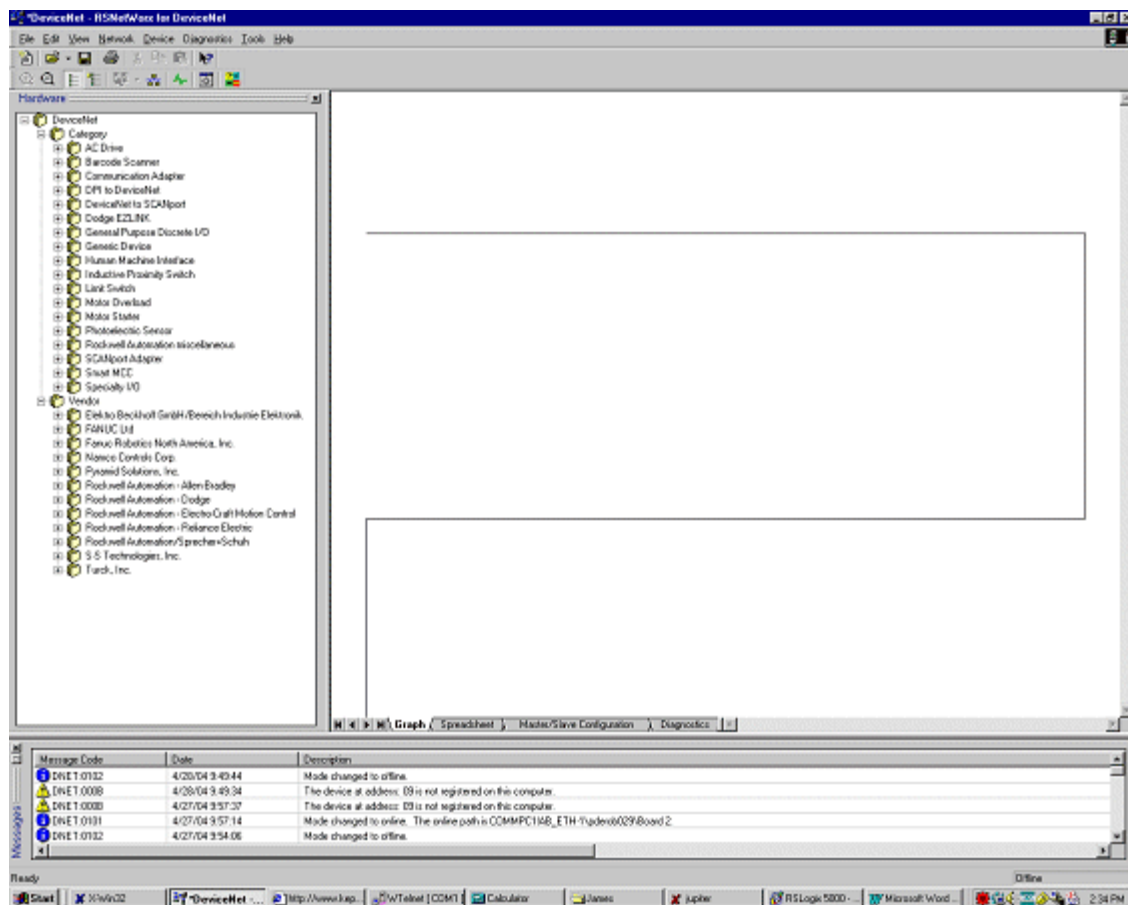

Fig. 5.4 (a) RSNetworx for DeviceNet First Screen

2   Select Properties under Network tab. You will see a screen similar to the following.
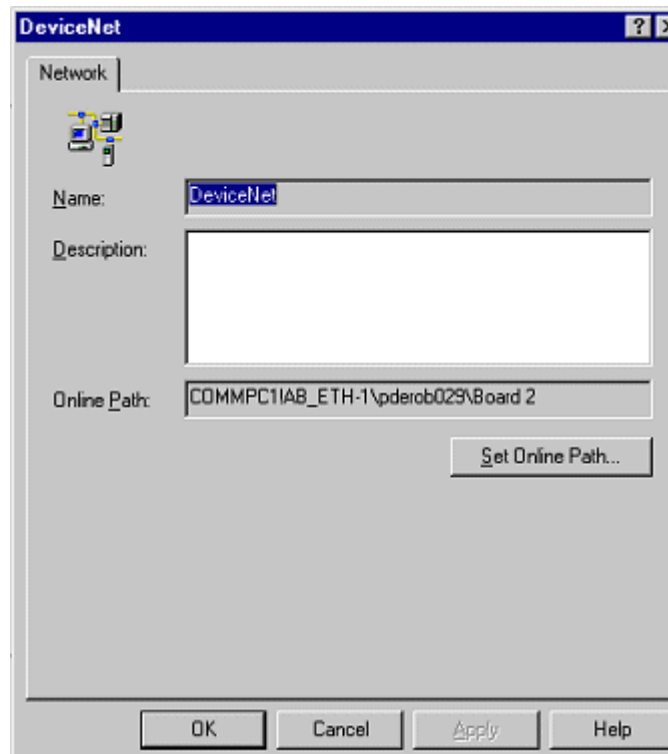

Fig. 5.4 (b) DeviceNet Network Screen

3   Click Set Online Path.. You will see a screen similar to the following.
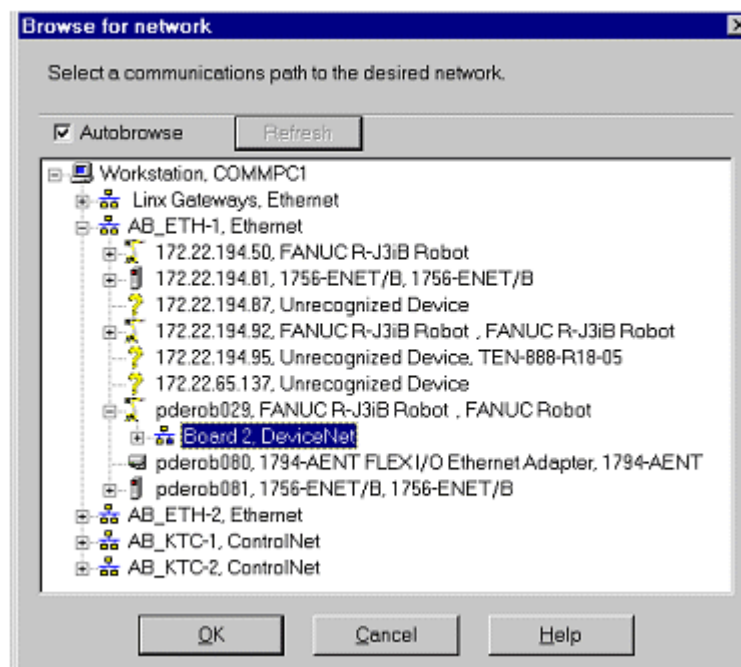

Fig. 5.4 (c) Set Online Path.. Screen

4   Find the robot under Ethernet channel. In this example, the robot named pderob029 is under AB-ETH1, EtherNet.

5   Select Board 2, DeviceNet under the robot (pderob029), and click OK. You should see new path in Online Path text box. This is shown as COMMPC1¥AB_ETH-1¥pderob029¥Board 2 in this example.

6   Click Apply, and then click OK. After the path is set, you are ready to browse the devices in DeviceNet.

7   Select Online under the Network tab.

8   Click OK to begin browsing the network. After this is done, you will see the screen similar to the following.
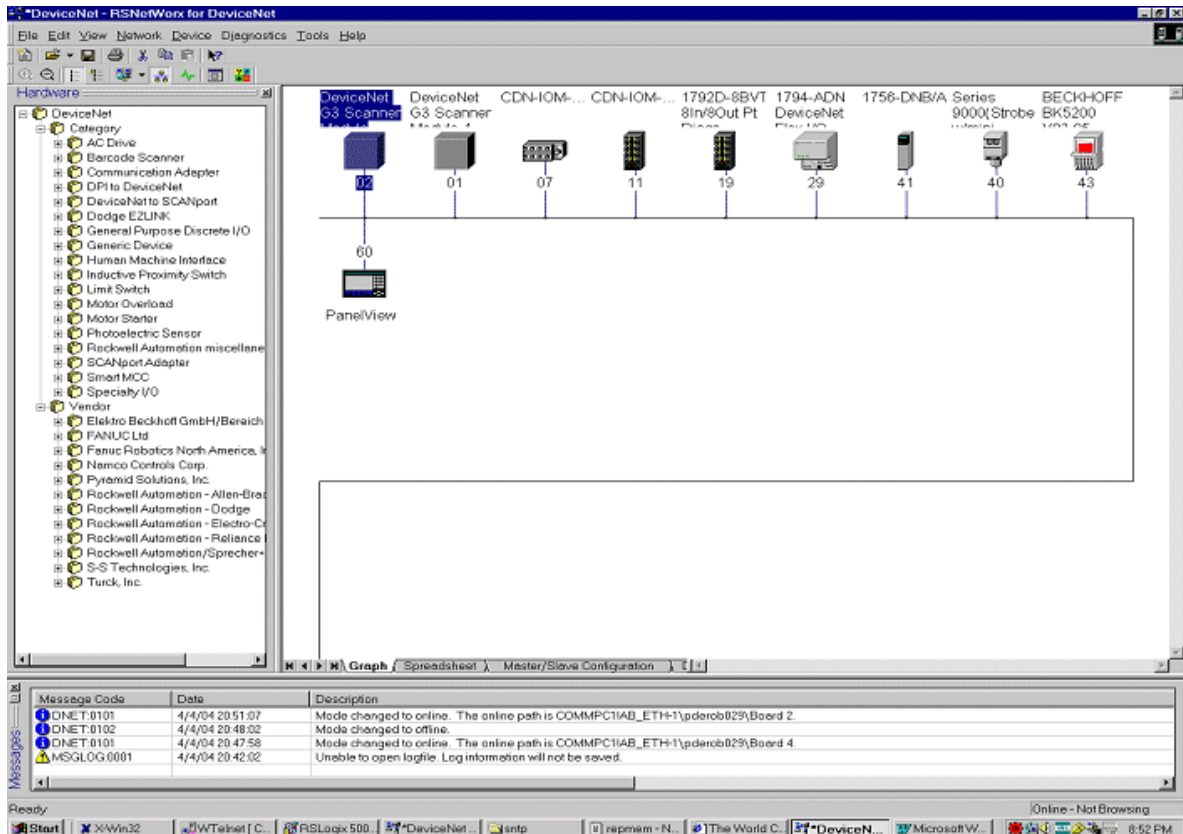

Fig. 5.4 (d) Local DeviceNet

9   At this point you can use the Class Instance Editor to get or set device parameters. For more information on how to use RSNetworx for DeviceNet, refer to the RSNetworx for DeviceNet Manual.

6   I/O CONFIGURATION

6.1   OVERVIEW

This chapter describes how to make EtherNet/IP I/O available within the robot by mapping it to digital, group, and UOP I/O points. Scanner connection I/O can also map to analog. Refer to Section 4.2.5.

This chapter also describes procedures for backing up and restoring the EtherNet/IP and I/O configurations.

6.1.1   I/O Size of Each Connection and I/O Configuration

Each connection (adapter or scanner) has input/output size setting, which determines the amount of I/O of each connection that can be assigned to I/O of the robot.
In some application software, the I/O is automatically configured when you turn on the controller.
The system variable $IO_AUTO_CFG (for digital I/O) and $IO_AUTO_UOP (for UOPI/O) controls this behavior.
If I/O is automatically configured, I/O configuration of each connection is made according to the input/output size of each connection.
If a connection was once configured and it is no longer used, please set its input/output size to 0 in addition to setting its ENABLE to FALSE, and removing its I/O assignment.
If input/output size is not 0, I/O assignment is made again by next power on.

6.2   MAPPING I/O ON THE ROBOT

The EtherNet/IP I/O can be mapped to digital, group, or UOP I/O points within the robot scanner connection. I/O can also map to analog. This is similar to mapping other I/O points on the robot where the rack, slot, and starting point number are used to map physical I/O to logical I/O within the I/O map.

All EtherNet/IP I/O uses rack 89. The slot number for each connection is shown in the EtherNet/IP Status screen.

Use Procedure 6-1 to map I/O on the robot.

Procedure 6-1 Mapping I/O on the Robot

Steps
1   Press the [MENU] key.
2   Select I/O.
3   Press F1, [TYPE], and select Digital, Group, UOP, or analog (analog is supported on scanner connections).
4   Press F2, [CONFIG].
5   Set the Range to the appropriate value. For analog, set the channel to the appropriate value.
6   Set the Rack to 89 and set the appropriate slot number and starting point as required.

> **NOTE**
> Refer to the Input/Output (I/O) Setup chapter in the application-specific Setup and Operations Manual for additional information on I/O configuration.

> **NOTE**
> See Procedure 4-3 for additional information on configuring Analog I/O on the controller.

In some application software the I/O is automatically configured when you turn on the controller. The system variable $IO_AUTO_CFG (for digital I/O) and $IO_AUTO_UOP (for UOP I/O) controls this behavior. If the system has already automatically configured the I/O, and sizes are changed, the I/O assignments can be cleared to force the system to remap all the I/O. This is done by clearing assignments (CLR_ASG). Use Procedure 6-2 to clear I/O assignments.

Procedure 6-2 Clearing I/O Assignments

Steps
1   Press the [MENU] key.
2   Select I/O.
3   Press F1, [TYPE].
4   Select Link Device.
5   Press F5, [CLR_ASG].
6   To remap all I/O, turn the controller off and back on.

> **NOTE**
> This clears all I/O assignments. The I/O will be remapped when you turn off then turn on the controller based on the settings of $IO_AUTO_CFG (for digital I/O) and $IO_AUTO_UOP (for UOP I/O).

6.3   BACKING UP AND RESTORING ETHERNET/IP AND I/O CONFIGURATION

There are two files which contain information on the configuration of EtherNet/IP and I/O mappings :
- DIOCFGSV.IO contains general I/O configuration and all I/O mappings (for example, mappings between EtherNet/IP and digital, group, and UOP I/O).
- SYSEIP.SV contains EtherNet/IP specific configuration including all adapter and scanner settings.

Use Procedure 6-3 to back up files manually.

Use Procedure 6-4 to do a full application backup, which includes DIOCFGSV.IO and SYSEIP.SV.

Procedure 6-3 Backing Up Files Manually

Steps
1   Select the default file device where files will be saved:
    a   Press the [MENU] key.
    b   Select File.
    c   Press F5, [UTIL], and choose SET DEVICE.
    d   Select the device to which you want to save the files.
2   Save DIOCFGSV.IO:
    a   Press the [MENU] key.
    b   Select I/O.
    c   Press F1, [TYPE], and choose DIGITAL.
    d   Press the [FCTN] key.
    e   Select Save to save DIOCFGSV.IO to the default device.

3   Save SYSEIP.SV:
    a   Press the [MENU] key.
    b   Select I/O.
    c   Press F1, [TYPE], and choose EtherNet/IP.
    d   Press the [FCTN] key.
    e   Select Save to save SYSEIP.SV to the default device.

Procedure 6-4 Performing a Full Application Backup

Steps
1   Select the default file device (where files will be saved):
    a   Press the [MENU] key.
    b   Select File.
    c   Press F5, [UTIL], and choose SET DEVICE.
    d   Select the device to which you want to save the files.
2   Press F4, [BACKUP], and choose "All of above".

7   EXPLICIT MESSAGING

7.1   OVERVIEW

The robot controller is an explicit message server and supports connected and unconnected explicit messaging. Up to six explicit message connections are supported. The EtherNet/IP Adapter option must be loaded to support this functionality. The following objects are supported by the controller:

- Identity Object (0x01)
- Message Router Object (0x02)
- Assembly Object (0x04)
- Connection Manager Object (0x06)
- Vendor Specific Register Object–Integers (0x6B)
- Vendor Specific Register Object–Reals (0x6C)
- Vendor Specific Register Object–Strings (0x6D)
- Vendor Specific Register Object–Position Registers: Cartesian (0x7B)
- Vendor Specific Register Object–Position Registers: Joint (0x7C)
- Vendor Specific Active Alarm Object (0xA0)
- Vendor Specific Alarm History Object (0xA1)
- Vendor Specific Motion Alarm Object (0xA2)
- Vendor Specific System Alarm Object (0xA3)
- Vendor Specific Application Alarm Object (0xA4)
- Vendor Specific Recovery Alarm Object (0xA5)
- Vendor Specific Communications Alarm Object (0xA6)
- Connection Configuration Object (0xF3)
- Port Object (0xF4)
- TCPIP Object (0xF5)
- Ethernet Link Object (0xF6)

This chapter does not go into the details of the standard CIP objects defined by ODVA. This chapter will instead document FANUC's Vendor Specific Alarm and Register objects, and the Assembly Object instances numbers for accessing I/O on the robot controller.

In general, no configuration is required on the robot controller to use explicit messaging. I/O must be configured on the robot if it is to be accessed through the Assembly Object.

NOTE
   Writing to a register is only supported by V7.20P11 or later.

NOTE
   Vendor Specific Register Object--Reals (0x6C) is only supported by V7.40 or later. Only Vendor Specific Register Object--Integers (0x6B) can be used before V7.40P.

NOTE
   Vendor Specific Register Object--Strings (0x6D) is only supported by V7.70P27 or later.

> **NOTE**
> Vendor Specific Register Object--Position Registers can be used in V8.10P or later.

7.2 ROBOT EXPLICIT MESSAGING CLIENT

7.2.1 Overview

> **NOTE**
> Robot Explicit Message Client is only supported by V7.40P or later.

The Robot Explicit Messaging Client allows you to send simple EtherNet/IP explicit messages to other EtherNet/IP enabled devices on the network. You can access this option from the main EtherNet/IP screen and execute queries from the teach pendant.

The explicit messaging feature implements the Get Attribute Single, and Set Attribute Single services.

Explicit messaging clients require up to five values for configuration. These values are usually described in hexadecimal notation, with the exception of the IP address. These values are shown in Table 7.2.1.

Table 7.2.1 Explicit Messaging Configuration Values

ITEM	DESCRIPTION
IP Address	The address of the remote device to be queried.
Class	The class of Object to which the explicit message is being sent.
Instance	The instance number defines which instance of the class will receive the message.
Attribute	Defines which attribute of the instance is being accessed.
Service	The action to be performed. The robot explicit messaging client only supports the Get Attribute Single, and Set Attribute Single services.

An explicit message configuration file can also be created that performs a batch of commands. (Refer to Procedure 7-3). Normally this is used with the Set Attributes Single service to configure a remote device.

Procedure 7-1 describes how to use the Get Attribute Single service. Procedure 7-2 describes how to use the Set Attribute Single service.

Procedure 7-1 Get Attribute Single Service

1   Press the [MENU] key.
2   Select I/O.
3   Press F1, [TYPE].
4   Select EtherNet I/P.
5   Press the [NEXT] key and then press F2, [EXP-MSG]. (If you scrolled to an active connection, that connection's IP address will be used as the default IP address). You will see a screen similar to the following.

```
I/O EtherNet/IP        JOINT 10 %
Explicit Message Query            1/8
      Input Mode:       Manual
      IP Addr:
      Class:            1
      Instance:         1
      Attribute:        1
      Service:          Get Att
      Value Size:       Byte(1)
      Value:            0
```

6   Select Input Mode, and choose Manual (the default).
7   Set the IP Addr field to the address of the remote device.
8   Set the Class, Instance, and Attribute fields. This information should either describe standard CIP objects, or be provided by the vendor of the remote device.
9   Select Service field and choose Get Att (the default).
10   Press F3, [EXEC] (you will have to cursor to a field other then Services and Value Size to see the F3, [EXEC] function). The robot will attempt to send the Explicit Message query to the device.
     A valid response or any errors will be displayed at the bottom of the screen.

Procedure 7-2 Set Attribute Single Service

1   Press the [MENU] key.
2   Select I/O.
3   Press F1, [TYPE].
4   Select EtherNet I/P.
5   Press the [NEXT] key and then press F2, [EXP-MSG]. (If you scrolled to an active connection, that connection's IP address will be used as the default IP address). You will see a screen similar to the following.

```
I/O EtherNet/IP        JOINT 10 %
Explicit Message Query            1/8
      Input Mode:       Manual
      IP Addr:
      Class:            1
      Instance:         1
      Attribute:        1
      Service:          Get Att
      Value Size:       Byte(1)
      Value:            0
```

6   Select Input Mode, and choose Manual (the default).
7   Set the IP Addr field to the address of the remote device.
8   Set the Class, Instance, and Attribute fields. This information should either describe standard CIP objects, or be provided by the vendor of the remote device.
9   Select Service field and choose Set Att.
10   Set the Value Size, and Value fields. The supported Value Size fields are Byte, Word, and Long.
11   Press F3, [EXEC] (you will have to cursor to a field other then Services and Value Size to see the F3, [EXEC] function). The robot will attempt to send the Explicit Message query to the device. The result of the query or any errors will be displayed at the bottom of the screen.

Procedure 7-3 Explicit Messaging Batch File Method

1   Press the [MENU] key.
2   Select I/O.
3   Press F1, [TYPE].
4   Select EtherNet I/P.
5   Press the [NEXT] key and then press F2, [EXP-MSG]. You will see a screen similar to the following.

```
I/O EtherNet/IP       JOINT 10 %
Explicit Message Query            1/8
Input Mode:           Manual
IP Addr:
Class:                1
Instance:             1
Attribute:            1
Service:              Get Att
Value Size:           Byte(1)
Value:                0
```

6   Move the cursor to Input Mode, and choose File. You will see a screen similar to the following.

```
I/O EtherNet/IP        JOINT 10 %
Explicit Message Query            1/2
Device: MC:¥
Input Mode:           File
Config File Name:     None
```

7   Move the cursor to the Config File Name field. Press F2, [DEVICE] to select the device where the explicit message config file is located.
8   Press F4, [CHOICE] to select the config file. (Only files with a .EM extension can be selected.)
9   Press F3, [EXEC] to execute the config file. The robot will attempt to send the Explicit Message query to the device. The result of the query or any errors will be displayed at the bottom of the screen.

7.2.2   Creating a Configuration File for the Batch File Method

The configuration file format is documented in this section. All files must be name with the .EM file extension (for example, MYCONFIG.EM). Lines beginning with '*' are comments. Comments and blank lines are ignored. See Batch File Example.

The following seven (7) lines MUST exist for each query followed by a space and corresponding value:

```
QUERY:
IPADDR:
CLASS:
INSTANCE:
ATTRIBUTE:
SIZE:
VALUE:
```

There can be multiple queries within a file. The Set Attribute Single service is assumed in all cases. Each Query begins with a query number, which is unique and generally sequential.

The Size field refers to the Data Size of the parameter (Value), the following three are supported.

- 1 (BYTE or 1 byte)
- 2 (WORD or 2 bytes)
- 4 (LONG or 4 bytes)

Batch File Example

```
* File Name: EMCFG.EM
* Author: Joe User
* Date: 03/15/2004
* File must be saved with an .EM extension
* Lines beginning with '*' are comments.
* Comments and blank lines are ignored.
* Following 7 lines MUST exist for each query.
* There can be multiple queries within a file.
* The ''SET ATT'' service is assumed in all cases.
* Each Query begins with a query number, which is
* unique and generally sequential.
* Size field refers to the Data Size of the
* parameter, the following three are supported
* 1 (BYTE or 1 byte), 2(WORD or 2 bytes), 4 (LONG or 4 bytes)
QUERY: 1
IPADDR: 172.22.200.147
CLASS: 15
INSTANCE: 2
ATTRIBUTE: 1
SIZE: 1
VALUE: 4
QUERY: 2
IPADDR: 172.22.200.147
CLASS: 15
INSTANCE: 3
ATTRIBUTE: 1
SIZE: 2
VALUE: 300
```

7.3 REMOTE EXPLICIT MESSAGING CLIENT CONFIGURATION

Explicit messaging clients require up to four values for configuration. These values are usually described in hexadecimal notation. These values are shown in Table 7.3.

Table 7.3 Configuration Values

ITEM	DESCRIPTION
Class	The class of Object to which the explicit message is being sent.
Instance	The instance number defines which instance of the class will receive the message.
Attribute	Defines which attribute of the instance is being accessed.
Service	The action to be performed.

These values are documented in this manual for all FANUC's Vendor Specific Objects.

Here is an example of configuring an Explicit Message in RSLogix5000. Note the Service Code, Class, Instance and Attribute fields.

Fig. 7.3 Message Configuration

7.4   VENDOR SPECIFIC REGISTER OBJECTS

Table 7.4 shows the brief description of FANUC register object model.

Table 7.4 FANUC Register Object Model

Register Type	Service	Class	Instance	Attribute
NUMREG (Int)	Get_Attr_Single	0x6B	1 (8 bits)	reg #
NUMREG (Int)	Get_Attr_All	0x6B	1 (8 bits)	N/A
NUMREG (Int)	Get_Attr_Block	0x6B	blk_size, 1 (16 bits)	start reg #
NUMREG (Int)	Set_Attr_Single	0x6B	1 (8 bits)	reg #
NUMREG (Int)	Set_Attr_All	0x6B	1 (8 bits)	N/A
NUMREG (Int)	Set_Attr_Block	0x6B	blk_size, 1 (16 bits)	start reg #
NUMREG (Real)	Get_Attr_Single	0x6C	1 (8 bits)	reg #
NUMREG (Real)	Get_Attr_All	0x6C	1 (8 bits)	N/A
NUMREG (Real)	Get_Attr_Block	0x6C	blk_size, 1 (16 bits)	start reg #
NUMREG (Real)	Set_Attr_Single	0x6C	1 (8 bits)	reg #
NUMREG (Real)	Set_Attr_All	0x6C	1 (8 bits)	N/A
NUMREG (Real)	Set_Attr_Block	0x6C	blk_size, 1 (16 bits)	start reg #
STRREG	Get_Attr_Single	0x6D	1 (8 bits)	reg #
STRREG	Get_Attr_All	0x6D	1 (8 bits)	N/A
STRREG	Get_Attr_Block	0x6D	blk_size, 1 (16 bits)	start reg #
STRREG	Set_Attr_Single	0x6D	1 (8 bits)	reg #
STRREG	Set_Attr_All	0x6D	1 (8 bits)	N/A
STRREG	Set_Attr_Block	0x6D	blk_size, 1 (16 bits)	start reg #
POSREG (CRT)	Get_Attr_Single	0x7B	group # (8 bits)	reg #
POSREG (CRT)	Get_Attr_All	0x7B	group # (8 bits)	N/A
POSREG (CRT)	Get_Attr_Block	0x7B	blk_size, group # (16 bits)	start reg #
POSREG (CRT)	Set_Attr_Single	0x7B	group # (8 bits)	reg #
POSREG (CRT)	Set_Attr_All	0x7B	group # (8 bits)	N/A
POSREG (CRT)	Set_Attr_Block	0x7B	blk_size, group # (16 bits)	start reg #
POSREG (JNT)	Get_Attr_Single	0x7C	group # (8 bits)	reg #
POSREG (JNT)	Get_Attr_All	0x7C	group # (8 bits)	N/A
POSREG (JNT)	Get_Attr_Block	0x7C	blk_size, group # (16 bits)	start reg #
POSREG (JNT)	Set_Attr_Single	0x7C	group # (8 bits)	reg #
POSREG (JNT)	Set_Attr_All	0x7C	group # (8 bits)	N/A
POSREG (JNT)	Set_Attr_Block	0x7C	blk_size, group # (16 bits)	start reg #
CURPOS (CRT)	Get_Attr_Single	0x7D	group # (8 bits)	1
CURJPOS (JNT)	Get_Attr_Single	0x7E	group # (8 bits)	1

7.4.1   Numeric Register Objects (0x6B and 0x6C)

Numeric registers can be read and written through FANUC's Registers Object. These registers on the robot controller can be one of two types: Integer type, or Real type. Normally, the type is transparent to the user. For example, if R[1] is set to a value of 49 (R[1] = 49), the numeric register will automatically be configured as an Integer type. However, if R[2] is set to a value of 1.61803 (R[2] = 1.61803), the numeric register will be automatically configured as a Real type.

Because of the format of the data transfer through explicit messaging, this automatic configuration cannot be done. Therefore, two Register Objects have been created: Register Object–Integers (0x6B), and Register Object–Reals (0x6C). To read or write a numeric register as an Integer value, the Register Object–Integers (0x6B) should be accessed. To read or write a numeric register as a Real value, the Register Object–Real (0x6C) should be accessed.

Note that when writing to the Register Object–Integers, the numeric register will be changed to the Integer type. Likewise, when writing to the Register Object–Reals, the numeric register will be changed to the Real type.

However, when reading a numeric register using the Register Object–Integers, if the numeric register is configured as a Real type, value is converted to the nearest integer. For example, if values are 5.4 and 5.5 then converted values are 5 and 6 respectively.

> **NOTE**
>   This conversion is supported by V8.20P17 or later, V8.30P31 or later, R-30*i*B Plus or later.

Likewise, when reading a numeric register using the Register Object–Real, if the numeric register is configured as an Integer type, returned value is converted in real. For example, if value is 4 then converted value is 4.0. Same rule applies when writing register back to robot controller.

> **NOTE**
>   FANUC's Register objects also allow reading and writing of the registers in blocks. The Get_Attribute_All service allows reading of up to the first 124 registers starting from first register. The Set_Attribute_All service allows writing of up to the first 115 registers starting from first register. The Get_Attribute_Block allows up to 124 registers for Reading and Set_Attribute_All allows writing up to 115 registers at a time starting from any register number.

7.4.1.1   Instance attributes

FANUC's Register Objects support a single instance: instance 1. Each attribute in the instance corresponds to a register. For example, attribute 1 corresponds to R[1] and attribute 5 corresponds to R[5]. Refer to Table 7.4.1.1.

Table 7.4.1.1 Instance Attributes

Attribute ID	Name	Data Type	Description of Attribute
1	R[1]	32-bit integer	Register 1
2	R[2]	32-bit integer	Register 2
...			
n-1	R[n-1]	32-bit integer	Register n-1
n*	R[n]	32-bit integer	Register n

*   Where n is the total number of registers on the controller.

7.4.1.2   Common services

FANUC's Register Objects provide the Common Services at the Instance level shown in Table 7.4.1.2 (a).
No Class level services are provided.

Table 7.4.1.2 (a) Common Services

Service Code	Service Name	Description of Service
0E hex	Get_Attribute_Single	Returns the content of the specified attribute.
01 hex	Get_Attribute_All	Returns a listing of the object's attributes (See the Get_Attribute_All definition below).
32 hex	Get_Attribute_Block	Returns specified block of registers values
10 hex	Set_Attribute_Single	Sets the specified attribute to the specified value.
02 hex	Set_Attribute_All	Sets all attributes starting from Attribute 1 to 115 or MAX registers on controller whichever is smaller.
33 hex	Set_Attribute_Block	Sets values to the specified block of registers up to 115 or MAX registesr on conttroler whichever is smaller.

At the Instance level, the attributes are returned in the order shown in Table 7.4.1.2 (b) using little-endian
byte-swapping.

Table 7.4.1.2 (b) Get_Attribute_All Response

32–bit integer	Byte 0	Byte 1	Byte 2	Byte 3
1	Register 1 (R[1])			
2	Register 2 (R[2])			
...				
n-1	Register n-1 (R[n-1])			
n*	Register n (R[n])			

*   Where n is the total number of registers on the controller or 124, whichever is smaller.

7.4.1.3   Errors

FANUC's Vendor Specific Register Objects will return the errors shown in Table 7.4.1.3.

Table 7.4.1.3 FANUC's Vendor Specific Register Object Errors

Error Status	Error Description
Undefined Attribute (0x14)	Returned when the Register requested does not exist.
Unsupported Service (0x08)	Returned when the requested service is unsupported.
Undefined Class Instance (0x05)	Returned when the requested instance number is unsupported.

7.4.1.4   Read single register

The examples below assume the numeric register(s) have type integer, and use object 0x6B. The same examples can be used for numeric registers of type Real by using Class 0x6C, instead of Class 0x6B.

To read R[5] from the controller, assuming R[5]'s type is an integer, the explicit message client would be configured with the values shown in Fig. 7.4.1.4.

Fig. 7.4.1.4 shows message block configuration in RSLogix5000 where my_reg is a 32–bits integer variable. The explicit message server on the robot controller would return R[5] to the client as a 32-bit integer. Refer to Table 7.4.1.2 (a) for Service Code. Refer to Table 7.4 for Class, Instance and Attribute.

Fig. 7.4.1.4 Read Register R[5]

7.4.1.5   Read all registers

To read up to the first 124 Registers from the controller, and assuming all these registers are configured as type integer, the explicit message client would be configured with the values shown in Fig. 7.4.1.5.
Fig. 7.4.1.5 is snapshot of RSLogix500 message block configuration to read all registers where my_regall is an array of 124 integers. The explicit message server on the robot controller would return up to the first 124 Registers as an array of integers as described in Section 7.4.1.2. Refer to Table 7.4.1.2 (a) for Service Code. Refer to Table 7.4 for Class, Instance and Attribute.

Fig. 7.4.1.5 Read All Registers

7.4.1.6   Read a block of registers

Register objects also provide the functionality to read or write a block of registers. Common services used for read or write a block are Get_Attribute_Block (0x32). In order to use this functionality Instance and attributes are used as follows.

Instance is 2 byte. It is divided into two pieces. First (lower) 8 bits carry instance number which could be maximum 255 (for now it is just place holder because FANUC's register supports only single class instance). Second (higher) 8 bits carry the number of registers to be read so maximum 255 but FANUC's register object allows 124 for reading at time. Please refer Table 7.4.1.6 for instance number calculation. For example, to read 10 registers starting from register number 6 in integer format see Fig. 7.4.1.6 for configuration details. Messaging server would return values of registers 6 to 15. Refer to Table 7.4.1.2 (a) for Service Code. Refer to Table 7.4 for Class, Instance and Attribute.

Table 7.4.1.6   Instance Numbers for Block Access

Block Size Decimal (HEX)	Instance (HEX)	Instance (Decimal)
1 (0x01)	0x0101	257
2 (0x2)	0x0201	513
3 (0x3)	0x0301	769
4 (0x4)	0x0401	1025
5 (0x5)	0x0501	1281

Fig. 7.4.1.6 Read 10 Registers from R[6]-R[15]

7.4.1.7   Write single register

To write the integer value 49 to R[5], the explicit message client would be configured with the values shown in Fig. 7.4.1.7 .

Fig. 7.4.1.7 is an snapshot of RSLogix5000 message block configuration. my_reg is 32–bits integer which has value 49. The explicit message server on the robot controller would write the value 49 to R[5] as a 32-bit integer. Refer to Table 7.4.1.2 (a) for Service Code. Refer to Table 7.4 for Class, Instance and Attribute.

Fig. 7.4.1.7 Write Value to R[5]

7.4.1.8   Write all registers

Similarly, all register can be written at once. Only 115 or total number of registers on controller (whichever is less) can be written, configuration parameters shown in Fig. 7.4.1.8. Explicit message client (e.g. PLC) would carry an array of 115 integers or real (460 bytes total) along with this configuration.

Fig. 7.4.1.8 is taken from RSLogix500 to write all registers to controllers. Source element is my_regall[115] which is an array of 115 DINT Source length is data size to be written. Service code is 0x02, class 0x6B for Integer, Instance is 1 and attribute has to be zero. Refer to Table 7.4.1.2 (a) for Service Code. Refer to Table 7.4 for Class, Instance and Attribute.

Fig. 7.4.1.8 Write All Registers

7.4.1.9   Write a block of registers

Similarly, a block of registers can be written to robot using this functionality. Service used for function is Set_Attribute_Block (0x33). Maximum 115 or maximum available registers in controller whichever is less can be written at a time. For example, to write 5 registers (integer) starting from 11 (0xB) following configuration is needed, see Fig. 7.4.1.9.

Please refer Fig. 7.4.1.9 for message block configuration to read or write registers in RSLogix5000. Refer to Table 7.4.1.2 (a) for Service Code. Refer to Table 7.4 for Class, Instance and Attribute.

Fig. 7.4.1.9 Writer Registers from R[11]-R[15]

7.4.2 String Register Object (0x6D)

String register object will provide similar functionality as numeric register objects. It provides the ability to read string data as well as write string data via Ethernet/IP explicit messaging.

7.4.2.1 Instance attributes

FANUC's string register object provides single class instance i.e. 1. Instance attributes supported are maximum string registers supported on controller. String register is of STRING or STRING2 or STRINGN type. The declaration of a variable of type STRING, STRING2, or STRINGN is equivalent to declaring a structured data type for the variable which allocates a UINT variable (first 4 bytes) containing the current size of the string in characters and an array of declared character size elements.

String register object is composed of first 4 bytes string length, 82 bytes long string and 2 bytes padding which totals to 88 bytes in single string register, refer Table 7.4.2.1 (a) and Fig. 7.4.2.1. So on reading single register, it returns 88 bytes of data which contains maximum 82 byte long string. This is done to be compatible with RockWell PLC string structure (RSLogix5000) as shown in Fig. 7.4.2.1.

Table 7.4.2.1 (a) String Format

0–3 Byte	4-85 Bytes	86-87 bytes
String Length	String Register n (SR[n])	Padding

Fig. 7.4.2.1 String Structure RSLogix5000

Please note that size of string shown in Fig. 7.4.2.1 also includes 2 byte padding.

Table 7.4.2.1 (b) Instance Attributes

Attribute ID	Name	Data Type	Description of Attribute
1	SR[1]	Structure containing unsigned integer and char array of 82 bytes and last 2 bytes is padding Array	String Register 1
2	SR[2]	Structure containing unsigned integer and char array of 82 bytes and last 2 bytes is padding Array	String Register 2
...			
n-1	SR[n-1]	Structure containing unsigned integer and char array of 82 bytes and last 2 bytes is padding Array	String Register n-1
n*	SR[n]	Structure containing unsigned integer and char array of 82 bytes and last 2 bytes is padding Array	String Register n

*   Where n is the total number of registers on the controller.

7.4.2.2   Common services

FANUC's Register Objects provide the Common Services at the Instance level shown in Table 7.4.2.2.
No Class level services are provided.

Table 7.4.2.2 Common Services

Service Code	Service Name	Description
0E hex	Get_Attribute_Single	Returns the content of the specified attribute i.e. 88 byte char array.
01 hex	Get_Attribute_All	Returns an array of 88 bytes char arrays. Maximum register returned are 5 or string registers supported on controller whichever is smaller.
32 hex	Get_Attribute_Block	Returns block of string registers starting from any register up to next 5 registers or maximum string registers supported on controller whichever is smaller.
10 hex	Set_Attribute_Single	Sets the specified attribute to the specified value.
02 hex	Set_Attribute_All	Sets all attributes starting from Attribute 1 to 5 or maximum string registers supported on controller whichever is smaller.
33 hex	Set_Attribute_Block	Sets values to block of string registers starting from any register up to next 5 registers or maximum string registers supported on controller whichever is smaller.

