MITSUBISHI

Mitsubishi Industrial Robot

CRn-500 series INSTRUCTION MANUAL

Ethernet Interface

MELFA

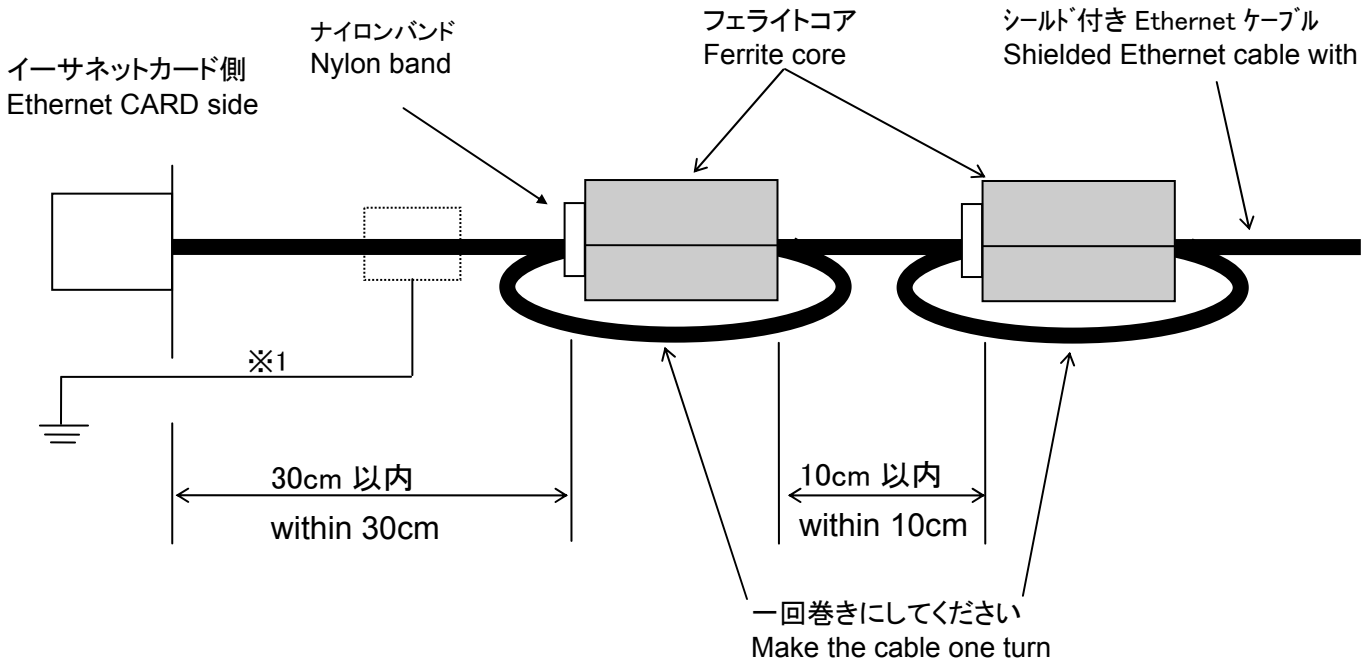
BFP-A8108-D

■ CE マーキング対策部品取付方法説明書
■ EMC Installation guideline and procedure

Ethernet ｹｰﾌﾞﾙへのﾌｪﾗｲﾄｺｱ取り付け要領
Coupling procedure of Ferrite core for Ethernet cable

ロボットコントローラ内蔵の Ethernet カードと Ethernet 機器（パソコンなど）を接続する Ethernet ケーブルに、添付のフェライトコアを下図のように 2 個取り付けてください。1 つはイーサネットカード側から 30cm 以内に、もう 1 つは1つ目のフェライトコアから 10cm以内に取り付けてください。
また、使用する Ethernet ケーブルは、シールド付きのものを使用ください。
それ以外は、ノイズによる誤動作を起こす可能性があります。

The two Ferrite cores should be installed to the Ethernet cable between controller and other Ethernet devices (See below).
Install one ferrite core in less than 30cm from the Ethernet card, and install another one in less than 10cm from that ferrite core.
Please use the shielded Ethernet cable under the environment of noise immunity.
If the customer do not install to the Ethernet cable with the Ferrite core, it will be become a trouble by Immunity and emission noise.



※1 注意 -Caution-
もし、ノイズによる影響を受けやすい環境下でのご使用の場合は、ケーブルのカバーを剥き、アース端子を利用してシールドを直接筐体のアースに落としてください。
If necessary, in case of under the environment of much immunity noise, remove the sheath of the Ethernet cable, and connect the shield that is inside a cable with the Earth [PE] terminal directly by cable.

■History

Print date	Instruction manual No.	Revision content
2000-04-17	BFP-A8108Z	First print
2000-04-24	BFP-A8108	Formal style
2000-07-06	BFP-A8108-A	The Real-time external control function was added.
2002-10-04	BFP-A8108-B	The section "1.5 Checking the robot controller software version" was added. The new function of the software version H7 of the controller was added. 1) The client function of the data link . 2) Add the current monitor of the real-time external control function. Change the structure of the communication packet. 3) Change the sample program.
2009-06-23	BFP-A8108-C	Ferrite core was added in Confirmation of product and EMC Installation guideline and procedure.
2009-09-30	BFP-A8108-D	The EC Declaration of Conformity was changed. (Correspond to the EMC directive; 2006/42/EC)

■ Preface

Thank you very much for employing Mitsubishi Electric Industrial Robot CRn-500 series.

The Ethernet interface is an option to add various network functions to the robot controller in combination with CRn-500 series controller. Before use, be sure to read through this document for sufficient understanding. Then make the most use of the Ethernet interface.

And also, the Ethernet interface corresponds from the software version E2 edition of the robot controller. Depending on the software version, a part of function of the Ethernet option does not operate. For details, refer to the Table "The software version and function of the controller".

(Refer to the "1.5 Confirming the software version of the robot controller " in this manual for confirming the version.)

Table: The software version and function of the controller

Software version of the robot controller	Controller communication function	Data link function (server)	Data link function (Server/Client)	Real-time external control function
A*, B*, C*, D*, E1	The Ethernet option does not operate.			
E2 to E4	O	O	X	X
F*, G*, H1 to H6	O	O	X	O
H7 or later	O	O	O	O

O ... Operate
X ... Don't operate

⚠ Safety Precautions

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

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

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

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

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

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

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

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

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

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

⚠**CAUTION**	Use the robot within the environment given in the specifications. Failure to do so could lead to a drop or reliability or faults. (Temperature, humidity, atmosphere, noise environment, etc.)
⚠**CAUTION**	Transport the robot with the designated transportation posture. Transporting the robot in a non-designated posture could lead to personal injuries or faults from dropping.
⚠**CAUTION**	Always use the robot installed on a secure table. Use in an instable posture could lead to positional deviation and vibration.
⚠**CAUTION**	Wire the cable as far away from noise sources as possible. If placed near a noise source, positional deviation or malfunction could occur.
⚠**CAUTION**	Do not apply excessive force on the connector or excessively bend the cable. Failure to observe this could lead to contact defects or wire breakage.
⚠**CAUTION**	Make sure that the workpiece weight, including the hand, does not exceed the rated load or tolerable torque. Exceeding these values could lead to alarms or faults.
⚠**WARNING**	Securely install the hand and tool, and securely grasp the workpiece. Failure to observe this could lead to personal injuries or damage if the object comes off or flies off during operation.
⚠**WARNING**	Securely ground the robot and controller. Failure to observe this could lead to malfunctioning by noise or to electric shock accidents.
⚠**CAUTION**	Indicate the operation state during robot operation. Failure to indicate the state could lead to operators approaching the robot or to incorrect operation.
⚠**WARNING**	When carrying out teaching work in the robot's movement range, always secure the priority right for the robot control. Failure to observe this could lead to personal injuries or damage if the robot is started with external commands.
⚠**CAUTION**	Keep the jog speed as low as possible, and always watch the robot. Failure to do so could lead to interference with the workpiece or peripheral devices.
⚠**CAUTION**	After editing the program, always confirm the operation with step operation before starting automatic operation. Failure to do so could lead to interference with peripheral devices because of programming mistakes, etc.
⚠**CAUTION**	Make sure that if the safety fence entrance door is opened during automatic operation, the door is locked or that the robot will automatically stop. Failure to do so could lead to personal injuries.
⚠**CAUTION**	Never carry out modifications based on personal judgments, or use non-designated maintenance parts. Failure to observe this could lead to faults or failures.
⚠**WARNING**	When the robot arm has to be moved by hand from an external area, do not place hands or fingers in the openings. Failure to observe this could lead to hands or fingers catching depending on the posture.
⚠**CAUTION**	Do not stop the robot or apply emergency stop by turning the robot controller's main power OFF. If the robot controller main power is turned OFF during automatic operation, the robot accuracy could be adversely affected.

Contents

1. Before use

This chapter describes the confirmation items and cautionary items which must be read before practical use of the Ethernet interface.

1.1. How to use the instruction manual

1.1.1. Content of instruction manual

Through the following configuration, this document introduces the functions which are added or changed in the Ethernet interface. For the functions and their operating methods provided in the standard robot controller, refer to "instruction manual" appended to the robot controller.

Table 1.1 Content of instruction manual

Chapter	Title	Content
1	Before use	In addition to the using method of the instruction manual, the confirmation items and cautionary items are introduced to use the Ethernet interface.
2	Preparation before use	The preparatory work is introduced to use the Ethernet interface. Referring to the chapter, install the interface card, apply the cabling and wiring and confirm the other setting items.
3	Trial for operation	Using the system configured in "This document/Chapter 2 Preparation before use", it introduces a series of the operating methods from the start-up to the stop. Referring to each introduction, understand the basic operating method.
4	Explanation of functions	The method to operate the Ethernet interface is introduced to each operation function. The details of each operation method are introduced in this chapter.
5	Appendix	Since the added errors when indexing the terms or using the Ethernet interface are herein described, refer to this chapter as necessary.

1.1.2. Symbols of instruction manual

This manual uses the symbols and their expressions as shown in table 1.2.

Table 1.2 Symbol of instruction manual

Symbol	Meaning
[JOINT]	If [ ] is added in the sentence as shown in the left, it means the key of the teaching pendant. (14) It means that (B) key is pressed with (A) key pressed.
[STEP/MOVE] + [+X(J1)] (A) (B)	It means that (B) key is pressed with (A) key pressed. This example (jog operation) means that [+X(J1)] key is pressed with [STEP/MOVE] pressed.
[STEP/MOVE] + ([ADD ↑] → [RPL ↓]) (A) (B) (C)	It means that (B) key is pressed and released with (A) key pressed, and then (C) is pressed. This example (position compensation) means that [ADD ↑] key is pressed and released with [STEP/MOVE] key pressed, and [RPL ↓] key is pressed.

1.2. Terms used in the instruction manual

The following terms are used in this document.

(1) Ethernet interface

The Ethernet interface is an option to add various network functions to the robot controller in combination with CRn-500 series controller.

(2) Network personal computer

The personal computer is a commercially available one which provides the network function, integrating the Ethernet interface card. Windows95/Windows98/ Me/WindowsNT4.0 Workstation/ Windows2000/ WindowsXP are applicable as the operating system.

(3) 10Base-5/10Base-T

These cable standards are specified by the Ethernet.

10Base-5 allows the installation of the equipment which is called the transceiver, being connected to the transceiver with the exclusive transceiver cable.

10Base-T is a connection system which uses the twist pair cable line, providing the equipment which is called the hub and allowing the network to be connected in the star arrangement with the hub centered. When the hub is used, the straight cable is used, and when two units are directly connected to each other one to one, the cross cable is used. Here, 10Base-T is currently popular since it is easier for cable wiring, relatively cheaper and easily available at the commercial shop.

(4) MELFA-BASICIV/MOVEMASTER command

This is a type of robot language.

The CRn-5xx controller is provided with either the MELFA-BASICIV language or MOVEMASTER command language. MELFA-BASICIV is a high-function language that allows the program to be described in the same manner as general BASIC. The MOVEMASTER command language has been popular with the Mitsubishi compact robot MOVEMASTER Series and E/EN Series, etc.

This option will function with either language.