7.4.2.3   Errors

Refer Section 7.4.1.3

7.4.2.4   Read single register

Fig. 7.4.2.4 shows message block configuration to read single string registers. Example screen shots are taken from RSLogix5000. To read string register 8, a string type variable"my_string_in" needs to be created to read the string. Please note that Instance value in Fig. 7.4.2.4 is decimal number and all others are hexadecimal numbers. Refer to Table 7.4.2.2 for Service Code. Refer to Table 7.4 for Class, Instance and Attribute.
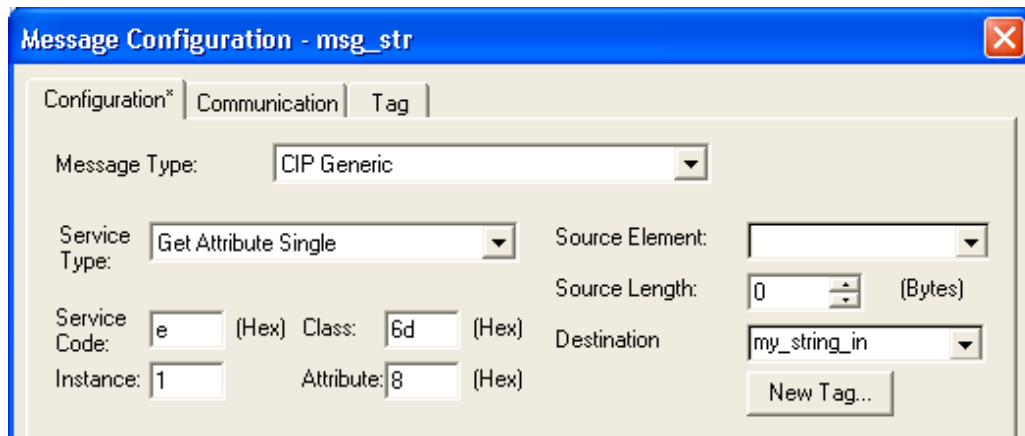
Fig. 7.4.2.4 Read String Register SR[8]

7.4.2.5   Read all register

Reading all registers is also supported for string registers. Reading all register allows to read first 5 STRING registers. This limitation is posed due to maintain compatibility with RSLogix5000. Fig. 7.4.2.5 shows the message block configuration to read all string registers where 'mystr' is an array of 5 STRING variables. Refer to Table 7.4.2.2 for Service Code. Refer to Table 7.4 for Class, Instance and Attribute.
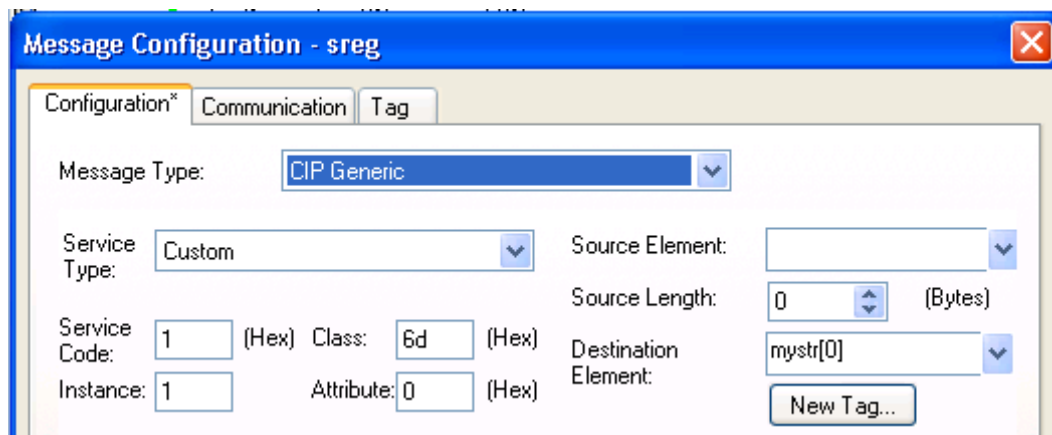
Fig. 7.4.2.5 Read All Register

7.4.2.6   Read a block of register

String registers also can be read in blocks. The block size limitation is 5 registers at a time. The advantage over read all register is that it allows to read from any register index to next 5 registers.
Please refer Fig. 7.4.2.6 for configuration details to read 5 registers starting from register number 5. Refer to Table 7.4.2.2 for Service Code. Refer to Table 7.4 for Class, Instance and Attribute.

> **NOTE**
> Please note that the data returned/consumed as a result of block read/write operation is always multiple of 88 which is 88 times block size. In case of read/write all data size is 440 bytes, if total string registers available on controller are 5 or more.
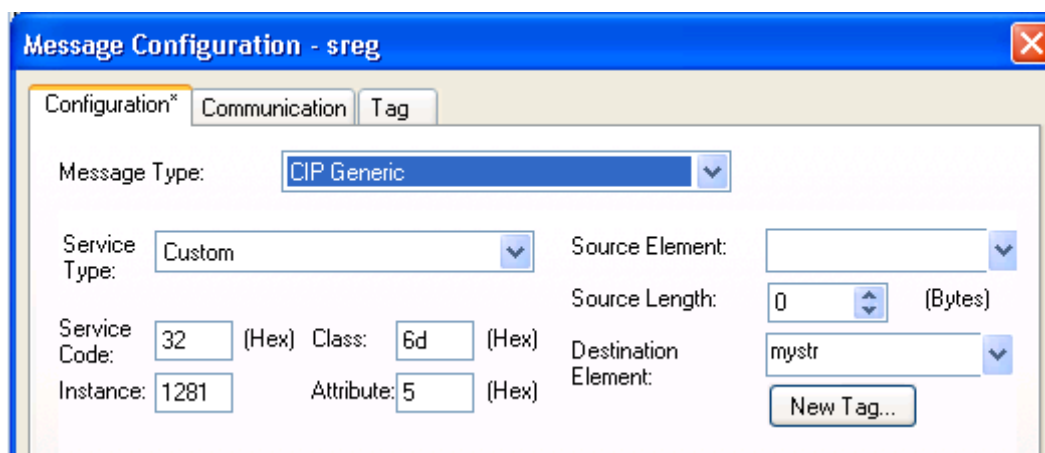
Fig. 7.4.2.6 Read a Block of Register

7.4.2.7   Write single register

For writing string to any specified string register, a variable of STRING (RSLogix5000) type or a structure as described in Fig. 7.4.2.7 needs to be created which carries string data. Refer to Table 7.4.2.2 for Service Code. Refer to Table 7.4 for Class, Instance and Attribute.
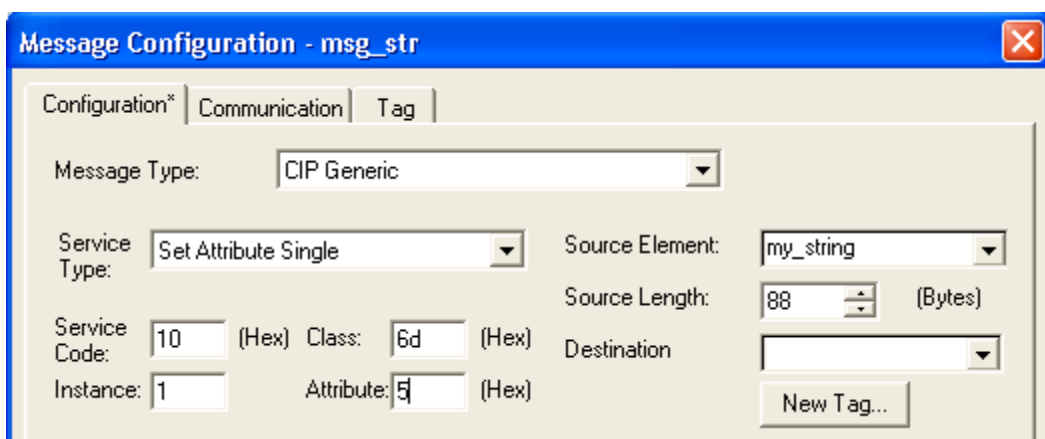
Fig. 7.4.2.7 Write String to SR[5]

7.4.2.8   Write all registers

This service allows to write strings to first 5 string registers from string register 1. Each string data could be 82 bytes long. Fig. 7.4.2.8 shows the message block configuration where 'mystr' is an array of 5 STRING data type.
Please refer Fig. 7.4.2.1 for STRING data type definition. Refer to Table 7.4.2.2 for Service Code. Refer to Table 7.4 for Class, Instance and Attribute.

Fig. 7.4.2.8 Write All Registers

7.4.2.9   Write a block of registers

This service is similar to write all except it allows to write string data starting from any registers to next 5 registers. Please refer Fig. 7.4.2.9 for message configuration details. Refer to Table 7.4.2.2 for Service Code. Refer to Table 7.4 for Class, Instance and Attribute.

Fig. 7.4.2.9 Write a Block of Registers

7.4.3   Position Register Object (0x7B, 0x7C, 0x7D, 0x7E)

Position register object provides similar functionality as numeric register objects. It provides the ability to read position data as well as writing the position data. This also has flexibility to read/write data in two representations Cartesian and Joint.

7.4.3.1   Instance attributes

FANUC's position register object provides single class instance i.e. 1. Instance attributes supported are maximum position registers supported in controller. Position register is a structure shown in Fig. 7.4.3.1 (a), Fig. 7.4.3.1 (b), Fig. 7.4.3.1 (c), and Fig. 7.4.3.1 (d) for joint and cartesian representations respectively. Joint representation can be accommodated in a structure consisting a DINT and 9 REAL numbers which supports up to 9 axes. If robot axes are less than 9 then remaining axes returned are uninitialized. The explicit messaging server returns 40 bytes data for joint position representation on reading and consumes 40 bytes position data on writing. On the other hand, cartesian position representation is 44 byte. The explicit messaging server returns 44 bytes of data on reading and consumes 44 bytes of data on writing.

Table 7.4.3.1 (a) R-30*i*A/R-30*i*B Structure Definition for Joint Position Representation

Member Names	Data Size	Data Type	Description
Structure Name: PRJNT9 (User defined)			
UT	1 Byte	Decimal	User Tool Number
UF	1 Byte	Decimal	User Frame Number
Dummy	2 Bytes	Decimal	Reserved
JNT_ANGLE[9]	36 Bytes	Float	Robot Axes: An array of 9 Real

Table 7.4.3.1 (b) R-30*i*A/R-30*i*B Instance Number Calculation

Block Size Decimal (HEX)	Instance (HEX) Group #1	Instance (Decimal) Group #1	Instance (HEX) Group #2	Instance (Decimal) Group #2
1 (0x01)	0x0101	257	0x0102	258
2 (0x02)	0x0201	513	0x0202	514
3 (0x03)	0x0301	769	0x0302	770
4 (0x04)	0x0401	1025	0x0402	1026
5 (0x05)	0x0501	1281	0x0502	1282

Table 7.4.3.1 (c) R-30*i*A/R-30*i*B Structure Definition for Cartesian Position Representation

Member Names	Data Size	Data Type	Description
Structure Name: POGREG_T (user defined)			
UT	1 Byte	Decimal	User Tool Number
UF	1 Byte	Decimal	User Frame Number
Dummy	2 Bytes	Decimal	Reserved
X	4 Bytes	Float	X mm
Y	4 Bytes	Float	Y mm
Z	4 Bytes	Float	Z mm
W	4 Bytes	Float	W Degree
P	4 Bytes	Float	P Degree
R	4 Bytes	Float	R Degree
Turn4	1 Byte	Decimal	Turn4
Turn5	1 Byte	Decimal	Turn5
Turn6	1 Byte	Decimal	Turn6
Reserved1–4	4–Bits	Bool	Reserved
Front	1 Bit	Bool	Front
Up	1 Bit	Bool	Up
Left	1 Bit	Bool	Left
Flip	1 Bit	Bool	flip
EXT[3]	12 Bytes	Float	Extended Axes

Fig. 7.4.3.1 (a) R-30*i*A/R-30*i*B Position Register Joint Mode

Fig. 7.4.3.1 (b) R-30*i*A/R-30*i*B Position Register Cartesian Mode

Table 7.4.3.1 (d) R-30*i*B Plus Structure Definition for Joint Position Representation

Structure Name: POSREGJ_T (User defined)			
Member Names	**Data Size**	**Data Type**	**Description**
UFRAME	2 Bytes	Decimal	User Frame Number
UTOOL	2 Bytes	Decimal	User Tool Number
JNT[9]	36 Bytes	Float	Robot Axes: An array of 9 Real

Table 7.4.3.1 (e) R-30*i*B Plus Instance Number Calculation

Block Size Decimal (HEX)	Instance (HEX) Group #1	Instance (Decimal) Group #1	Instance (HEX) Group #2	Instance (Decimal) Group #2
1 (0x01)	0x0101	257	0x0102	258
2 (0x02)	0x0201	513	0x0202	514
3 (0x03)	0x0301	769	0x0302	770
4 (0x04)	0x0401	1025	0x0402	1026
5 (0x05)	0x0501	1281	0x0502	1282

Table 7.4.3.1 (f) R-30*i*B Plus Structure Definition for Cartesian Position Representation

Structure Name: POGREG_T (user defined)			
Member Names	**Data Size**	**Data Type**	**Description**
UFRAME	2 Byte	Decimal	User Frame Number
UTOOL	2 Byte	Decimal	User Tool Number
X	4 Bytes	Float	X mm
Y	4 Bytes	Float	Y mm
Z	4 Bytes	Float	Z mm
W	4 Bytes	Float	W Degree
P	4 Bytes	Float	P Degree
R	4 Bytes	Float	R Degree
Turn1	1 Byte	Decimal	Turn1
Turn2	1 Byte	Decimal	Turn2
Turn3	1 Byte	Decimal	Turn3
Reserved1–4	4–Bits	Bool	Reserved
Front	1 Bit	Bool	Front
Up	1 Bit	Bool	Up
Left	1 Bit	Bool	Left
Flip	1 Bit	Bool	flip
EXT[3]	12 Bytes	Float	Extended Axes

Name:	POSREGJ_T				
Description:	FANUC Position Register (Joint Mode)				

Members: Data Type Size: 40 byte(s)

	Name	Data Type	Style	Description	External Access
	uframe	INT	Decimal		Read/Write
	utool	INT	Decimal		Read/Write
	joint	REAL[9]	Float		Read/Write

Fig. 7.4.3.1 (c) R-30*i*B Plus Position Register Joint Mode

Fig. 7.4.3.1 (d) R-30*i*B Plus Position Register Cartesian Mode

7.4.3.2   Common services

FANUC's Register Objects provide the Common Services at the Instance level shown in Table 7.4.3.2. No Class level services are provided.

Table 7.4.3.2 Common Services

Service Code	Service Name	Description
0E hex	Get_Attribute_Single	Returns the content of the specified attribute i.e. 40 or 44 bytes for Joint or Cartesian representation respectively.
01 hex	Get_Attribute_All	Returns an array of position registers. Maximum register returned are 10 or position registers supported on controller whichever is smaller.
32 hex	Get_Attribute_Block	Returns block of position registers starting from any register up to next 10 registers or maximum position registers supported on controller whichever is smaller.
10 hex	Set_Attribute_Single	Sets the specified attribute to the specified value.
02 hex	Set_Attribute_All	Sets all attribute starting from Attribute 1 to 10 or maximum position registers supported on controller whichever is smaller.
33 hex	Set_Attribute_Block	Sets values to block of position registers starting from any register up to next 10 registers or maximum position registers supported on controller whichever is smaller.

> **NOTE**
> When reading or writing of position registers in any representation i.e. joint or cartesian, the position will be converted so that the position representation currently used in the position register does not change.

7.4.3.3   Errors

Refer Section 7.4.1.3 for EIP related errors and for other errors refer to error code manual for complete set of error details.

7.4.3.4   Read single register

Reading position register is similar to numeric or string registers except instance number represents the group number of the robot having multiple groups so instance number is always nonzero positive integer. This object allows to read registers from multiple groups of the robot. Fig. 7.4.3.4(a) shows the configuration details to read position register 3 in joint representation. Refer to Table 7.4.3.2 for Service Code. Refer to Table 7.4 for Class, Instance and Attribute.

Fig. 7.4.3.4 (a) Read Single Register in Joint Mode

Similarly Fig. 7.4.3.4 (b) shows the configuration for read position register 8 in cartesian mode. Please remember that Joint representation is a 40 bytes and Cartesian representation is 44 byte data.

Fig. 7.4.3.4 (b) Read Single Register in Cartesian Mode

7.4.3.5   Read all registers

Reading all registers is also supported for position registers. Reading procedure is similar to Numeric or String Registers except class and data type. Reading all position registers is limited to maximum of 10 or number of registers on controller whichever is SMALLER starting from register 1. Please refer Fig. 7.4.3.5 for configuration details. prrd is an array of structures PRCRT a user define data type shown in Fig. 7.4.3.1 (b) and Fig. 7.4.3.1 (d) for cartesian representation. Refer to Table 7.4.3.2 for Service Code. Refer to Table 7.4 for Class, Instance and Attribute.

Fig. 7.4.3.5 Read All Registers

7.4.3.6   Read a block of registers

Position registers also can read or written in blocks. The block size limitation is 10 registers at a time. This functionality provides the flexibility to set start register index of the block of registers i.e. read 23 to 32 or 50 to 59, etc.

> **NOTE**
> Please note that the data returned/consumed as a result of block read/write operation is always multiple of 40 (joint) or 44 (cartesian) which is 40 or 44 times block size for joint and cartesian respectively. In case of read/write all data size is 400 (joint) or 440 (cartesian) bytes, if total position registers available on controller are 10 or more.

For example, message block configuration is shown in Fig. 7.4.3.6 for reading 8 position registers of group 1 starting from register number 10 in cartesian mode. Instance 2049 is decimal equivalent of 0x0801 and 'prrd' is an array of structure defined in Fig. 7.4.3.1 (b) and Fig. 7.4.3.1 (d) to hold position data in cartesian mode. Refer to Table 7.4.3.2 for Service Code. Refer to Table 7.4 for Class, Instance and Attribute.

Fig. 7.4.3.6 Read a Block of Registers

7.4.3.7   Read current position (CURPOS or CURJPOS)

This functionality also allows to read current position of robot in two modes: joint and cartesian. Configuration is same as reading a single position register with index 1 except class 0x7D is used for cartesian and 0x7E is used for joint. Please refer Fig. 7.4.3.7 (a) for cartesian mode and Fig. 7.4.3.7 (b) for joint mode in RSLogix5000 message block configuration. Refer to Table 7.4.3.2 for Service Code. Refer to Table 7.4 for Class, Instance and Attribute.

Fig. 7.4.3.7 (a) Read CURPOS

Fig. 7.4.3.7 (b) Read CURJPOS

7.4.3.8   Write single register

This service provides the flexibility to write position data to position registers in two modes cartesian and joint. Fig. 7.4.3.8 shows the configuration parameters. Here prwr is a user defined data structure as shown in Fig. 7.4.3.1 (a) and Fig. 7.4.3.1 (c) to hold the position data for joint representation. Refer to Table 7.4.3.2 for Service Code. Refer to Table 7.4 for Class, Instance and Attribute.

Fig. 7.4.3.8 Write Single Register in Joint Mode

> **NOTE**
> Please note that UT/UF values MUST be 255 or 0 which indicates current UT/UF in the robot. The UT/UF values cannot be changed because the current UT/UF values are stored in the position registers.

- 73 -

7.4.3.9   Write all registers

This service allows to write position data to all position registers limited to maximum of 10 from register 1. Position data for each register is 44 or 40 bytes long for cartesian and joint representation respectively. Fig. 7.4.3.9 shows the message block configuration where 'prwr' is an array of 10 data structures to hold cartesian position data. Please refer Fig. 7.4.3.1 (b) and Fig. 7.4.3.1 (d) for cartesian position data structure. Refer to Table 7.4.3.2 for Service Code. Refer to Table 7.4 for Class, Instance and Attribute.

Fig. 7.4.3.9 Write All Registers

7.4.3.10   Write a block of registers

Fig. 7.4.3.10 (a) is message block configuration for writing 8 position registers of group 1 starting from register index 32 (0x20) in cartesian mode where "prwr" is an array of structure defined in Fig. 7.4.3.1 (b) and Fig. 7.4.3.1 (d) to carry position data. Refer to Table 7.4.3.2 for Service Code. Refer to Table 7.4 for Class, Instance and Attribute.

Fig. 7.4.3.10 (a) Write a Block of Registers

Similarly a block of registers can be written to group #2 of robot. Fig. 7.4.3.10 (b) shows the message block configuration for writing 5 position registers of group #2 starting from register index 5 in joint mode. Instance 1282 is decimal equivalent of 0x0502, see Table 7.4.3.1 (b) and Table 7.4.3.1 (e). Variable "prwr" is an array of structure defined in Fig. 7.4.3.1 (a) and Fig. 7.4.3.1 (c).

Fig. 7.4.3.10 (b) Write a Block of Registers to Group #2

7.5 VENDOR SPECIFIC ACTIVE ALARM OBJECT (0xA0)

Information about Active Alarms can be read through FANUC's Active Alarm Object. Each instance of the object corresponds to an active alarm. For example, instance 1 corresponds to the most recent Active Alarm, and instance 5 corresponds to the 5th most recent Active Alarm.

7.5.1 Instance Attributes

Table 7.5.1(a) R-30iA/R-30iB Instance Attributes

Attribute ID	Name	Data Type	Description of Attribute
1	Alarm ID	16-bit integer	The Alarm ID, or Alarm Code.
2	Alarm Number	16-bit integer	The Alarm Number
3	Alarm ID Cause Code	16-bit integer	The Cause Code of the Alarm ID.
4	Alarm Num Cause Code	16-bit integer	The Cause Code of the Alarm Number.
5	Alarm Severity	16-bit integer	The Alarm Severity.
6	Time Stamp	32-bit integer	The Alarm Time Stamp in 32-bit MS-DOS format.
7	Date/Time String	24 characters	The Alarm Time Stamp in a human readable string.
8	Alarm Message	80 characters	The Alarm Message in a human readable string.
9	Cause Code Message	80 characters	The Alarm Cause Code Message in a human readable string.
10	Alarm Severity String	24 characters	The Alarm Severity in a human readable string.

Table 7.5.1(b) R-30iB Plus Instance Attributes

Attribute ID	Name	Data Type	Description of Attribute
1	Alarm ID	16-bit integer	The Alarm ID, or Alarm Code.
2	Alarm Number	16-bit integer	The Alarm Number
3	Alarm ID Cause Code	16-bit integer	The Cause Code of the Alarm ID.
4	Alarm Num Cause Code	16-bit integer	The Cause Code of the Alarm Number.
5	Alarm Severity	16-bit integer	The Alarm Severity.
6	Time Stamp	32-bit integer	The Alarm Time Stamp in 32-bit MS-DOS format.
7	Date/Time String	28 characters long: first 4 bytes length and rest string	The Alarm Time Stamp in a human readable string.
8	Alarm Message	88 characters long: first 4 bytes length and rest string	The Alarm Message in a human readable string.
9	Cause Code Message	88 characters long: first 4 bytes length and rest string	The Alarm Cause Code Message in a human readable string.
10	Alarm Severity String	28 characters long: first 4 bytes length and rest string	The Alarm Severity in a human readable string.

> **NOTE**
> If you need alarm string output in legacy style (R-30iB and before), please set EIP_CFGOPTIONS = 0x800 and cycle power. Now you will see alarm outputs in standard C string which means no length in first 4 bytes. Above support has been added to directly use RockWell STRING data type in RSLogix5000 or equivalent. IN legacy, string out for attribute ID 7–10 will be 4–8 bytes short i.e. 24, 80, 80, 24 respectively.

7.5.2   Common Services

FANUC's Active Alarm Object provides the following Common Services at the Instance level. No Class level services are provided. Refer to Table 7.5.2.

Table 7.5.2 Common Services

Service Code	Service Name	Description of Service
0E hex	Get_Attribute_Single	Returns the content of the specified attribute.
01 hex	Get_Attribute_All	Returns a listing of the object's attributes (See the Get_Attribute_All definition below).

7.5.2.1   Get_Attribute_All response

At the Instance level, the attributes are returned in the order shown in Table 7.5.2.1 using little-endian byte-swapping for 16-bit and 32-bit integers. Response is 248 bytes long including 2 byte padding at the end.

Table 7.5.2.1 (a) R-30*i*A/R-30*i*B Get_Attribute_All Responses

32-bit integer	Byte 0	Byte 1	Byte 2	Byte 3
1	Alarm ID		Alarm Number	
2	Alarm ID Cause Code		Alarm Num Cause Code	
3	Alarm Severity		PAD (All Zeros)	
4	Time Stamp			
5	Date/Time String (24 bytes)			
...	...			
12	Alarm Message (80 bytes)			
...	...			
34	Cause Code Message (80 bytes)			
...	...			
56	Alarm Severity String (24 bytes)			
...	...			

Table 7.5.2.1 (b) R-30*i*B Plus Get_Attribute_All Responses

32-bit integer	Byte 0	Byte 1	Byte 2	Byte 3
1	Alarm ID		Alarm Number	
2	Alarm ID Cause Code		Alarm Num Cause Code	
3	Alarm Severity		PAD (All Zeros)	
4	Time Stamp			
5	Date/Time String (28 bytes)			
...	...			
12	Alarm Message (88 bytes)			
...	...			
34	Cause Code Message (88 bytes)			
...	...			
56	Alarm Severity String (28 bytes)			
...	...			

7.5.3   Errors

FANUC's Vendor Specific Active Alarm Object will return the errors shown in Table 7.5.3.

Table 7.5.3 Errors

Error Status	Error Description
Undefined Attribute (0x14)	Returned when the Alarm Attribute requested does not exist.
Unsupported Service (0x08)	Returned when the requested service is unsupported.
Undefined Class Instance (0x05)	Returned when the requested instance number does not exists. For example, if there is only 1 single active alarm, requesting the Active Alarm Object instance 2 will cause this error to be returned.

7.5.4   Examples

7.5.4.1   Read most recent active alarm cause code

To read the most recent active alarm's cause code from the controller, the explicit message client would be configured with the values shown in Table 7.5.4.1.

Table 7.5.4.1 Read Most Recent Active Alarm Cause Code

Class	0xA0
Instance	0x01
Attribute	0x03
Service	0x0E

The explicit message server on the robot controller would return the cause code to the client as a 16-bit integer.

7.5.4.2   Read all alarm information from the second most recent active alarm

To read all alarm information about the second most recent active alarm from the controller, the explicit message client would be configured with the values shown in Table 7.5.4.2.

Table 7.5.4.2 Read All Alarm Information from the Second Most Recent Active Alarm

Class	0xA0
Instance	0x02
Attribute	Not Required
Service	0x01

The explicit message server on the robot controller would return all Registers as described in Section 7.5.2.1.

7.6 VENDOC SPECIFIC ALARM HISTORY OBJECT (0xA1)

Information about the Alarm History can be read through FANUC's Alarm History Object. Each instance of the object corresponds to an alarm in the history. For example, instance 1 corresponds to the most recent Alarm in the alarm history, and instance 5 corresponds to the 5th most recent Alarm.

7.6.1 Instance Attributes

The instance attributes are identical to those of FANUC's Active Alarm Object (0xA0) and are documented in Section 7.5.1.

7.6.2 Common Services

The common services are identical to those of FANUC's Active Alarm Object (0xA0) and are documented in Section 7.5.2.

7.6.3 Errors

The errors are identical to those of FANUC's Active Alarm Object (0xA0) and are documented in Section 7.5.3.

7.6.4 Examples

7.6.4.1 Read most recent alarm cause code

To read the most recent alarm's cause code from the controller, the explicit message client would be configured with the values shown in Table 7.6.4.1.

Table 7.6.4.1 Read Most Recent Alarm Cause Code

Class	0xA1
Instance	0x01
Attribute	0x03
Service	0x0E

The explicit message server on the robot controller would return the cause code to the client as a 16-bit integer.

7.6.4.2 Real all alarm information from the second most recent alarm

To read all alarm information about the second most recent alarm from the controller, the explicit message client would be configured with the values shown in Table 7.6.4.2.

Table 7.6.4.2 Real All Alarm Information from the Second Most Recent Alarm

Class	0xA1
Instance	0x02
Attribute	Not Required
Service	0x01

The explicit message server on the robot controller would return all Registers as described in Section 7.5.2.1.

7.7   VENDOR SPECIFIC MOTION ALARM OBJECT (0xA2)

Information about Motion Alarms can be read through FANUC's Motion Alarm Object. Each instance of the object corresponds to a motion alarm. For example, instance 1 corresponds to the most recent motion alarm, and instance 5 corresponds to the 5th most recent motion alarm.

7.7.1   Instance Attributes

The instance attributes are identical to those of FANUC's Active Alarm Object (0xA0) and are documented in Section 7.5.1.

7.7.2   Common Services

The common services are identical to those of FANUC's Active Alarm Object (0xA0) and are documented in Section 7.5.2.

7.7.3   Errors

The errors are identical to those of FANUC's Active Alarm Object (0xA0) and are documented in Section 7.5.3.

7.7.4   Examples

7.7.4.1   Read most recent motion alarm cause code

To read the most recent motion alarm's cause code from the controller, the explicit message client would be configured with the values shown in Table 7.7.4.1.

Table 7.7.4.1 Read Most Recent Motion Alarm Cause Code

Class	0xA2
Instance	0x01
Attribute	0x03
Service	0x0E

The explicit message server on the robot controller would return the cause code to the client as a 16-bit integer.

7.7.4.2   Read all alarm information from the second most recent motion alarm

To read all alarm information about the second most recent motion alarm from the controller, the explicit message client would be configured with the values shown in Table 7.7.4.2.

Table 7.7.4.2 Read All Alarm Information from the Second Most Recent Motion Alarm

Class	0xA2
Instance	0x02
Attribute	Not Required
Service	0x01

The explicit message server on the robot controller would return all Registers as described in Section 7.5.2.1.

7.8　VENDOR SPECIFIC SYSTEM ALARM OBJECT (0xA3)

Information about System Alarms can be read through FANUC's System Alarm Object. Each instance of the object corresponds to a system alarm. For example, instance 1 corresponds to the most recent system alarm, and instance 5 corresponds to the 5th most recent system alarm.

7.8.1　Instance Attributes

The instance attributes are identical to those of FANUC's Active Alarm Object (0xA0) and are documented in Section 7.5.1.

7.8.2　Common Services

The common services are identical to those of FANUC's Active Alarm Object (0xA0) and are documented in Section 7.5.2.

7.8.3　Errors

The errors are identical to those of FANUC's Active Alarm Object (0xA0) and are documented in Section 7.5.3.

7.8.4　Examples

7.8.4.1　Read most recent system alarm cause code

To read the most recent system alarm's cause code from the controller, the explicit message client would be configured with the following values:

Table 7.8.4.1 Read Most Recent System Alarm Cause Code

Class	0xA3
Instance	0x01
Attribute	0x03
Service	0x0E

The explicit message server on the robot controller would return the cause code to the client as a 16-bit integer.

7.8.4.2　Read all alarm information from the second most recent system alarm

To read all alarm information about the second most recent system alarm from the controller, the explicit message client would be configured with the following values:

Table 7.8.4.2 Read All Alarm Information from the Second Most Recent System Alarm

Class	0xA3
Instance	0x02
Attribute	Not Required
Service	0x01

The explicit message server on the robot controller would return all Registers as described in Section 7.5.2.1.

7.9   7.9 VENDOR SPECIFIC APPLICATION ALARM OBJECT (0xA4)

Information about Application Alarms can be read through FANUC's Application Alarm Object. Each instance of the object corresponds to an application alarm. For example, instance 1 corresponds to the most recent application alarm, and instance 5 corresponds to the 5th most recent application alarm.

7.9.1   Instance Attributes

The instance attributes are identical to those of FANUC's Active Alarm Object (0xA0) and are documented in Section 7.5.1.

7.9.2   Common Services

The common services are identical to those of FANUC's Active Alarm Object (0xA0) and are documented in Section 7.5.2.

7.9.3   Errors

The errors are identical to those of FANUC's Active Alarm Object (0xA0) and are documented in Section 7.5.3.

7.9.4   Examples

7.9.4.1   Read most recent application alarm cause code

To read the most recent alarm's cause code from the controller, the explicit message client would be configured with the values shown in Table 7.9.4.1.

Table 7.9.4.1 Read Most Recent Application Alarm Cause Code

Class	0xA4
Instance	0x01
Attribute	0x03
Service	0x0E

The explicit message server on the robot controller would return the cause code to the client as a 16-bit integer.

7.9.4.2   Read all alarm information from the second most recent application alarm

To read all alarm information about the second most recent application alarm from the controller, the explicit message client would be configured with the values shown in Table 7.9.4.2.

Table 7.9.4.2 Read All Alarm Information from the Second Most Recent Application Alarm

Class	0xA4
Instance	0x02
Attribute	Not Required
Service	0x01

The explicit message server on the robot controller would return all Registers as described in Section 7.5.2.1.

7.10   VENDOR SPECIFIC RECOVERY ALARM OBJECT (0xA5)

Information about Recovery Alarms can be read through FANUC's Recovery Alarm Object. Each instance of the object corresponds to a recovery alarm. For example, instance 1 corresponds to the most recent recovery alarm, and instance 5 corresponds to the 5th most recent recovery alarm.

7.10.1   Instance Attributes

The instance attributes are identical to those of FANUC's Active Alarm Object (0xA0) and are documented in Section 7.5.1.

7.10.2   Common Services

The common services are identical to those of FANUC's Active Alarm Object (0xA0) and are documented in Section 7.5.2.

7.10.3   Errors

The errors are identical to those of FANUC's Active Alarm Object (0xA0) and are documented in Section 7.5.3.

7.10.4   Examples

7.10.4.1   Read most recent recovery alarm cause code

To read the most recent recovery alarm's cause code from the controller, the explicit message client would be configured with the values shown in Table 7.10.4.1.

Table 7.10.4.1 Read Most Recent Recovery Alarm Cause Code

Class	0xA5
Instance	0x01
Attribute	0x03
Service	0x0E

The explicit message server on the robot controller would return the cause code to the client as a 16-bit integer.

7.10.4.2   Read all alarm information from the second most recent recovery alarm

To read all alarm information about the second most recent recovery alarm from the controller, the explicit message client would be configured with the values shown in Table 7.10.4.2.

Table 7.10.4.2 Read All Alarm Information from the Second Most Recent Recovery Alarm

Class	0xA5
Instance	0x02
Attribute	Not Required
Service	0x01

The explicit message server on the robot controller would return all Registers as described in Section 7.5.2.1.

7.11   VENDOR SPECIFIC COMMUNICATIONS ALARM OBJECT (0xA6)

Information about Communications Alarms can be read through FANUC's Communications Alarm Object. Each instance of the object corresponds to a communications alarm. For example, instance 1 corresponds to the most recent communications alarm and instance 5 corresponds to the 5th most recent communications alarm.

7.11.1   Instance Attributes

The instance attributes are identical to those of FANUC's Active Alarm Object (0xA0) and are documented in Section 7.5.1.

7.11.2   Common Services

The common services are identical to those of FANUC's Active Alarm Object (0xA0) and are documented in Section 7.5.2.

7.11.3   Errors

The errors are identical to those of FANUC's Active Alarm Object (0xA0) and are documented in Section 7.5.3.

7.11.4   Examples

7.11.4.1   Read most recent communication alarm cause code

To read the most recent alarm's cause code from the controller, the explicit message client would be configured with the values shown in Table 7.11.4.1.

Table 7.11.4.1 Read Most Recent Communication Alarm Cause Code

Class	0xA6
Instance	0x01
Attribute	0x03
Service	0x0E

The explicit message server on the robot controller would return the cause code to the client as a 16-bit integer.

7.11.4.2   Read all alarm information from the second most recent communications alarm

To read the most recent alarm's cause code from the controller, the explicit message client would be configured with the values shown in Table 7.11.4.2.

Table 7.11.4.2 Read All Alarm Information from the Second Most Recent Communications Alarm

Class	0xA6
Instance	0x02
Attribute	Not Required
Service	0x0E

The explicit message server on the robot controller would return all Registers as described in Section 7.5.2.1.

7.12   ACCESSING I/O USING EXPLICIT MESSAGING

7.12.1   Accessing I/O Specific to an Implicit EtherNet/IP Connection

I/O can be accessed through ODVA's standard Assembly Object. Table 7.12.1 (a) describes the Assembly Instance numbers that can be used to read or write I/O specific to an Implicit EtherNet/IP connection configured on a FANUC robot controller.

> **NOTE**
> I/O cannot be written to the writable assembly instances using explicit messaging while an active implicit connection to the instance is running.

Detailed instructions for mapping I/O on the robot controller can be found in Section 6.2 of this manual.

Table 7.12.1(a) Accessing I/O

Instance Number	Read/Write	Input/Output	Slot
101	r	output	1
102	r	output	2
103	r	output	3
104	r	output	4
105	r	output	5
106	r	output	6
107	r	output	7
108	r	output	8
109	r	output	9
110	r	output	10
111	r	output	11
112	r	output	12
113	r	output	13
114	r	output	14
115	r	output	15
116	r	output	16
117	r	output	17
118	r	output	18
119	r	output	19
120	r	output	20
121	r	output	21
122	r	output	22
123	r	output	23
124	r	output	24
125	r	output	25
126	r	output	26
127	r	output	27
128	r	output	28
129	r	output	29
130	r	output	30
131	r	output	31
132	r	output	32
1101	r	output	33
1102	r	output	34
1103	r	output	35
1104	r	output	36
1105	r	output	37

Instance Number	Read/Write	Input/Output	Slot
1106	r	output	38
1107	r	output	39
1108	r	output	40
1109	r	output	41
1110	r	output	42
1111	r	output	43
1112	r	output	44
1113	r	output	45
1114	r	output	46
1115	r	output	47
1116	r	output	48
1117	r	output	49
1118	r	output	50
1119	r	output	51
1120	r	output	52
1121	r	output	53
1122	r	output	54
1123	r	output	55
1124	r	output	56
1125	r	output	57
1126	r	output	58
1127	r	output	59
1128	r	output	60
1129	r	output	61
1130	r	output	62
1131	r	output	63
1132	r	output	64
151	r/w*	input	1
152	r/w*	input	2
153	r/w*	input	3
154	r/w*	input	4
155	r/w*	input	5
156	r/w*	input	6
157	r/w*	input	7
158	r/w*	input	8
159	r/w*	input	9
160	r/w*	input	10
161	r/w*	input	11
162	r/w*	input	12
163	r/w*	input	13
164	r/w*	input	14
165	r/w*	input	15
166	r/w*	input	16
167	r/w*	input	17
168	r/w*	input	18
169	r/w*	input	19
170	r/w*	input	20
171	r/w*	input	21
172	r/w*	input	22
173	r/w*	input	23
174	r/w*	input	24
175	r/w*	input	25
176	r/w*	input	26

Instance Number	Read/Write	Input/Output	Slot
177	r/w*	input	27
178	r/w*	input	28
179	r/w*	input	29
180	r/w*	input	30
181	r/w*	input	31
182	r/w*	input	32
1151	r/w*	input	33
1152	r/w*	input	34
1153	r/w*	input	35
1154	r/w*	input	36
1155	r/w*	input	37
1156	r/w*	input	38
1157	r/w*	input	39
1158	r/w*	input	40
1159	r/w*	input	41
1160	r/w*	input	42
1161	r/w*	input	43
1162	r/w*	input	44
1163	r/w*	input	45
1164	r/w*	input	46
1165	r/w*	input	47
1166	r/w*	input	48
1167	r/w*	input	49
1168	r/w*	input	50
1169	r/w*	input	51
1170	r/w*	input	52
1171	r/w*	input	53
1172	r/w*	input	54
1173	r/w*	input	55
1174	r/w*	input	56
1175	r/w*	input	57
1176	r/w*	input	58
1177	r/w*	input	59
1178	r/w*	input	60
1179	r/w*	input	61
1180	r/w*	input	62
1181	r/w*	input	63
1182	r/w*	input	64

NOTE
  The slots after slot 33 can only be used with the R-30*i*B Plus Controller.

* Only instances corresponding to Adapter connections are writable using Explicit Messaging; however, these instances will not be writable while an active implicit connection to the instance is running. If the corresponding connection is a Scanner connection, then the instance will be read-only.

For example, suppose the controller is configured with one Adapter connection on slot 1 with a 16-bit word of input and a 16-bit word of output, which are mapped to DI[1-16] and to DO[1-16] respectively. Once an implicit connection is establish to the adapter, the output values in DO[1-16] can be accessed through explicit messaging with the values shown in Table 7.12.1 (b).

Table 7.12.1 (b) Output Values

Class	0x04
Instance	0x65
Attribute	0x03
Service	0x0E

And the input values in DI[1-16] can be accessed through explicit messaging with the values Table 7.12.1 (c).

Table 7.12.1 (c) Input Values

Class	0x04
Instance	0x97
Attribute	0x03
Service	0x0E

7.12.2   Accessing General I/O

In addition to I/O mapped from EtherNet/IP connections, other types of I/O can be read with explicit messaging using the following Assembly Object Instance numbers.

Table 7.12.2 (a) Accessing General I/O

I/O Type	Instance Number (hexadecimal)
Digital input	0x320
Digital output	0x321
Analog input	0x322
Analog output	0x323
Tool output	0x324
PLC input	0x325
PLC output	0x326
Robot digital input	0x327
Robot digital output	0x328
Brake output	0x329
Operator panel input	0x32a
Operator panel output	0x32b
Teach pendant digital input	0x32d
Teach pendant digital output	0x32e
Weld input	0x32f
Weld output	0x330
Group input	0x331
Group output	0x332
User operator panel input	0x333
User operator panel output	0x334
Laser DIN	0x335
Laser DOUT	0x336
Laser AIN	0x337
Laser AOUT	0x338
Weld stick input	0x339
Weld stick output	0x33a
Memory image boolean	0x33b
Memory image DIN	0x33c
Dummy boolean port type	0x33d
Dummy numeric port type	0x33e
Process axes (ISDT)	0x33f
Internal operator panel input	0x340
Internal operator panel output	0x341

I/O Type	Instance Number (hexadecimal)
Flag (F[ ])	0x342
Marker (M[ ])	0x343

For example, the values shown in Table 7.12.2 (b) would access all Digital Outputs (DOs) with explicit messaging.

Table 7.12.2 (b) Accessing Digital Outputs

Class	0x04
Instance	0x321
Attribute	0x03
Service	0x0E

7.13   USING EXPLICIT MESSAGING IN RSLogix 5000

This section steps through an example of how to configure an I/O read and write operation on a robot controller using RSLogix5000. In this example, an I/O read and write is done on Rack 89 Slot 1 of the robot controller every 1000ms.

Three rungs are created in our main program. The first rung is a timer rung. This timer, mtime, will trigger read and write messages to be sent to the robot controller every 1000ms. The next two rungs have MSG blocks, where the individual Explicit Messages will be defined. Fig. 7.13(a) shows the three rungs.
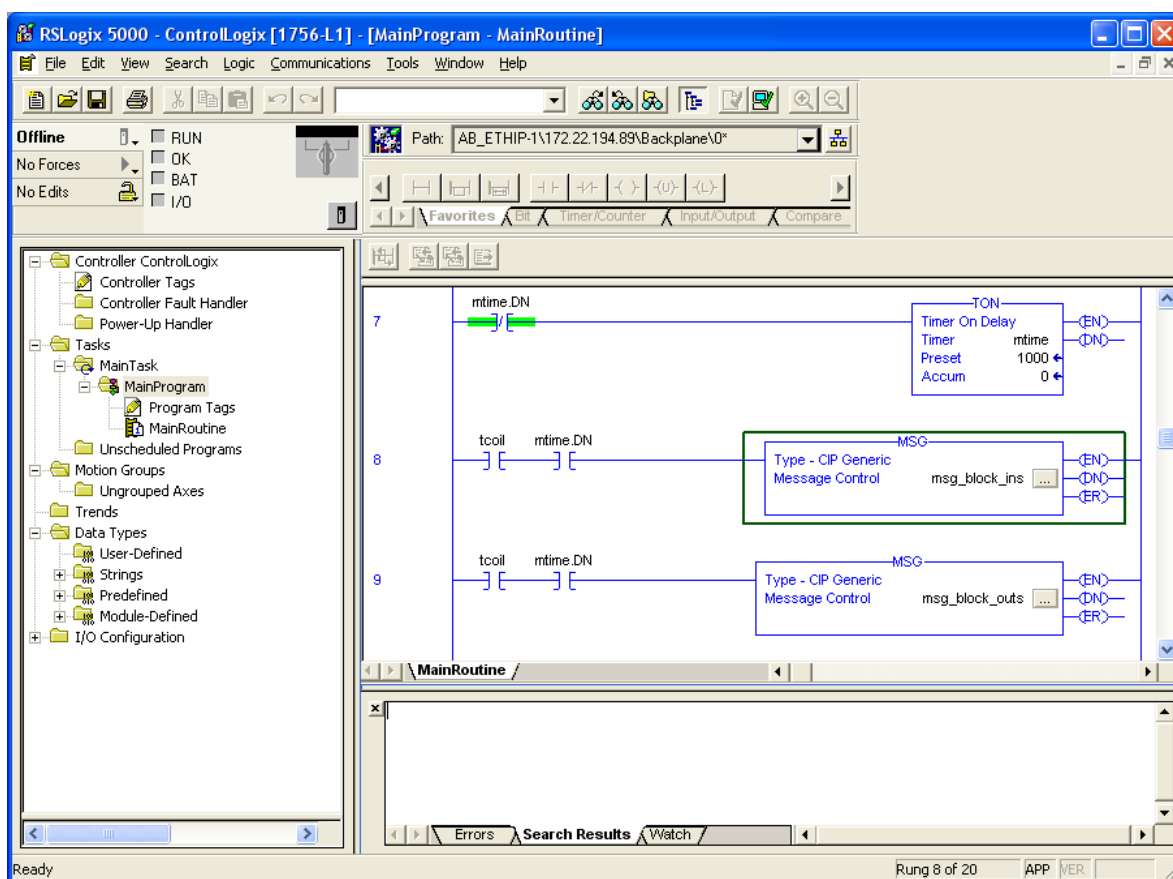
Fig. 7.13 (a) RsLogix 5000 Example Rungs

> **NOTE**
> We have created the element tcoil to turn the sending of the two Explicit Messages on and off.

To add a message block, you will need to add a MSG ladder element. Fig. 7.13 (b) shows an example of adding a MSG ladder element.
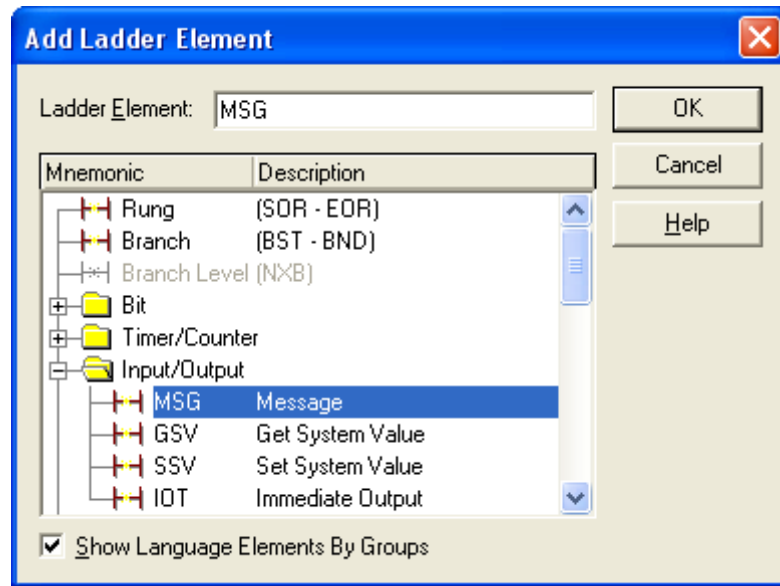
Fig. 7.13 (b) RSLogix 5000 Add MSG Block

Configuration of the message block requires the Class, Instance, Attribute, and Service values as discussed in Section 7.3 of this manual. For example, to read the robot controller outputs at Rack 89 Slot 1, we would access the Assembly Class (0x04), Attribute 3 (0x03), Instance 101 (0x65), and Service Get_Attribute_Single (0x0e). See Section 7.12 for more details.

Thus, once configured, the MSG block should look similar to Fig. 7.13(c). You will need to create a destination for the robot controller outputs to be read into. In this example, we created an array named robot_douts.
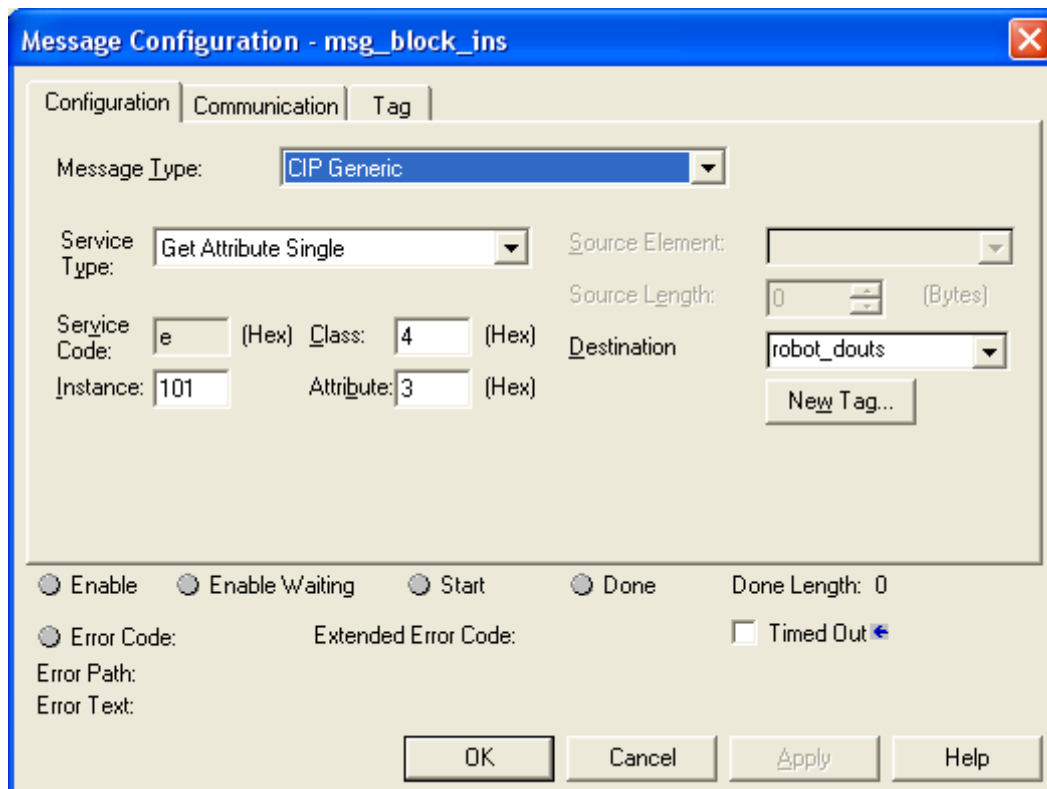
Fig. 7.13 (c) MSG Block: Read Robot DOUTs

Next, click on the Communications tab. From this tab you should be able to browse to the device to which you want to connect. In Fig. 7.13 (d), we browsed to and are connecting to a device named pderob224. Note that the robot controller must already be configured in RSLogix5000's I/O Configuration for the Browse button to successfully find the robot.
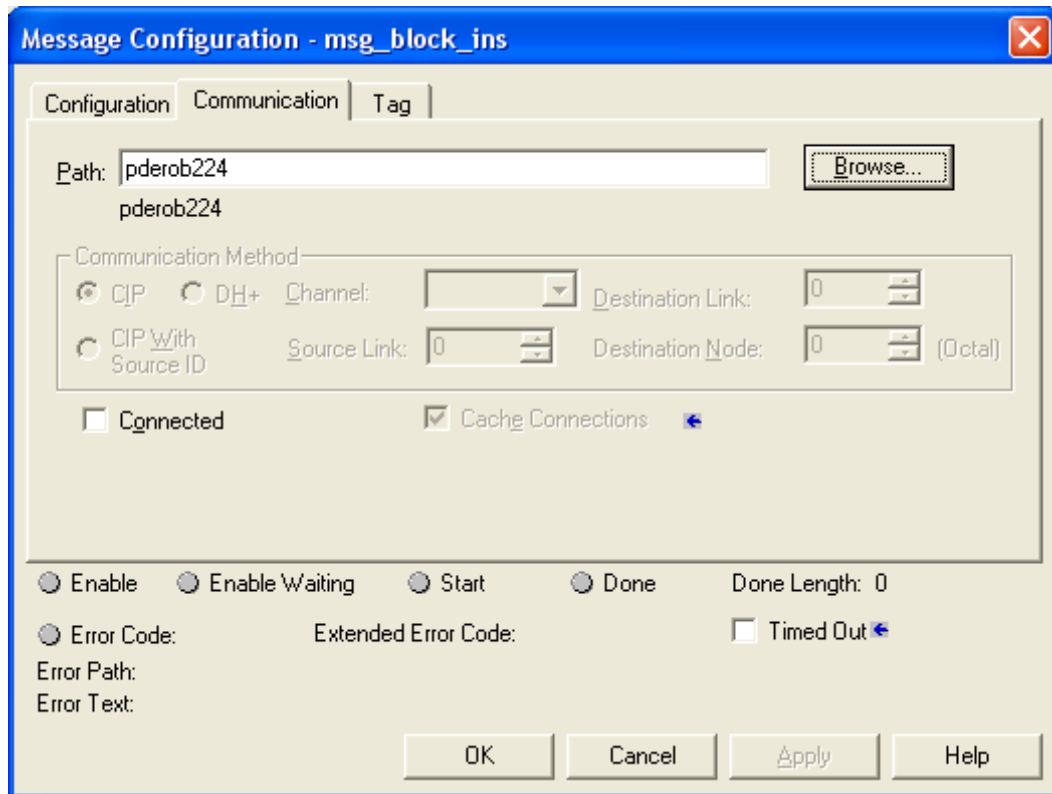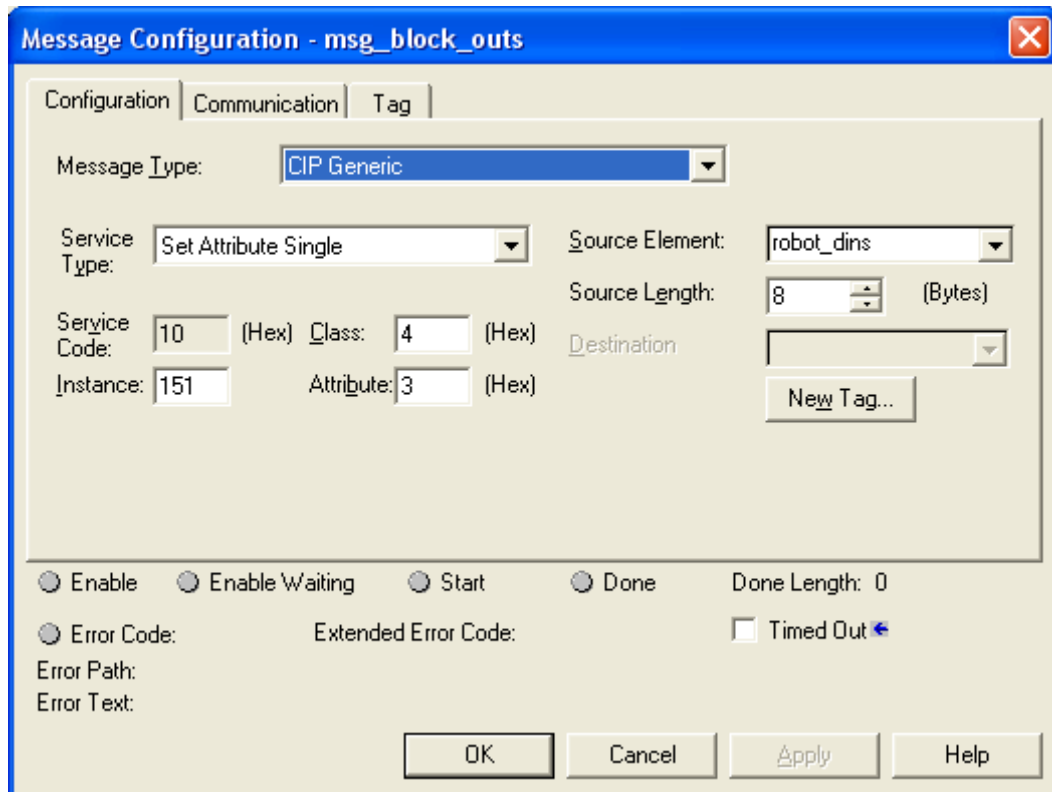
Fig. 7.13 (d) MSG Block Communication Tab

Documentation for RSLogix5000 also provides two additional ways of expressing the Path. First, the path can be an expression of comma-separated values that indicate the route for the MSG starting at the Ethernet module on the PLC and ending at the target device. For example "ENET,2,192.168.1.224" would be a path from the ENET module, port 2 on the ENET module (this value should always be 2), to the IP address of the robot controller.

Secondly, the same path could be expressed as "1,2,2,192.168.1.224". Where the 1 represents the slot number of the processor in the rack, and the 2 in the second position represents the slot number of the ENET module. The 2 in the third position would represent port two on the ENET module, and the fourth position contains the IP address of the robot.

Fig. 7.13 (e) shows the MSG block used to write to the robot controller's inputs. In this case we use Class 0x04, Instance 151 (0x97), Attribute 0x03, and Service Set_Attribute_Single (0x10). We also created the array robot_dins to write to the robot controller.

Fig. 7.13 (e) MSG Block: Write Robot DINs

8   NETWORK DESIGN AND PERFORMANCE

8.1   NETWORK DESIGN CONSIDERATIONS

Good network design is critical for reliable operation. It is important to pay special attention to wiring guidelines and environmental conditions affecting the cable system and equipment. It is also necessary to control network traffic to avoid wasted network bandwidth and device resources.

Keep in mind the following wiring guidelines and environmental considerations:
- Use category 5 twisted pair (or better) rated for 100-BaseTX Ethernet applications and the application environment. Consider shielded versus unshielded twisted pair cabling.
- Pay careful attention to wiring guidelines such as maximum length from the switch to the device (100 meters).
- Do not exceed recommended bending radius of specific cabling being used.
- Use connectors appropriate to the environment. There are various industrial Ethernet connectors in addition to the standard open RJ45 that should be used where applicable. For example, connectors are available with IP65 or IP67 ratings. M12 4-pin D-coded Connectors are included in the Ethernet/IP specification.
- Route the wire runs away from electrical or magnetic interference or cross at ninety degrees to minimize induced noise on the Ethernet network.

Keep the following in mind as you manage network traffic:
- Control or eliminate collisions by limiting the collision domain.
- Control broadcast traffic by limiting the broadcast domain.
- Control multicast traffic with multicast aware switches (support for IGMP snooping).
- Use QOS (Quality of Service) techniques in very demanding applications.

Collisions are a traditional concern on an Ethernet network but can be completely avoided by using switches—rather than hubs—and full duplex connections. It is critical to use switches and full duplex connections for any Ethernet I/O network, because it reduces the collision domain to only one device so that no collisions will occur. The robot interface will autonegotiate by default and use the fastest connection possible. Normally this is 100Mbps and full duplex. The robot can be set for a specific connection speed and duplex (refer to the chapter titled "Setting Up TCP/IP" in the "Ethernet Function OPERATOR'S MANUAL (B-82974EN)"). However be very careful that both ends of the connection use the same speed and duplex mode. Be careful not to set one end of a connection for autonegotiate and set the other end to a specific speed duplex – both ends must autonegotiate, or both ends must be fixed to the same settings.

The LEDs near the RJ45 connector on the robot will confirm a connection link (refer to the appendix titled "Diagnostic Information" in the "Ethernet Function OPERATOR'S MANUAL (B-82974EN)" for details on the LEDs). Link State can be confirmed using the TCP/IP status Host Comm screen by following Procedure 8-1.

Procedure 8-1 Verifying Link State

1   Press the [MENU] key.
2   Select Setup.
3   Press F1, [TYPE] and select Host Comm.
4   Select TCP/IP.
5   Toggle to the correct port (port #1 or port #2) by pressing F3, [PORT].
6   Press the [NEXT] key, then F2, [STATUS].

Broadcast traffic is traffic that all nodes on the subnet must listen for and in some cases respond to. Excessive broadcast traffic wastes network bandwidth and wastes resources in all effected nodes. The broadcast domain is the range of devices (typically the entire subnet) that must listen to all broadcasts. Limit the broadcast domain to only the control devices (for example, EtherNet/IP nodes) by using a separate subnet for the control equipment or by using VLANs (virtual LANs) supported by some higher end switches. If the EtherNet/IP network is completely isolated as a separate control network this is not a concern. However, when connecting into larger networks this becomes important.

Some network environments have a significant amount of multicast traffic. A basic layer 2 switch will treat multicast traffic like broadcast traffic and forward to all ports in the switch wasting network bandwidth and node resources on traffic which is ultimately dropped for the nodes that are not interested in the multicast traffic. Switches that support "IGMP snooping" will selectively send multicast traffic only to the nodes which have joined a particular group. EtherNet/IP UDP packet has a TTL (time to link) value of one. You will not be able to route I/O traffic across more than one switch.

Quality of Service (QOS) techniques provide mechanisms to prioritize network traffic. Generally on an Ethernet network all packets are equal. Packets can be dropped or delayed within network infrastructure equipment (for example, switches) in the presence of excessive traffic. Which packets are dropped or delayed is random.

QOS is a term covering several different approaches to prioritizing packets including:
- MAC layer (layer 2) prioritization (IEEE 802.1p).
- IP layer (layer 3) prioritization using source/destination IP addresses.
- Transport layer (layer 4) prioritization using source/destination ports.

These QOS mechanisms are generally implemented within the network infrastructure equipment and are beyond the scope of this manual. Some form of QOS should be considered on complex networks requiring the highest possible level of determinism in I/O exchanges within the control network.

It is important to select the proper switch in order for the network to function correctly. The switch should support :
- 100 Mbps baud rate
- Full duplex connections
- Port auto-negotiation
- Environmental specifications appropriate for the application (for example, temperature)
- Power supply requirements and redundancy (for example, support for 24vdc or 120vac and support for a second redundant power supply if warranted)

NOTE
If there is a significant amount of multicast traffic, the switch should support IGMP snooping (multicast aware). Please consider this when Ethernet/IP and/or RIPE (robot ring) traffic exists.

NOTE
If the control network will be part of a larger network, the control network should be on a separate VLAN or subnet. This can be done within the control switch or possibly based on how the larger network connects to the control switch.

Some examples of switch products are:
- Cisco 2955 (industrialized version of 2950) – www.cisco.com
- Hischmann MICE (modular industrial switch) – www.hirschmann.de
- Phoenix Contact (managed/unmanaged industrial switch) – www.ethernetrail.com
- N-Tron 508TX-A, 8 port industrial switch with advanced firmware – www.n-tron.com

8.2   I/O RESPONSE TIME

The system response time is the amount of time it takes an I/O signal to propagate through the system to its destination and back again. For a Controller -to- PLC system this time would be from the time an output is sent to the time a modified input is read. The system response time depends on many factors including:

- The Actual Packet Interval (API is based on RPI)
- PLC Ladder Scan Time
- No lost or delayed packets due to excessive traffic or noise

To calculate the response time, keep in mind that the response time is asynchronous but has a deterministic upper limit. After a signal is set in the I/O Image, it will take a maximum of one API before it gets transmitted to any node on the network.

Fig. 8.2 shows a case where a DO is transferred to a PLC and back as a DI. In this case, after the DO is set in the I/O Image, it will take a maximum of one API, t(api), to get the DO to the Ethernet transceiver. After the DO is in the Ethernet transceiver, it is sent to the destination (PLC) at wire speed, t(wire) (assumes full duplex link so no collisions).
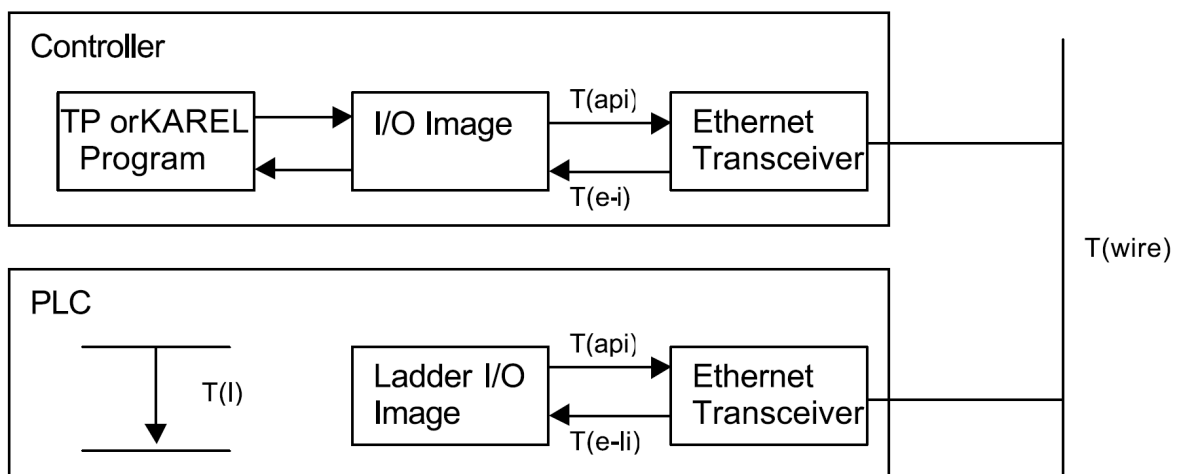
Fig. 8.2 EtherNet/IP Response Time Diagram

In the PLC, the inputs are scanned into the Ladder I/O Image, which fixes them for the entire Ladder Scan, t(e-li). The inputs are processed during the ladder scan t(l) and then set back to the PLC's Ethernet transceiver at it's API rate.

The PLC outputs are transferred at Ethernet wire speed back to the controller. They are then transferred to the I/O Image t(e-i) where they can be read by the KAREL or teach pendant program.

```
T(Controller -to- PLC -to- Controller) = t(api) + t(wire) + t(e-li) + t(l) +
      t(api) + t(wire) + t(e-i)
```

- t(api) = KAREL or teach pendant outputs get immediately set in the I/O Image. The time necessary to get to them to Ethernet transceiver can be a maximum of one API. In this example it is assumed robot->plc API and plc->robot API values are the same.
- t(wire) = The time it takes for the packet to traverse the network including any switch delays due to queueing.
- t(e-l i) = After the signal is in the PLC Ethernet transceiver, it must get processed through the PLC network stack and placed in an appropriate ladder image data file to be accessed by the PLC Ladder.

- t(l) = The input value needs to be fixed for the entire scan in the ladder. The PLC Scan can usually be obtained by examining an appropriate status register in the PLC. After the signal has been processed, the reverse process must take place.
- T(e-i) = After the signal is in the robot Ethernet transceiver, it must get processed through the robot network stack and placed in I/O image area to be accessed by the TP or Karel program.

For example, using V7.10 or higher and a ControlLogix PLC over a simple network with a 20ms API, the following times were calculated. This example assumes wire time is negligible (100Mbps network, no switch delays), input packets are processed through the network stack and into image area within 1ms, and ladder scan time is 5ms.

```
T(Controller - to - PLC - to - Controller
) = t(api) + t(wire) + t(e-li) + t(l) + t(api) + t(wire) + t(e-i)
```

```
T(Controller - to - PLC - to - Controller) = 20ms + 0ms + 1ms + 5ms +
20ms + 0ms + 1ms
```

```
T(Controller - to - PLC - to - Controller) = 47ms ***
```

*** This value assumes no delayed/lost packets due to excessive traffic or noise.

Your actual PLC ladder scan times might vary from the example. Most PLCs offer the capability to get the actual scan time from a programmer or monitor.

This example assumes the packet is not delayed or dropped in a network switch or at the source/destination node. Packets can be dropped due to the following reasons:
- Excessive traffic can cause queue delays or dropped packets in the switch or source/destination nodes depending on extent of traffic and queue sizes.
- Packet corruption due to noise can cause a bad CRC check on the packet (a packet with a bad CRC is dropped).

The maximum upper limit is based on EtherNet/IP timeout values. Timeouts will occur when a consumer does not receive data from a producer within a multiple of the API. Typically this timeout value is 3-4 times the API value. If a timeout occurs, an error is posted. The error severity and last state I/O behavior can be configured. Refer to Section 3.2.3 for adapter.

9   DIAGNOSTICS AND TROUBLESHOOTING

9.1   VERIFYING NETWORK CONNECTIONS

There are two basic tools for verifying network connections:
- Ethernet status LEDs
- PING

The LEDs and PING utility are basic tools but they give a good indication of whether or not devices are able to communicate on the network. If the LINK LED is off, or if PING times out, then no other network functionality will work for that device.

Refer to Section 9.1.1 for more information about Ethernet status LEDs.

Refer to Section 9.1.2 for more information about the PING utility.

9.1.1   Ethernet Status LEDs

The Ethernet status LEDs at the Ethernet RJ45 connector on the robot will indicate if the robot is connected. Most Ethernet switches and other equipment will have similar LEDs indicating a physical connection. If the LINK LED is off then there is no Ethernet connectivity at all. This generally implies a disconnected or bad cable or bad connections. The speed and duplex must match between the robot and the switch. For more information about the Ethernet status LEDs, refer to the appendix titled "Diagnostic Information" in the "Ethernet Function OPERATOR'S MANUAL (B-82974EN)". Details on auto-negotiating and manually setting speed and duplex level can be found in the chapter titled " Setting Up TCP/IP" in the "Ethernet Function OPERATOR'S MANUAL (B-82974EN)". The robot will auto-negotiate by default and should not be changed in most cases.

9.1.2   PING Utility

PING is a network utility that sends a request to a specific IP address and expects a response. The request is essentially "Can you hear me?" The destination node will send a response that it received the request. The requesting node will either receive the response or timeout. PING is a basic network utility that is included with most operating systems, such as Windows and Unix, and is also supported on the robot. Even devices that do not support generating PING requests (for example, an EtherNet/IP block with no user interface) will respond to the PING request.

The robot supports PING directly from the EtherNet/IP status screen. Use Procedure 9-1.

The PING utility is also available on the robot to PING any name or IP address. Use Procedure 9-2.

The PING utility is also available from any windows PC. Use Procedure 9-3.

Procedure 9-1 Using PING from the EtherNet/IP Status Screen

Steps
1 Press the [MENU] key.
2 Select I/O.
3 Press F1, [TYPE].
4 Select EtherNet/IP.
5 Move the cursor to the connection with the device you want to PING.
6 Press F2, [PING].

The prompt line on the teach pendant will indicate if the PING was successful or if the PING request timed out.

> **NOTE**
> This function only works on the adapter connection (connection #1) if there is a scanner connected.

Procedure 9-2 Using PING from the Host Comm Screen

Steps
1 Press the [MENU] key.
2 Select Setup.
3 Press F1, [TYPE].
4 Select Host Comm.
5 Move the cursor to select PING in the Protocol List and press the [ENTER] key.
6 Enter the name or IP address of the node to PING.
7 Press F2, [PING].

The prompt line on the teach pendant will indicate if the PING was successful or if the PING request timed out.
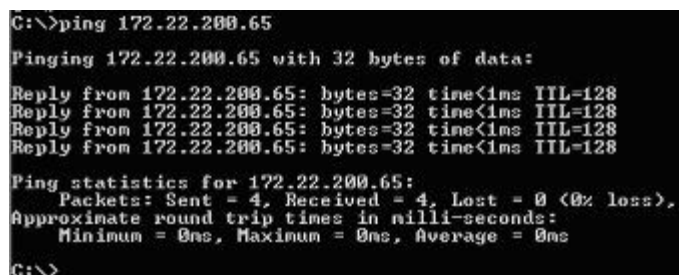
Procedure 9-3 Using PING on a Windows PC

Steps
1 Open a DOS command prompt.
2 Type the following command, replacing the IP address with the IP address you want to PING, and press the [ENTER] key.

```
PING 192.168.0.10
```

The following image shows a successful PING.

The following image shows an unsuccessful PING.

```
C:\>ping 172.22.200.240

Pinging 172.22.200.240 with 32 bytes of data:

Request timed out.
Request timed out.
Request timed out.
Request timed out.

Ping statistics for 172.22.200.240:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),

C:\>
```

If the LINK LED is on but a PING request fails it usually indicates a problem with IP address configuration. Either no IP address is configured, or the combination of IP address and subnet mask is inconsistent for the network. Refer to the chapter titled "Setting Up TCP/IP" in the "Ethernet Function OPERATOR'S MANUAL (B-82974EN)" for details on configuring the IP address and subnet mask for the robot.

9.2   ERROR CODES

The following error codes are defined by the EtherNet/IP January 2005 Specification. When a controller scanner connection fails when establishing a connection to a target device, the controller posts an error in the following format (PRIO-350 is the error code, and PRIO-358 is the cause code).

```
PRIO-350 EtherNet/IP Scanner Error (#)
PRIO-358 EtherNet/IP Fwd Open Fail (0x#)
```

The PRIO-350 code (#) specifies on which connection the error has occurred. The PRIO-358 code (0x#) specifies the **extended status** of the error returned by the target device (in hexadecimal format).

Table 9.2 lists the descriptions of the extended status error codes:

> **NOTE**
> EtherNet/IP alarms are documented in the "OPERATOR'S MANUAL (Alarm Code List) (B-83284EN-1)". For the R-30*i*A/R-30*i*A Mate Controller, refer to the "R-30*i*A/R-30*i*A Mate Controller OPERATOR'S MANUAL (Alarm Code List) (B-83124EN-6)".

Table 9.2 Forward Open Failure Error codes

GENERAL STATUS	EXTENDED STATUS	DESCRIPTION
0x01	0x0100	Connection in Use or Duplicate Forward Open
0x01	0x0103	Transport Class and Trigger combination not supported
0x01	0x0106	Ownership Conflict
0x01	0x0107	Connection not found at target application.
0x01	0x0108	Invalid Connection Type. Indicates a problem with either the Connection Type or Priority of the Connection
0x01	0x0109	Invalid Connection Size
0x01	0x0110	Device not configured
0x01	0x0111	RPI not supported. Might also indicate problem with connection timeout multiplier or production inhibit time.
0x01	0x0113	Connection Manager cannot support any more connections

GENERAL STATUS	EXTENDED STATUS	DESCRIPTION
0x01	0x0114	Either the Vendor Id or the Product Code in the key segment did not match the device
0x01	0x0115	Product Type in the key segment did not match the device
0x01	0x0116	Major or Minor Revision information in the key segment did not match the device
0x01	0x0117	Invalid Connection Point
0x01	0x0118	Invalid Configuration Format
0x01	0x0119	Connection request fails since there is no controlling connection currently open.
0x01	0x011A	Target Application cannot support any more connections
0x01	0x011B	RPI is smaller than the Production Inhibit Time.
0x01	0x0203	Connection cannot be closed since the connection has timed out
0x01	0x0204	Unconnected Send timed out waiting for a response.
0x01	0x0205	Parameter Error in Unconnected Send Service
0x01	0x0206	Message too large for Unconnected message service
0x01	0x0207	Unconnected acknowledge without reply
0x01	0x0301	No buffer memory available
0x01	0x0302	Network Bandwidth not available for data
0x01	0x0303	No screeners available
0x01	0x0304	Not Configured to send real-time data
0x01	0x0311	Port specified in Port Segment Not Available
0x01	0x0312	Link Address specified in Port Segment Not Available
0x01	0x0315	Invalid Segment Type or Segment Value in Path
0x01	0x0316	Error in close path
0x01	0x0317	Scheduling not specified
0x01	0x0318	Link Address to Self Invalid
0x01	0x0319	Resources on Secondary Unavailable
0x01	0x031A	Connection already established
0x01	0x031B	Direct connection already established
0x01	0x031C	Miscellaneous
0x01	0x031D	Redundant connection mismatch
0x01	0x031E	No more consumer resources available in the producing module
0x01	0x031F	No connection resources exist for target path
0x01	0x320 — 0x7FF	Vendor specific

10   ETHERNET/IP ENHANCED DATA ACCESS

10.1   Overview

Ethernet/IP Enhanced Data Access (EDA) allows users to configure data to be sent to PLC over standard Ethernet/IP I/O (Implicit) or Explicit connection. It provides an easy method to set up and communicate production data that is typically not mapped as normal I/O. Data structures of user's choice and default PLC logic can be directly imported into the PLC. Variable names are imported in the form of comments defined in the robot into the PLC data structures without retyping the comments in the PLC. Available data types include Numeric Registers, String Registers, Position Registers, KAREL and SYSTEM Variables (All atomic data types, KAREL String, XYZWPR, XYZWPREXT, JOINTPOS and CURPOS) and a combination of any of these data types. It provides two modes to exchange data:

- Implicit Connection
- Explicit Connection

Implicit connection allows data exchange along with standard I/O. However, an explicit connection supports only non-I/O data (Numeric Registers, String Registers, Position Registers, CURPOS, KAREL and System variables). CURPOS has read only access (send to PLC) through this feature.

The minimum RPI (Requested Packet Interval) for Implicit connections is 8ms. Below (Table 10.1) are recommended access intervals for Implicit and Explicit connections. Please note that bigger data requires higher interval considering other loads in the systems. If that is the not case, EIP/EDA may cause a time out when other operations like MD: backups are performed from the TP screen for example.

Table 10.1 Access Intervals

Connection Type	Access Time/RPI
Implicit	32ms
Explicit	100ms

> **NOTE**
> Please note that the I/O exchange can be achieved through standard Explicit Messaging in general.