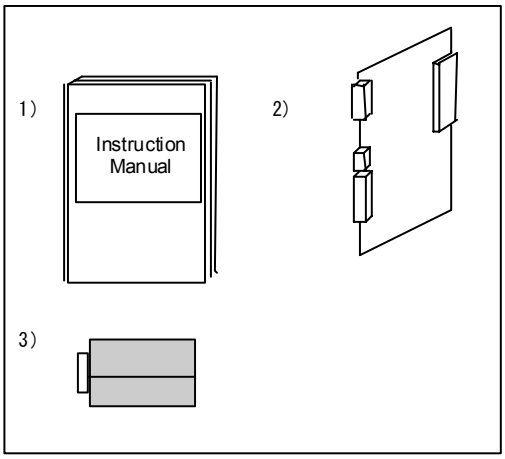
⚠CAUTION

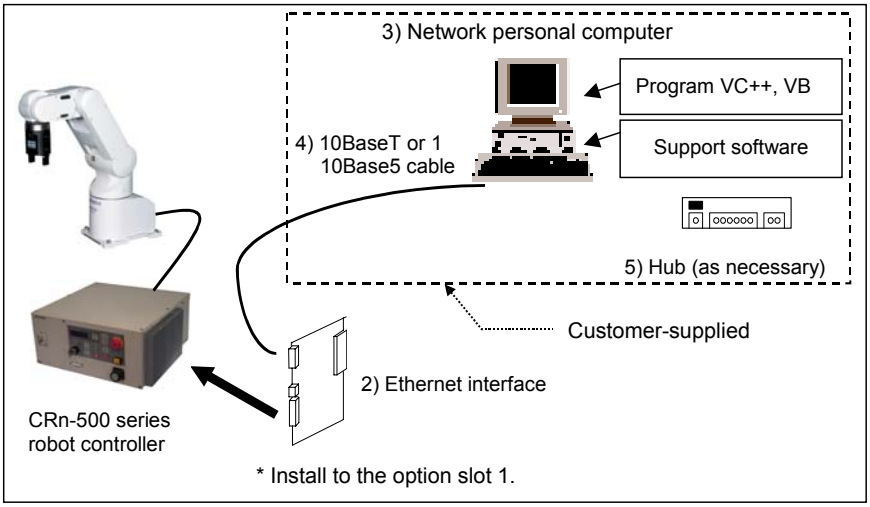
The MOVEMASTER commands can be used only with some robot models (RV-1A/RV-2AJ, etc.). Thus, only MELFA-BASICIV may be provided depending on the model being used.
Refer to the instruction manual enclosed with the robot in use for details on which language can be used.
As a default, the language is set to MELFA-BASICIV. The parameter RLNG must be changed to change the robot language. Refer to the enclosed instruction manual for details.

1.3. Confirmation of product

The standard configuration of the product supplied by the customer is as follows. Confirm the configuration.

No.	Part name	Type	Qty.
1)	Instruction manual (this document)	BFP－A8108	1
2)	Ethernet interface card	HR533	1
3)	Ferrite core	E045R301334	2





In addition to the standard robot system configuration, the following is necessary. These devices are separately procured by the customer.

No.	Part name	Type	Qty.
3)	Network personal computer (Network interface is necessary.)	Personal computer operated by Windows95/98/Me/WindowsNT4.0 Workstation/Windows2000/WindowsXP. In addition, the computer with TCP/IP network functions, such as LinuxOS . (Operation is not verified)	1 or more
4)	Ethernet cable (Select the straight cable or cross cable depending on the connection system.)	10Base-T or10Base-5	1 or more

Prepare the following as necessary.

5)	Hub (Necessary if it is used in the LAN environment.)	(Goods on the market)	1
6)	Robot controller programming aiding tool corresponding to Windows for NARC controller of our company	(An optional) Personal computer Support Software	1
7)	Application for network communication program production corresponding to Windows	(Goods on the market) Microsoft. VisualC++5.0/6.0, etc.	1

1.4. Ethernet interface

1.4.1. Function of Ethernet interface

The Ethernet interface has the following functions.

(1)The connection with 10baseT or 10base5 is supported.

(2)TCP/IP protocol is used to allow the communication with the personal computer on the Ethernet.

(3)This one card alone can be installed on one controller. The card is installed in the optional slot 1.

(4)The sampling program (corresponding to Microsoft Visual Basic Version 5.0) of the personal computer is equipped.

The following is provided as the samples. (Refer to Chapter 5 Appendix.)

• The data link function is used to transmit and receive the variables of personal computer and robot (characters and numerical values). (OPEN/INPUT#/PRINT#)

Here, approve that the result of the operation of the application which the customer produces on the basis of the sample is out of the responsibility with our company.

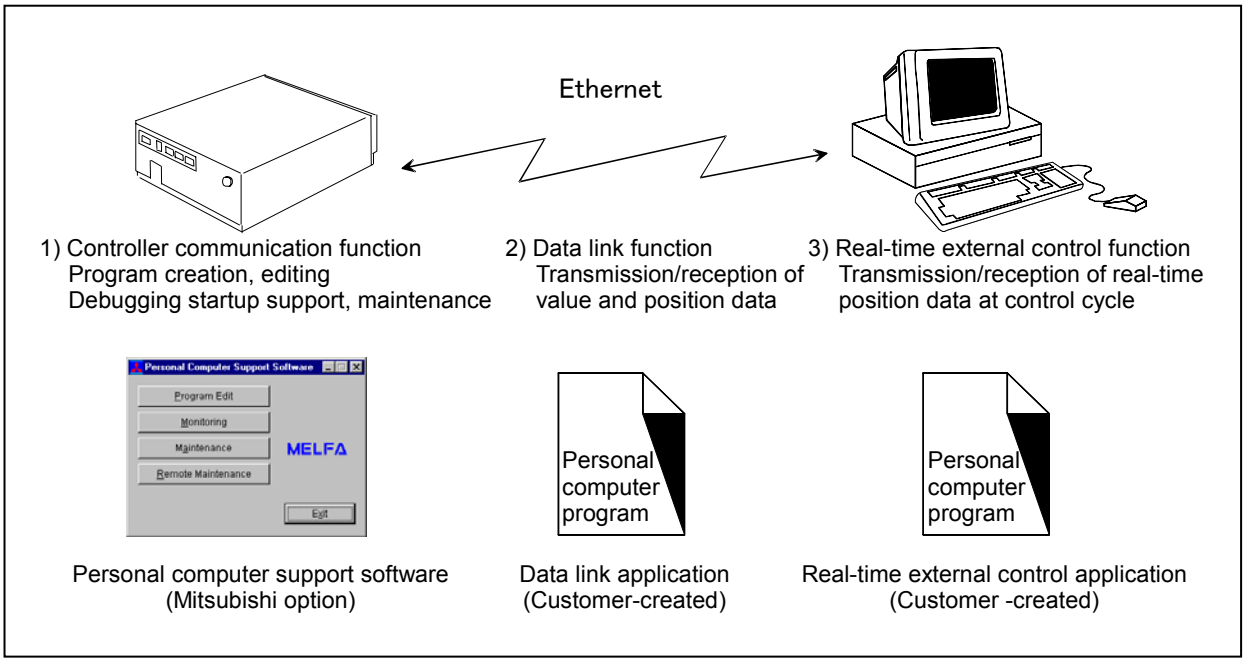
(5)The three Ethernet functions are described below.

Refer to the section "4. Explanation of each function" for details on each function.

No.	Outline of function	Remarks	Reference page
1)	Controller communication function Data can be communicated with the robot controller via Ethernet. (Program upload/download, status monitor, etc.) Personal computer support software (optional) is available as an application.	* Communication with up to 16 clients is possible.	Chapter 1 General Chapter 2 General Chapter 3. 1 Chapter 4. 1 Chapter 5. 1
2)	Data link function The value and position data can be linked between the robot program and personal computer using MELFA-BASICIV language (OPEN/PRINT/INPUT command).	* By changing the communication open destination COM No., communication with applications in up to 8 clients is possible.	Chapter 1 General Chapter 2 General Chapter 3. 2 Chapter 4. 2 Chapter 5. 1 Chapter 5. 2.1
3)	Real-time external control function The position command data can be retrieved and operated at the robot motion control cycle unit. Joint, XYZ or motor pulse can be designated for the position data. It is also possible to monitor the input/output signals or output the signals simultaneously. Control is started with the MXT command (MELFA-BASICIV language and MOVEMASTER command). This function is valid only with the following models. *RP-1AH/3AH/5AH Series *RV-1A *RV-4A/3AL/4AC/3ALC Series	* The user must create an application program on the personal computer side to control the robot. Communication is carried out one-on-one.	Chapter 1 General Chapter 2 General Chapter 3. 3 Chapter 4. 3 Chapter 5. 1 Chapter 5. 2.2

* The personal computer used to communicate with the robot controller must be located on the same network.

Communication cannot be carried out over firewalls (from internet) or over gateways (from different adjacent network, etc.). Consider operation with a method that communicates via a server (i.e., HTTP server, etc.) connected to the same network as the robot controller. Pay special attention to safety and response in this case.



1.5. Checking the robot controller software version

The Ethernet interface is compatible from robot controller software version E2. The robot controller software version E5 and above must be installed to use the real-time external control function. Check the controller software version with the following procedure before starting use.

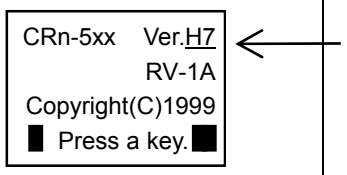
When using the controller software version A*, B*, C*, D* or E1, the functions will not activate even when the Ethernet interface board is installed. Contact Mitsubishi in this case.

* Checking the software version on the teaching pendant screen

Set the teaching pendant to "DISABLE", and turn ON the robot controller power.

No. 2 shown below indicates the controller software version.

No.	Teaching pendant screen display	Explanation
1	P28TB Ver.B2 Robot System Teaching Box JUST A MINUTE.	First, the teaching pendant software version will appear briefly. Teaching pendant software version: B2 Wait for several seconds.

2		Next, the controller software version will appear. Controller software version: H7 *) For example, Ver. A* will appear for Version A (*: Value 1 or higher)

2. Preparation before use

What is done before use is described.

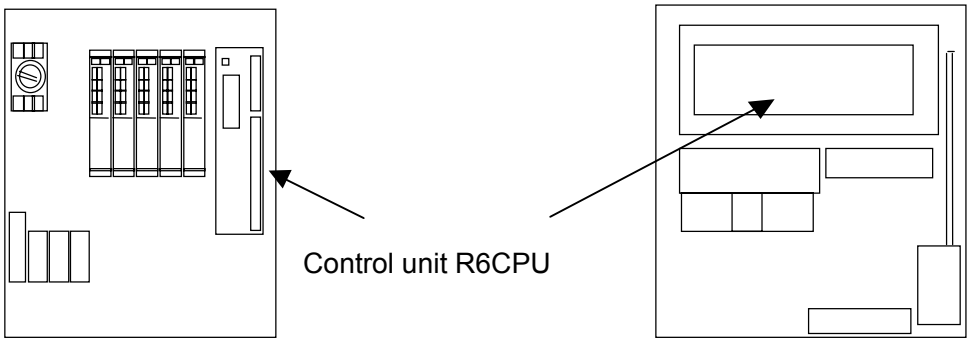
Installation of Ethernet interface … Refer to 2.1.

↓

Connection of Ethernet cable … Refer to 2.2.

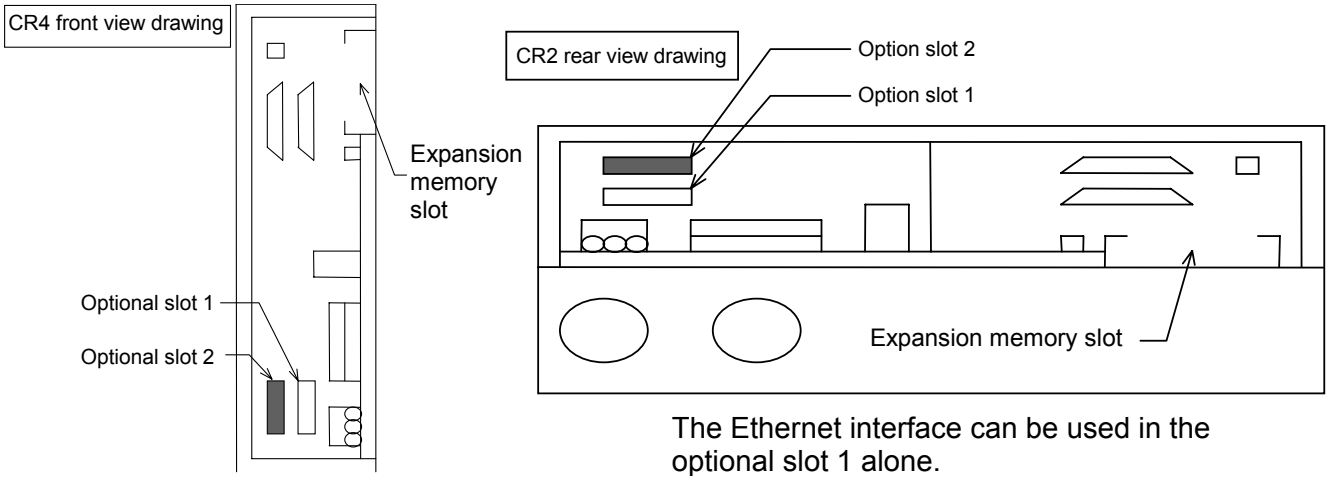
↓

Parameter setting … Refer to 2.3.



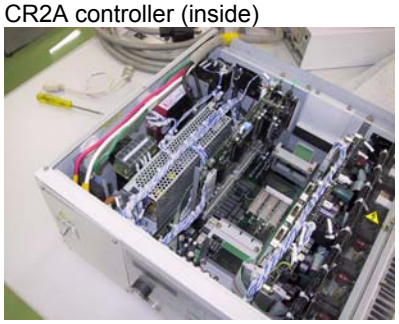
CR4 controller (Front view)

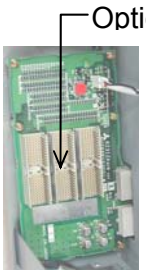
CR2 controller (Top view: inside)

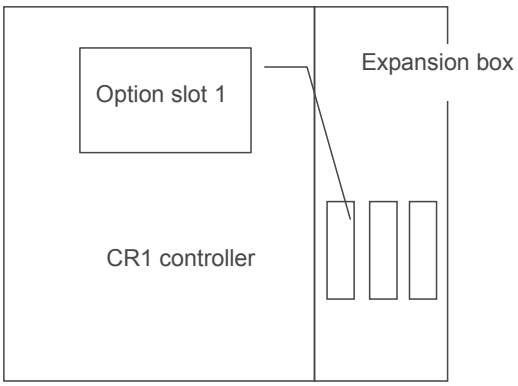


The Ethernet interface can be used in the optional slot 1 alone.