Fig. 10.1 shows the data setup and access process.
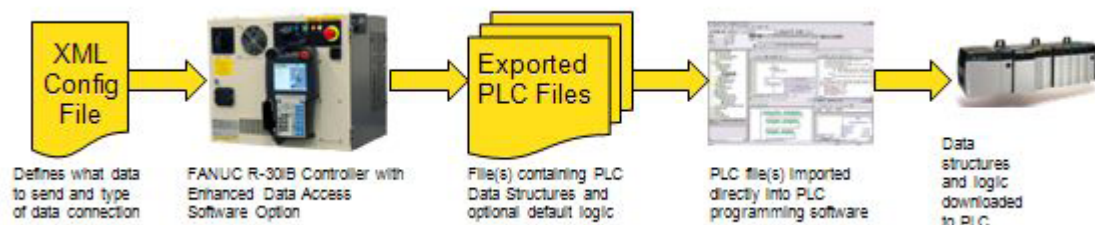
Fig. 10.1 EDA Process
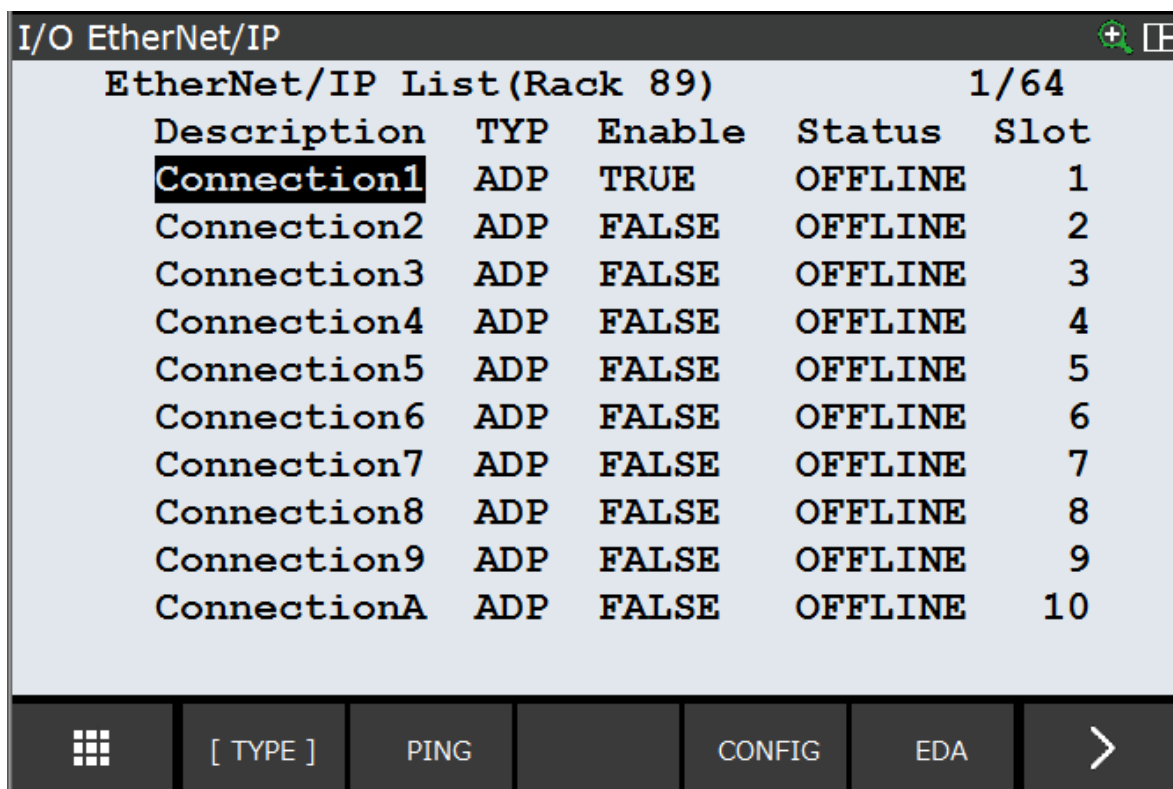
10.2   Setup and Configuration

This option (R822) requires the Ethernet/IP Adapter (R784) option. Each user defined data type (UDT) needs to be configured in the robot for respective adapter (implicit) or explicit messaging. This configuration initializes respective connections or attributes in case of explicit messaging. The export file is generated based on the user configuration. The export file creates appropriate data structures, controller tags, and rungs in the PLC on import. There are two ways to create this configuration.

1   Configure using TP
2   Configure using PC based Text Editor

10.2.1   Configuration using TP

The TP EDA configuration screen allows users to create or modify configuration file. Go to MENU ⇒IO ⇒Ethernet/IP, you will see screen as shown in Fig. 10.2.1(a). Now press F5, [EDA] to display the EDA screens. If you don't see F5, [EDA], move the cursor to the Description column. EDA label will appear at F5. Enhanced Data Access screen is divided in multiple sub-screens.

```
I/O EtherNet/IP                                       🔍 ⊞
     EtherNet/IP List(Rack 89)                  1/64
     Description    TYP    Enable    Status    Slot
     Connection1    ADP    TRUE      OFFLINE     1
     Connection2    ADP    FALSE     OFFLINE     2
     Connection3    ADP    FALSE     OFFLINE     3
     Connection4    ADP    FALSE     OFFLINE     4
     Connection5    ADP    FALSE     OFFLINE     5
     Connection6    ADP    FALSE     OFFLINE     6
     Connection7    ADP    FALSE     OFFLINE     7
     Connection8    ADP    FALSE     OFFLINE     8
     Connection9    ADP    FALSE     OFFLINE     9
     ConnectionA    ADP    FALSE     OFFLINE    10


     ▦      [ TYPE ]    PING              CONFIG    EDA      ＞
```

Fig. 10.2.1 (a) Ethernet/IP Main Screen

The EDA screen looks as shown in Fig. 10.2.1(d). It provides information about Export type which is set to 'Both' by default and changes as per configuration. You don't need to change it unless configuration is done for Implicit and Explicit both modes and user wants to generate export file for only one of them. The next item is the PLC configuration software (e.g. RSLogix5000) information. It has name, revision and version. Revision is the most important item. It is the Schema Revision which MUST match with your RSLogix schema revision. Otherwise, the import operation of exported .L5X file will fail. Usually, the schema revision doesn't change frequently. You can check the schema revision by exporting MainProgram in RSLogix5000. Open up any project in RSLogix5000 and right click on MainProgram (or any other program) then select Export Program... as shown in Fig. 10.2.1 (b). Please make sure the program is NOT inhibited. Otherwise you won't see the Export Program... in the right click menu.
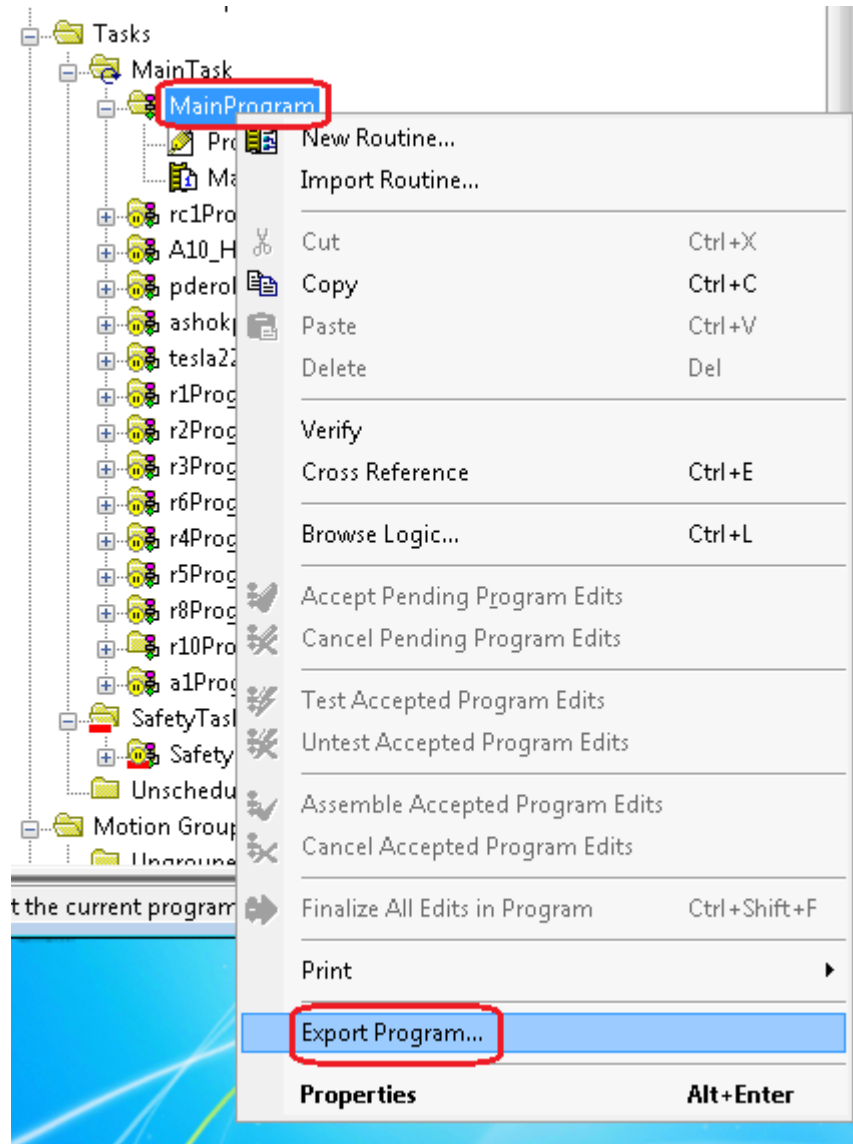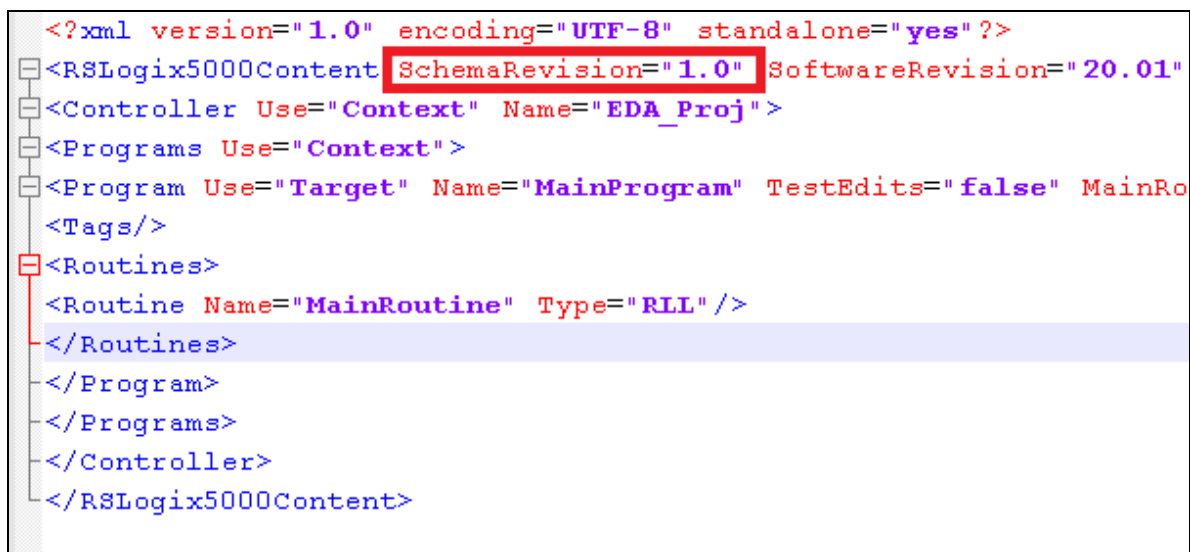
Fig. 10.2.1 (b) Export Program

Now you will be prompted to save the file. Once the file is saved, open MainProgram.L5X or whatever name you gave it and look for SchemaRevision="1.0" as shown in Fig. 10.2.1 (c). It is in the second line of the file which supported schema revision.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<RSLogix5000Content SchemaRevision="1.0" SoftwareRevision="20.01"
<Controller Use="Context" Name="EDA_Proj">
<Programs Use="Context">
<Program Use="Target" Name="MainProgram" TestEdits="false" MainRo
<Tags/>
<Routines>
<Routine Name="MainRoutine" Type="RLL"/>
</Routines>
</Program>
</Programs>
</Controller>
</RSLogix5000Content>
```
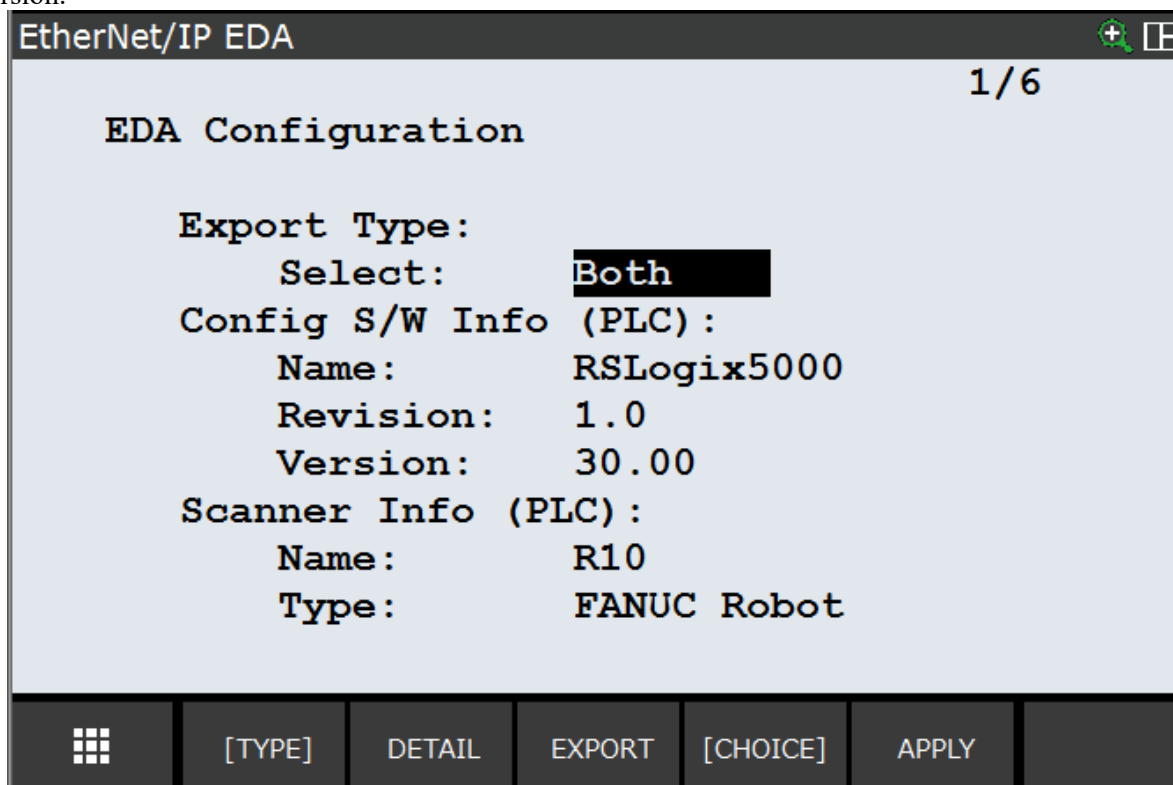
Fig. 10.2.1 (c) Schema Revision

Another mandatory element on this screen is Scanner Info which provides the scanner connection name and method type to be used to create the scanner connection. There are two ways to create a scanner using Generic Ethernet Profile (Catalog name: Ethernet Module) and using Add-on-Profile (AOP) for FANUC Robot (Catalog name: FANUC RobotPlus). If type is set FANUC Robot and you create connection using Ethernet Module after importing, tags won't get updated until some manual changes are done in rungs. Export type has three categories based on connection modes:

- Implicit: add non-I/O data with I/O connection
- Explicit: add non-I/O data with explicit messaging
- Both: add non-I/O data with Implicit and Explicit connections

This screen comes up with default info which can be modified if not up-to-date with the PLC software version.

```
EtherNet/IP EDA                                    🔍 ⊞
                                              1/6

     EDA Configuration

         Export Type:
             Select:        Both
         Config S/W Info (PLC):
             Name:          RSLogix5000
             Revision:      1.0
             Version:       30.00
         Scanner Info (PLC):
             Name:          R10
             Type:          FANUC Robot


   ▦      [TYPE]    DETAIL   EXPORT   [CHOICE]   APPLY
```

Fig. 10.2.1 (d) PLC Configuration Information

The next screen (Fig. 10.2.1(e)) provides information about the file device/path where configuration is saved. File path points to default file path set in file screen. F2, [DEVICE] allows to change device. Please note that F2, [DEVICE] only allows you to change the device NOT the subdirectories. Alternatively, if you wish to save the file to a folder in the device, you need to go to the FILE MENU and create/navigate to the folder and come back to EDA screen. Now you will see the same folder is set in the file path. Pre-created or manually created configuration files can directly be loaded from this screen using F3, [LOAD]. Set the appropriate Device/Path and cursor down to Config File. Press F4, [CHOICE] to see the list of all XML files in the device/path. Now you can select appropriate file to be loaded. Press F3, [LOAD] to load the selected file and cycle power to take effect. You can also create a new configuration file by pointing the configuration file to NONE. When you press F5, [APPLY], then a text box appears to enter new file name. Please note that the file name must have the extension ".XML". This screen also provides the summary of data sizes per connection and connection state. Rung flag is YES by default which means create rungs when EXPORT is done. It can be set YES or NO. If it is set to NO then only UDT file (<Connection Name>DataType.L5X i.e. r10DataType.L5X) is created. No controller tags and rungs will be created. Please note that after creating a new configuration file, it needs to be loaded using F3, [LOAD] and power cycle is required to take effect.

Fig. 10.2.1 (e) EDA Configuration Summary
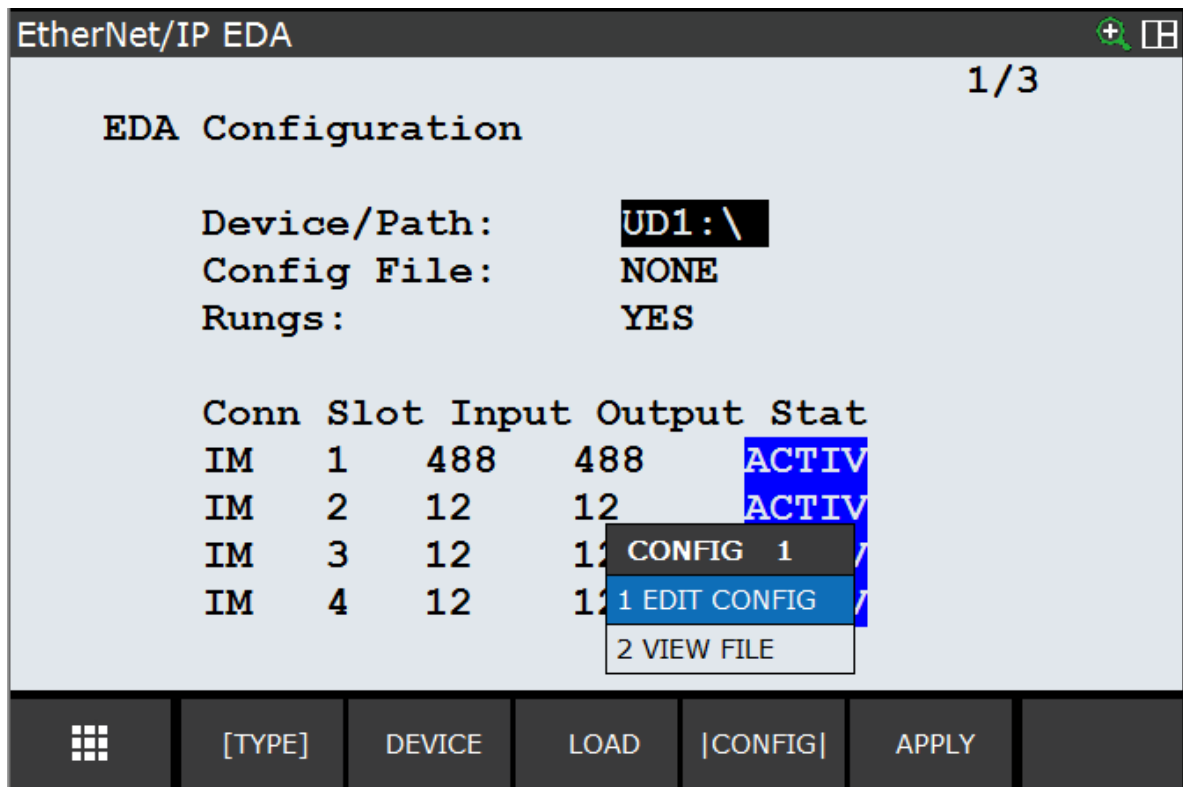
Fig. 10.2.1 (f) Edit/View Configuration File

If you click on EDIT CONFIG in the Fig. 10.2.1 (f), it will prompt (Fig. 10.2.1 (h)) to check if corresponding adapters are set to FALSE you are planning to add non-I/O data on. If you select YES, then it takes to EDIT (CONFIG) screen (Fig. 10.2.1 (j)). Otherwise it stays in summary screen. This screen allows you to insert or modify (limited) or delete any items. Items could be Numeric, String, Position registers, or CURPOS, and KAREL or SYSTEM variables. Please note that SYSTEM variables have read access from this feature. When you insert a SYSTEM variable, it automatically changes to OUTPUT type. CURPOS and SYSVAR can only be OUTPUT type i.e. only reading from the controller is permitted. Maximum of 500 items can be inserted from this screen.

If device selected is not present, you will see below warning and will not be allowed to switch to EDIT screen. So make sure device is plugged in. In Fig. 10.2.1 (g), UD1: is not plugged in.
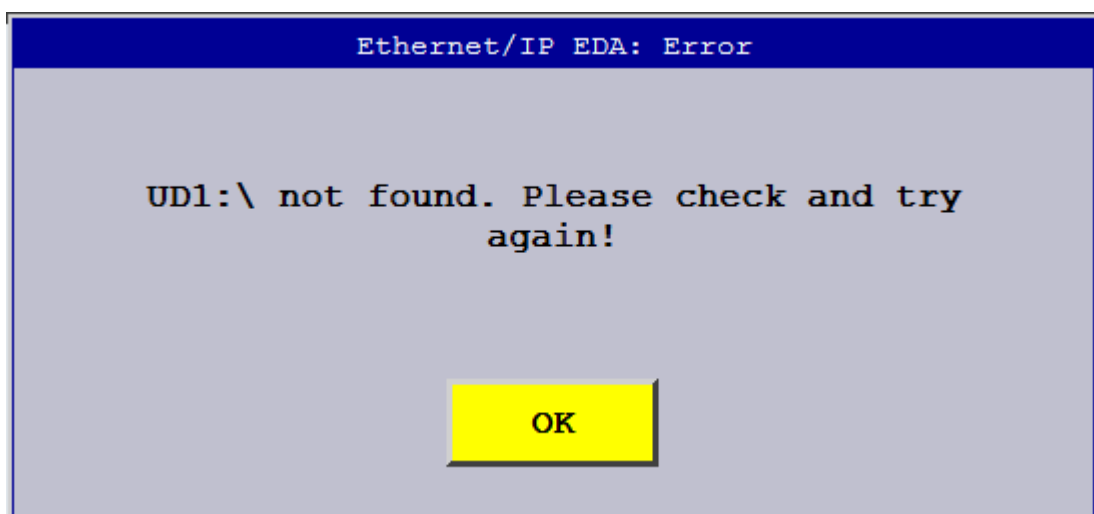
Fig. 10.2.1 (g) Device Check

> ⚠ **WARNING**
> Do not ignore this warning. Otherwise you will not be able to APPLY new configuration and configuration will be lost once you leave the screen without APPLYing.

```
Ethernet/IP EDA: ADP Online Check

APPLY is not allowed if ADP is enabled
(TRUE). Make sure to disable (FALSE) the
one you are adding data to. Is it
disabled?


        YES                 NO
```
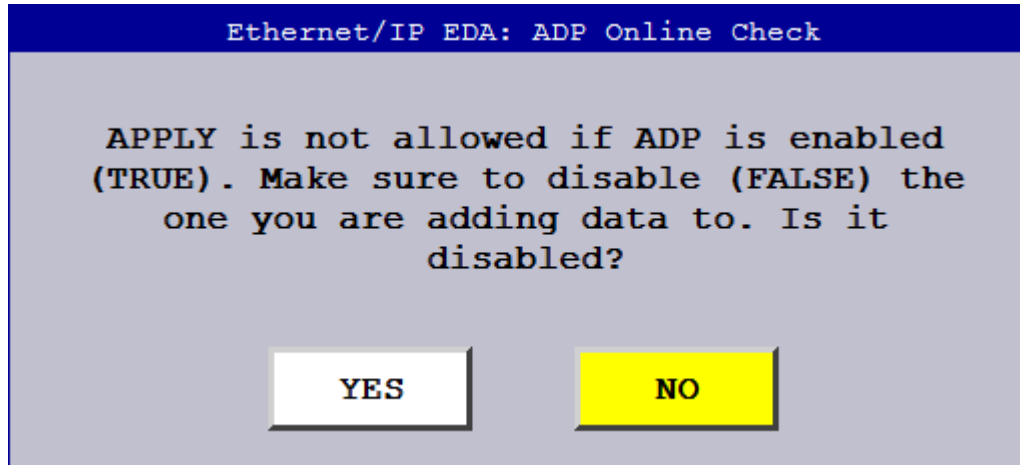
Fig. 10.2.1 (h) Adapter Check

Column definition of each item type is explained in Table 10.2.1.

Table 10.2.1 Column Header Description

Conn	Connection Type: Implicit (IM) or Explicit (EM)
Slt	Slot or Attribute Number
IO	Input (IN) or Output (OUT)
Var	Variable Type: • Numeric Register (R) • String Register (SR) • Position Register (PR) • Current Position (CP) • KAREL Variable (KL) • SYSTEM variable (SV)
Strt	Start Index (applies only to registers)
Totl	Total number of registers (applies only to registers)
Rep	Representation: • Numeric Register (R): 0 (Integer) or 1 (Real) • String Register (SR): Not applicable • Position Register (PR): 0 (Cartesian) or 1 (Joint) • Current Position— CURPOS: 0 (Cartesian) or CURJPOS : 1 (Joint) • KAREL Variable (KL): : Not applicable • SYSTEM variable (SV): : Not applicable
Group	Position Register (PR) and CURPOS: Group number (Not applicable to all others)

> **NOTE**
> Please note that KAREL/SYSTEM program and variable columns are self descriptive when inserted. Strt column header represents program name in case KAREL or SYSTEM variables and next field represents variable name.

> **NOTE**
> Please do NOT add data when ADP is set to TRUE.

If you apply new configuration just added and corresponding ADP is set to TRUE then you will see below prompt and won't be able to apply the configuration. If you leave the screen without APPLYing configuration, you will lose all new configurations. So better you pay attention to the warning you got just before switching to EDIT screen. For example, if you are trying to add NUMREG to slot #1 (IM) then ADP #1 must be set to FALSE before anything can be added (and applied) to it. Main purpose of this is to avoid adding data in running ADP which can lead to controller crashes. Please note that this warning is generated only for implicit connections.
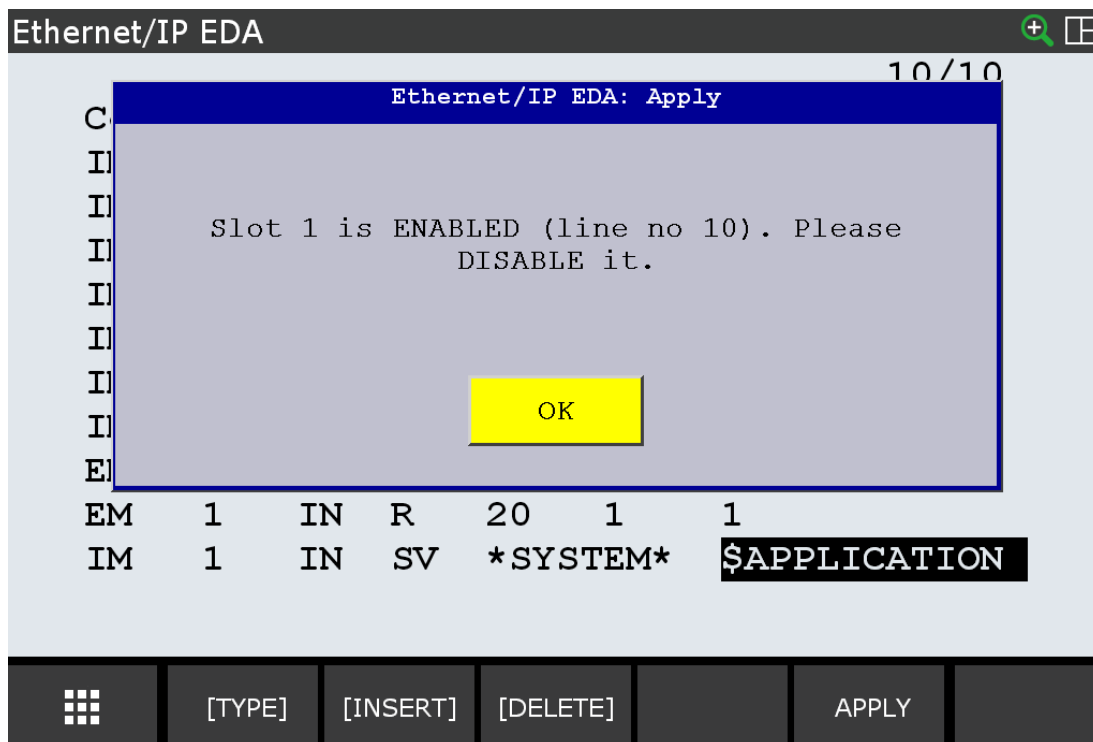
Fig. 10.2.1 (i) Slot Enabled
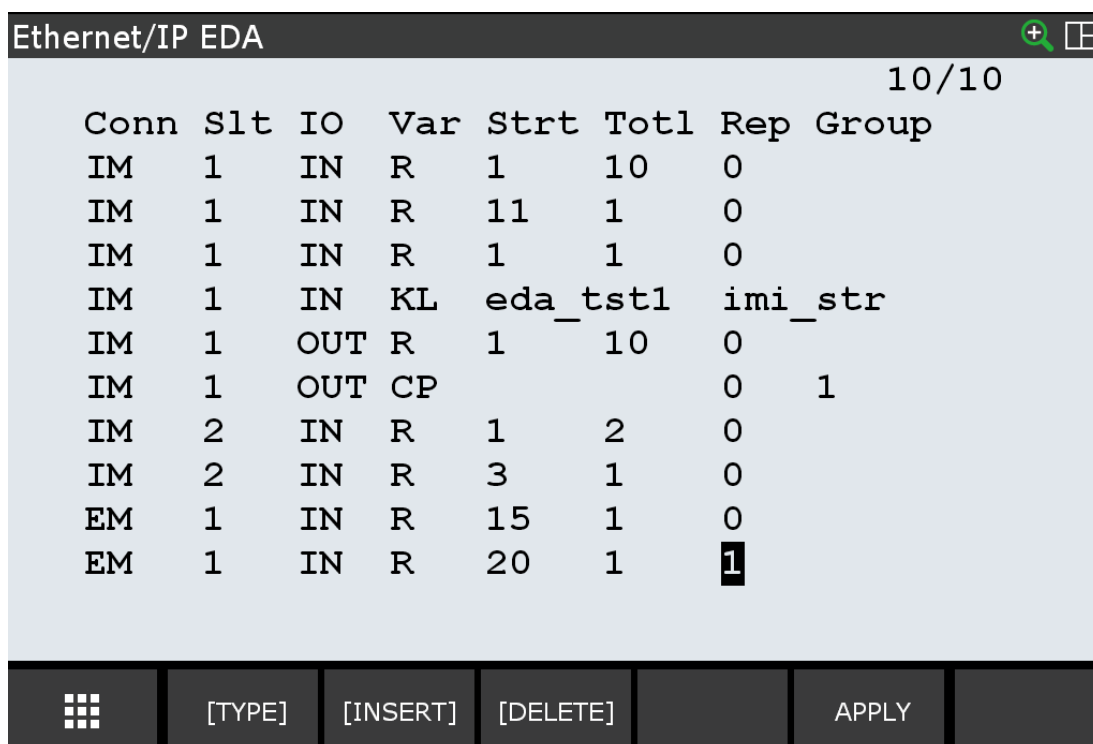
Fig. 10.2.1 (j) EDIT Screen

You can hit F2, [INSERT] to insert items using pull-up menu as shown in Fig. 10.2.1(k). When KAREL variables are inserted/changed, you need to enter program and variable names manually. In case of SYSTEM variables, program names appear automatically and are not editable. Variable names need to be entered manually in this as well. If variable is wrong, it is prompted during APPLY process. Please note that RSLogix5000 doesn't allow tag names more than 40 characters long. So if variable names exceed this limit, names are shortened automatically. Please refer Section 10.3.3 for details.
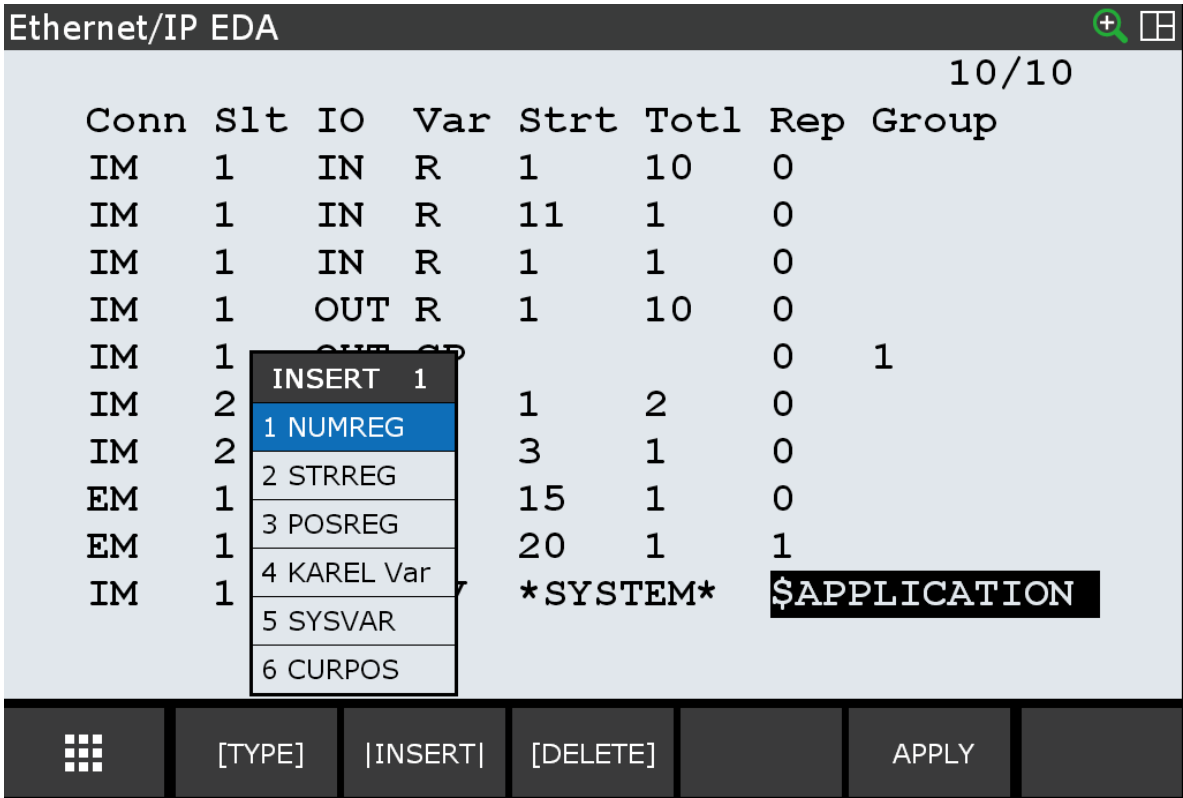
Fig. 10.2.1 (k) INSERT Items

You get to this screen (Fig. 10.2.1 (l)) when you try to change the slot number. Slot number and associated structure can be changed at this screen. If no structure name is provided, then name defaults to SLOT<slot number> or ATTR<attr number> and final structures becomes R10_SLOT1_T and R10_ATTR1_T for implicit or explicit connection respectively where R10 is configured connection name.

Fig. 10.2.1 (l) Slot Number and Structure Name

Similarly user gets to this screen (Fig. 10.2.1 (m)) when s/he tries to change IN/OUT through F4, [CHOICE] key. Item can be added to Input or Output per choice in the screen. Also, user has choice to configure structure name for input/output structures. If user does not provide structure name, it defaults as follows:

- Implicit: INPUT<slot number> or OUTPUT<slot number> but final structures are constructed using this string and connection name to make it unique per robot. For example, if connection name configured is R10 then final structures becomes R10_INPUT1_T and R10_OUTPUT1_T for input and output respectively.
- Explicit: EM_INPUT<slot number> or EM_OUTPUT<slot number>. In this case, final structures become R10_EM_INPUT1_T and R10_EM_OUTPUT1_T for input and output respectively.

Fig. 10.2.1 (m) I/O Type and Structure Name

Once slot and I/O configuration is done hit F2, [SAVE] or PREV key which takes back to configuration screen.

Similarly any item or set of items can be deleted from existing configuration. F3, [DELETE] allows to mark items for deletion (Fig. 10.2.1 (n). Items are marked with red background in Var column as shown in Fig. 10.2.1 (o). Also, user can unmark items which marked accidentally. Once all items are marked for deletion, then you can delete marked set.
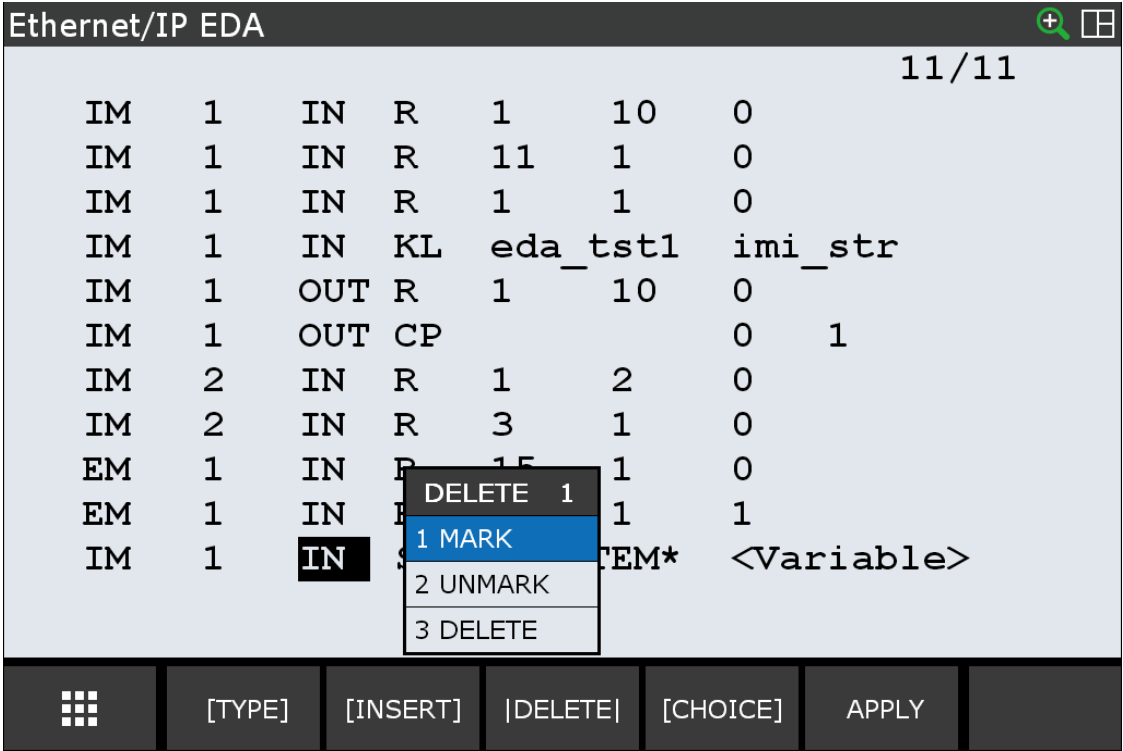
Fig. 10.2.1 (n) DELETE Items
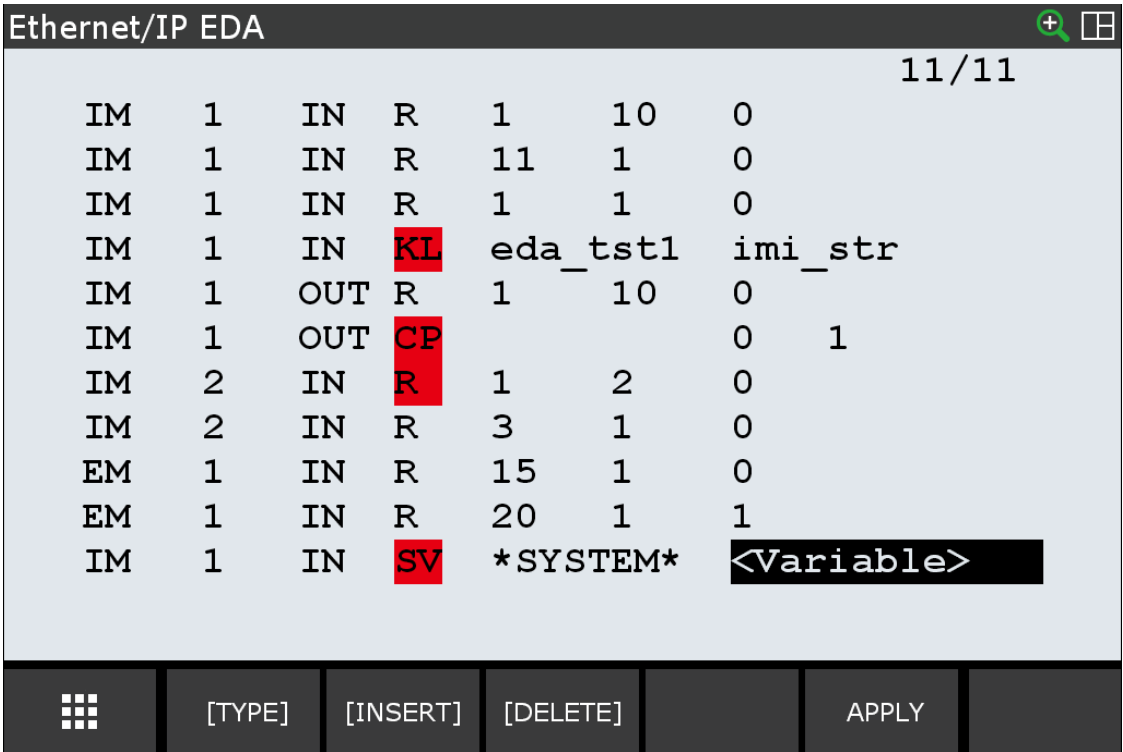
Fig. 10.2.1 (o) Marking Items for Deletion

Once configuration is done, you MUST apply all changes. You can save last file as <file_name>.BCK in same directory by hitting YES in prompt box (Fig. 10.2.1(p)).
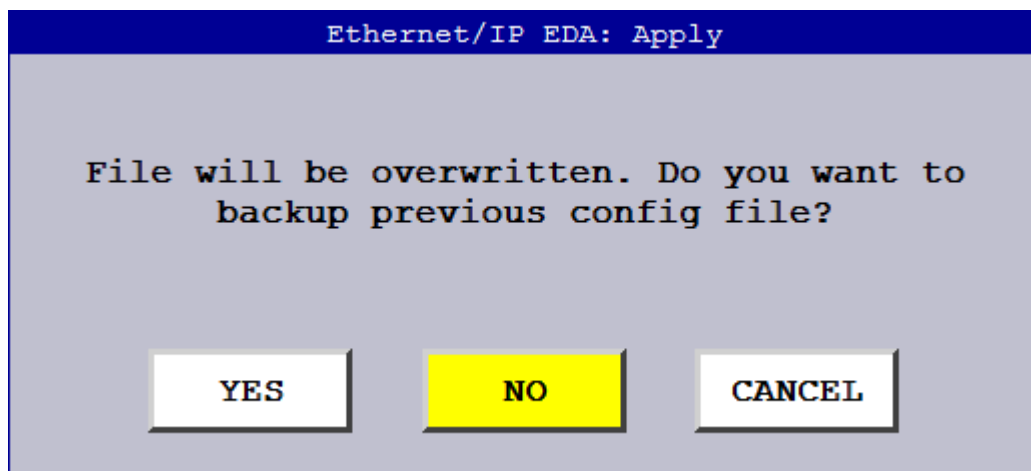
Fig. 10.2.1 (p) Applying Changes

If any item is invalid then it is prompted to show the line number as shown in Fig. 10.2.1 (q). This error shows that program or variable name is invalid in line number 13. This error also appears when read only variable is configured for input. Please note that errors are prompted one-by-one. Once all errors are corrected message shown in Fig. 10.2.1 (p) appears if existing configuration is modified. Otherwise you will be prompted to enter new file name as shown in Fig. 10.2.1 (u).
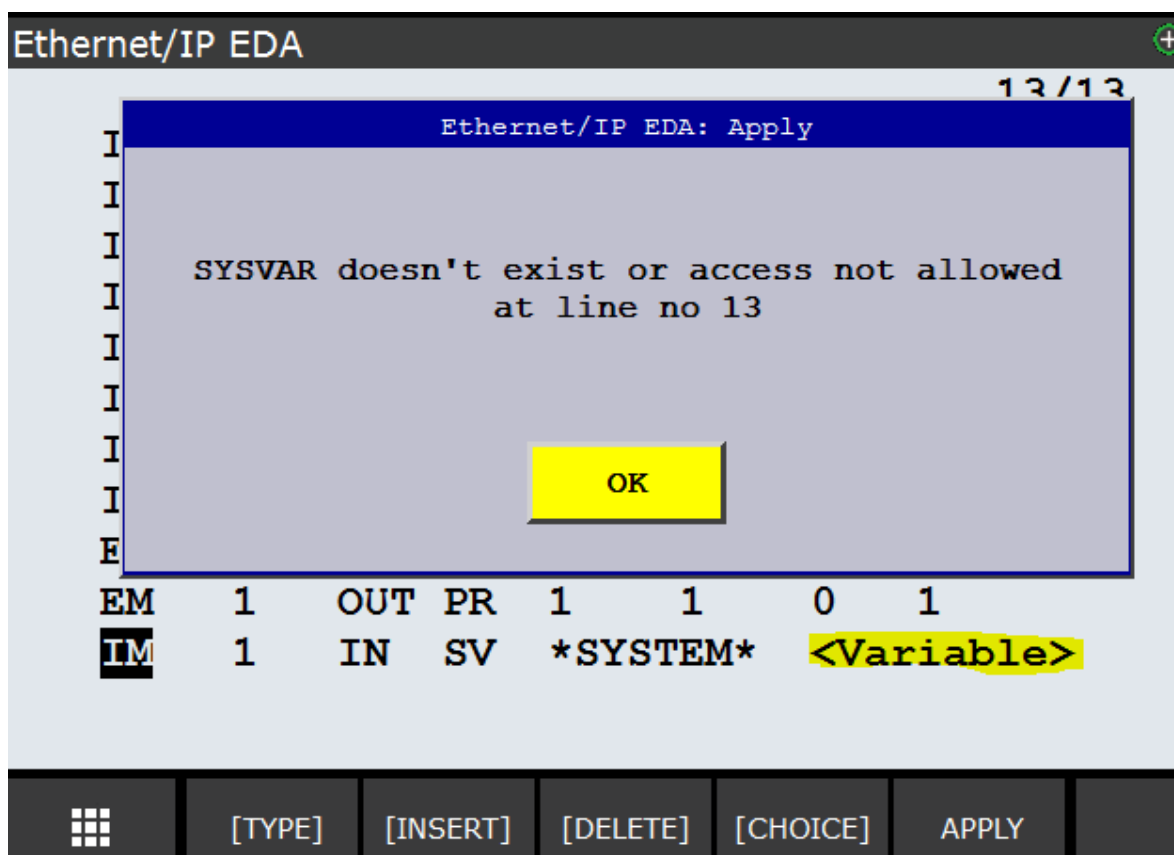
Fig. 10.2.1 (q) Invalid Item

Similarly when user tries to apply without selecting the KAREL program name and KAREL variable name, this error appears (Fig. 10.2.1 (r)).
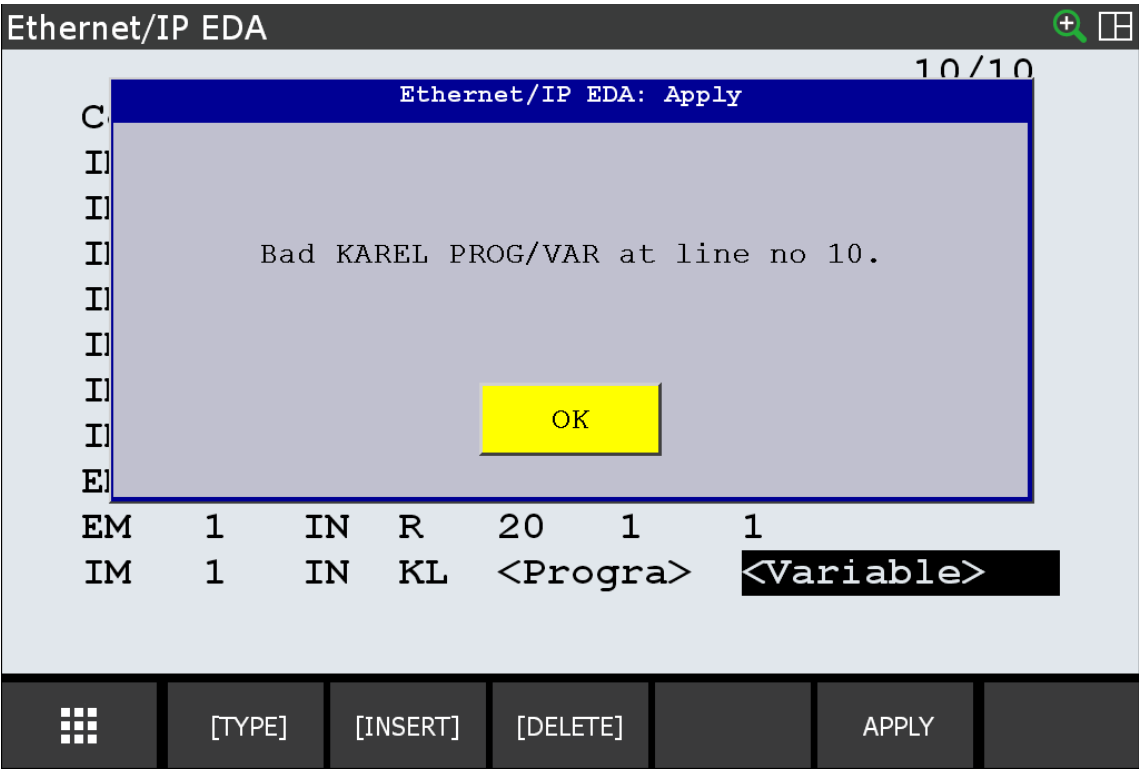
Fig. 10.2.1 (r) Invalid KAREL Program Name/Variable Name

Currently RockWell PLC does not support multidimensional arrays. This is warning to remind you to check if support has been added in your version. Otherwise this will lead to import error in RSLogix 5000.
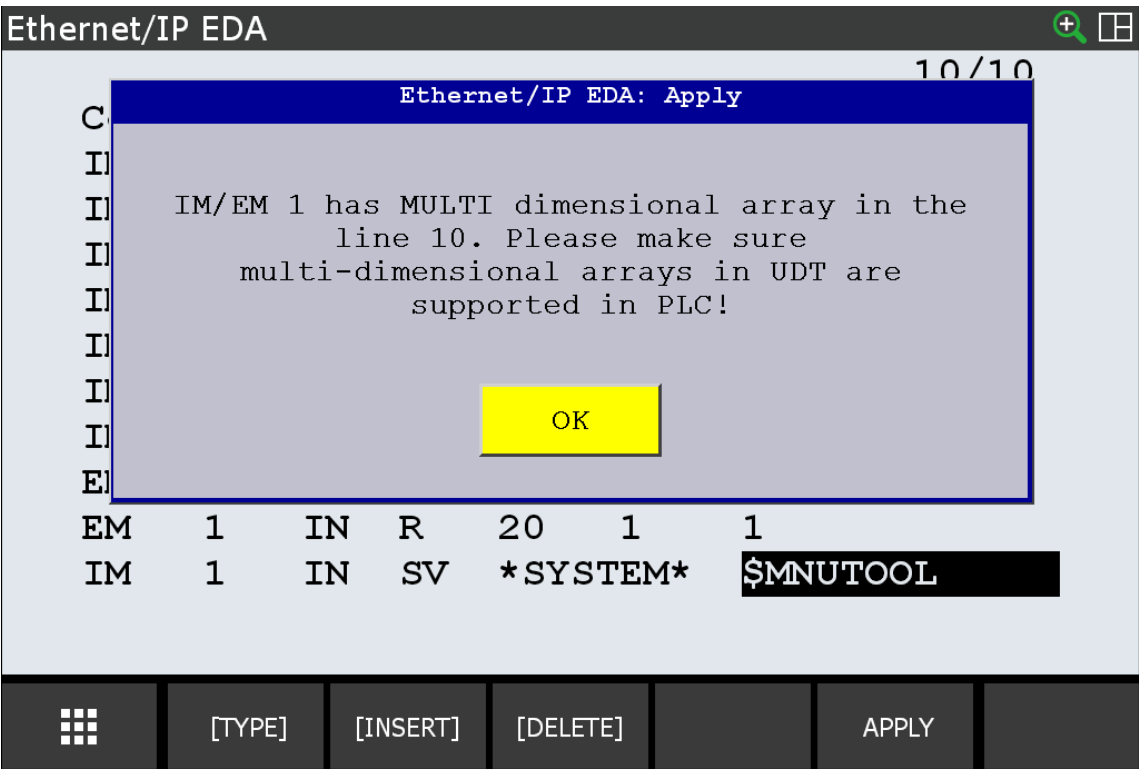
Fig. 10.2.1 (s) Multi-Dimensional Arrays

There are limited data types are supported in EDA. If unsupported data type is added in configuration, you will see below warning and item will be skipped in APPLY process.
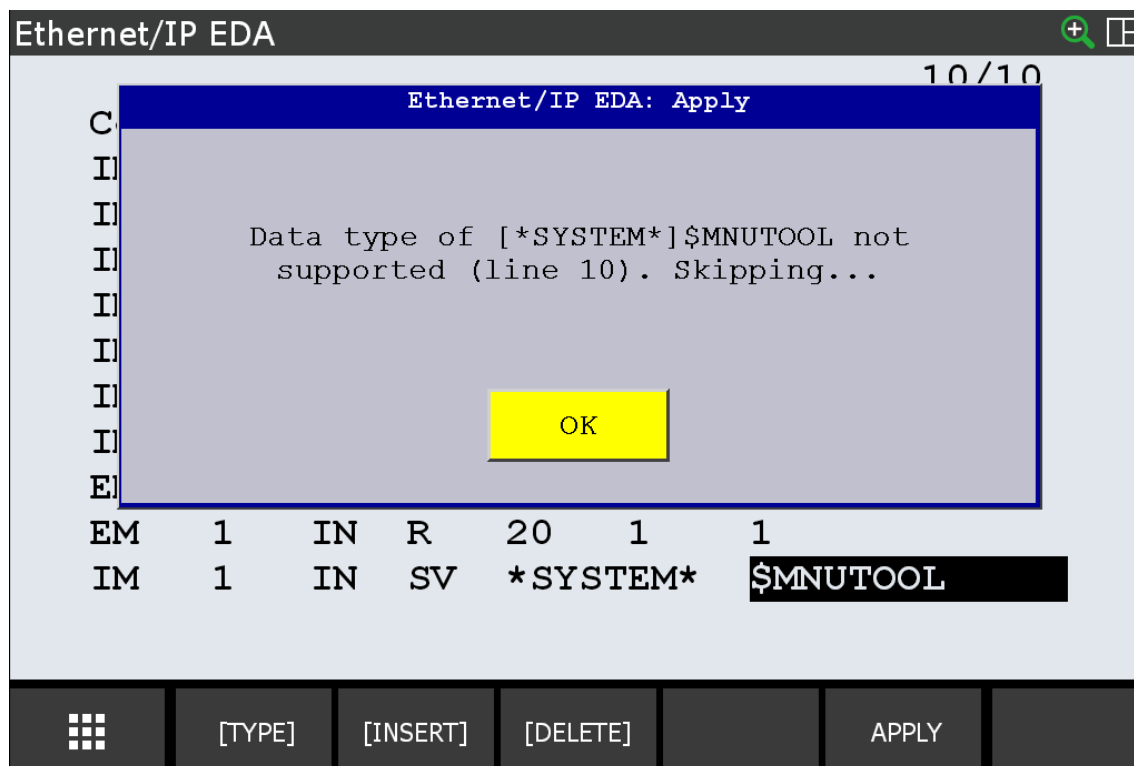
Fig. 10.2.1 (t) Unsupported Data Type

Now you are prompted to enter new file name as shown in Fig. 10.2.1 (u) if configuration file is pointing to NONE in summary screen. New file is set as default configuration going forward.
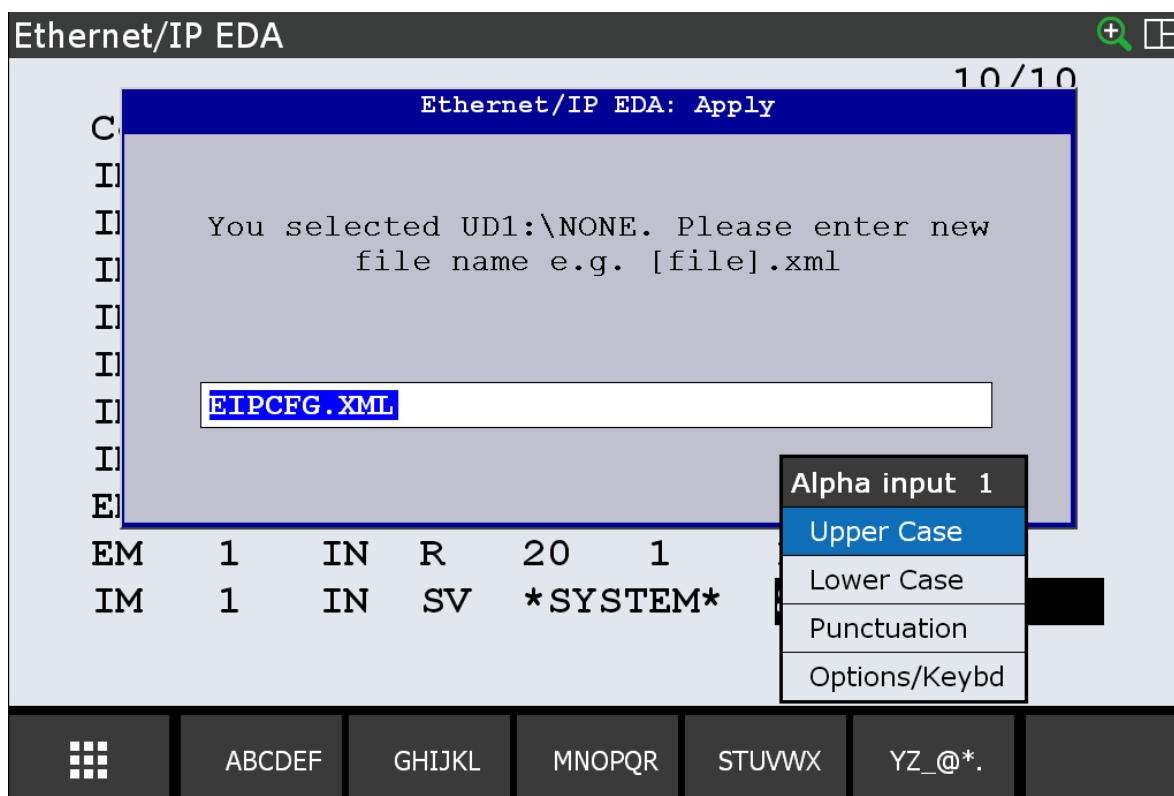
Fig. 10.2.1 (u) New File Name

- 115 -

If apply is successful, you MUST load recent configuration file if you want to use it and cycle power as advised in Fig. 10.2.1 (v).
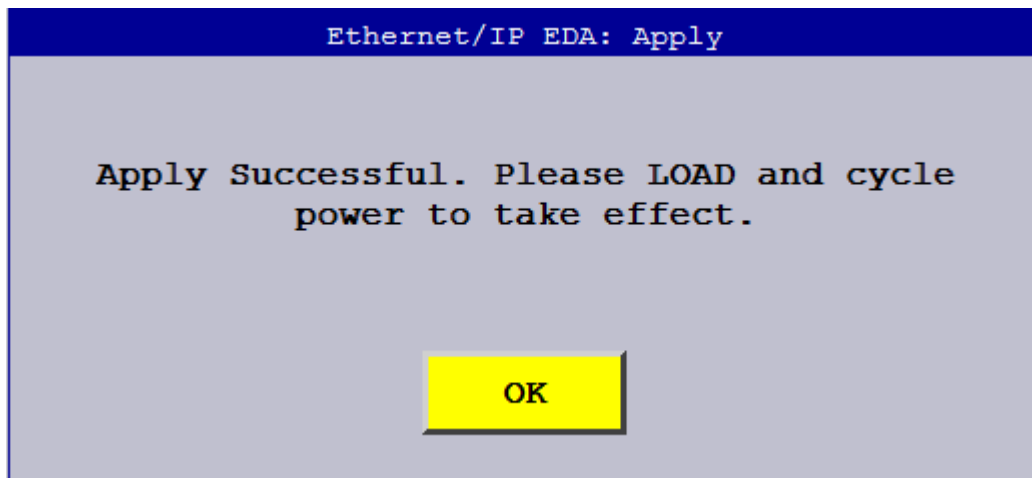
Fig. 10.2.1 (v) Apply Successful

Please note that if configuration size exceeds the allowed limits per connection, Stat column marks the corresponding connection in INVAL (red). Valid configuration size is marked with PEND (yellow) if existing connection is modified, otherwise ACTIVE (blue) as shown in Fig. 10.2.1 (w).
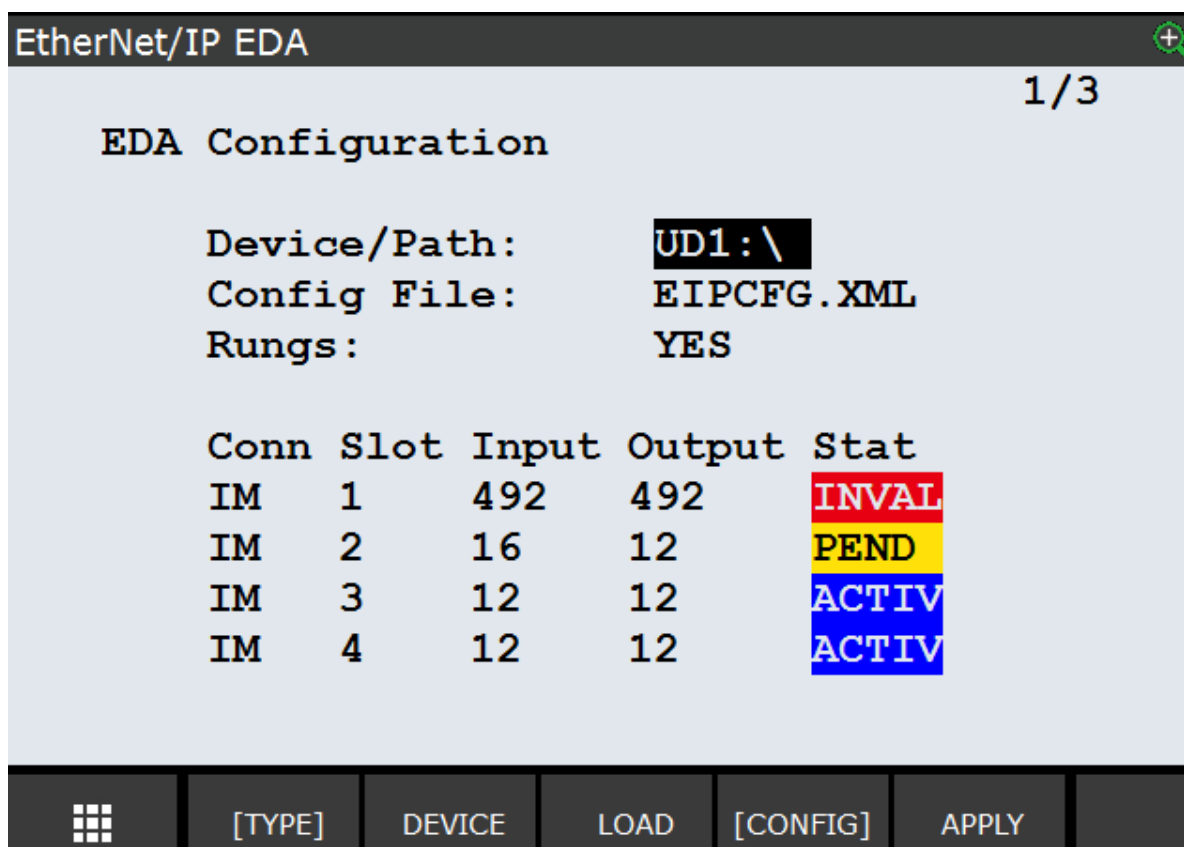
Fig. 10.2.1 (w) Stat Definition

Click on VIEW FILE in the Fig. 10.2.1 (f), it displays the configuration file in XML format as shown in Fig. 10.2.1 (x). Header shows the file location e.g. UD1:¥EIPCFG.XML. Press F3, [EXIT] to return to configuration summary.
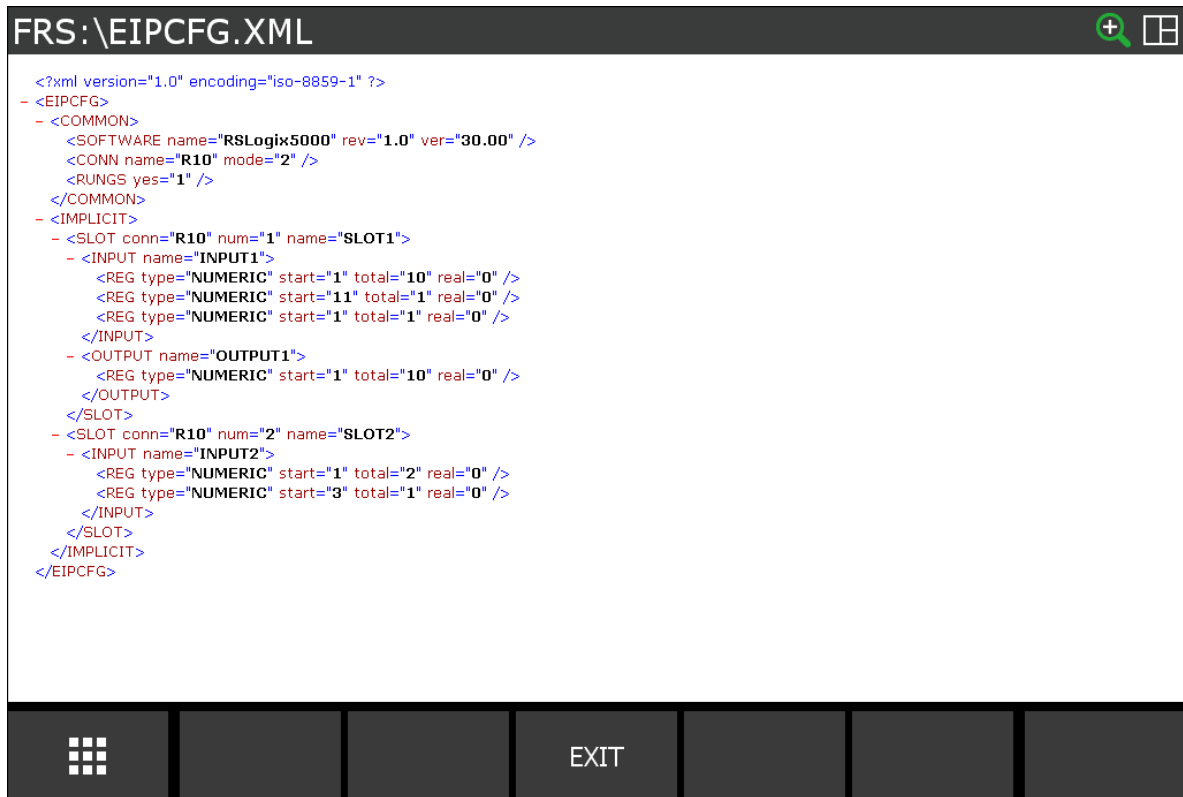
Fig. 10.2.1 (x) Display Configuration File

10.2.2   Configuration using PC Text Editor

Tag definition is given in Table 10.2.2.

Table 10.2.2 Tag Definitions

Tag Name	Tag Parent	Tag Depth	Tag Description
<EIPCFG>	None	0	This is root tag of the configuration file
<COMMON>	<EIPCFG>	1	Contains common information about PLC software, structure name, and connection name

<SOFTWARE .../>	<COMMON>	2	This tag has attributes name, ver, and rev to specify software name, version, and schema revision respectively. For example,
			`<SOFTWARE name="RSLogix5000`
			`                  rev="1.0" ver="20.1" />`

<CONN .../>	<COMMON>	2	This tag provides connection name. This MUST match with connection name created in the PLC. This has two attributes name which is scanner connection name when you create scanner connection in PLC. Other attribute is mode which is method type for creating scanner connection. RSLogix5000 offers two ways to create scanner now: using generic Ethernet profile (Ethernet Module) which is mode 1 and FANUC AOP (FANUC Robot) which is mode 2.For example, `<CONN name=" R10"  mode="1"/>`

- 117 -

Tag Name	Tag Parent	Tag Depth	Tag Description
<RUNG .../>	<COMMON>	2	This tag allows rungs creation including controller tags. For example, `<RUNG yes="1" />`
<IMPLICIT>	<EIPCFG>	1	This tag includes configuration for implicit connection.
<EXPLICIT>	<EIPCFG>	1	Same as <IMPLICT> except it has configuration information for explicit connection.

<SLOT>	<IMPLICIT>	2	Carries slot number and variable configuration to be included in structure. For example,
			`<SLOT conn="R10" num="1"`
			`      name="slot1>`

<ATTR>	<EXPLICIT>	2	Same as <SLOT> except it has configuration information for explicit connection. For example,
			`<ATTR num=4" name="attr1"`
			`      class="6E" inst="1">`

<INPUT>	<SLOT> or <ATTR>	3	This tag carries input configuration information. It has input structure name as its attribute. For example, `<INPUT name="in struct">`

<REG ..../>	<INPUT> or <OUTPUT>	4	This tag defines register configuration for NUMERIC, STRING, and POSITION registers. For example,
			`<REG type="NUMERIC" start="1"`
			`      total="5" real="0" />`
			`<REG type="STRING" start="5"`
			`      total="2" />`
			`<REG type="POSITION" start="1"`
			`      total="2" jpos"0"`
			`      group="1" />`
			`<REG type="CURPOS" jpos"0"`
			`      group="1" />`

<KAREL .../>	<INPUT> or <OUTPUT>	4	Contains KAREL or SYSTEM variable information.
			For example,
			`<KAREL var="recipe"`
			`      prog="prod1" />`
			`<KAREL var="$fast_clock"`
			`      prog="*SYSTEM*" />`

Below is sample configuration file.

> ⚠ **WARNING**
> Please do not copy this file directly from this document. Sometimes some hidden characters get copied and causes configuration failure. You can do MD: backup and find sample eipcfg.xml which you can modify and load back.

Sample Configuration File (EIPCFG.XML)

```xml
<?xml version=" 1.0"  ?>
<EIPCFG>
  <!- Common Information for both implicit and explicit connection. - >

  <COMMON>
    <!- PLC software information ->

    <SOFTWARE name="RSLogix5000" rev="1.0" ver="20.1" />
    <!- Connection name, PLC program name, PLC routine name
                      will be based on following name. - >

    <CONN name=" R10"  mode="1"/>
      <!- Would you want to create rungs for this configuration? - >

      <RUNG yes=" 1"  />
  </COMMON>
  <!- Configuration data for implicit connection for all slots - >

  <IMPLICIT>
    <!- Configuration info for specific slot - >

    <SLOT conn=" R10" num=" 1"  name=" slot1_struct" >
      <!- Input configuration information - >

      <INPUT name=" im_in" >
        <!- One Numeric Register, Integer, starting from 1 - >

        <REG type ="NUMERIC" start="1" total="1" real="0"/>
        <!- Two String Registers starting from 1 - >

        <REG type ="STRING" start="1" total="2"/>
        <!- Two Position Registers in Cartesian
                        starting from 4 from group 1 - >

        <REG type ="POSITION" start="1" total="2" jpos="0" group="1"/>
        <!- KAREL Variable name and Program name - >

        <KAREL name="in_recipe" prog="prod1" />
        <!- Five Numeric Registers, real, starting from 2 - >

        <REG type ="NUMERIC" start="2" total="5" real="1"/>
      </INPUT>
      <!- Out configuration details - >

      <OUTPUT name=" im_out>
        <REG type ="NUMERIC" start="7" total="1" real="0"/>
        <REG type ="STRING" start="15" total="2"/>
        <REG type ="POSITION" start="10" total="2" jpos="0" group="1"/>
        <KAREL name="out_recipe" prog="prod1" />
        <REG type ="NUMERIC" start="8" total="5" real="0"/>
      </OUTPUT>
    </SLOT>
  </IMPLICIT>
  <!- Configuration for explicit connection - >

  <EXPLICIT>
    <!- Configuration for attribute 1 - >
    <ATTR num=" 1"  name=" attr1_struct"  class=" 6E"  inst=" 1" >
      <INPUT name=" em_in" >
        <REG type ="NUMERIC" start="13" total="1" real="0"/>
        <REG type ="STRING" start="22" total="2"/>
        <!- Two Position Registers in Joint mode starting
                        from 3 from group 2 - >

        <REG type ="POSITION" start="3" total="2" jpos="1" group="2"/>
        <!- CURPOS in Cartesian mode from group 1 - >

        <REG type ="CURPOS" jpos="0" group="1"/>
```

```
       <KAREL name="in_recipe" prog="prod2" />
       <REG type ="NUMERIC" start="14" total="5" real="0"/>
    </INPUT>
   <OUTPUT name="em_out>
     <REG type ="NUMERIC" start="19" total="1" real="0"/>
     <REG type ="STRING" start="15" total="2"/>
     <REG type ="POSITION" start="5" total="2" jpos="0" group="1"/>
     <KAREL name="out_recipe" prog="prod2" />
     <REG type ="NUMERIC" start="20" total="5" real="0"/>
   </OUTPUT>
  </ATTR>
  <!- Configuration for attribute 2 - >