CR2A controller (inside)







Install the CR1 controller in the expansion option box. Refer to the separate manual "Controller setup, basic operation, and maintenance" for detail.

2.1. Installation of Ethernet interface

The Ethernet interface is installed in the controller. For details of the removal, etc. of the controller box cover, refer to the instruction manual of the controller.

⚠CAUTION

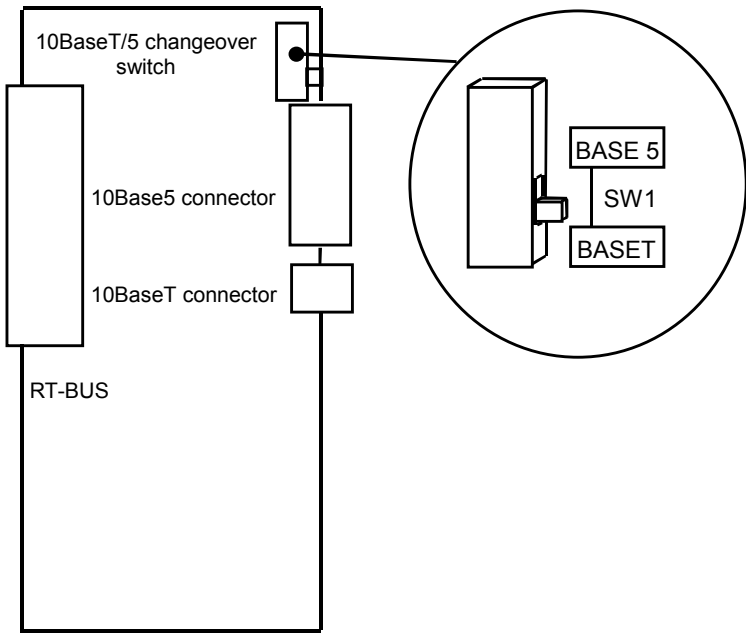
Since the card uses the electronic parts, they may sometimes be destroyed by static electricity.
Reading through the cautionary items described on the bag which packs the interface card, carefully handle the card.

2.1.1. 10BaseT/5 changeover switch setting

Depending on the type of the applied cable, set 10BaseT/5 changeover switch SW1.

For 10BaseT, set the changeover switch at "BASET" (lower side), and for 10Base5, set the changeover switch at "BASE5" (upper side).

SW1 is located at the upper right of the Ethernet interface board. (Refer to the following drawing.)

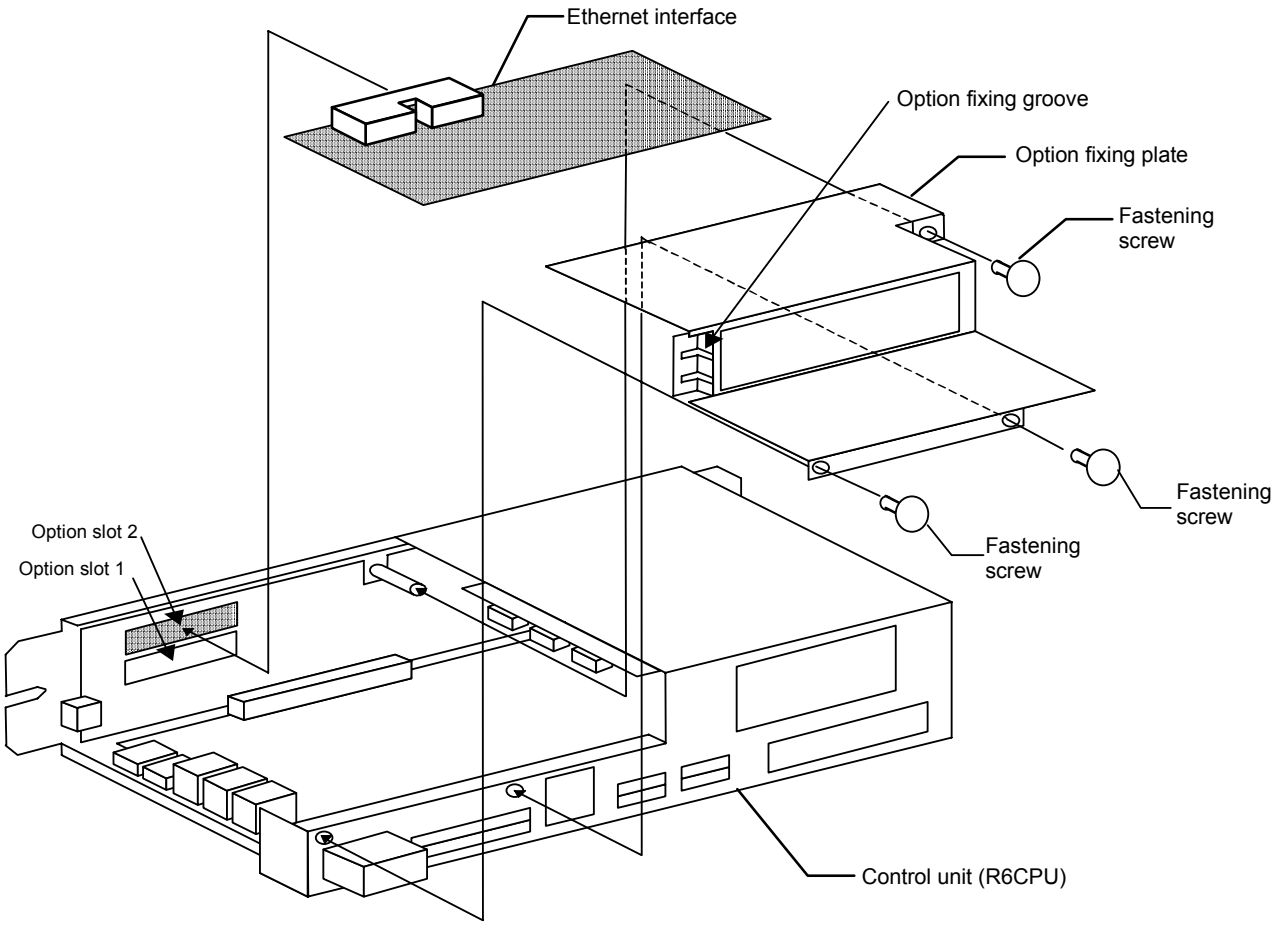


2.1.2. Installation of interface card on the controller

The procedure to install the Ethernet interface is herein described.
When using the CR1 controller, refer to "Installation of optional device" of the instruction manual of "CR1 controller, controller setup, basic operation and maintenance".
The Ethernet interface is installed in the control unit (R6CPU unit) of the controller or in the optional slot 1 (OPT1) of the expansion option box. For details of the control unit (R6CPU unit), refer to the instruction manual "controller setup, basic operation and maintenance".

Procedure to install the Ethernet interface

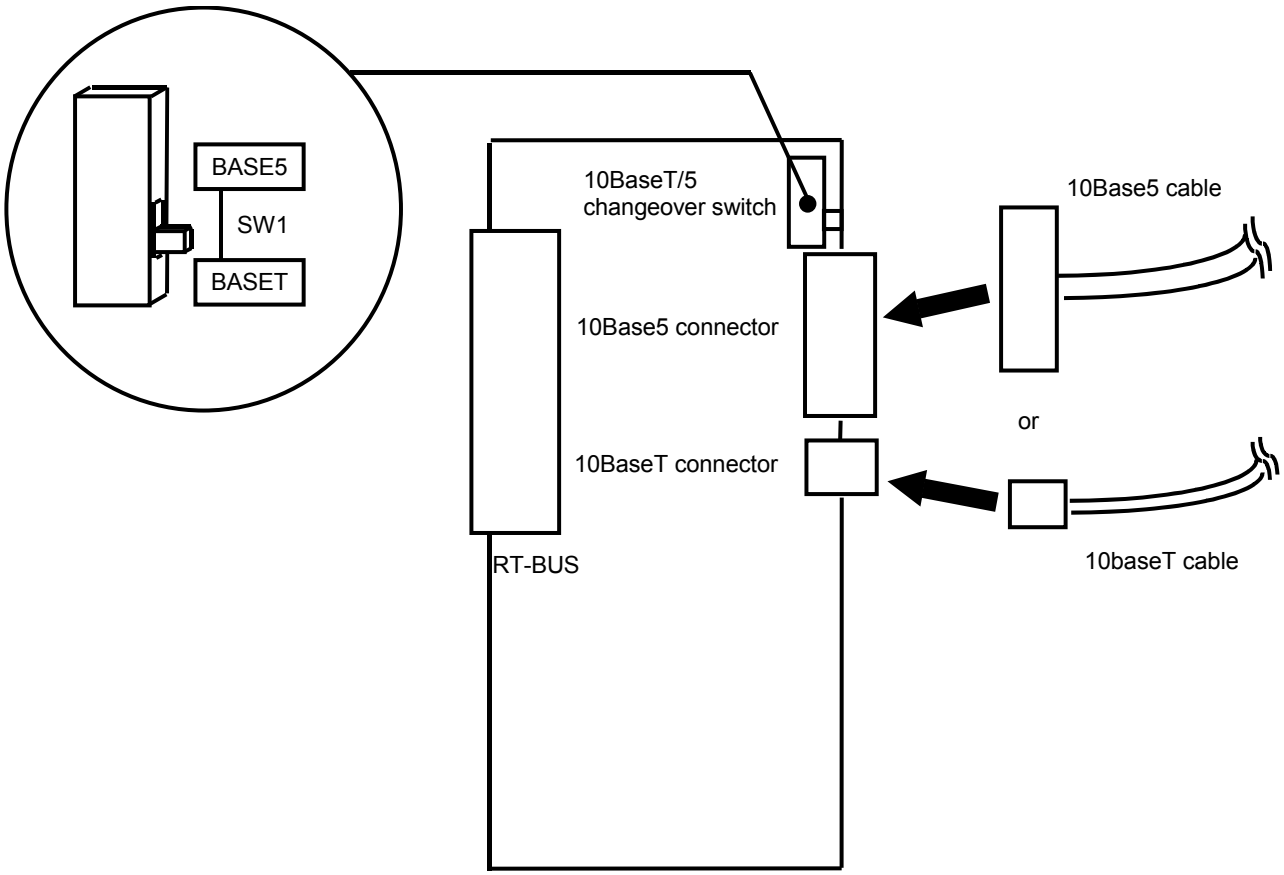
(1) Remove the optional fixing plate of the control unit (R6CPU). (Three fastening screws)
(2) Insert the Ethernet interface to the optional slot 1 (OPT1).
(3) Install the option fixing plate, engaging the end of the Ethernet interface into the option fixing groove. Reversing procedure (1), tighten the fastening screws (3 places) for fixation.
(4) Referring to "2.2 Connection of Ethernet cable", connect the Ethernet cable to the Ethernet interface.
(5) Process the outlet port of the Ethernet cable connected. For details, refer to the instruction manual "controller setup, basic operation and maintenance" of each controller.



2.2. Connection of Ethernet cable

As shown below, connect the Ethernet cable of 10BaseT or 10Base5 to the connector of the interface card.

When the hub is used, use the straight cable. Or when the personal computer and controller are connected to each other one to one, use the cross cable.



2.3. Parameter setting

Before use, it is necessary to set the following parameters. The parameters which are set on the robot controller are shown in the following list. For the method to set the parameter, refer to the instruction manual of the controller.

⚠CAUTION After changing the parameters, turn the power supply of the controller from OFF to ON. Unless this is done, the changed parameters will not become valid.

2.3.1. Parameter list

The parameters are listed below. For details of the parameters, refer to "2.3.2 Details of parameters".