  <ATTR num=" 2"  name=" attr2_struct"  class=" 6E"  inst=" 1" >
    <OUTPUT name=" em_sys_out>
      <!- System variable information - >
      <KAREL name="$APPLICATION" prog="*SYSTEM*" />
    </OUTPUT>
   </ATTR>
  </EXPLICIT>
</EIPCFG>
```

10.3   Structure Names and Controller Tags

10.3.1   Structure Hierarchy

Advance hierarchy (Fig. 10.3.1 (a)) reduces the number of elements against to legacy construction. This feature is enabled by default. This feature permits you to utilize full allowable I/O and 511 non-I/O data. . However 511 limits can never be used as maximum elements allowed in the configuration screen (EDIT) is 500 which is more than enough for any application. Each INPUT or OUTPUT structure contains DIN or DOUT structure which carries all I/O data. Each of these structures contains multiple DWORD structures where each DWORD is 32 elements structure and each element is one I/O point. So each DWORD carries 32 I/O points.
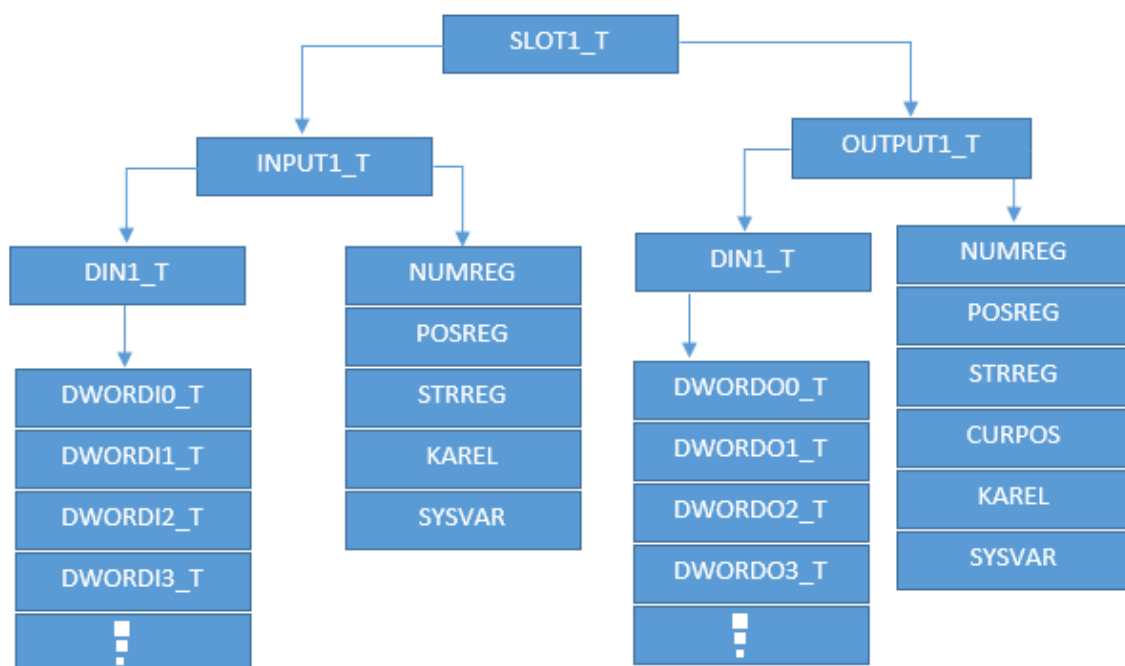
Fig. 10.3.1 (a) Advanced Hierarchy

In order to use legacy style hierarchy, please set $EIPCFG.$OPTIONS = 1024 and cycle power to take effect. This is default behavior in previous generation controllers. Legacy hierarchy has all structure members under INPUT or OUTPUT at same level. Each bit in I/O is counted towards one structure element. Each I/O point needs to be an element because element name is derived from I/O comment. Making I/O an array will lose comment names. All non-I/O data is one structure for each data. The downside of this approach is higher number of structure elements for larger I/O size which might hit the Rockwell's hard limit of 512 elements per structure. In other words, any user defined structures in Rockwell world can have maximum of 512 elements in single structure. Below in Fig. 10.3.1 (b) is hierarchy for legacy structure construction. For example, if INPUT configured is 60 bytes and number of numeric registers on slot 1 is 20 then total elements in INPUT structure become 60 x (8 + 1) + 20 = 560 which is over 512. So import will fail for this configuration. Ultimately I/O size needs to be reduced to accommodate other non-I/O data.
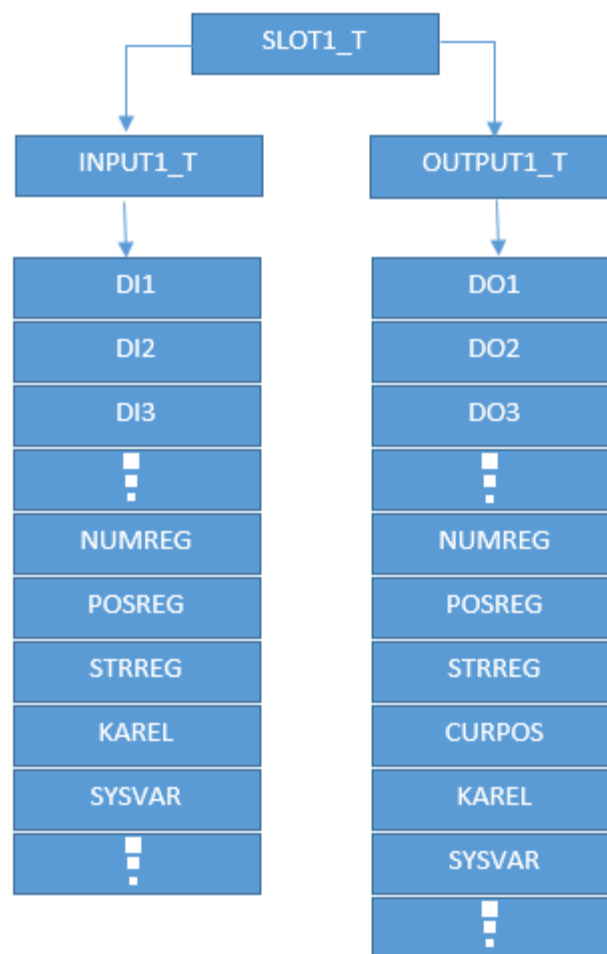
Fig. 10.3.1 (b) Legacy Hierarchy

10.3.2   Structure Names

Names configured in the slots or attributes are structure names per slot or attributes basis. Format of each structure is <conn name>_<strcut name>_T. For example, if <SLOT conn="R10" num="1" name="slot1"> then structure name for slot 1 is R10_SLOT1_T. System changes case to upper and appends "_T" just to differentiate from controller tag names. These structures are reusable for any controller connection if same UDT has to be used. If names are not configured or not provided in configuration file, system provides default names "SLOTx" or "ATTRx" for implicit or explicit respectively. Please see all default structure names in Table 10.3.2.

Table 10.3.2 Default Structure Names

Category	Implicit Structure (with conn name R10 and slot #1)	Explicit Structure (with conn name R10 and attr #1)
Slot	R10_SLOT1_T	R10_ATTR1_T
Input	R10_INPUT1_T	R10_EM_INPUT1_T
Output	R10_OUTPUT1_T	R10_EM_OUTPUT1_T
I/O Input	R10_DIN1_T	N/A
I/O Output	R10_DOUT1_T	N/A
Input DWORD	R10_DWORDI1n_T where is member # in R10_DIN1_T starting from 0.	N/A
Output DWORD	R10_DWORDO1n_T where is member # in R10_DOUT1_T starting from 0.	N/A
Numeric Register	DINT (RockWell double integer type)	Same as in Implicit
String Register	STRING (RockWell STRING data type)	Same as in Implicit
Position Register	POSREG_T for Cartesian and POSREGJ_T for joint representation type	Same as in Implicit
Current Position	Same as Position register types	Same as in Implicit
KAREL/SYSVAR	These structures are as same as defined in Robot controller	Same as in Implicit

10.3.3   Controller Tag Name

Controller tag names are derived from connection name and slot/attr numbers. For example if connection name is "R10" then tag name for slot 1 and attr 1 after importing file into PLC becomes r10_s1 for implicit connection and r10_a1 for explicit connection respectively. Tag name for input and output data are predefined names in and out respectively. In advanced hierarchy, there are some additional fixed tag names like din, dout, dowrdi<n>, dowrdo<n> where n is dword index starting from 0. There are some extra controller tags created to control explicit connection as defined in Table 10.3.3

Table 10.3.3 Explicit Connection specific Controller Tags

Tag Type	Tag Name
ON/OFF flag	r10_a1_tcoil
Timer	r10_a1_timer
Message Block for Write	r10_a1_msgout
Message Block for Read	r10_a1_msgin

Long Variable Names: RockWell allows only 40 characters long tags. So if variable names are more than 40 characters, robot shrinks the names keeping first and last part of the name. Middle parts are made LxLx... where *x* is last digit in the part. In case of array, it represents array index (if single digit otherwise 1's place of index) and level number (if structure is not numbered, otherwise numbering itself. Single digit rule applies as in arrays) in case of structure. For example, if variable names are
line1[1].wind5orzdt[2].wind6orzdt[2].wind7orzdt[2].wind8orzdt[2].wind9orzdt[2].itp1, ,
line2[1].wind5orzdt5.wind6orzdt4.wind7orzdt3.wind8orzdt2.wind9orzdt1.itp2 and
line3[1].wind5orzdt.wind6orzdt.wind7orzdt.wind8orzdt.wind9orzdt.itp3 then resulting variable names
become line11_L2L2L2L2L2_itp1, line21_L5L4L3L2L1_itp2, and line31_L1L2L3L4L5_itp3.

10.3.4   Registers Name

Register names are composed based on comments and register notations. For example, tag name for R[1] with comment "data rate", becomes data_rate_R1. Similarly tag names for String and Position registers are created.

Table 10.3.4 Registers Naming Convention

Registers	Comments	Tag Names
R[4]		R4
R[7]	delay min	delay_min_R7
SR[8]		SR8
SR[10]	error name	error_name_SR10
PR[2]		PR2_G1 (for group 1)
PR[6]	origin two	origin_two_PR6_G2 (for group 2)

10.3.5   I/O Name

I/O names are also created based on comments and mapping. When I/O is mapped, comments are considered in following order GIO > UOP > DIO. Please refer Table 10.3.5 for example.

Table 10.3.5 I/O Naming Convention

I/O		Mapping		Tag name

DI[1–8]	comment1–8	GI[1]	data rate	```data_rate_DI1_GO1_1```
				```data_rate_DI2_GO1_2```
				```data_rate_DI3_GO1_4```
				```data_rate_DI4_GO1_8```
				```data_rate_DI5_GO1_16```
				```data_rate_DI6_GO1_32```
				```data_rate_DI7_GO1_64```
				```data_rate_DI8_GO1_128```

DO[9–16]		GO[1]		```DO9_GO1_1```
				```DO10_GO1_2```
				```DO11_GO1_4```
				```DO12_GO1_8```
				```DO13_GO1_16```
				```DO14_GO1_32```
				```DO15_GO1_64```
				```DO16_GO1_128```

DI[17]	stop bit	UI[17]	Panel Input	Panel_Input_DI17_UI17
DO[17]	start bit	UO[1]	Cmd Enabled	Cmd_Enabled_DO17_UO1

10.3.6   KAREL/SYSTEM Variable Name

KAREL and SYSTEM variable names are used same as in the controller.

10.3.7   Current Position (CURPOS)

This variable represents robot current position in configured group and representation. In KAREL terms, it is CURPOS and CURJPOS. Size of the position is same as corresponding position registers.

10.4   10.4 Loading Configuration File

Go to main Ethernet/IP screen (MENU >> I/O >> Ethernet/IP). Now press F5, [EDA]. Press F2, [DETAIL] that gets to screen shown in Fig. 10.2.1(w). This screen comes up with default file device. Select the device using F2, [DEVICE] if file is not in default file device. Now select the appropriate configuration file. Press F3, [LOAD] to load configuration file. It prompts a message to cycle power to take effect on success. Otherwise appropriate error message gets prompted. This file can be loaded from File Menu as well but file name MUST be eipcfg.xml. It also can be loaded through FTP to MD: device.

10.5   Exporting Configuration File for PLC

Export process creates importable configuration file for Rockwell PLC. However only RSLogix5000 v20 or higher version supports configuration via xml file. Please make sure Robot configuration file is loaded successfully and power cycle is complete as described in Section 10.2. Go to main EDA screen and choose appropriate export type as shown in Fig. 10.5 (a). Now hit F3, [EXPORT], PLC configuration file gets created in default file device on success. Otherwise appropriate error message pops up as shown in Fig. 10.5 (c).

There are three export categories to export configuration file. If user wants to create configuration for only implicit or explicit or both connections.

Fig. 10.5 (a) Export Type

If export operation is successful, it is displayed in TP line as shown in Fig. 10.5 (b). Configuration sizes can be seen in summary screen per connection basis for all configuration including implicit and explicit connections. Generated export file is named as <Connection Name>Program.L5X if rungs creation is configured. Otherwise it is named <Connection Name>DataType.L5X. For example, if connection name is R10 then file name is R10Program.L5X or R10DataType.L5X for with rungs or without rungs respectively.

Fig. 10.5 (b) Export Operation

Export error in Fig. 10.5 (c) (Illegal port number) describes that I/O is not mapped correctly. So please go to digital I/O screen and make sure I/O is properly mapped for configured I/O points. If EDA configuration is wrong, it is caught during APPLY process unless configuration file is manually created and loaded. However even file is manually created and loaded, during power bad variables are ignored. So it is very unlikely if any wrong variable show up at this stage. TP configuration does not allow invalid variable names.

Fig. 10.5 (c) Export Error

Configuration size error only occurs if user ignores stat information in summary screen as shown (INVALID in RED) in Fig. 10.2.1 (w). So if you ignore this error, then you will see export error as in Fig. 10.5 (d).
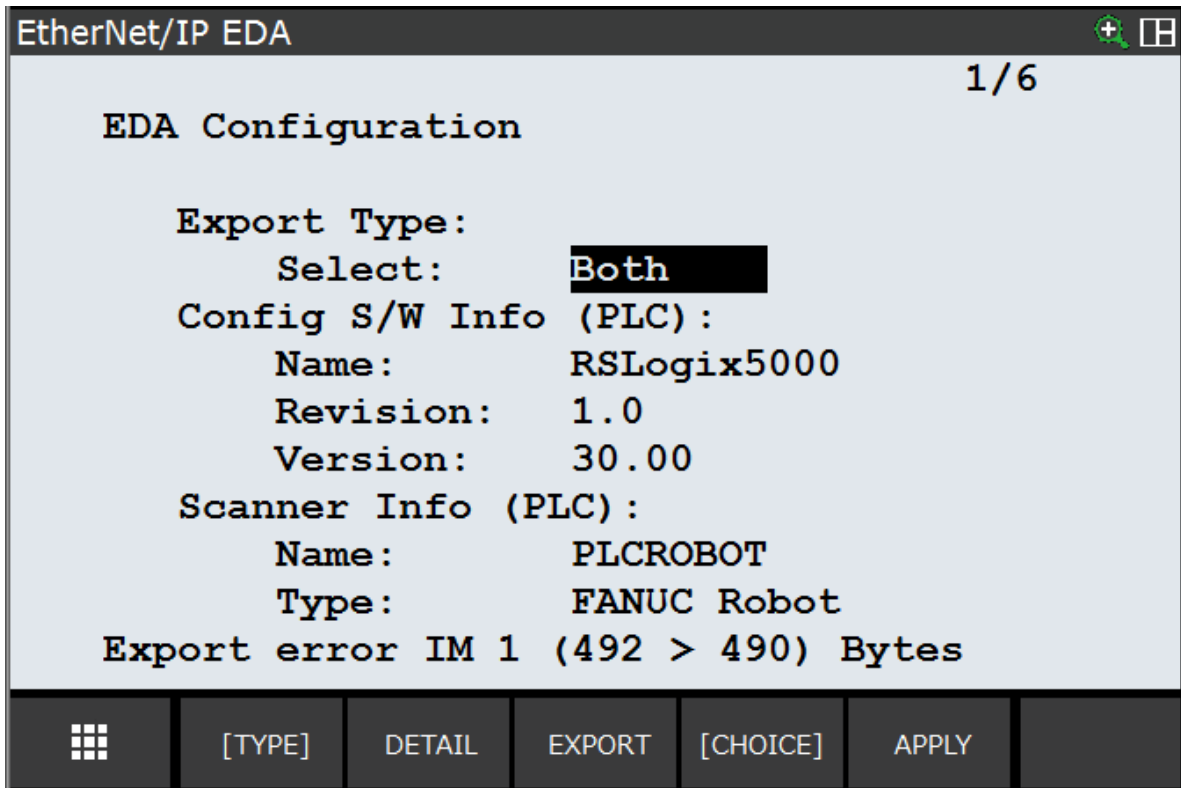
Fig. 10.5 (d) Data Size Exceeded

10.6 Importing Export File in PLC Config Software (e.g. RSLogix5000)

After the export file is successfully created as explained in Section 10.5, transfer it to a PC in which the PLC configuration tool is loaded. Please make sure RSLogix 5000 version is v20 or higher.

Follow the below steps to import configuration file generated by robot controller. These steps are described using RSLogix 500 v20 with FANUC Robotics AOP v1.36 for example purpose. Very first thing to create is scanner connection because import process will look for appropriate connection name. At this point only connection existence with appropriate name is important not the configuration. So just create the connection with default configuration if using AOP (FANUC RobotPlus). If you serach FANUC then you will find two AOPs for FANUC Robot. Make sure to select FANUC RobotPlus as shown in Fig. 10.6 (a). Otherwise use below data for default configuration if using generic Ethernet profile (Ethernet Module). Connection name is the one you configured in Robot EDA configuration screen. Just in case you forgot, it is prefix in L5X file name i.e. r10 in r10program.l5x. Make sure you configured right number of connections i.e. 4 in our example. Click Change button in Fig. 10.6 (b) and select 4 for No. of Standard Connections from the drop down menu.

Table 10.6

Type	Assembly Instances	Data Size
Input	101	8 (8–bits)
Output	151	8 (8–bits)
Config	100	0
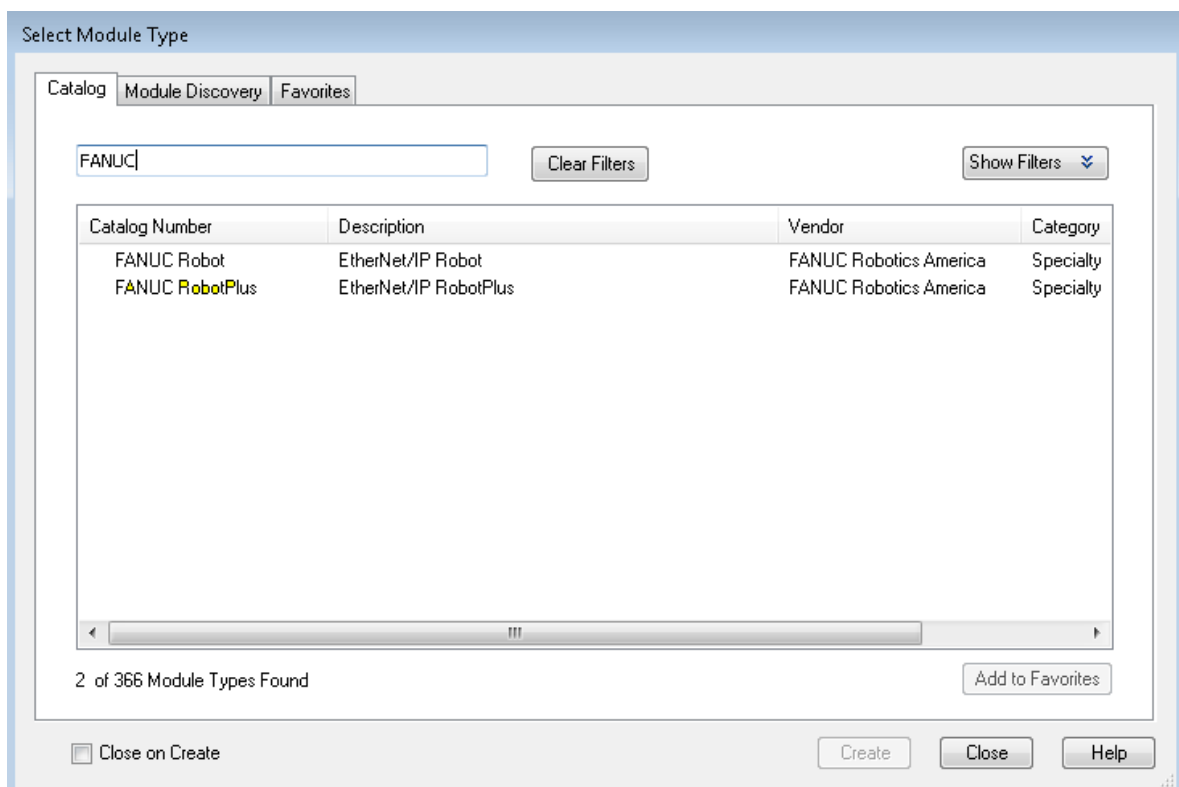
Fig. 10.6 (a) Select AOP for FANUC RobotPlus
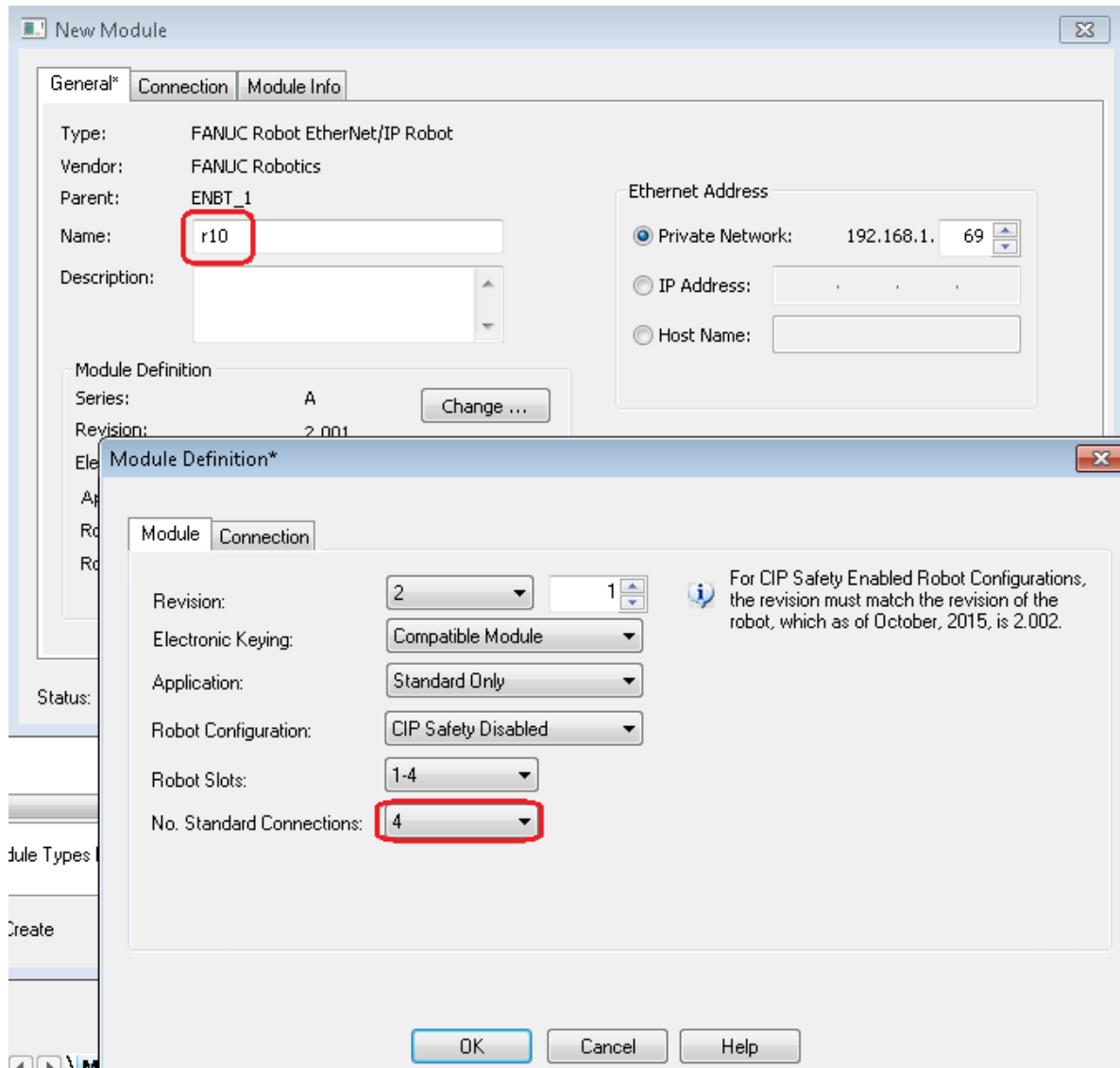
Fig. 10.6 (b) Create Scanner Connection

Now click Connection tab in Fig. 10.6 (c) to see the default I/O sizes as you need to come back and change these to match the actual configuration after import process is completed.
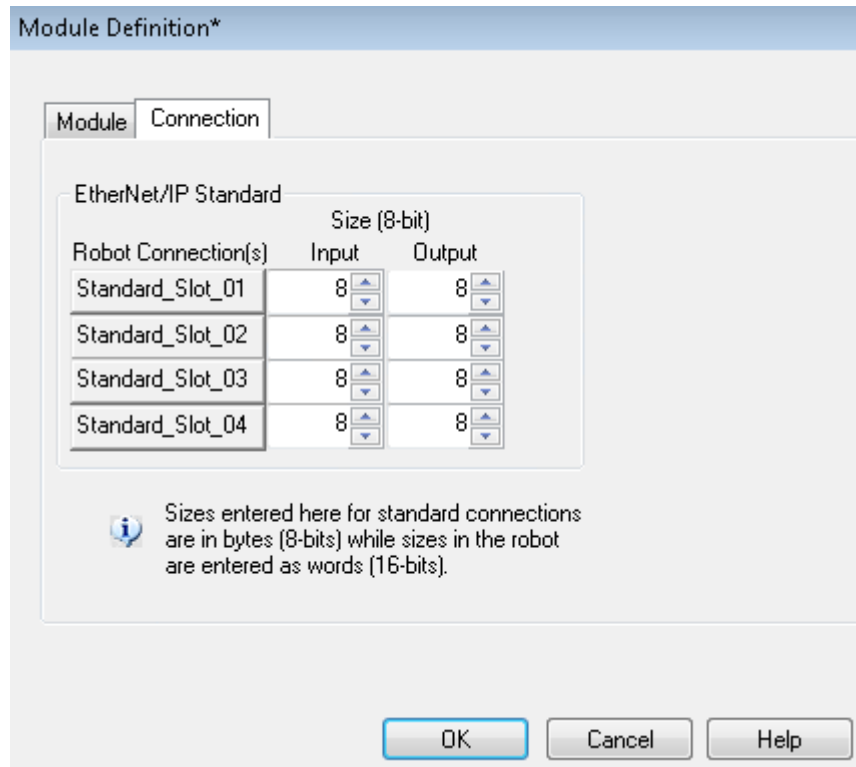

Fig. 10.6 (c) Default I/O Sizes

If you are using Ethernet Module then fill the below module properties (Fig. 10.6 (d)) using configuration data from Table 10.6.
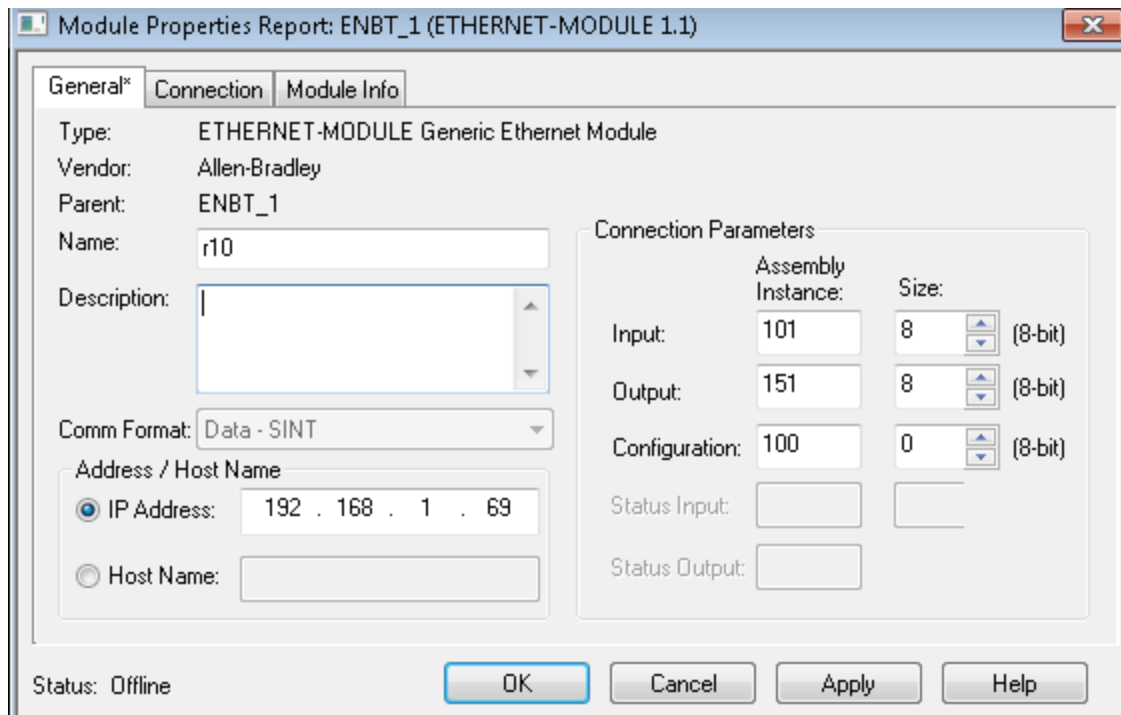

Fig. 10.6 (d) Using Generic Ethernet Profile (Ethernet Module)

Once connection creation complete, you will see FANUC Robot/A r10 if used AOP and ETHERNET-MODULE r10 (pderob159 instead of r10 in Fig. 10.6 (e) if used generic Ethernet profile.
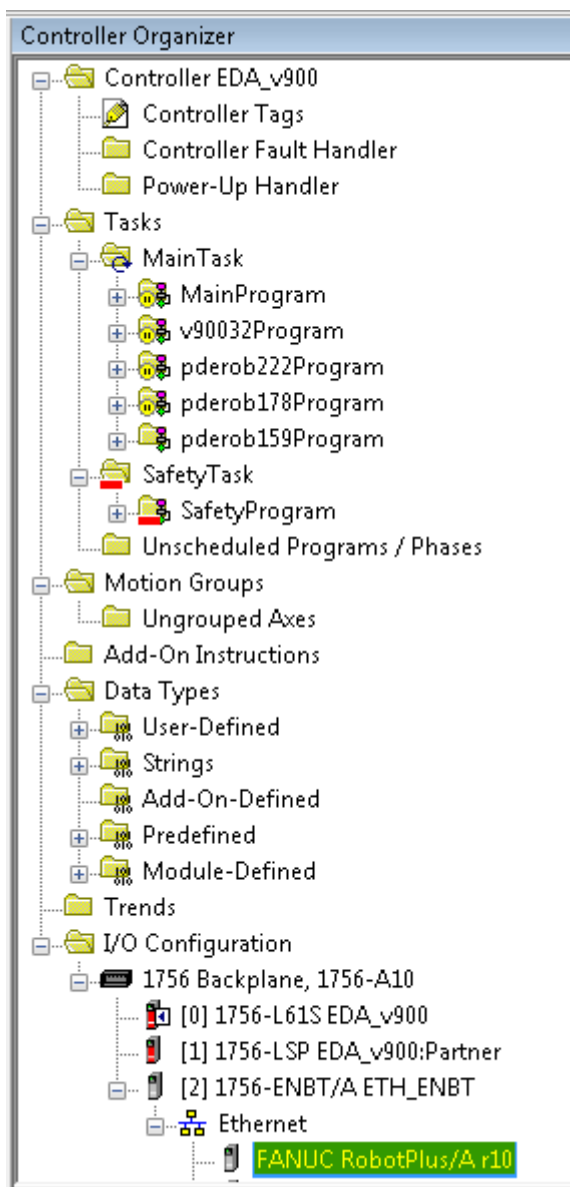
Fig. 10.6 (e) Connection

Now it is time to import L5X file. Right click on Main Task in controller organizer. Select Import Program as shown in Fig. 10.6 (f).
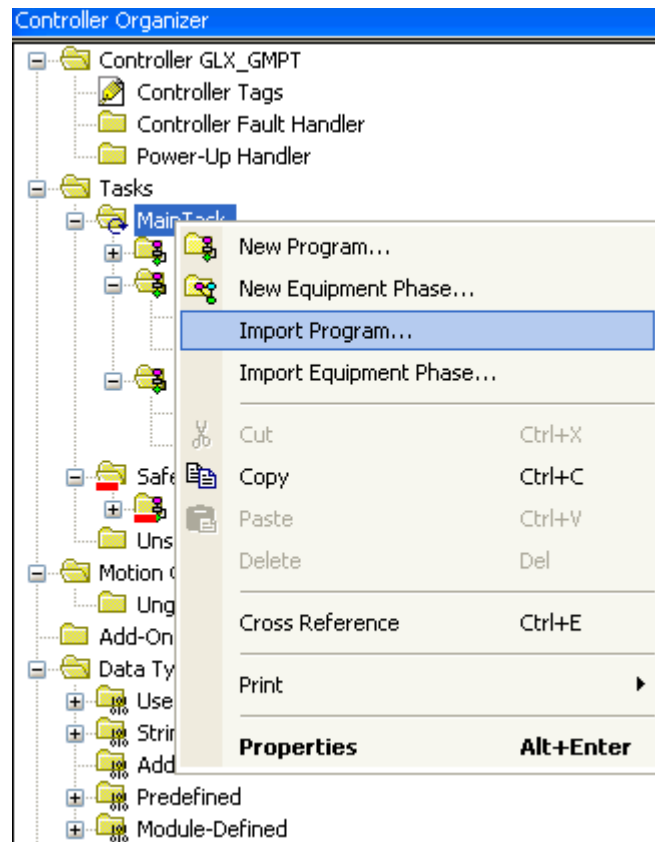
Fig. 10.6 (f) Import PLC Configuration File

After clicking on Import Program, file browser pops up to locate L5X file. Please choose appropriate file and click Import. Example file name is r10program.l5x highlighted in Fig. 10.6 (g).
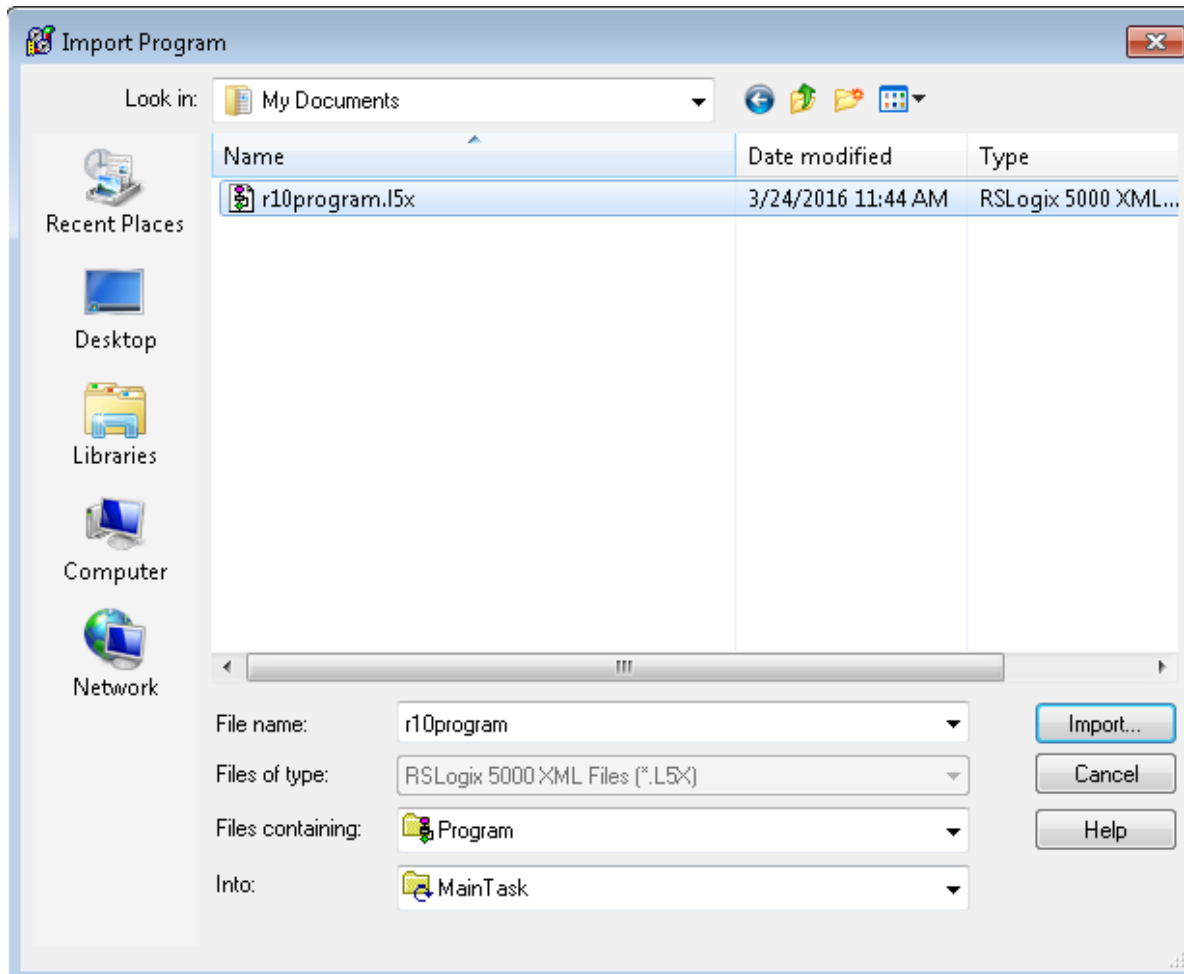
Fig. 10.6 (g) Locate L5X File

Now you will see import content in the left pane in Fig. 10.6 (h). On the right pane, you can still change the program name if you want special name for the program. Similarly you can change the routine name via routine configuration by clicking Routine in left pane. If program and routine already exist with same names, it allows you to use existing or overwrite. Other important items are Tags and Data Types.
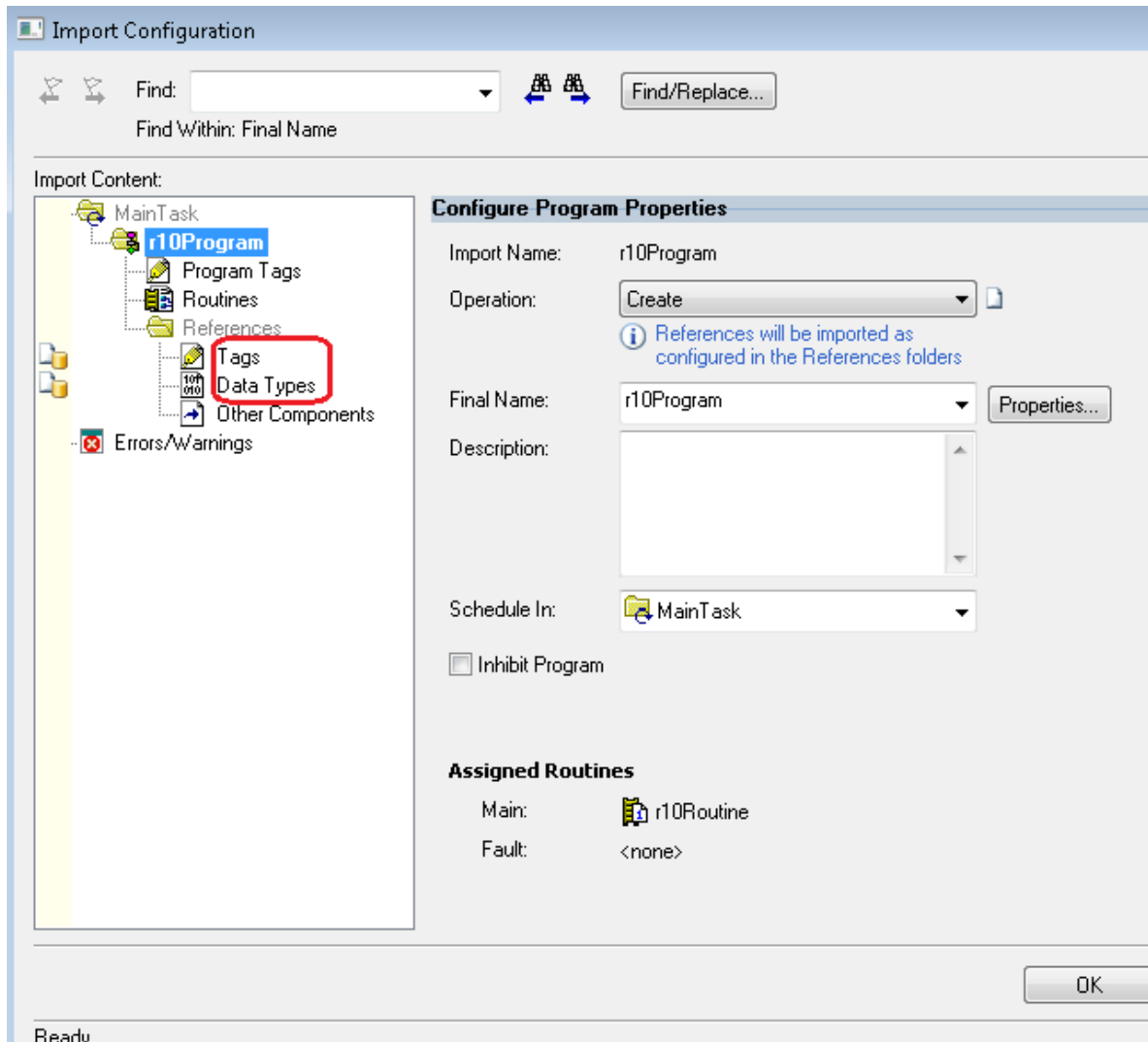
Fig. 10.6 (h) Import Dialog

- 134 -

Click on Tags in Fig. 10.6 (i) and you will see new tags being created in left pane. Top tags those say "Use Existing" which are the tags created when you created scanner connection as first step. So leave them as is. Bottom 4 are controller tags for each connection 1 - 4. If they already exist, you will see "Use Existing" by default but change this to "Overwrite" as there may be changes to structure. Change to tags can be identified by hovering mouse on to the symbol next to it as shown in Fig. 10.6 (j).
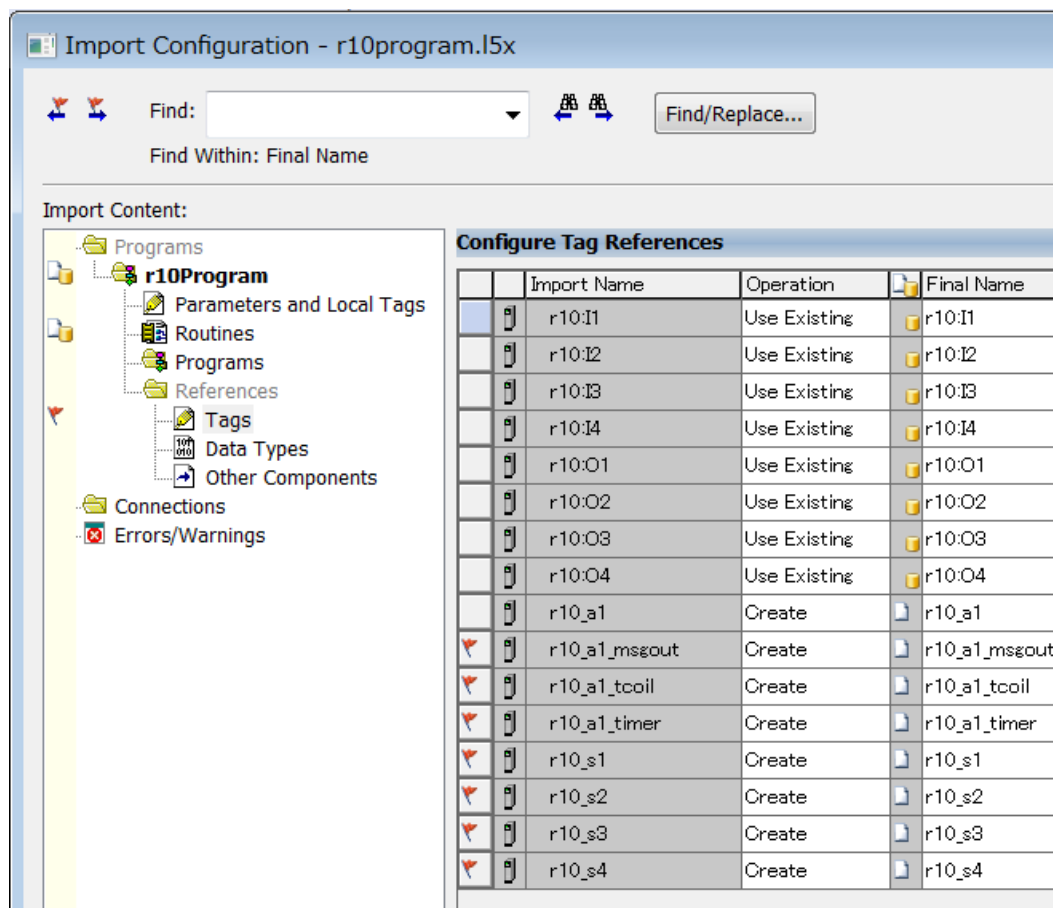
Fig. 10.6 (i) Controller Tags
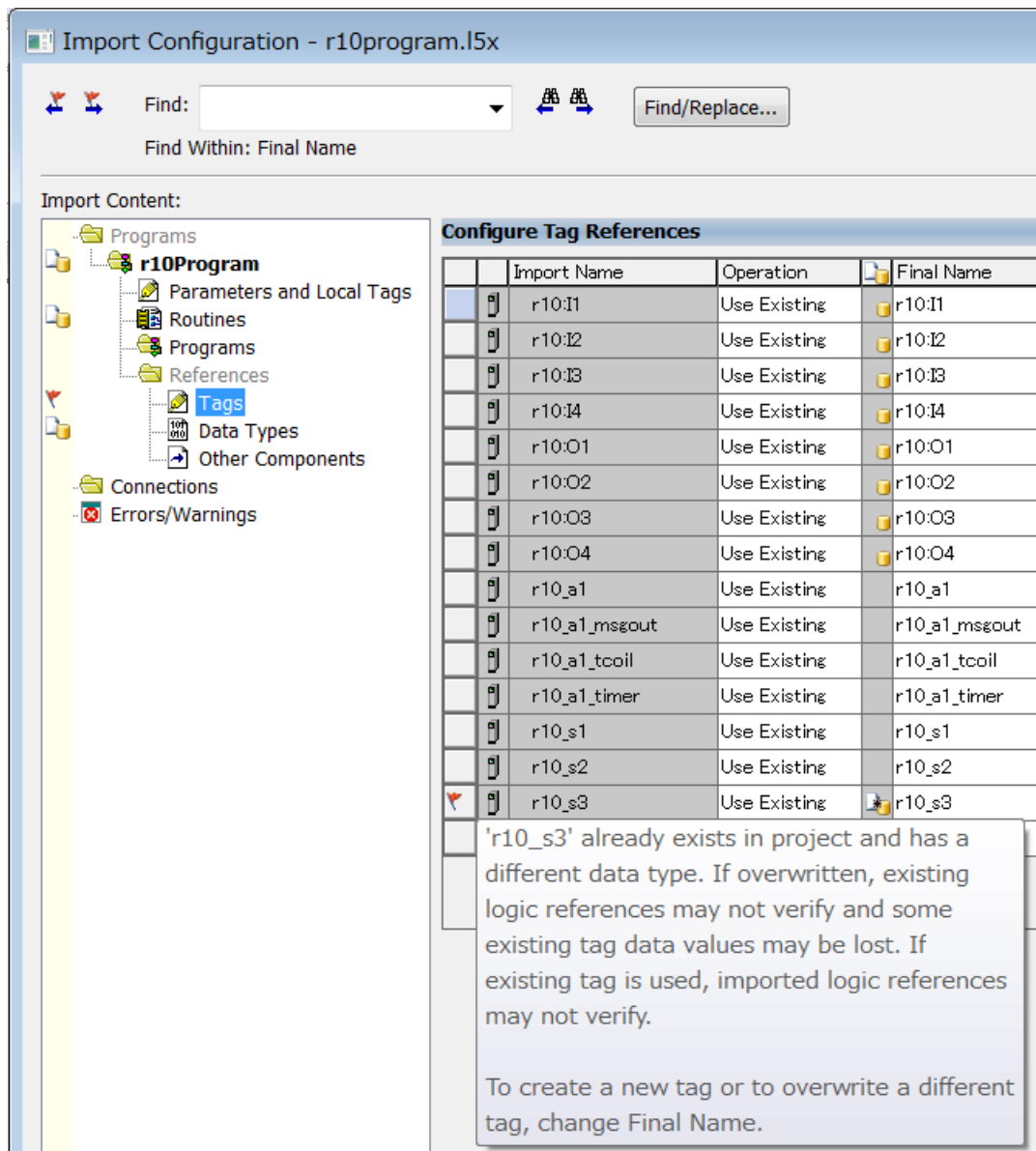
Fig. 10.6 (j) Tag Warning

Click on Data Types in the left pane in Fig. 10.6 (k). It shows Data Types configuration for new data types. It also allows to choose if user wants to use existing or overwrite if existing is different than new data type being created as shown in Fig. 10.6 (l).
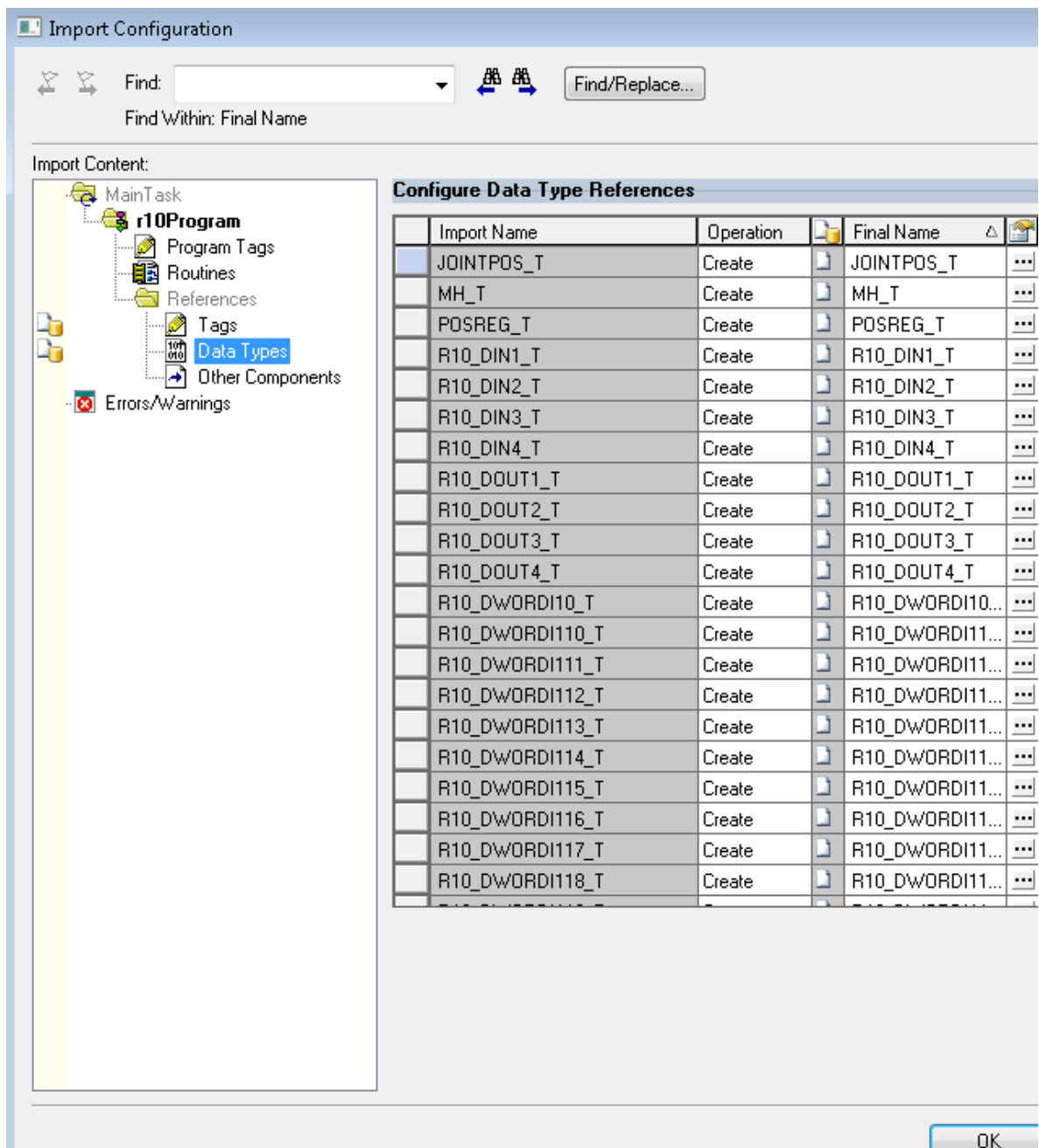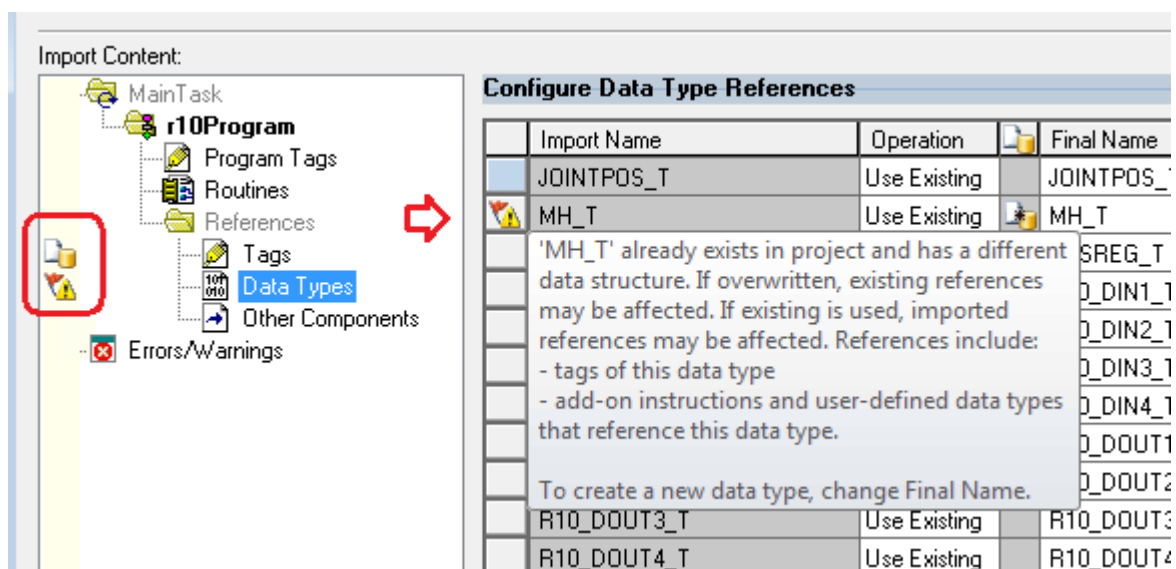

Fig. 10.6 (k) User Defined Data Types

Fig. 10.6 (l) Structure Warning

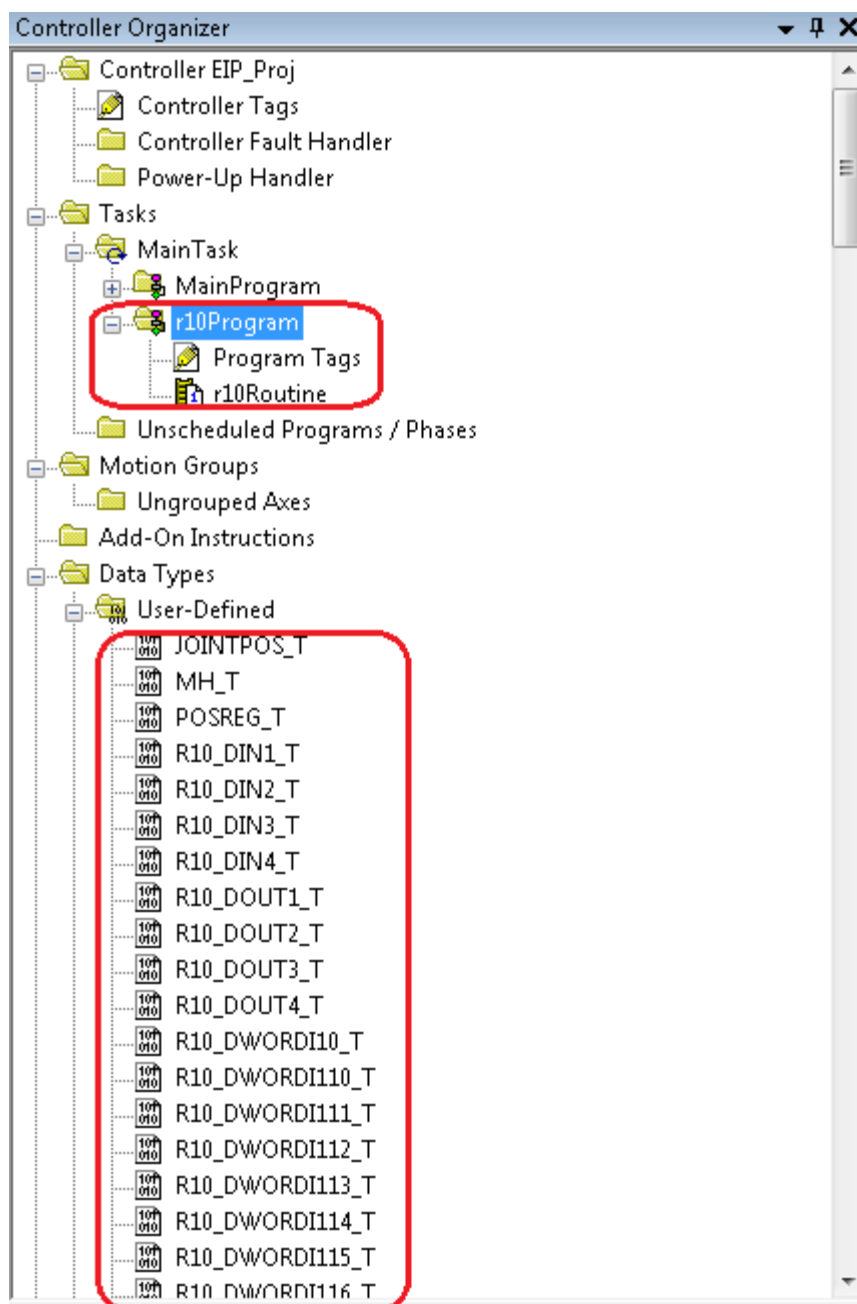Fig. 10.6 (m) shows the tree view of newly created program, routine, and user defined data types.

Fig. 10.6(m) Imported Data Items

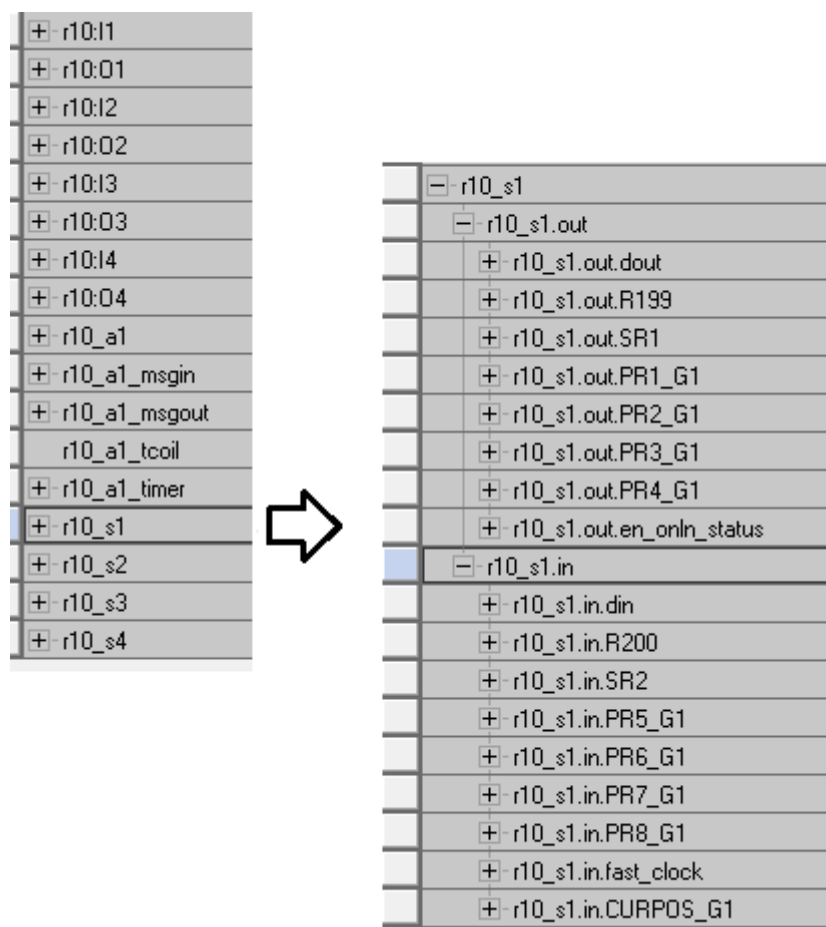Fig. 10.6 (n) shows final controller tags created for given export file.

Fig. 10.6 (n) Imported Controller Tags

Fig. 10.6 (o) shows the KAREL structure imported from robot. It exactly looks as in robot.

Name:	MH_T

Description:	FANUC KAREL Structure

Members: Data Type Size: 368 byte(s)

	Name	Data Type	Style	Description	External Access
	PKUFNUM	DINT	Decimal		Read/Write
	PLUFNUM	DINT	Decimal		Read/Write
	UTNUM	DINT	Decimal		Read/Write
	TOTALPKS	DINT	Decimal		Read/Write
	TOTALDRPS	DINT	Decimal		Read/Write
	PKDELAY	REAL	Float		Read/Write
	PLDELAY	REAL	Float		Read/Write
	PAYLOAD	DINT	Decimal		Read/Write
⊞	PKPOS	XYZWPR_T[10]			Read/Write
⊞	PLAPPR	XYZWPR_T			Read/Write
⊞	PLRET	XYZWPR_T			Read/Write

Fig. 10.6 (o) KAREL Structure

- 141 -

Fig. 10.6 (p) shows KAREL position XYZWPR and it is fixed structure across any FANUC robot. So it can be reused for any FANUC robot.

Name: XYZWPR_T

Description:
KAREL Position
(XYZWPR)

Members: Data Type Size: 28 byte(s)

	Name	Data Type	Style	Description	External Access
	X	REAL	Float		Read/Write
	Y	REAL	Float		Read/Write
	Z	REAL	Float		Read/Write
	W	REAL	Float		Read/Write
	P	REAL	Float		Read/Write
	R	REAL	Float		Read/Write
	turn1	SINT	Decimal		Read/Write
	turn2	SINT	Decimal		Read/Write
	turn3	SINT	Decimal		Read/Write
	front	BOOL	Binary		Read/Write
	up	BOOL	Binary		Read/Write
	left	BOOL	Binary		Read/Write
	flip	BOOL	Binary		Read/Write
1of 010					

Fig. 10.6 (p) KAREL XYZWPR

Position Register looks like KAREL XYZWPR_T except it has 4 bytes for User Frame (uframe) and User Tool (utool) number along with ext[3] for extended axis. On the other hand RXYZWPR_T is XYZWPR_T plus ext[3]. See Fig. 10.6 (q).

Name: POSREG_T

Description: FANUC Position Register (Cartesian Mode)

Members: Data Type Size: 44 byte(s)

Name	Data Type	Style	Description	External Access
uframe	INT	Decimal		Read/Write
utool	INT	Decimal		Read/Write
X	REAL	Float		Read/Write
Y	REAL	Float		Read/Write
Z	REAL	Float		Read/Write
W	REAL	Float		Read/Write
P	REAL	Float		Read/Write
R	REAL	Float		Read/Write
turn1	SINT	Decimal		Read/Write
turn2	SINT	Decimal		Read/Write
turn3	SINT	Decimal		Read/Write
front	BOOL	Decimal		Read/Write
up	BOOL	Decimal		Read/Write
left	BOOL	Decimal		Read/Write
flip	BOOL	Decimal		Read/Write
ext	REAL[3]	Float		Read/Write

Move Up   Move Down   OK   Cancel   Apply

Fig. 10.6 (q) Position Register Structure (POSREG_T)

JOINTPOS_T is KAREL joint position structure. POSREGJ_T is joint position register structure which is uframe and utool along with array of 9 real numbers for axes including extended axes as shown in Fig. 10.6 (s).

Name: JOINTPOS_T

Description: KAREL Position (JOINTPOS)

Members: Data Type Size: 36 byte(s)

Name	Data Type	Style	Description	External Access
joint	REAL[9]	Float		Read/Write

Fig. 10.6 (r) JOINTPOS_T

Fig. 10.6 (s) Joint Position Register Structure (POSREGJ_T)

STRING and Numeric Register Data Types: STRING register data type is RockWell STRING type and numeric register is DINT or REAL type for integer and real (float) respectively.

Fig. 10.6 (t) describes FANUC KAREL string structure. This is also fixed structure and reusable across FANUC robots.

Fig. 10.6 (t) KAREL String

Fig. 10.6 (u) shows the ladder logic created from the import process. Also, please note that import process creates a note on originator/scanner connection configuration information to be used for connection creation. Rung 0 through Rung 7 in are created (two for each connection: input and output) to copy data back and forth for implicit connection.
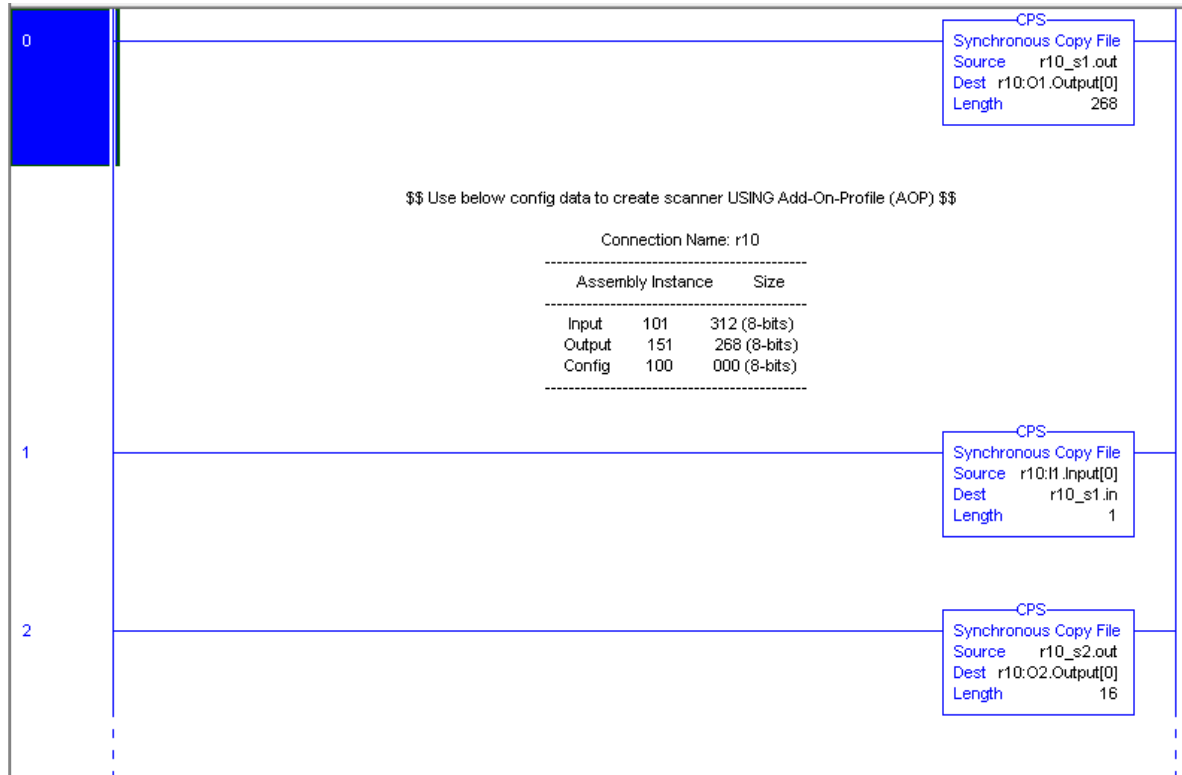
Fig. 10.6(u) Ladder Logic - Implicit

Rung 8 to 10 are created (Fig. 10.6(v)) to handle explicit connections and a timer (Rung 8) to execute rungs periodically. Default timer is set to 1000ms (1 Sec).
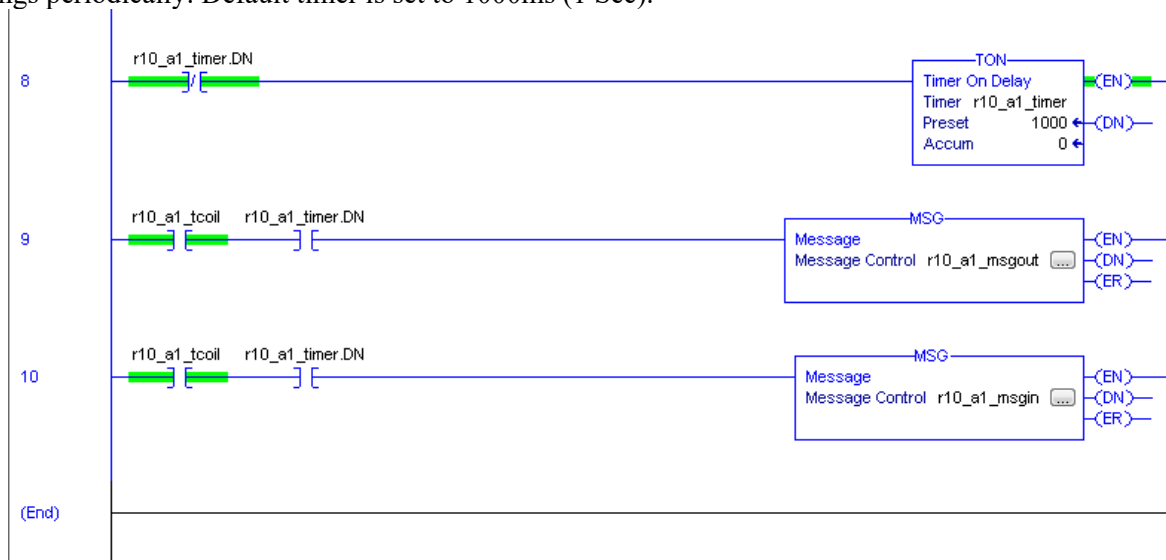
Fig. 10.6(v) Ladder Logic- Explicit

Now update the connection configuration parameters shown in the ladder logic as comment. If you have used AOP to create connections then only I/O sizes needs to be updated. Otherwise you will need to update assembly instances as well. Please note that Comm Format MUST be Data-SINT (8‑ bit/Byte) if

using generic Ethernet profile to create the connection. Otherwise data tearing may happen. Now you are ready to download program to the PLC and get it going after successful download.

10.7   Data Conversion (REAL vs INT)

This data conversion is only applicable to NUMERIC registers. When requested register is INTEGER and configured as a REAL, value is converted to the nearest INTEGER. For example, if values are 5.4 and 5.5 in controller then converted values are 5 and 6 respectively. Likewise, when requested numeric register REAL and if the numeric register is configured as an INTEGER, returned value is converted to REAL. For example, if value is 4 then converted value is 4.0. Same rule applies when writing register back to robot controller.

10.8   Limitations

This section lists potential capabilities and limitations.

10.8.1   I/O Size Limitation

Implicit Connection can only transfer 490 bytes of input and 490 bytes of output per connection.

Explicit connection is allowed to read 496 bytes of data from robot and 464 bytes of data can be written back to robot from the PLC. This limitation applies per message block (Connection).

10.8.2   Connection Limitation

- FANUC Add-On-Profile (FANUC RobotPlus): Four implicit (scanner) connections.
- Generic Ethernet Profile (Ethernet Module): One implicit (scanner) connection.
- The robot can handle at most 6 explicit connections in connected mode and up to 32 in unconnected mode.

10.8.3   Other Limitations

- EDA does not support variable-sized (e.g. ARRAY[*] of INTEGER) arrays in KAREL programs.
- Ordering of tags in configuration will not be preserved in UDT. All same type of tags will be grouped together as a result of import.
- No I/O exchange is supported over explicit connection for Enhanced Data Access.

10.9　Typical Robot Data Type Size

Table 10.9 shows up typical robot data type sizes for reference.

Table 10.9 Robot Data Type and Size

Robot Data Type	Robot Data Size (Bytes)
I/O Input	1 byte per 8 inputs
I/O Output	1 byte per 8 outputs
Numeric Data Register	4 bytes
String Data Register	88 bytes
Position Data Register	40 bytes (Joint) or 44 bytes (Cartesian)
KAREL BOOLEAN	1 byte per 8 Boolean
KAREL INTEGER	4 bytes
KAREL REAL	4 bytes
KAREL STRING	4 byte header plus 1 byte x length of string e.g. for STRING[20] size is 24 bytes, STRING[9] size is 16 bytes. If declared size is not multiple of 4 bytes, it is padded to next multiple.
KAREL POSITION	24 bytes (Joint), 28 bytes (Cartesian) and 40 byte (Redundant Cartesian)
KAREL Structure	Sum of individual items

> **NOTE**
> Please note that individual items of data structure are padded to next multiple of 4 if they are not. So whole structure size is always multiple of 4. This is Rockwell PLC's strict format rule.

10.10　General Errors

There are EDA specific alarms. It has following two alarms to show up error in string format. Table 10.10 (a) shows EDA alarms for implicit connections while Table 10.10(b) shows the general error status.

Table 10.10 (a) Implicit Errors

Error String	Description
PRIO 234 EtherNet/IP UDT Write Error: %s	%s is replaced with variable name and error status that has error in writing its content
PRIO 235 EtherNet/IP UDT Read Error: %s	%s is replaced with variable name and error status that has error in reading its content

Table 10.10 (b) General Status Errors

Error Code	Description
0xD0002	I/O not mapped or Illegal port number
0x0400	Null Pointer
0x0401	Bad Register Index
0x0402	Bad Position Representation
0x0403	Variable Read Error
0x0404	Variable not accessible
0x0405	Export Error
0x0406	Bad KAREL/SYSTEM variable name
0x0407	Configuration exceeds allowed data limit per connection
0x0408	User Tool or User Frame is bad
0x0409	Bad Array Index
0x0410	Error Unknown
0x0411	Invalid group number

APPENDIX

A   THIRD-PARTY CONFIGURATION TOOLS

A.1   Tools Overview

Robot Scanner connections can be configured from the EtherNet/IP Interface screens, or from third party tools such as RSNetWorx for EtherNet/IP using the Connection Configuration Object (CCO). Certain devices require detailed configuration data, and can only be added to the robot scanlist by using offline tools such as an Allen Bradley Flex I/O block with attached modules. Other devices can be configured through both interfaces, such as another FANUC Robot, or an RJ-Lynx I/O block. To use the offline tools, an EDS file for each device is required.

It is recommended that either all scanlist configurations be done entirely from the *i*Pendant, or be done entirely from RSNetWorx for EtherNet/IP. In certain situations RSNetWorx for EtherNet/IP version 4.11 might delete scanlist entries configured through the *i*Pendant.

> **NOTE**
> TIP: Some third party tools cannot import scanlist configurations from the robot
> controller unless both the configured revision numbers (major and minor) exactly
> match the revision numbers in the EDS file that the third party tool had loaded
> for the corresponding device. To set the revision numbers, see Section 4.2.4.

Procedure A-1 Configure the Robot Scanners Using RSNetworX for EtherNet/IP

Steps
1   From the EtherNet/IP Interface Status screen, create and configure the desired number of scanner connections. See Section 4.2.3.
2   Perform a Controlled start. Refer to the application-specific Setup and Operations Manual.
3   Configure an AB_ETH driver in RsLinx. The configuration screen should be similar to the screen shown in Fig. A.1 (a).
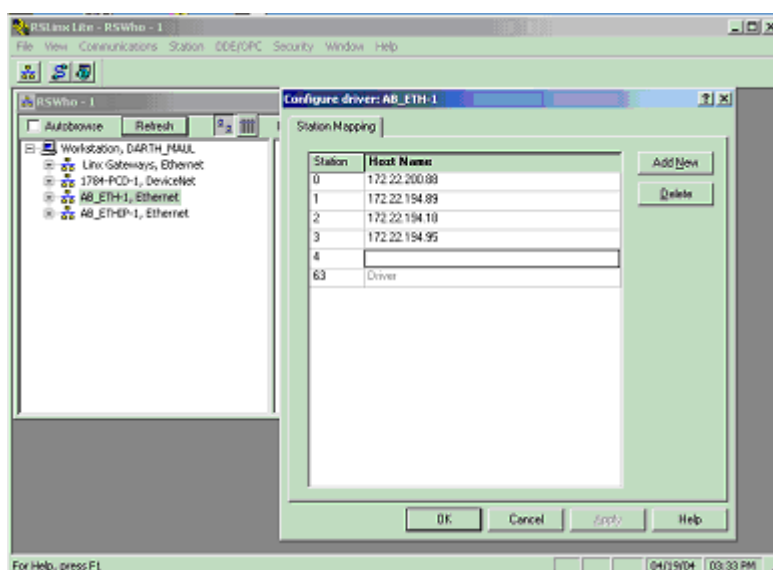

Fig. A.1 (a) Configuring the Driver

4   In RSNetWorx, under Tools, select the EDS Wizard and register the FANUC Robot EDS file.
5   Create an EtherNet/IP project in RSNetWorx that includes the devices configured in RSLinx.
6   Right click on the robot icon, and select Scanlist Configuration. You will see a screen similar to the one shown in Fig. A.1 (b).
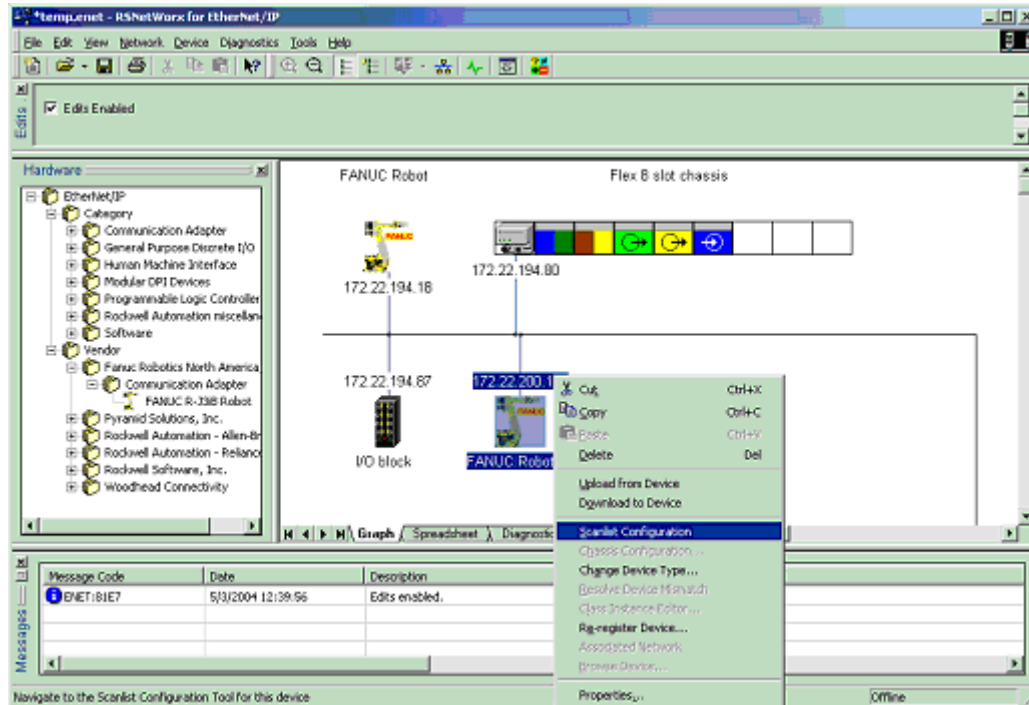

Fig. A.1 (b) Scanlist Configuration Screen

7   A new window appears entitle "FANUC Robot – Scanlist Configuration". In this configuration window, right click over the device you want to add to the robot's scanlist, and select Insert Connection. See Fig. A.1 (c).
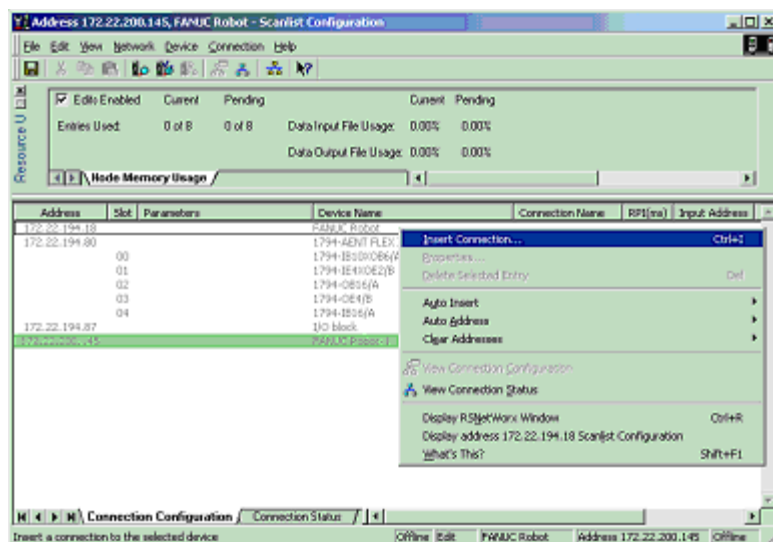
Fig. A.1 (c) Insert Connection Screen

8   Configure the connection properties and click on OK. Pay specific attention to the selecting the appropriate Connection Name, Input Size, and Output Size. Other configurable values might include RPI, and Target to Scanner Transmission Mode. Note that the robot does not use values inserted for Input Address or Output Address. The screens below show a FANUC Robot being added to the scanlist of another FANUC Robot.
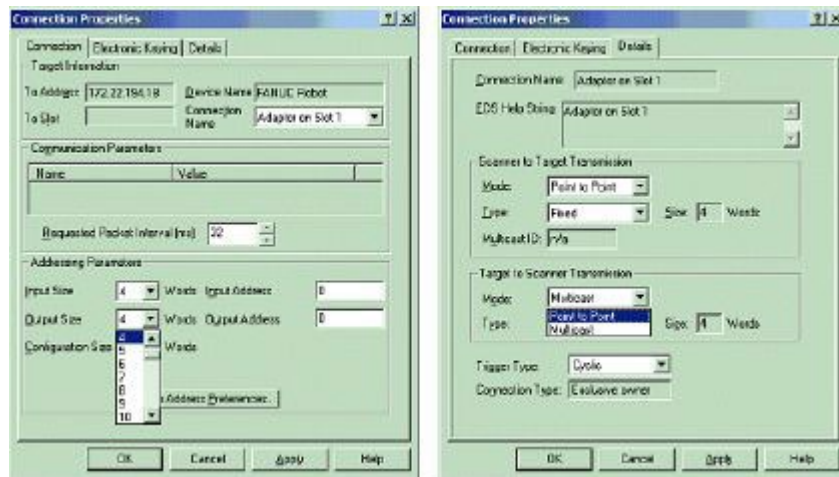

Fig. A.1 (d) Adding a Robot

9   In the Scanlist Configuration window, select Device/Download to Device.

10   When finished, Cold start the controller.

B. KAREL PROGRAMS FOR ETHERNET/IP SCANNER QUICK CONNECT

B.1 OVERVIEW

The EtherNet/IP Scanner option installs the following KAREL programs:
- EN_OFFLN Allows a teach pendant program to turn an EtherNet/IP scanner connection off
- EN_ONLN Allows a teach pendant program to turn an EtherNet/IP scanner connection on
- EN_AROFF - Allows a teach pendant program to turn off auto-reconnect for an EtherNet/IP scanner connection.
- EN_ARON - Allows a teach pendant program to turn on auto-reconnect for an EtherNet/IP scanner connection.
- EN_STCHK - Allows a teach pendant program to check the status of an EtherNet/IP scanner connection.

B.2 KAREL PROGRAM DESCRIPTIONS AND PARAMETERS

The following are the KAREL program descriptions and parameters.

EN_OFFLN (INTEGER slot_number)

This program allows a teach pendant program to turn an EtherNet/IP scanner connection offline. This program takes the slot number as an argument. The valid values for a slot number are 1 through 32. For example, 1 corresponds to the EtherNet/IP Connection on slot 1, rack 89, or the first connection displayed on the EtherNet/IP Status Screen. There is no difference between this call and disabling the connection from the teach pendant.

EN_ONLN (INTEGER slot_number, INTEGER <wait_time>)

This program allows a teach pendant program to turn an EtherNet/IP scanner connection online. This program takes the slot number as an argument. The valid values for a slot number are 1 through 32. For example, 1 corresponds to the EtherNet/IP Connection on slot 1, rack 89, or the first connection displayed on the EtherNet/IP Status Screen. The optional argument, wait_time, is used as follows:
- If wait_time is not used. If wait_time is not explicitly specified (it is an optional argument), its value will be defaulted to 15 and EN_ONLN follows the: if wait_time is not 0 rule.
- If wait_time is not 0. The EtherNet/IP scanner connection will be enabled. Auto-reconnect will also be enabled, causing the scanner to attempt to make a connection to the adapter device every 2 seconds until successful. Note that EN_ONLN will block and will not return until a successful connection is made, or until the user aborts the teach pendant program. An alarm will be posted if wait_time seconds pass before a connection is established. After the alarm is posted and the robot faults, a reset/resume from either the PLC or teach pendant will restart/resume the program inside of the EN_ONLN call and the wait_time timer will be reset. Before EN_ONLN returns, auto-reconnect will set to its original state (its state before EN_ONLN was called).
- If wait_time is used and set to 0. Auto-reconnect will not be enabled‑ the user must explicitly enable Auto-reconnect if needed. The EtherNet/IP connection will be enabled and the call will return immediately (will not block). The application or user programs can then use EN_STCHK to check the status if it needs to confirm the status of the connection.

There is difference between this call and enabling the connection from the teach pendant. Call to this macro forces scanner device (if Quick Connect mode enabled) to wait for GRATUITOUS ARP packet from target device before starting the connection process.

EN_AROFF (INTEGER slot_number)

This program allows a teach pendant program to turn off auto-reconnect for an EtherNet/IP scanner connection. This program takes the slot number as an argument. The valid values for a slot number are 1 through 64. For example, 1 corresponds to the EtherNet/IP Connection on slot 1, rack 89, or the first connection displayed on the EtherNet/IP Status Screen.

There is no difference between this call and disabling auto-reconnect from the teach pendant.

EN_ARON (INTEGER slot_ number)

This program allows a teach pendant program to turn on auto-reconnect for an EtherNet/IP scanner connection. This program takes the slot number as an argument. The valid values for a slot number are 1 through 32. For example, 1 corresponds to the EtherNet/IP Connection on slot 1, rack 89, or the first connection displayed on the EtherNet/IP Status Screen.

Enabling auto-reconnect has the following side effects. While enabled, all EtherNet/IP alarms relating to connection establishment and connection time- outs for this slot number will be masked (will not be posted). The EtherNet/IP scanner corresponding to the slot number will attempt to make a connection to the adapter device every 2 seconds until successful. Before each retry, the ARP cache in the TCP/IP stack will be flushed of the target IP address. Also, the status on the teach pendant will become encapsulated in < and > as in <STATUS>, for example.

There is no difference between this call and enabling auto-reconnect from the teach pendant.

EN_STCHK (INTEGER slot_number, INTEGER register_number)

This program allows a teach pendant program to check the status of an EtherNet/IP connection. This program takes the slot number, and register_number as arguments. The valid values for a slot number are 1 through 64. For example, 1 corresponds to the EtherNet/IP Connection on slot 1, rack 89, or the first connection displayed on the EtherNet/IP Status Screen. The possible status values returned in the register_number are:

- 0 Offline
- 1 Error
- 2 Pending
- 3 Enabled but not connected or trying to connection
- 4 Enabled but not connected. Is trying to connect.
- 5 Online and connected but I/O is not being received from adapter
- 6 Online and I/O is being exchanged

> **NOTE**
> When a connection is taken offline, if a background application were to access I/O belonging to that connection, an unassigned port alarm would be posted. This should be taken into consideration by the teach pendant programmer when using the EN_OFFLN program.

EN_QCON (INTEGER slot_ number)

This program allows a teach pendant program to enable Quick Connect feature in target configured at given slot number (EtherNet/IP scanner connection). This program takes the slot number as an argument. The valid values for a slot number are 1 through 64. For example, 1 corresponds to the EtherNet/IP Connection on slot 1, rack 89, or the first connection displayed on the EtherNet/IP Status Screen.

EN_QCOFF (INTEGER slot_ number)

This program allows a teach pendant program to disable Quick Connect feature in target configured at given slot number (EtherNet/IP scanner connection). This program takes the slot number as an argument. The valid values for a slot number are 1 through 64. For example, 1 corresponds to the EtherNet/IP Connection on slot 1, rack 89, or the first connection displayed on the EtherNet/IP Status Screen.

EN_QCCHK (INTEGER slot_ number, INTEGERE reg)

This program allows a teach pendant program to enable Quick Connect feature in target configured at given slot number (EtherNet/IP scanner connection). This program takes the slot number as an argument and numeric register number to save the status of Quick connect i.e. 1 or 0 for enabled or disabled respectively. So if QC is enabled given register number is set 1 i.e. R[5] = 1 if register number is 5. The valid values for a slot number are 1 through 64. For example, 1 corresponds to the EtherNet/IP Connection on slot 1, rack 89, or the first connection displayed on the EtherNet/IP Status Screen.

B.3   USING KAREL PROGRAMS IN TEACH PENDANT PROGRAMS

Procedure B-1 shows how to use the EN_STCHK KAREL program. The other programs listed in this section can be used in the same way.

Procedure B-1 Placing the Call to the KAREL Program in the Teach Pendant Program

1   Press the [SELECT] key.
2   Display the appropriate list of programs. If F1, [TYPE], is not displayed on the screen, press the [NEXT] key, until it is displayed.
    a   Press F1, [TYPE].
    b   Select the list you want:
3   Move the cursor to the name of the program you want to modify and press the [ENTER] key.
4   Turn the teach pendant ON/OFF switch to ON.
5   Select F4, [INST].
6   Select Call from the list of options that appear at the top of the screen.
7   Select Call Program and press the [ENTER] key.
8   Press F3, [KAREL] to display the available KAREL programs at the top of the screen.
9   Select EN_STCHK and press the [ENTER] key.
10   Place the cursor to the right of the word EN_STCHK.
11   Press F4, [CHOICE].
12   Select Constant from the list at the top of the screen and press the [ENTER] key.
13   Type the Slot Number and press the [ENTER] key.
14   Press F4, [CHOICE].
15   Select Constant from the list at the top of the screen and press the [ENTER] key.
16   Type the Register Number for the result of the device status check and press the [ENTER] key.
    The finished line in the teach pendant program should look like the following:

```
CALL EN_STCHK (2,50)
```

NOTE
    50 is the register number for result (R[50]).

B.4   EXAMPLES USING ETHERNET/IP MACROS

B.4.1   Overview

Generally, EtherNet/IP macros are used to support tool change applications. The following examples demonstrate using Auto-Reconnect and connection Offline/Online macros in tool changing applications.

The connection offline/online macros are similar to manually taking the connection offline or online in the teach pendant screens but are called programmatically through a teach pendant program.

A single EtherNet/IP scanner connection can be used to connect to different types of I/O blocks (different electronic keying) that may be used on the various tools if the I/O sizes for these blocks are the same. In these cases, the electronic keying parameters must be set to 0 in the corresponding scanner configuration screen.

Turning on Auto-Reconnect means that the robot will automatically try to reconnect to the target device if the connection is lost. Without Auto-Reconnect enabled, the robot will fault if an EtherNet/IP scanner connection is lost, and reset must be pressed to retry the connection. With auto-reconnect enabled, the robot will not fault and will continuously try to reconnect to the device. Auto- reconnect should be turned off if a tool change is not underway (when you do not expect the connection to be lost) so that unexpected connection problems are not masked.

B.4.2   Individual Examples

The example below turns on Auto-Reconnect for the second connection (the connection corresponding to EtherNet/IP slot 2). This call can be executed just before the tool is to be physically disconnected to prevent the robot from faulting once the disconnection occurs. Alternatively, call EN_OFFLN to disable the EtherNet/IP scanner connection before the tool is to be physically disconnected, and execute this call when the tool is physically reconnected, but before EN_ONLN is called.