O ... Setting is necessary
- ... Setting is unnecessary

Parameter list

Parameter name	Details	Number of elements	Default value	Controller communication function	Data link function	Real-time control function
NETIP	IP address of robot controller	Character string 1	“192.168.0.1”	O	O	O
NETMSK	Sub-net-mask	Character string 1	“255.255.255.0”	O	O	O
NETPORT	Port No. Range 0 to 32767 For function expansion (reserved), ---------- Correspond to OPT 11-19 of COMDEV (OPT11) (OPT12) (OPT13) (OPT14) (OPT15) (OPT16) (OPT17) (OPT18) (OPT19)	Numerical value 10	10000, 10001, 10002, 10003, 10004, 10005, 10006, 10007, 10008, 10009	O	O	O
CPRCE11 CPRCE12 CPRCE13 CPRCE14 CPRCE15 CPRCE16 CPRCE17 CPRCE18 CPRCE19	Protocol 0: No-procedure 1: Procedure, 2: Data link (1: Procedure has currently no function.) Correspond to OPT 11-19 of COMDEV (OPT11) (OPT12) (OPT13) (OPT14) (OPT15) (OPT16) (OPT17) (OPT18) (OPT19)	Numerical value 9	0 0 0 0 0 0 0 0 0	-	O	-
COMDEV	Definition of device corresponding to COM1: to 8 Definition of device corresponding to COM1:, Definition of device corresponding to COM2:, Definition of device corresponding to COM3:, Definition of device corresponding to COM4:, Definition of device corresponding to COM5:, Definition of device corresponding to COM6:, Definition of device corresponding to COM7:, Definition of device corresponding to COM8: .	Character string 8	RS232C, , , , , , ,	-	O	-

Parameter name	Details	Number of elements	Default value	Controller communication function	Data link function	Real-time control function
	When the data link is applied, setting is necessary. OPT11 to OPT19 are allocated. Here, RS-232C of the controller is previously allocated to COM1: .					
NETMODE The software version H7 or later.	Server designation (1: Server, 0: Client) (OPT11) (OPT12) (OPT13) (OPT14) (OPT15) (OPT16) (OPT17) (OPT18) (OPT19)	Numerical value 9	1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1	-	O	-
NETHSTIP The software version H7 or later.	The IP address of the data communication destination server. * It is valid if specified as the client by NETMODE only. (OPT11) (OPT12) (OPT13) (OPT14) (OPT15) (OPT16) (OPT17) (OPT18) (OPT19)	Character string 9 .	192.168.0.2 , 192.168.0.3 , 192.168.0.4 , 192.168.0.5 , 192.168.0.6 , 192.168.0.7 , 192.168.0.8 , 192.168.0.9 , 192.168.0.10	-	O	-
	Communication destination IP address for real-time external control command					
MXTCOM1	The address to set up in the communication point number 1.	Value 1	192.168.0.2	-	-	O When the MOVEMASTER COMMAND is used
MXTCOM2	The address to set up in the communication point number 2.	Value 1	192.168.0.3			
MXTCOM3	The address to set up in the communication point number 3.	Value 1	192.168.0.4			
MXTTOUT	Timeout time for executing real-time external control command (Multiple of 7.1msec, Set -1 to disable timeout)	Value 1 (0-32767)	-1	-	-	O

2.3.2. Details of parameters

The parameters are herein described in details.

(1) NETIP (IP address of robot controller)

The IP address of the robot controller is set. IP address is like the address of the mail.

The format of IP address is composed of 4 numbers of 0 to 255 and the dot (.) between the numbers.

For example, it is set as 192.168.0.1 or 10.97.11.31.

If the controller and network personal computer are directly connected to each other one-to-one, it is allowed to set default value (a random value) but if it is connected to the local area network (LAN), IP address must be set as instructed by the manager of customer's LAN system.

If any IP addresses are overlapped, the function will not properly operate. Therefore, take care to prevent it from being overlapped with another during setting.

The personal computer used for communication with the robot controller must be located on the same network.

(2) NETMSK (sub-net-mask)

Set the sub-net-mask of the robot controller. Among the IP addresses, the sub-net-mask is set to define the sub-net-work.

The format of the sub-net-mask is composed of 4 numbers of 0 to 255 and the dot (.) between the numbers.

For example, it is set as 255.255.255.0 or 255.255.0.0.

As usual, it is allowed to set default value. If it is connected to the local area network (LAN), the sub-net-mask must be set as instructed by the manager of customer's LAN system.

(3) NETPORT (port No.)

The port No. of the robot controller is set. The port No. is like the name of the mail.

For the nine elements, the port numbers are each expressed with a value.

The first element (element No. 1) is used for real-time control.

The second to ninth elements (elements No. 2 to 9) are used for the support software or data link.

Normally, the default value does not need to be changed. Make sure that the port numbers are not duplicated.

(4) CRRCE11 to 19 (protocol)

When using the data link function, the setup is necessary.

Sets the protocol (procedure) for communication. The protocol has three kinds of no-procedure, procedure and data link.

0... No-procedure: The protocol is applied to use the personal computer Support Software .

1... Procedure: Reserved. (Since it is not any function, don't set it by mistake.)

2... Data link: The protocol is used to use OPEN/INPUT/PRINT commands for communication.

(5) COMDEV (Definition of devices corresponding to COM1: to 8)

When using the data link function, the setup is necessary.

Definition of device corresponding to COM1: to 8 is set. COM1: to 8 is used for OPEN command of the robot program.

Be sure to set it only when the data link is specified on setting of the protocol (CPRCE11 to 19).

The setting values of the Ethernet interface option correspond to the port Nos. which are set at the parameter NETPORT.

* In the following parameters NETOPORT (n) and COMDEV(n), n indicates the element No. of that parameter.

n	The device name set up by COMDEV(n)	Port number
1	OPT11	The port number specified by NETPORT(2)
2	OPT12	The port number specified by NETPORT(3)
3	OPT13	The port number specified by NETPORT(4)
4	OPT14	The port number specified by NETPORT(5)
5	OPT15	The port number specified by NETPORT(6)
6	OPT16	The port number specified by NETPORT(7)
7	OPT17	The port number specified by NETPORT(8)
8	OPT18	The port number specified by NETPORT(9)
9	OPT19	The port number specified by NETPORT(10)

For example, if the port No. specified at NETPORT(3) is allocated to the data link of COM:3, the following will be applied.

COMDEV(3) = OPT13 * OPT13 is set at 3rd element of COMDEV.

CPRCE13 = 2 * Set up as a data link.

(6) NETMODE (server specification). * The software version H7 or later.
Set up, when using the data link function.
Set the TCP/IP communication in the data link function of the robot controller as the server or the client.
It is necessary to change with the application of the equipment connected to the robot controller.
This function corresponds to the software version H7 or later.
In the version older than H7, the robot controller operates only as a server.

(7) NETHSTIP (The IP address of the server of the data communication point). * The software version H7 or later .
Set up, when using the robot controller as a client by the data link function.
Specify the IP address of the partner server which the robot controller connects by the data link function.
Set up, when only set the robot controller to the client by server specification of NETMODE.

(8) MXTCOM1 to 3 (Communication destination IP address for real-time external control command)
This is set only when using the real-time external control function with the MOVEMASTER command robot language.
Designate the IP address for the robot controller communication destination personal computer.

(9) MXTTOUT (Timeout setting for executing real-time external control command)
This is changed when using real-time external control command and setting the timeout time for communication with the robot controller.
Set a multiple of the approx. 7.11msec control cycle.
When the real-time external control command is executed, the timeout time during which no communication data is received by the robot controller from the personal computer is counted up. If the count reaches the value set in MXTTOUT, the operation will stop with the error (#7820). For example, to generate an error when there is no communication for approx. 7 seconds, set 1000.
This setting is set to -1 (timeout disabled) as the default.

2.3.3. Example of setting of parameter 1 (When the Support Software is used)

The setting example to use the Support Software is shown below.

Set the parameters for the robot controller, and the network for the personal computer OS being used.

List Conditional example 1

IP address of robot controller	192.168.0.1
IP address of personal computer	192.168.0.2
Port No. of robot controller	10001

Set the robot controller parameters as shown below.

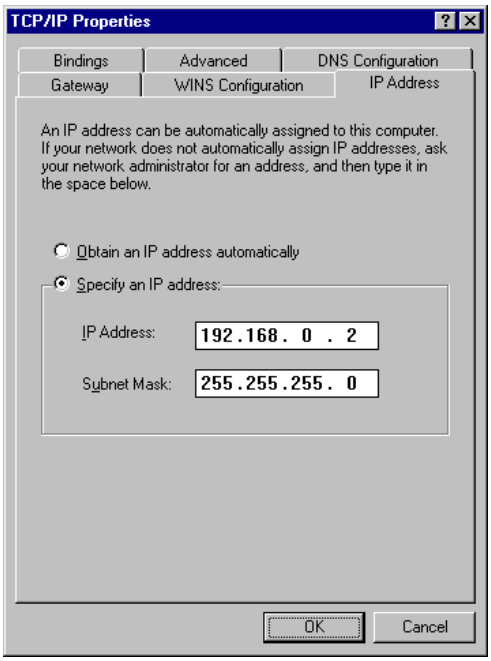
If the default settings are to be used, the parameters do not need to be changed.

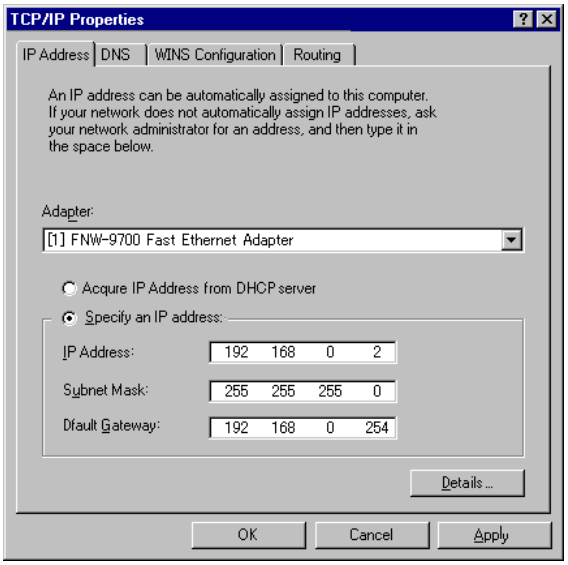
List Parameter change example 1

Parameter name to be changed	Before/after change	Parameter value
NETIP	Before	192.168.0.1
	After	192.168.0.1 (With the default value.)
NETPORT	Before	10001
	After	10001 (With the default value.)

Next, set the personal computer IP address to 192.168.0.2. Set this value on the Network Properties screen.

Windows 95 (lower left screen), Windows NT4.0 (lower right screen)





The personal computer IP address is set with the Windows TCP/IP Property Network setting (property in network computer). Because the set-up screen differs with versions of Windows, refer to the manuals enclosed with Windows, etc., for details on setting this address.

Refer to the instruction manuals enclosed with the personal computer support software for details on setting and using the personal computer support software.

2.3.4. Example of setting of parameter 2-1

(When the data link function is used: When the controller is the server)

Shows the example of the setting, when the controller is server by the data link function.

List Example of conditions 2-1

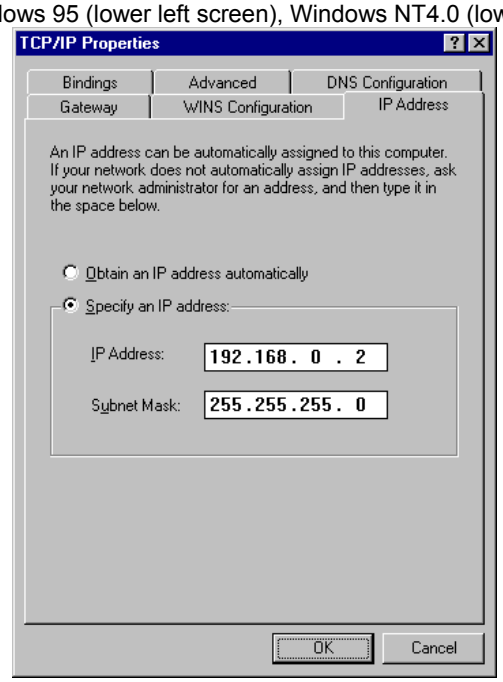
Robot controller IP address	192.168.0.1
Personal computer IP address	192.168.0.2
Robot controller port No.	10003
Communication line No. <For MELFA-BASICIV> OPEN command COM No. <For MOVEMASTER command> OPN command line No.	COM3: 3

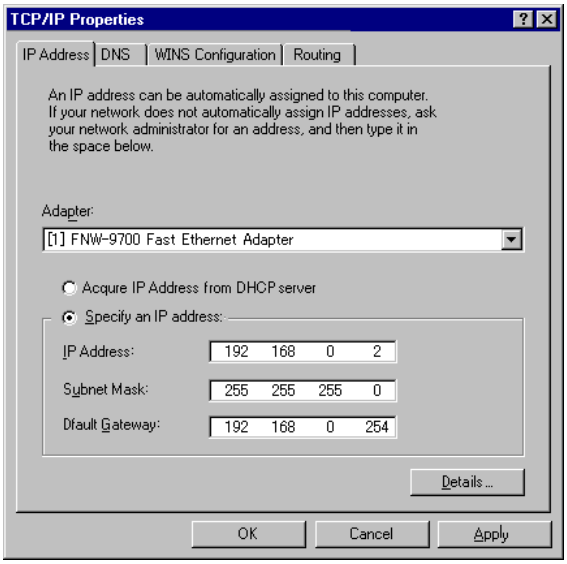
List Example of parameter changes 2-1

Name of parameter to change	Before/after changes	Parameter value
NETIP	Before	192.168.0.1
	after	〃 (Default value)
NETPORT	Before	10000,10001,10002,10003,10004,10005,10006,10007,10008,10009
	after	〃 (Default value)
CPRCE13	Before	0
	after	2
COMDEV	Before	RS232, , , , , ,
	after	RS232, , OPT13, , , , ,

Next, set the personal computer IP address to 192.168.0.2. Set this value on the Network Properties screen.

Windows 95 (lower left screen), Windows NT4.0 (lower right screen)





The personal computer IP address is set with the Windows TCP/IP Property Network setting (property in network computer). Because the set-up screen differs with versions of Windows, refer to the manuals enclosed with Windows, etc., for details on setting this address.

Refer to the instruction manuals enclosed with the personal computer support software for details on setting and using the personal computer support software.

2.3.5. Example of setting parameters 2-2

(When the data link function is used: When the controller is the client)

Shows the example of the setting, when the controller is client by the data link function.

List Example of conditions 2-2

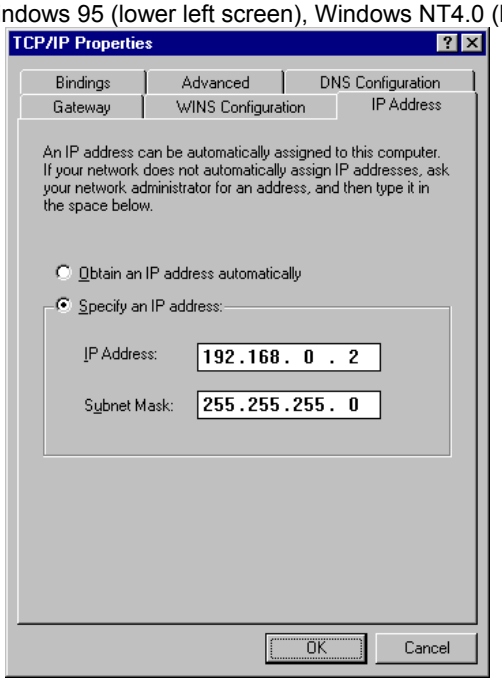
Robot controller IP address	192.168.0.1
Personal computer IP address	192.168.0.2
Robot controller port No.	10003
Communication line No. <For MELFA-BASICIV> OPEN command COM No. <For MOVEMASTER command> OPN command line No.	 COM3: 3

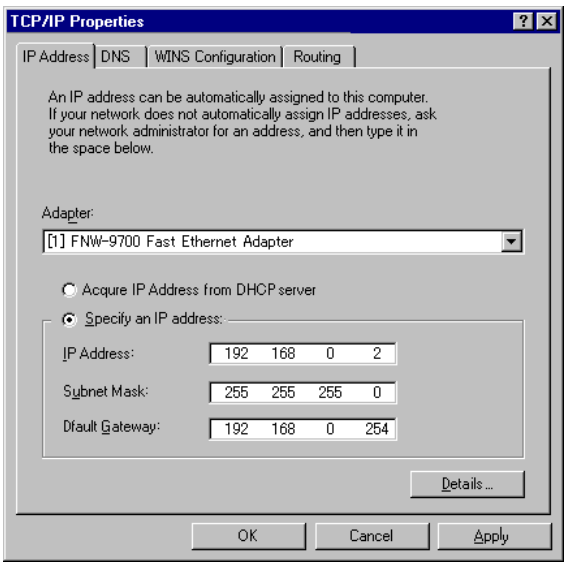
List Example of parameter changes 2-2

Name of parameter to change	Before/after changes	Parameter value
NEITP	Before	192.168.0.1
	After	192.168.0.1 (Default value)
NETPORT	Before	10000,10001,10002,10003,10004,10005,10006,10007,10008,10009
	After	10000,10001,10002,10003,10004,10005,10006,10007,10008,10009 (Default value)
CPRCE13	Before	0
	After	2
COMDEV	Before	RS232, , , , , ,
	After	RS232, , OPT13, , , , ,
NETMODE	Before	1,1,1,1,1,1,1,1,1,1
	After	1,1,0,1,1,1,1,1,1,1
NETHSTIP	Before	192.168.0.2, 192.168.0.3, 192.168.0.4, 192.168.0.5, 192.168.0.6, 192.168.0.7, 192.168.0.8, 192.168.0.9, 192.168.0.10
	After	192.168.0.2, 192.168.0.3, 192.168.0.2, 192.168.0.5, 192.168.0.6, 192.168.0.7, 192.168.0.8, 192.168.0.9, 192.168.0.10

Next, set the personal computer IP address to 192.168.0.2. Set this value on the Network Properties screen.

Windows 95 (lower left screen), Windows NT4.0 (lower right screen)





The personal computer IP address is set with the Windows TCP/IP Property Network setting (property in network computer). Because the set-up screen differs with versions of Windows, refer to the manuals enclosed with Windows, etc., for details on setting this address.

Refer to the instruction manuals enclosed with the personal computer support software for details on setting and using the personal computer support software.

2.3.6. Example of setting parameters 3 (for using the real-time external control function)

An example of the settings for using the real-time external control function is shown below.

List Example of conditions 3

Robot controller IP address	192.168.0.1
Personal computer IP address	192.168.0.2
Robot controller port No.	10000

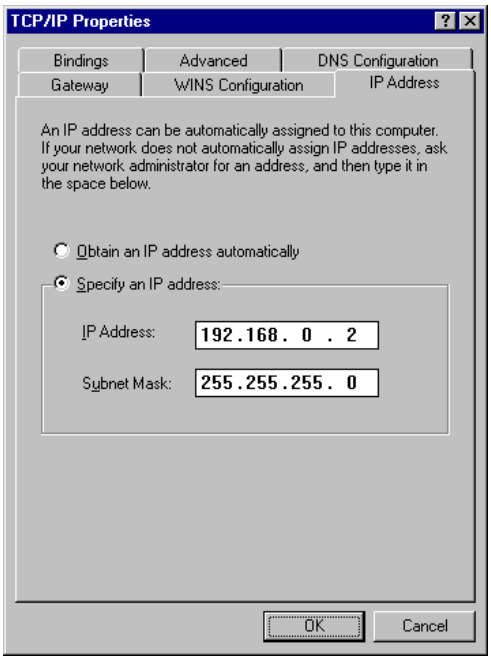
List Example of parameter changes 3

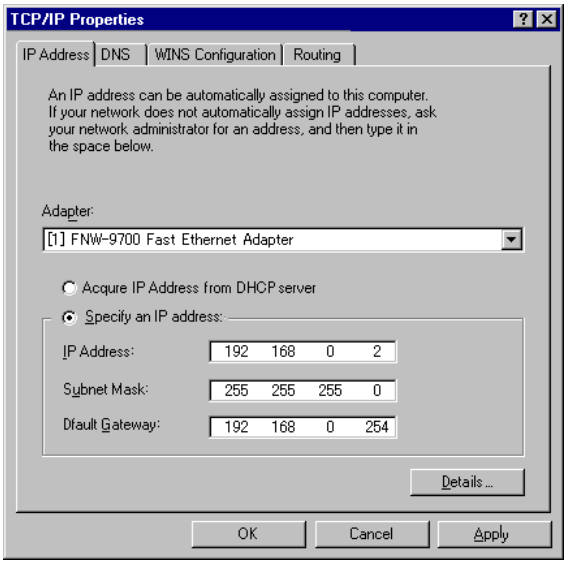
Name of parameter to change	Before/after changes	Parameter value
NEITP	Before	192.168.0.1
	after	192.168.0.1 (Default value)
NETPORT	Before	10000,10001,10002,10003,10004,10005,10006,10007,10008,10009
	after	10000,10001,10002,10003,10004,10005,10006,10007,10008,10009 (Default value)
MXTTOUT	Before	-1
	after	-1 (Default value)
MXTCOM1*	Before	192.168.0.2
	after	192.168.0.2 (Default value)

* MXTCOM1 is used only when setting the robot language to MOVEMASTER command.

Next, set the personal computer IP address to 192.168.0.2. Set this value on the Network Properties screen.

Windows 95 (lower left screen), Windows NT4.0 (lower right screen)





The personal computer IP address is set with the Windows TCP/IP Property Network setting (property in network computer). Refer to the manuals enclosed with Windows, etc., for details on setting this address.

Refer to the instruction manuals enclosed with the personal computer support software for details on setting and using the personal computer support software.

2.4. Connection confirmation

Before use, confirm the following items again.

Connection confirmation

No.	Confirmation item	Check
1	Is the teaching pendant securely fixed?	
2	Is the Ethernet cable properly connected between the controller and personal computer? (Refer to 2.2 in this manual.)	
3	Is any proper Ethernet cable used? (This cross cable is used to connect the personal computer and controller one-on-one. When using a hub with LAN, use a straight cable.)	
4	Is the parameter of the controller properly set? (Refer to 2.3 in this manual.)	
5	Is the power supply of the controller turned off once after the parameter is set?	

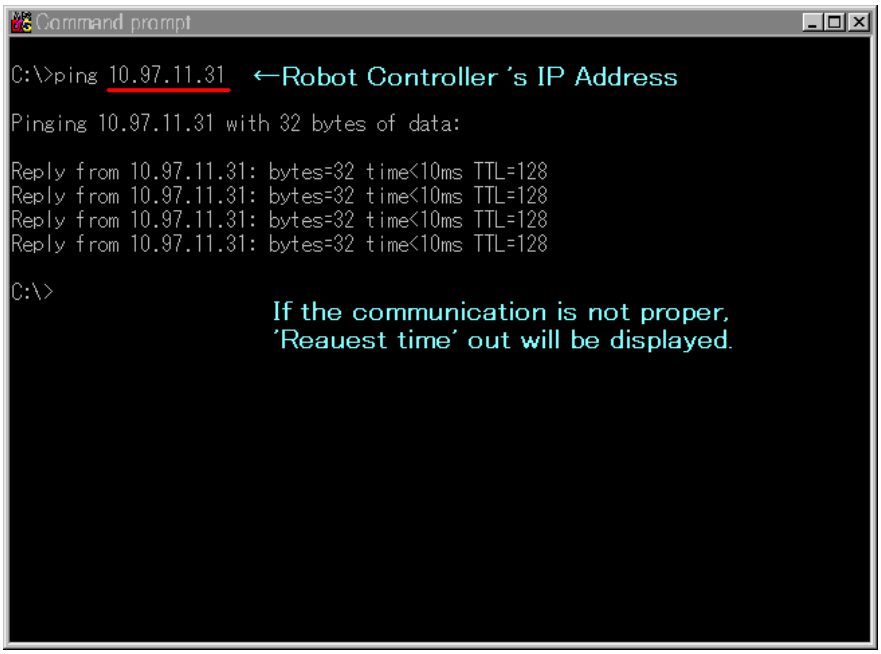
2.4.1. Checking the connection with the Windows ping command

The method for checking the connection with the Windows ping command is shown below.

Start up the " MS-DOS Prompt " from the Windows " Start " - " Programs " menu, and designate the robot controller IP address as shown below.

If the communication is normal, " Reply from ... " will appear as shown below.

If the communication is abnormal, " Request time out " will appear.

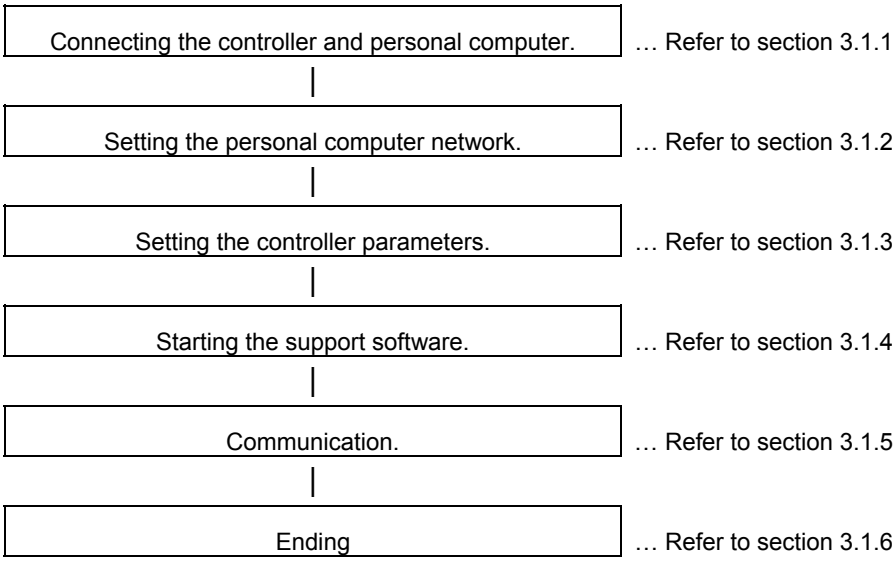


3. Operation

This chapter explains the methods for using the three Ethernet option functions with a system in which the controller and network personal computer are connected with a one-on-one cross cable.

3.1. Controller communication function

The operations for communicating with the personal computer support software are explained in this section.



3.1.1. Connecting the controller and personal computer

Connect the controller and personal computer with a 10 BaseT cross cable.

Refer to the connection described in section "2.2 Ethernet cable".

3.1.2. Setting the personal computer network

Refer to section "2.3.3 Example of setting the parameters 1 (for using the support software)" and set the network.

3.1.3. Setting the controller parameters

Turn ON the robot controller power, and set the parameters as shown below.

If the default settings are to be used, the parameters do not need to be changed.