```
1: CALL EN_ARON(2) ;
```

The next example enables, or brings online, the second connection (the connection corresponding to EtherNet/IP slot 2). This call should be executed when the tool is physically reconnected.

```
1: CALL EN_ONLN(2) ;
```

The example below checks the status of the second connection (the connection corresponding to EtherNet/IP slot 2). The status is placed in register 5, R[5]. The connection is not established and exchanging I/O until the status is equal to the value 6.

```
1: CALL EN_STCHK(2,5) ;
```

The next example turns off Auto-Reconnect for the second connection (the connection corresponding to EtherNet/IP slot 2). This call should be executed after an EtherNet/IP scanner connection has been established. Any problem with this EtherNet/IP connection at this point would be a valid error and will now fault the robot.

```
1: CALL EN_AROFF(2) ;
```

The example below disables, or takes offline, the second connection (the connection corresponding to EtherNet/IP slot 2). This call may be executed just before the tool is to be physically disconnected.

```
1: CALL EN_OFFLN(2) ;
```

The example below enables, the second connection (the connection corresponding to EtherNet/IP slot 2). This call may be executed just before the tool is to be physically connected.

```
1: CALL EN_QCON(2) ;
```

The example below disables, the second connection (the connection corresponding to EtherNet/IP slot 2).

```
1: CALL EN_QCOFF(2) ;
```

The example below enables, the second connection (the connection corresponding to EtherNet/IP slot 2). This call may be executed just after enabling the QC to make sure target has QC enabled.

```
1: CALL EN_QCCHK(2, 5) ;
```

B.4.3   Advanced Examples

It can take up to 300ms for an EtherNet/IP adapter device to power-up and become ready to exchange I/O. The following example can be done after moving away from the tool changer nest to help cycle time by allowing the device power- up and connection time to be done in parallel with the robot motion. The following logic will check for the device to go online for up to 15 seconds. If the device status becomes online in less than 15 seconds, the robot will resume immediately after the online status is obtained. If the device is still not online after 15 seconds a User Alarm is posted and the robot will fault. Note that in line #14, the option argument wait_time is set to 0 for EN_ONLN. Care must be used when setting a device online just after it is reconnected. If it is not fully powered up and available for reconnection by the scanner, an alarm might be generated. To avoid this problem, auto-reconnect is generally enabled while setting the device online, and then disabled once the device comes online. The following logic assumes auto-reconnect has already been enabled.

```
 1: TIMER[1]=RESET ;
 2: TIMER[1]=START ;
 3: LBL[1] ;
 4: CALL EN_QCCHK(2,5)
 5: IF R[5]=1,JMP LBL[3] ;
 6: IF (TIMER[1]>10),JMP LBL[2] ;
 7: CALL EN_QCON(2)
 8: WAIT .10(sec) ;
 9: JMP LBL[1] ;
10: LBL[2] ;
11: UALM[1] ;
12: LBL[3] ;
13: TIMER[1]=RESET ;
14: CALL EN_ONLN(2,0)
15: LBL[4] ;
16: WAIT .10(sec) ;
17: CALL EN_STCHK(2,50) ;
18: IF R[50]=6,JMP LBL[5] ;
19: IF (TIMER[1]<15),JMP LBL[4] ;
20: UALM[2] ;
21: JMP LBL[4] ;
22: LBL[5] ;
23: TIMER[1]=STOP ;
24: CALL EN_AROFF(2) ;
```

The same functionality of the above logic can also be achieved by setting the optional parameter wait_time of the EN_ONLN program to a non-zero value as seen in the logic below. When the wait_time parameter is not set, it will default to 15. In this case, EN_ONLN does not return until an EtherNet/IP scanner connection has been established. Please note that line #3‑ #12 enables the QC if not already. It loops for 10 seconds before it gives up and posts user alarm 1. This is just an example so please rewrite the program as needed for your application.

```
1: CALL EN_ONLN(2,15)
2: CALL EN_AROFF(2) ;
```

OR

```
1: CALL EN_ONLN(2)
2: CALL EN_AROFF(2) ;
```

Below is an outline of how a tool change may occur and a recommended sequence of calls to programmatically handle the tool change.

*   Tool is connected and exchanging I/O with EtherNet/IP scanner connection 2.
*   A tool change is scheduled to occur.

```
CALL EN_OFLN(2) ;
```

*   Physically disconnect the tool.
*   Physically connect a new tool.

```
CALL EN_ARON(2) ;
CALL EN_ONLN(2) ;
```

*   When EN_ONLN returns, tool is connected and exchanging I/O with EtherNet/IP scanner connection 2.

```
CALL EN_AROFF(2) ;
```

C   EtherNet/IP SAFETY SETUP PROCEDURE

C.1   OVERVIEW

This chapter describes how to connect the robot controller to the Safety PLC and confirm the connection status on EtherNet/IP Safety. There are the following 3 steps:

- Robot Controller Configuration (Section C.2)
- Safety PLC Configuration
- Confirmation of the connection status (Section C.3)

This chapter describes how to configure on the robot. This chapter also describes how to confirm of the connection status by the robot controller. Refer to your Safety PLC manual for the Safety PLC configuration.

⚠ **CAUTION**
   The following procedure is just an example. This is the procedure for R-30*i*B controller with system software version V8.30P/41. The procedure can be different depending on the software version or type of Safety PLC.

NOTE
   For EtherNet/IP Safety function that exchanges safety signals on EtherNet/IP, please read "R-30*i*A/R-30*i*A Mate controller Dual Check Safety Function (ISO 13849-1:2006 COMPLIANT) OPERATOR'S MANUAL (B-83104EN)" or "R-30*i*B/R-30*i*B Mate/R-30*i*B Plus/R-30*i*B Mate Plus/R-30*i*B Compact Plus CONTROLLER Dual Check Safety Function OPERATOR'S MANUAL (B-83184EN) in addition to this manual.

C.2   ROBOT CONTROLLER CONFIGURATION

Perform the following steps to configure on the robot controller.

- Setting of the IP address (Subsection C.2.1)
- Setting of the Ethernet port (Subsection C.2.2)
- Setting of the CIP-Safety parameters (Subsection C.2.3)

In this procedure, setup status as shown below.

Ethernet port	Port 2
Robot controller	192.168.250.2
Subnet mask	255.255.255.0

> ⚠ **CAUTION**
> In general, these IP addresses are determined by the network administrator. To find out what addresses to assign, contact the network administrator of your organization.
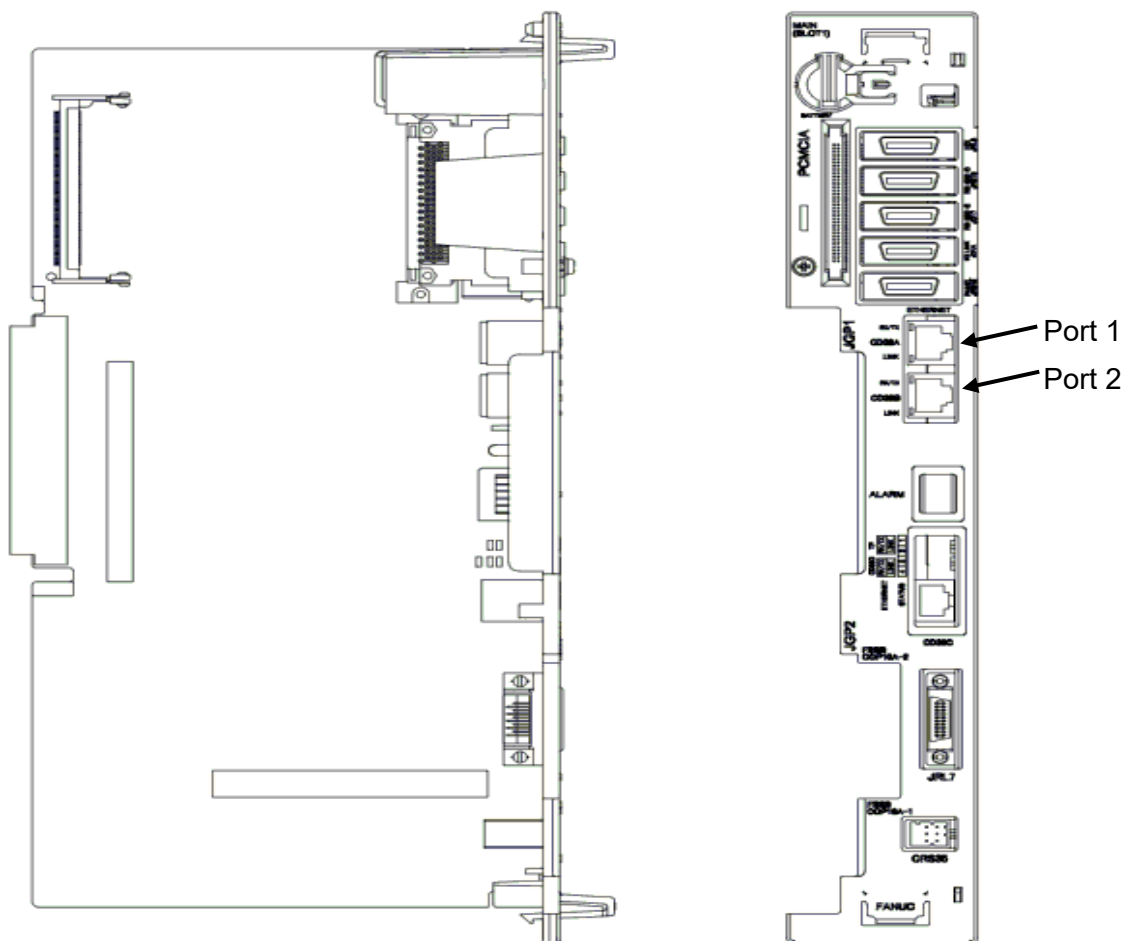
C.2.1   Setting of the IP address

In this subsection, IP address and subnet of the robot controller are set.

1   Press the [MENU] key.
2   Select "SETUP".
3   Press F1, [TYPE].
4   Select "Host Comm".

```
Busy   Step   Hold  ⬤Fault  SRVO-223 DSP dry run (9,1)                    10%
Run   ⇄ I/O   Prod  Trigr                              T1    JOINT

SETUP Protocols                                            ∧        ▱

                                                  1/11

           Protocol   Description
        1  TCP/IP     TCP/IP Detailed Setup
        2  TELNET     Telnet Protocol
        3  SM         Socket Messaging Device
        4  PC SHARE   PC Share Setup
        5  PROXY      Proxy Server
        6  PPP        Point to Point Protocol
        7  PING       Ping Protocol
        8  HTTP       HTTP Authentication
        9  FTP        File Transfer Protocol
       10  SMTP       EMAIL Setup




        [ TYPE ]              DETAIL    [ SHOW ]
```
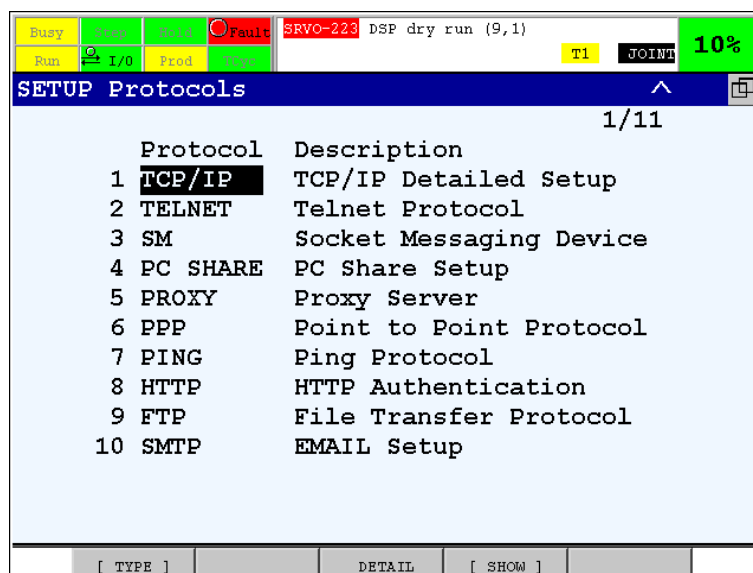
5   Move the cursor to the "TCP/IP", and press the [ENTER] key.
6   Press F3, [PORT].   Check IP addr selected is "Port#2".
7   Move the cursor to the "Port#2 IP addr" and enter the IP address.
8   Move the cursor to the "Subnet Mask" and enter the subnet mask.

```
Busy   Step   Hold  ⬤Fault  SYST-219 Need to apply to DCS param (1)      10%
Run   ⇄ I/O   Prod  Trigr                              T1    JOINT

SETUP Host Comm                                            ∧        ▱
     TCP/IP                                        2/40
          Robot name:              ROBOT
          Port#2 IP addr:      192.168.250.2
          Subnet Mask:         255.255.255.0
          Board address:   00:e0:e4:1f:16:bc
          Router IP addr:        10.18.47.1

          Host Name (LOCAL) Internet Address
          1 **********     ******************
          2 **********     ******************
          3 **********     ******************
          4 **********     ******************




        [ TYPE ]   DHCP     PORT     PING   ⑳ HELP    >
```
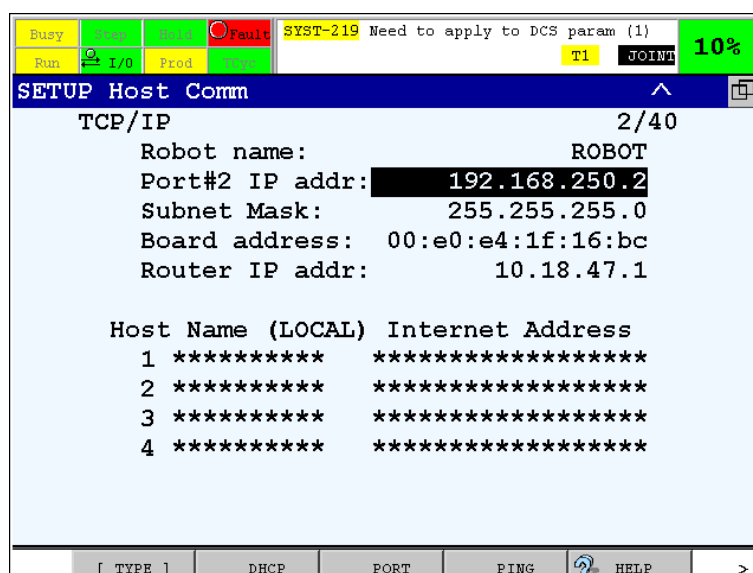
9   Cycle power of the robot controller.

> **⚠ CAUTION**
> In setting the IP address, do not insert any unnecessary spaces or "0". If an
> unnecessary space or "0" is inserted, communication cannot be established.

C.2.2   Setting of the Ethernet port

In this subsection, the Ethernet port for EtherNet/IP Safety is set.

1   Press the [MENU] key.
2   Select "I/O".
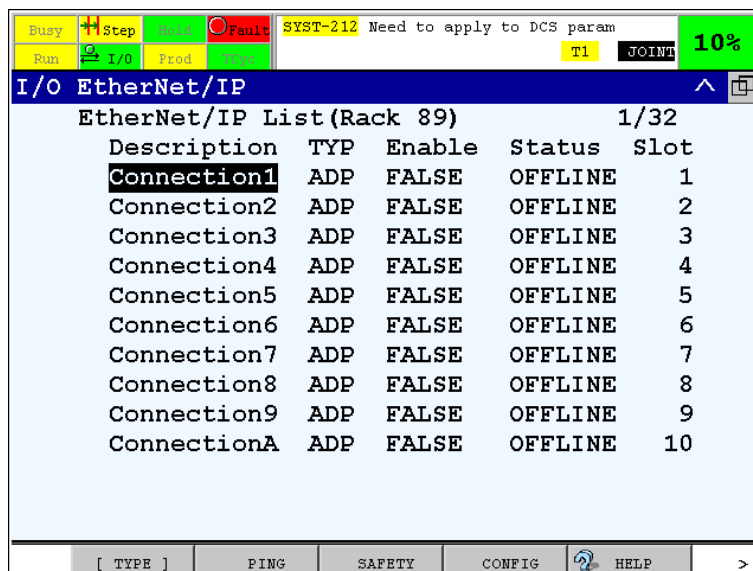
3   Press F1, [TYPE].
4   Select "EtherNet/IP".
5   Press F3, [SAFETY].
6   Move the cursor to the "Ethernet Interface" and press F4, [CHOICE].
7   Select "Port 2".
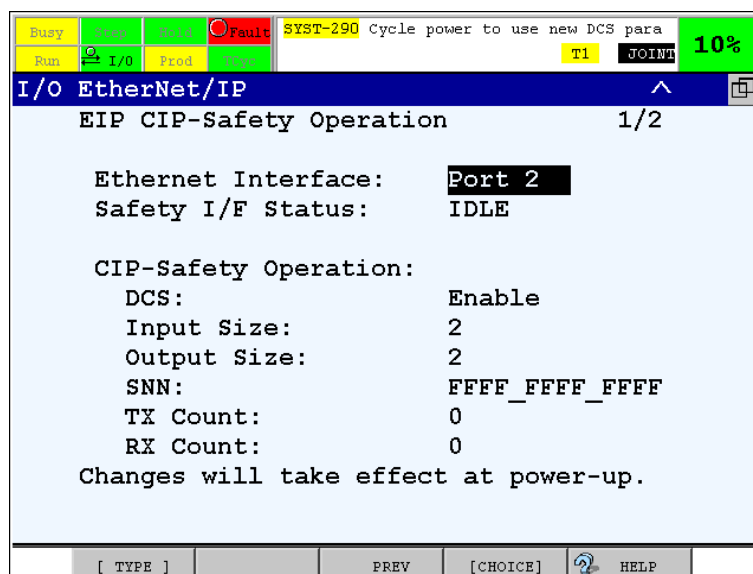
8   Cycle power of the robot controller.

C.2.3   Setting of the CIP-Safety parameters

In this subsection, the CIP Safety I/O size is set. Change the CIP Safety I/O size if needed. The default safety I/O size is 2 bytes in and 2 bytes out. The CIP Safety I/O size must be equal to the number of I/O specified in the configuration of safety PLC. In this procedure, setup status as shown below.

Input size	4 bytes
Output size	4 bytes

1   Press the [MENU] key.
2   Select "System".
3   Press F1, [TYPE].
4   Select "DCS".
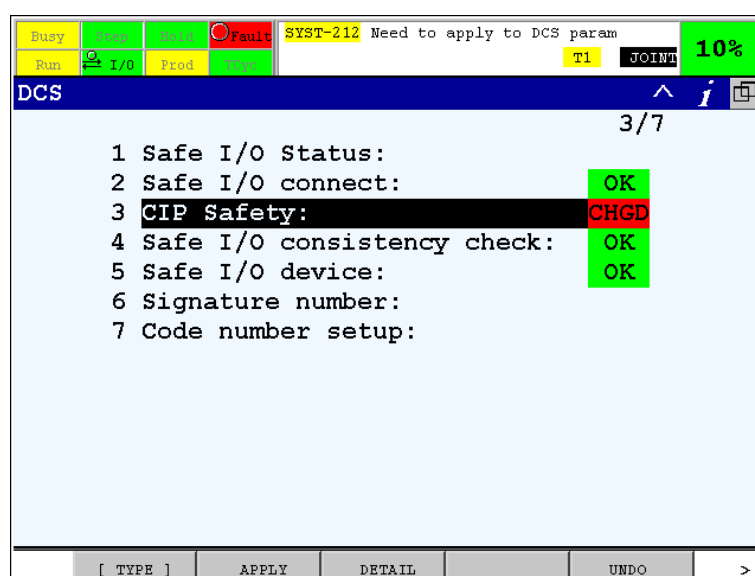
5   Move the cursor to the "CIP Safety" and press the [ENTER] key.
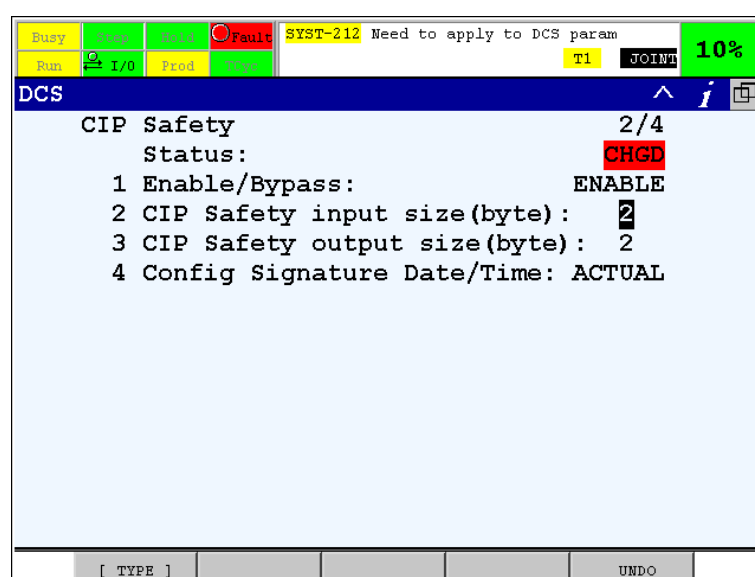
6   Move the cursor to the "CIP Safety input size (byte)" and press the [ENTER] key.
7   The message "Do you want to change setting?" is displayed. Press F4, "Yes".
8   Press the [4] key and press the [ENTER] key.
9   Move the cursor to the "CIP Safety input size (byte)" and press the [ENTER] key.
10   Press the [4] key and press the [ENTER] key.

Busy	Step	Hold	⬤Fault	SYST-212 Need to apply to DCS param		⚠ⓘ	10%
Run	I/O	Prod	Tgge		T1	J⚠	

DCS   ∧ ⅈ 🗗

```
      CIP Safety                            3/4
          Status:                        CHGD
      1 Enable/Bypass:                   ENABLE
      2 CIP Safety input size(byte):      4
      3 CIP Safety output size(byte):     4
      4 Config Signature Date/Time: ACTUAL
```

| [TYPE] | | | | UNDO | |

11   Press the [PREV] key.

Busy	Step	Hold	⬤Fault	SYST-212 Need to apply to DCS param		⚠ⓘ	10%
Run	I/O	Prod	Tgge		T1	JOINT	

DCS   ∧ ⅈ 🗗

```
                                         3/7
      1 Safe I/O Status:
      2 Safe I/O connect:                OK
      3 CIP Safety:                      CHGD
      4 Safe I/O consistency check:      OK
      5 Safe I/O device:                 OK
      6 Signature number:
      7 Code number setup:
```

| [TYPE] | APPLY | DETAIL | | UNDO | > |

- 165 -

12   Press F2, [APPLY].
13   Enter the code number. (Default code number is "1111".)
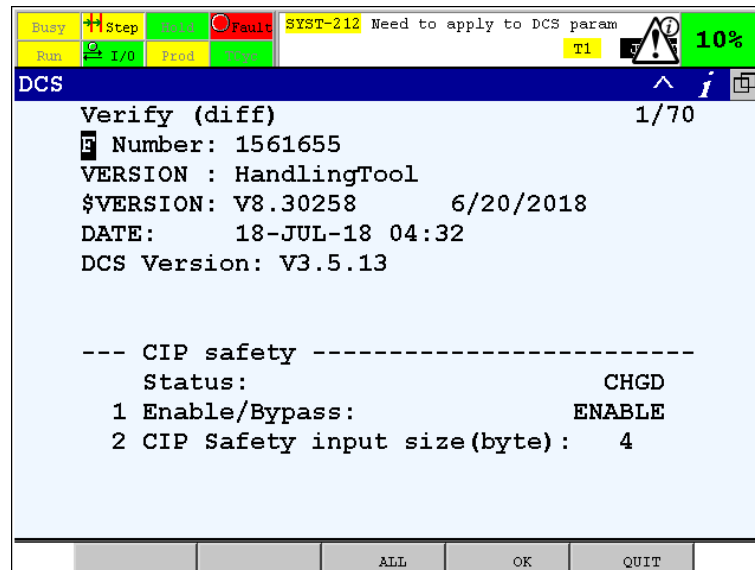
14   Press F4, [OK] after review the differences.
15   Cycle power of the robot controller.

C.3   CONFIRMATION OF THE CONNECTION STATUS

The communication status can be checked in the following screens.
- EIP CIP-Safety Operation screen (Subsection C.3.1)
- Safe I/O status screen (Subsection C.3.2)

C.3.1   EIP CIP-Safety Operation screen

The connection status can be checked in the EIP CIP-Safety Operation screen.

1   Press the [MENU] key.
2   Select "I/O".
3   Press F1, [TYPE].
4   Select "EtherNet/IP".

| Busy | Step | Hold | Fault | SYST-212 Need to apply to DCS param | | 10% |
| Run | I/O | Prod | | | T1 JOINT | |

```
I/O EtherNet/IP                                    ∧ 回
      EtherNet/IP List(Rack 89)              1/32
      Description   TYP   Enable   Status   Slot
      Connection1   ADP   FALSE    OFFLINE     1
      Connection2   ADP   FALSE    OFFLINE     2
      Connection3   ADP   FALSE    OFFLINE     3
      Connection4   ADP   FALSE    OFFLINE     4
      Connection5   ADP   FALSE    OFFLINE     5
      Connection6   ADP   FALSE    OFFLINE     6
      Connection7   ADP   FALSE    OFFLINE     7
      Connection8   ADP   FALSE    OFFLINE     8
      Connection9   ADP   FALSE    OFFLINE     9
      ConnectionA   ADP   FALSE    OFFLINE    10


      [ TYPE ]      PING      SAFETY    CONFIG    HELP      >
```

5   Press F3, [SAFETY].

| Busy | Step | Hold | Fault | SRVO-012 Power failure recovery | | 10% |
| Run | I/O | Prod | | | T1 | |

```
I/O EtherNet/IP                                    ∧ 回
      EIP CIP-Safety Operation               1/2

      Ethernet Interface:      Port 2
      Safety I/F Status:       RUNNING

      CIP-Safety Operation:
        DCS:                   Enable
        Input Size:            4
        Output Size:           4
        SNN:                   0004_0000_0001
        TX Count:              1436
        RX Count:              991



      [ TYPE ]              PREV    [CHOICE]    HELP
```

- 167 -

6   Check Safety I/F Status. Running, Idle or Pending is displayed. The following are the detail.

Running	There is an active safety I/O connection.
Idle	Safety PLC did not or could not make a safety I/O connection to the robot.
Pending	The configuration steps have not yet been completed. A power cycle is required to make changes active.

In this screen, the following parameters can also be confirmed.

DCS	Enabled or Bypassed.
Input size	1-8 bytes. This is set in DCS menu. (Subsection C.2.3)
Output size	1-8 bytes. This is set in DCS menu. (Subsection C.2.3)
SNN	Safety Network Number. This is set from safety PLC configuration software. When SNN shows "FFFF_FFFF_FFFF" or "????" it means the SNN has not been set from Safety PLC configuration software. Please follow the information in the Safety PLC manual.
TX Count	Transmit Count (packets sent to PLC). This count is dynamically updated so can be observed to increment when connection is active.
RX Count	Receive Count (packets received from PLC). This count is dynamically updated so can be observed to increment when connection is active.

C.3.2   Safe I/O status screen

The Safety signal status can be checked in the Safe I/O status screen.

1   Press the [MENU] key.
2   Select "SYSTEM".
3   Press F1, [TYPE].
4   Select "DCS".
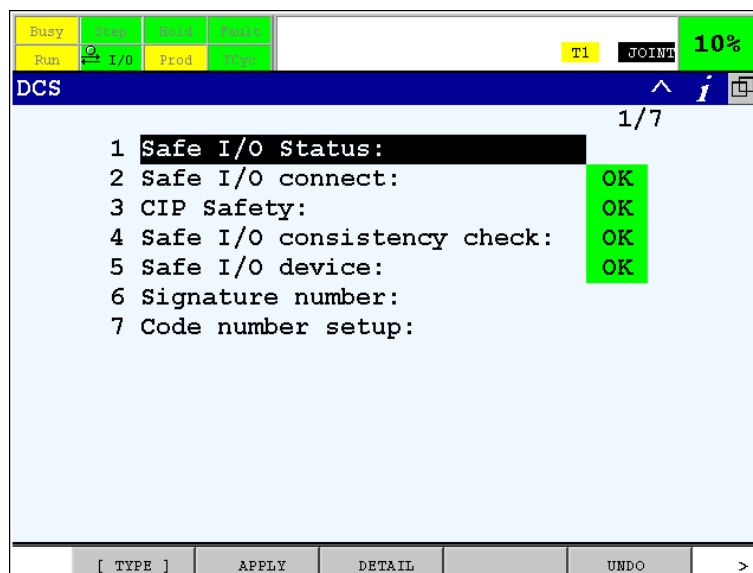
5   Move the cursor to the "Safe I/O Status" and press the [ENTER] key.

6   Press F2, [DATA].

7   Select "CSI".

8   Check that CSI status is changed on this screen when safety output signal corresponding to the CSI is
changed from Safety PLC configuration software.

NOTE
   Refer to your Safety PLC manual for the Safety PLC configuration.

INDEX

REVISION RECORD

Edition	Date	Contents
04	Oct., 2022	• Addition of R-30iB Mini Plus. • Addition of NOTE about Large Forward Open in 2.2 SPECIFICATION OVERVIEW. • Addition of description regarding RPI and Reconnect in 4 SCANNER CONFIGURATION. • 7 EXPLICIT MESSAGING has been changed. • 9 DIAGNOSTICS AND TROUBLESHOOTING has been changed. • Correction of errors.
03	Nov., 2018	• Addition of R-30iB Mate, R-30iB Plus, R-30iB Mate Plus and R-30iB Compact Plus. • 4.2.6 Ethernet/IP Scanner Lite and 10 EtherNet/IP ENHANCED DATA ACCESS have been added. • C EtherNet/IP SAFETY SETUP PROCEDURE has been added. • Correction of errors.
02	Sep., 2012	• Revised as the manual for R-30iA/R-30iA Mate/R-30iB controller.
01	Feb., 2008	

B-82854EN/04