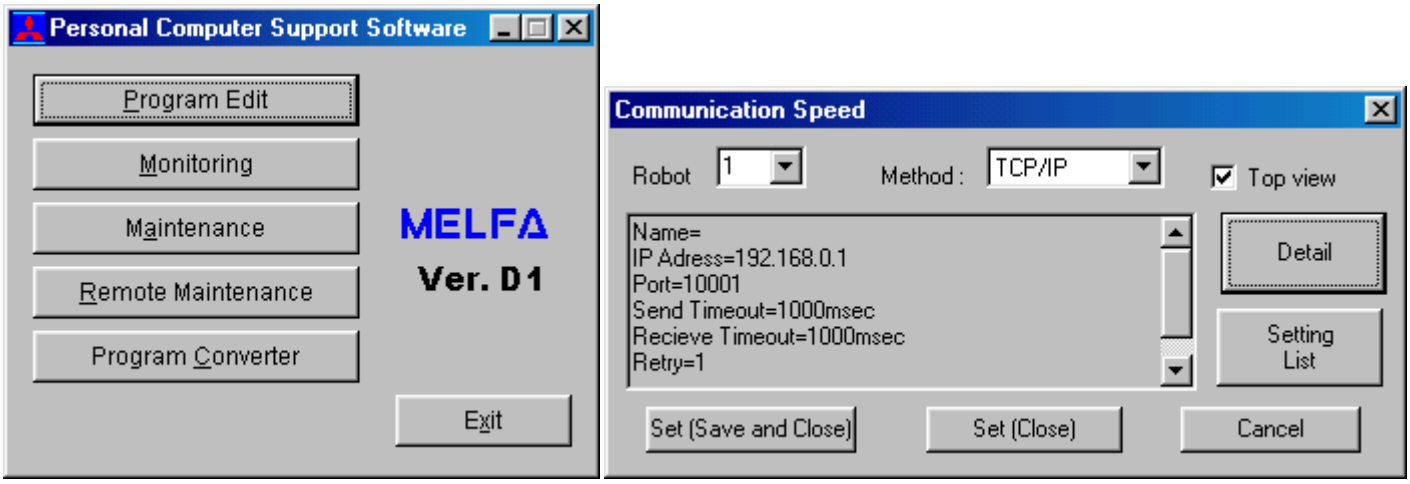
Name of parameter to change	Before/after changes	Parameter value
NETIP	Before	192.168.0.1
	After	192.168.0.1 (Default value)
NETPORT	Before	10000,10001,10002,10003,10004,10005,10006,10007,10008,10009
	After	10000,10001,10002,10003,10004,10005,10006,10007,10008,10009 (Default value)

After setting the parameters, turn the robot controller power OFF and ON.

Refer to the instruction manual enclosed with the robot controller for details on setting the parameters.

3.1.4. Setting the personal computer support software communication

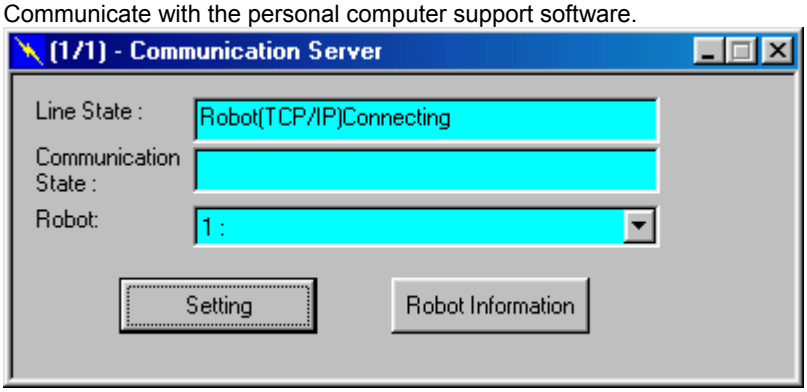
Start the personal computer support software and make the communication settings. Set the communication method to TCP/IP, and the IP Address to 192.168.0.1.



Refer to the instruction manual enclosed with the personal computer support software for details on setting the personal computer support software.

3.1.5. Communication

Communicate with the personal computer support software.



Communication can be carried out with the Ethernet interface TCP/IP in the same manner as RS-232-C communication.

Refer to the instruction manual enclosed with the personal computer support software for details on using the personal computer support software.

If communication is not possible, refer to section "2.4 Checking the connection" and check the state.

⚠CAUTION

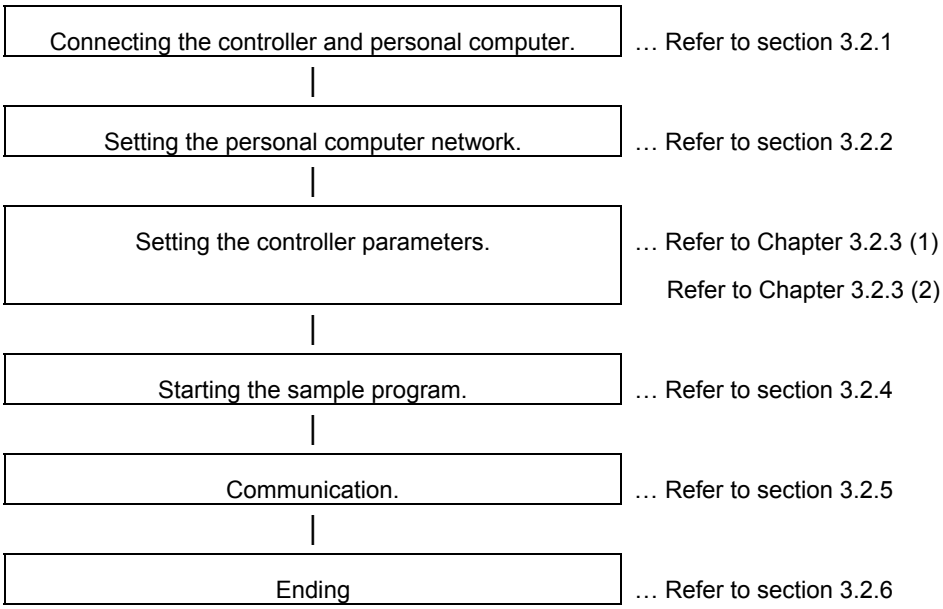
When the robot controller power is turned OFF and ON, the connection will be disconnected and communication will be disabled.
In this case, end the application software on the personal computer once, and then restart.

3.2. Data link function

This section explains the operations for starting the sample program given in "5.2.1 Sample program for data link function" and communicating with a system in which the controller and network personal computer are connected with a one-on-one cross cable.

The controller can be specified as the client from the software version H7 edition of the controller. (Following table)

Software version	H6 or earlier	H7 or later
TCP/IP connection configuration	Controller = Server fixation. Personal computer = Client fixation.	Controller = Server Personal computer = Client
		Controller = Client Personal computer = Server



3.2.1. Connect the controller and personal computer.

Connect the controller and personal computer with a 10 BaseT cross cable.

Refer to the connection described in section "2.2 Ethernet cable".

3.2.2. Setting the personal computer network.

Set one of the following clauses as reference corresponding to the customer's system configuration. (The controller is the server or the client)

- 2.3.4 Example of setting of parameter 2-1 (When the data link function is used: When the controller is the server.)
- 2.3.5E Example of setting of parameter 2-1 (When the data link function is used: When the controller is the client.)

3.2.3. Setting the controller parameters.

The contents of the setting of parameter differ, when the robot controller is specified as server and client of TCP/IP connection.

Turn ON the robot controller power, and set the parameters as shown below.

The NETIP/NETPORT parameters do not need to be changed when using the default values.

After setting the parameters, turn the robot controller power OFF and ON.

Refer to the instruction manual enclosed with the robot controller for details on setting the parameters.

(1) When the controller is specified as the server

Parameter name to be changed	Before/after change	Parameter value
NETIP	Before	192.168.0.1
	After	192.168.0.1 (Default value)
NETPORT	Before	10000,10001,10002,10003,10004,10005,10006,10007,10008,10009
	After	10000,10001,10002,10003,10004,10005,10006,10007,10008,10009 (Default value)
CPRCE13	Before	0
	After	2
COMDEV	Before	RS232, , , , , ,
	After	RS232, , OPT13, , , , ,

(2) When the controller is specified as the client

Parameter name to be changed	Before/after change	Parameter value
NETIP	Before	192.168.0.1
	After	192.168.0.1 (Default value)
NETPORT	Before	10000,10001,10002,10003,10004,10005,10006,10007,10008,10009
	After	10000,10001,10002,10003,10004,10005,10006,10007,10008,10009 (Default value)
CPRCE13	Before	0
	After	2
COMDEV	Before	RS232, , , , , , ,
	After	RS232, , OPT13, , , , ,
NETMODE	Before	1,1,1,1,1,1,1,1,1,1
	After	1,1,0,1,1,1,1,1,1,1
NETHSTIP	Before	192.168.0.2, 192.168.0.3, 192.168.0.4, 192.168.0.5, 192.168.0.6, 192.168.0.7, 192.168.0.8, 192.168.0.9, 192.168.0.10
	After	192.168.0.2, 192.168.0.3, 192.168.0.2, 192.168.0.5, 192.168.0.6, 192.168.0.7, 192.168.0.8, 192.168.0.9, 192.168.0.10

3.2.4. Starting the sample program

The test program is an example for establishing a data link between the robot and personal computer. COM3 is used.

(1) Using the teaching pendant or personal computer support software, register the following robot program with an appropriate program name. Either the MELFA-BASICIV or MOVEMASTER command can be used as the robot language. MELFA-BASICIV is set as the default. The parameter RLNG must be changed to change the robot language. Refer to the instruction manual enclosed with the robot controller for details. The MOVEMASTER commands can be used only with some robot models (RV-1A/RV-2AJ, etc.). Thus, only MELFA-BASICIV may be provided depending on the model being used.

<Robot program>

1) Example for MELFA-BASICIV

```
10 OPEN "COM3:" AS #1          ' Open as communication line COM3
20 PRINT #1,"START"            ' Send START character string
30 INPUT #1,DTATA              ' Wait for reception of value in DATA variable
40 IF DATA<0 THEN GOTO 70      ' If DATA is negative, jump to line 70 and end
50 PRINT #1,"DATA=";DATA       ' Reply DATA = value
60 GOTO 30                     ' Jump to line 30 and repeat
70 PRINT #1,”END"              '. Send END character string
80 END                         ' End
```

2) Example for MOVEMASTER command

```
10 OPN 1,3                     ' Open as communication line No. 3
20 SC $1,”START”                ‘ Set START characters in character string $1
25 CR $1,1                     ' Send START character string
30 INP 1,1,0                   ' Wait for reception of value in counter 1
40 CP 1                        ' Set counter 1 value in internal register
45 SM 0,70                     ‘ If value is negative, jump to line 70 and end
50 CR 1,1                      ' Reply counter 1 value
60 GT 30                       ' Jump to line 30 and repeat
70 SC $1,”END”                  ' Set END characters in character string $1
75 CR $1,1                     ‘ Send END character string
80 ED                            ' End
```

(2) Start the personal computer data link program

Refer to section "5.2.1 Sample program for data link function" and create the execution file. (The created execution file will be sample.exe.)

Start Windows Explorer, and double-click on sample.exe.

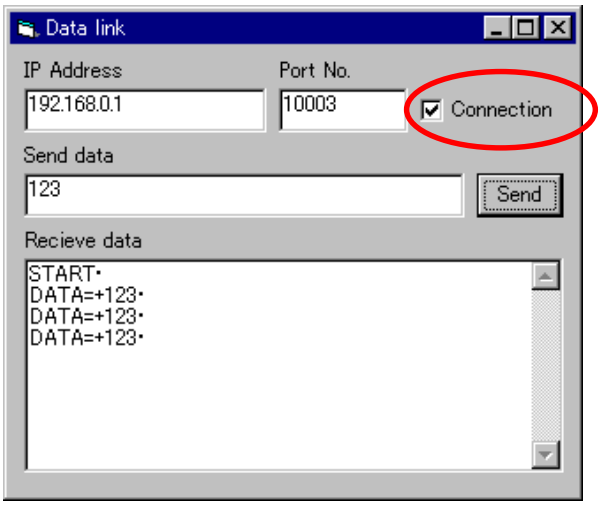
Set the IP address and port No., click on the connection check box, and open the communication line with the controller.

If the Send button is not validated, check that the IP address matches NETIP set with the controller.

If the button is still not validated, refer to section "2.4 Checking the connection", and check the connection cable or restart the controller and sample.exe.

(3) Start the robot program.

Press the START button on the robot controller's operating panel, and start the robot program.



3.2.5. Communication

(1) When the robot controller program is started, first the following data will be sent to the personal computer.

"START"(CR) (CR) indicates the CR code.

(2) When the personal computer receives the data, the characters will appear in the received data area.

(3) Send value data from the personal computer.

For example, input the value data 123 in the transmission data area, and click on the Send button with the mouse.

(4) When the robot controller receives the value data in the DATA variable, it will reply data to the personal computer.

DATA=123 will appear in the personal computer's received data area.

If communication cannot be carried out correctly, refer to section "2.4 Checking the connection" and check.

⚠CAUTION

When the robot controller power is turned OFF and ON, the connection will be disconnected and communication will be disabled.
In this case, end the application software on the personal computer once, and then restart.

3.2.6. Ending

(1) Press the END button on the robot controller operating panel, and enter cycle operation.

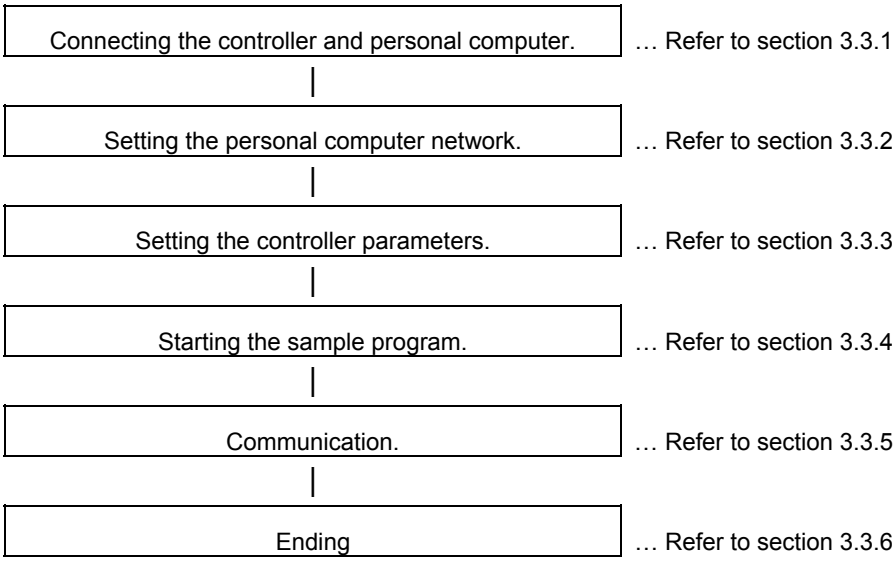
(2) Input the value -1 from the personal computer, and end the program.

(3) End the personal computer's sample program.

(4) Turn OFF the robot controller's power.

3.3. Real-time external control function

This section explains the operations for starting the sample program given in "5.2.2 Sample program for real-time external control function" and communicating with a system in which the controller and network personal computer are connected with a one-on-one cross cable.



3.3.1. Connecting the controller and personal computer

Connect the controller and personal computer with a 10 BaseT cross cable.
Refer to the connection described in section "2.2 Ethernet cable".

3.3.2. Setting the personal computer network

Refer to section "2.3.5 Example of setting the parameters 3 (for using the real-time external control function)" and set the network.

3.3.3. Setting the controller parameters

Turn ON the robot controller power, and set the parameters as shown below.
If the default settings are to be used, the parameters do not need to be changed.

After setting the parameters, turn the robot controller power OFF and ON.
Refer to the instruction manual enclosed with the robot controller for details on setting the parameters.

Name of parameter to change	Before/after changes	Parameter value
NETIP	Before	192.168.0.1
	After	192.168.0.1 (Default value)
NETPORT	Before	10000,10001,10002,10003,10004,10005,10006,10007,10008,10009
	After	10000,10001,10002,10003,10004,10005,10006,10007,10008,10009 (Default value)
MATTOUT	Before	-1
	After	-1 (Default value)
MXTCOM1*	Before	192.168.0.2
	After	192.168.0.2 (Default value)

*MXTCOM1 is used only when the robot language is set to MOVEMASTER command. It is not used with MELFA-BASICIV.

3.3.4. Starting the sample program

The test program is an example of communicating in real-time between the robot and personal computer. The XYZ position data X axis or joint position data J1 axis is commanded from the personal computer to the robot and controlled.

(1) Using the teaching pendant or personal computer support software, register the following robot program with an appropriate program name. Ether the MELFA-BASICIV or MOVEMASTER command can be used as the robot language. MELFA-BASICIV is set as the default. The parameter RLNG must be changed to change the robot language. Refer to the instruction manual enclosed with the robot controller for details. The MOVEMASTER commands can be used only with some robot models (RV-1A/RV-2AJ, etc.). Thus, only MELFA-BASICIV may be provided depending on the model being used.

<Robot program>

1) Example for MELFA-BASICIV

```
10 OPEN "ENET; 192.168.0.2" AS #1    ' Designate personal computer side IP address as Ethernet in file No. 1
20 MOV P1                            ' Move to default position P1 (teach random position as P1)
30 MXT1,0                            ' Move according to command value issued from file No. 1
                                       Current XYZ position is replied from controller to personal computer
40 MOV P1                            ' After external control mode ends, move to default position P1 with joint
                                       interpolation
50 HLT                               ' Halt
60 END                               ' End
```

2) Example for MOVEMASTER command

```
10 MO 1                              ' Move to default position 1 (teach random position as 1)
20 MXT 1,0                           ' Move according to command value issued from communication
                                       destination No. 1
                                     ' Receive XYZ data from the personal computer
30 MO 1                              ' After external control mode ends, move to default position 1 with joint
                                       interpolation
40 HLT                               ' Halt
50 ED                                ' End
```

(2) Start the robot program.

Press the START button on the robot controller's operating panel, and start the robot program.

The robot will move to the default position P1, and real-time external control will be executed with the MXT command.

(3) Start the personal computer's real-time external control sample program.

Refer to section "5.2.2 Sample program for real-time external control function" and create the execution file. (The created execution file will be sample.exe.)

Start Windows Explorer, and double-click on sample.exe.

3.3.5. Moving the robot

Specify and input the following values for the numerical value displayed on the screen according to the message of the sample program.

*The IP address (192.168.0.1) of the robot controller of the connection point

*The port number (10001)

*The data type of command

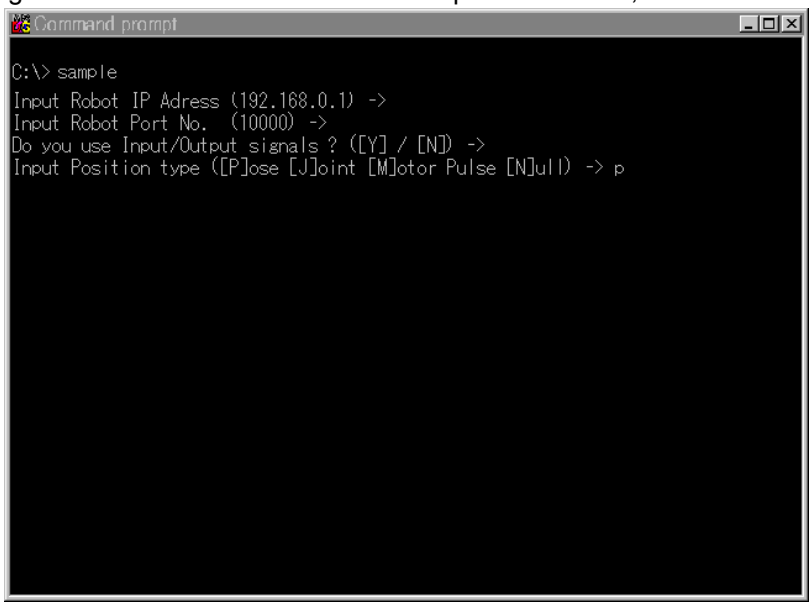
*The data type of monitoring (The version is H7 or later), etc

Fit the data type of command to the argument of the MXT command of the robot program

Key operation is as follows. For details, refer to the sample program.

Key	Contents
Z or X .	The robot moves.
C	The instruction value is set to 0 and the robot stops.
D	Each time the MOVE key is pressed, change the display / un-displaying of the monitor data
ENTER	End the MXT command.

If the amount of instructions becomes too large or the movement range of the robot is exceeded, an error is generated and the robot controller stops. In this case, reset the robot controller.



If communication cannot be carried out correctly, refer to section "2.4 Checking the connection", and check the connection cable or restart the controller and sample.exe.

⚠CAUTION

When the robot controller power is turned OFF and ON, the connection will be disconnected and communication will be disabled.
In this case, end the application software on the personal computer once, and then restart.

3.3.6. Ending

(1) Press the END button on the robot controller operating panel, and enter cycle operation.

(2) End the personal computer's sample program.

When the [ENTER] key is pressed, the MXT command will end, the robot will return to the default position, and the robot program will stop.

The sample program will also end.

(3) Turn OFF the robot controller's power.

4. Explanation of functions

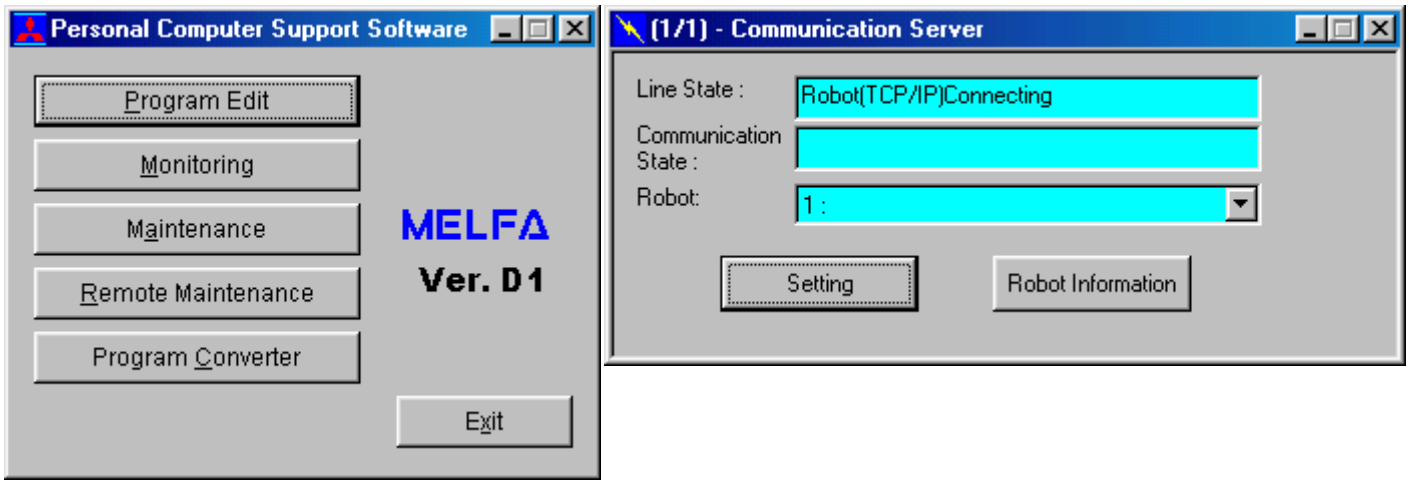
This chapter describes the detailed functions of the Ethernet interface.

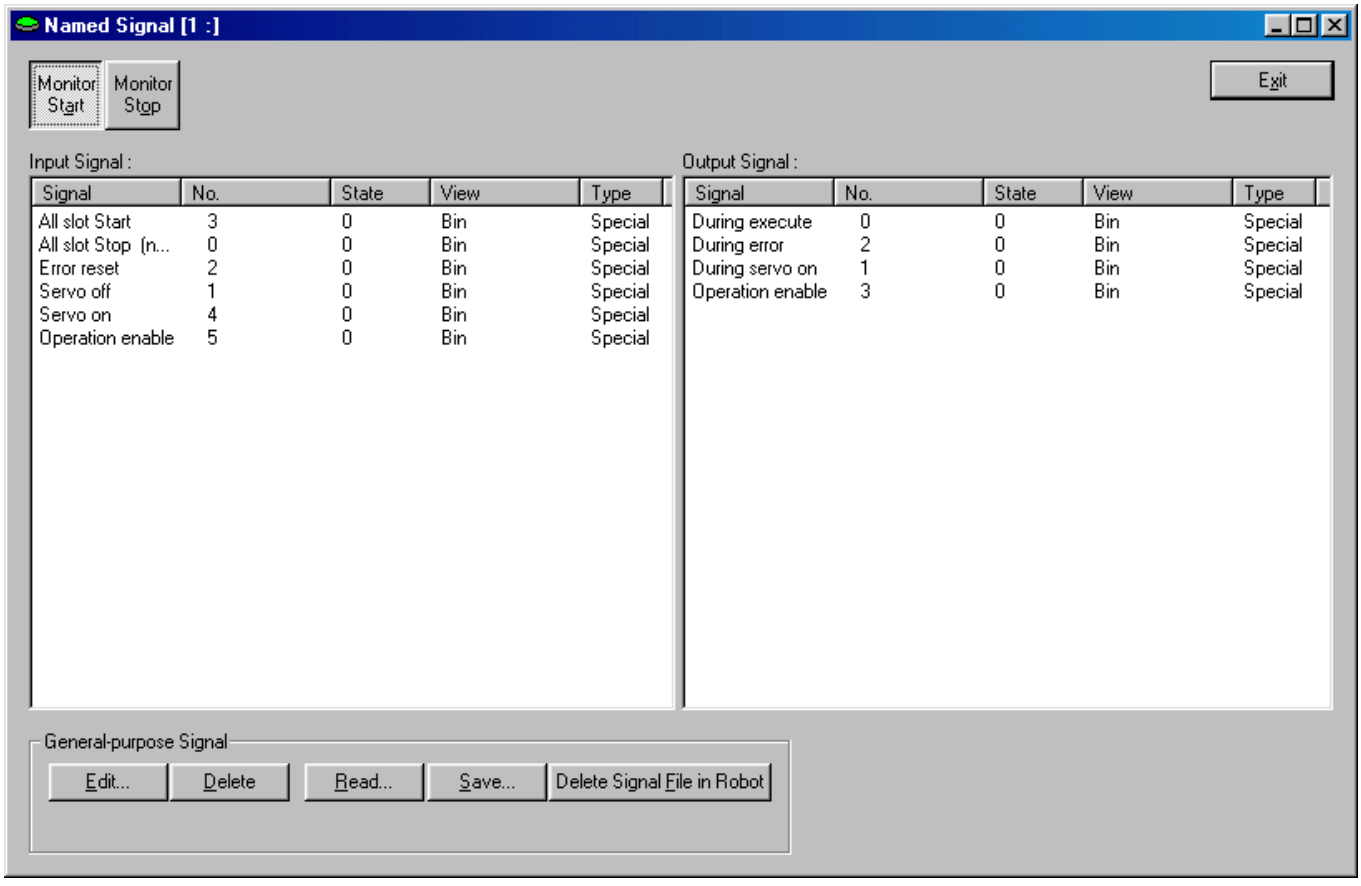
4.1. Controller communication function

・Communication via the network of the personal computer is used like the Support software which corresponds to the existing RS-232C.

・The Support Software enables all functions such as the up down load and status monitor, etc. of the program of the robot.

* It can be used with high speed and away in comparison with the RS232C.



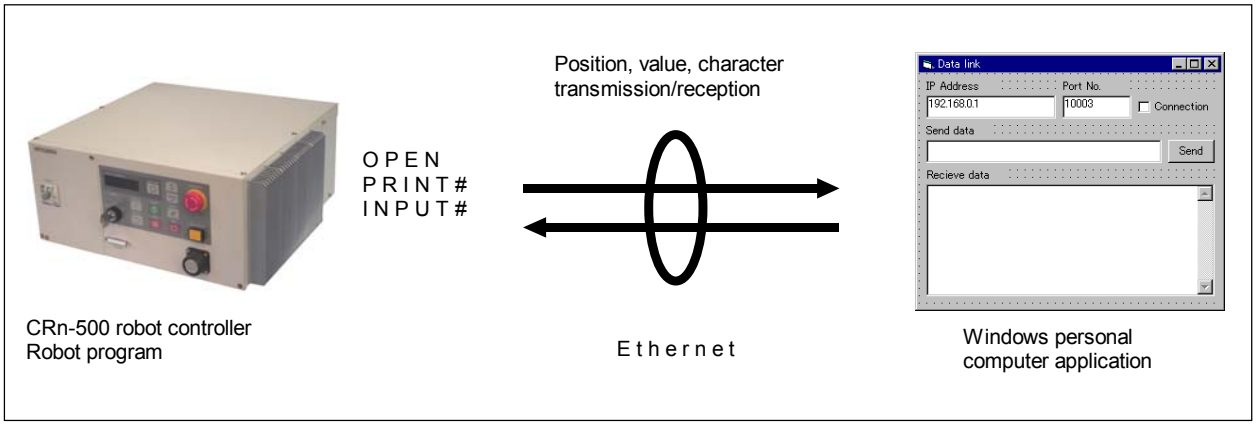


4.2. Data link function

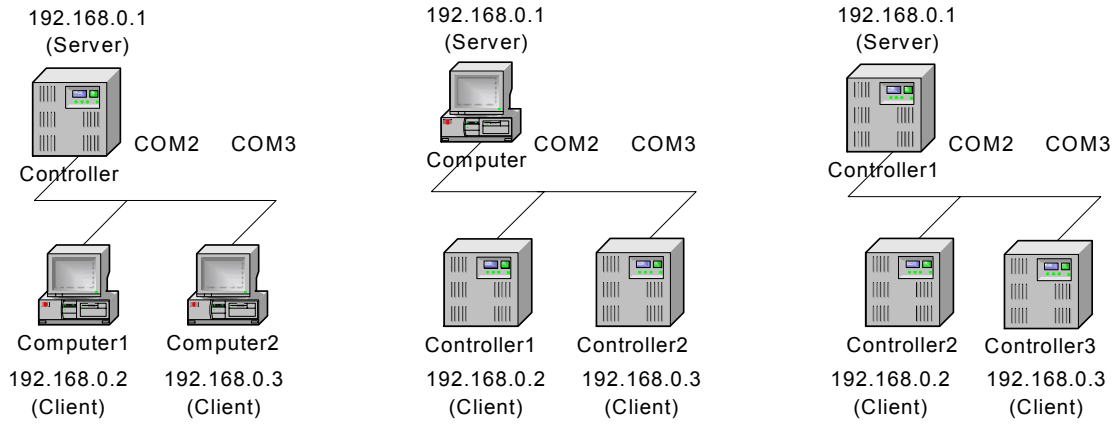
Like the data link communication with RS-232C, OPEN/PRINT/INPUT of the robot language can be also used in the Ethernet.
For each robot language, refer to the instruction manual appended to the robot controller.

[Statement example] To set port No. 10003 as communication destination and open as #1
Set parameter COMDEV (element No. 3) to OPT13, NETPORT to 10003.

```
100 OPEN “COM3:” AS #1        ’Set port No.
110 INPUT #1, C1$             'Read
120 PRINT #1, ”Reply”, C1$    ‘Writing
130 CLOSE #1                  ‘Line closing
140 HLT                       ‘Stop
```



The data link function of the Ethernet interface has the two kinds shown below.
*Uses the robot controller as the server.
*Uses the robot controller as the client.
In addition, to set it as the client, it is necessary for the software version of the robot controller to be H7 or later.
Choose corresponding to the customer's system such as the example of the following figure.



Two or more clients are not connectable with the one line number COMn.
Change the line number, when using the robot controller as the server and connecting two or more clients.

4.2.1. MELFA-BASICIV Commands

This section describes the robot language (MELFA-BASICIV).
The commands described in this section have been added in software version H7 or later. These commands cannot be used in software version H6 or earlier. For more information about OPEN, CLOSE, INPUT# and PRINT# used for data linking, refer to the CR1/CR2/CR4/CR7/CR8 Controller INSTRUCTION MANUAL Detailed explanations of functions and operations.

⚠CAUTION An error occurs if a syntax check of this robot language is performed in Mitsubishi personal computer support software (A*, B*, C* and D1 editions) available from August, 2002. Therefore, do not perform any syntax check in the personal computer support software.

M_OPEN ** Software version H7 or later*

[Function]
Indicates whether or not the file has been opened.

[Fomat]

<Numeric variable> = M_OPEN [(<file number>)]

[Terminology]

<Numeric variable>	Specify a numeric variable to be assigned.
<File number>	Specify a file number constant between 1 and 8 for the communication line that was opened by the OPEN instruction. If omitted, 1 is set. If 9 or higher is specified, an error occurs when executed.

[Reference Program]

```
10 ' Client Program ----------
100 M1=0
110 M_TIMER(1)=0                                  'Resets the timer to 0.
120 OPEN "COM2:" AS #1                            'Opens the line.
130 IF M_TIMER(1)>10000.0 THEN 240                'Jumps when 10 seconds elapses.
140 IF M_OPEN(1)<>1 THEN GOTO 120                 'Loops if no connection is made.
145 DEF ACT 1,M_OPEN(1)=0 GOSUB 300               'Monitors the down state of the server using an interrupt.
146 ACT 1=1                                       'Starts monitoring.
150 M1=M1+1
160 IF M1<10 THEN C1$="MELFA" ELSE C1$="END"      'Sends END after sending the "MELFA" string nine times.
170 PRINT #1,C1$                                  'Sends a character string.
180 INPUT #1,C2$                                  'Receives a character string.
190 IF C1$="END" THEN 210                         'Jumps to CLOSE after sending "END."
200 GOTO 150                                      'Loops.
210 CLOSE #1                                      'Closes the line.
220 HLT                                           'Halts the program.
230 END                                           'Ends.
240 ERROR 9100                                    'Generates error 9100 if no connection can be made to the
                                                   server.
250 CLOSE #1
260 HLT
270 END
280 ERROR 9101                                    'Generates error 9101 if the server is down during
processing.
290 CLOSE #1
300 HLT
310 END
```

[Explanation]

(1) This command is used in a combination with the OPEN instruction. The following lists the meanings and values for the types of the files specified by the OPEN instruction.

Type of file to be opened	Meaning		Value
File	Indicates whether or not the file has been opened. 1 is always returned after executing the OPEN instruction.		1: Already opened. -1: The file number is undefined (not opened).
Communication line RS232C	Indicates the status of the counterpart of the RS232C cable communication. The CTS signal input status is returned as is. The power off status and cable disconnection status of the counterpart can be determined. (Mitsubishi genuine cable specification: Can be used only when the RTS signal of the counterpart is enabled using model name RS-MAXY-CBL/RS-AT-RCBL .)		1: Already connected (CTS signal is ON). 0: Not connected (CTS signal is OFF). -1: The file number is undefined (not opened).
Communication line Ethernet	Indicates whether or not connection is made with the counterpart.	For server setting	1: Client is already connected. 0: Client is not connected. -1: The file number is undefined (not opened).
		For client setting	1: Already connected to the server. (Connection has been made.) 0: Not connected to the server. (Connection has not been made. Equivalent to when the server is down after being opened.) -1: The file number is undefined. (When the file has not been opened, or has been opened while the server is down.)

[Related Instruction]

OPEN

[Related Parameters]

COMDEV, CPRE**, NETMODE

C_COM

** Software version H7 or later*

[Function]

Sets the parameters for the line to be opened by the OPEN instruction. This is used when the communication destination is changed frequently.

* Character string type

* Only for a client with the Ethernet option.

[Fomat]

C_COM (<communication line number>) = “ETH: <server side IP address> [, <port number>]”

[Terminology]

ETH:	An identifier to indicate that the target is an Ethernet
<Communication line number>	The number of the COM to be specified by the OPEN instruction (The line type is assigned by the COMDEV parameter.) Specify 1 through 8.
<Server side IP address>	Server side IP address (May be omitted.)
<Port number>	Port number on the server side (If omitted, the set value of the NETPORT parameter is used.)

[Reference Program]

Example when the Ethernet option is installed in an option slot and OPT12 is set in the second element of the COMDEV parameter

```
100 C_COM(2)="ETH:192.168.0.10,10010"  ' Set the IP address of the communication destination server
                                         corresponding to communication line COM2
110 OPEN "COM2:" AS #1                 ' As 192.168.0.10 and the port number as 10010, and then open the line.
120 IF M_OPEN(1)<>1 THEN 110           ‘ Loops if unable to connect to the server.
130 PRINT #1, "HELLO"                  ‘ Sends a character string.
140 INPUT #1, C1$                      ‘ Receives a character string.
150 CLOSE #1                           ‘ Closes the line.
160 C_COM(2)="ETH:192.168.0.11,10011"  ‘ Set the IP address of the communication destination server
                                         corresponding to communication line COM2
170 OPEN "COM2:" AS #1                 ‘ As 192.168.0.11 and the port number as 10011, and then open the line.
180 IF M_OPEN(1)<>1 THEN 170           ‘ Loops if unable to connect to the server.
190 PRINT #1, C1$                      ‘ Sends a character string.
200 INPUT #1, C2$                      ‘ Receives a character string.
210 CLOSE #1                           ‘ Closes the line.
220 HLT                                ‘ Halts the program.
230 END                                ‘ Ends.
```

[Description]

(1) It is not necessary to use this command when the communication counterpart of the robot controller is specified with the NETHSTIP and NETPORT parameters and the specified communication counterpart will not be changed at all.

(2) Currently, this function is valid only for a client of a data link with the Ethernet option.

(3) Because the communication parameters of the OPEN instruction are set, it is necessary to execute this command before the OPEN instruction.

(4) When the power is turned on, the set values specified by the NETHSTIP and NETPORT parameters are used. When this command is executed, the values specified by the parameters of this command are changed temporarily. They are valid until the power is turned off. When the power is turned on again, the values revert to the original values set by the parameters.

(5) If this command is executed after the OPEN instruction, the current open status will not change. In such a case, it is necessary to close the line with the CLOSE instruction once, and then execute the OPEN instruction again.

(6) If an incorrect syntax is used, an error occurs when the program is executed, not when the program is edited.

[Related Parameters]

NETHSTIP, NETPORT

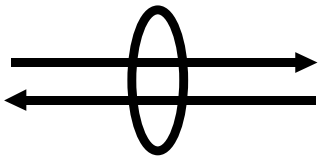
4.3. Real-time external control function

The robot motion movement control can retrieve the position command at real-time in cycle units, and move to the commanded position. It is also possible to monitor the input/output signals or output the signals simultaneously.
Using the robot language MXT command, real-time communication (command/monitor) is carried out with communication.

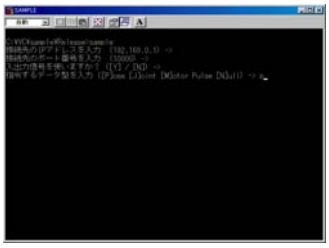
CRn-500 robot controller
Robot program

OPEN
PRINT#
INPUT#

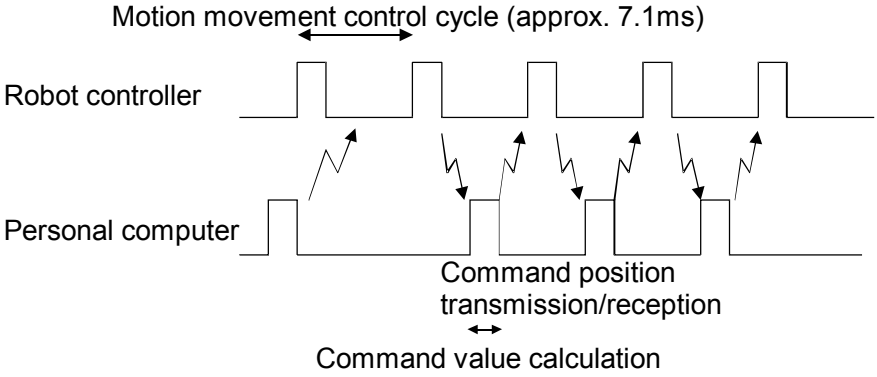
Real-time external control packet
data * transmission/reception



Ethernet



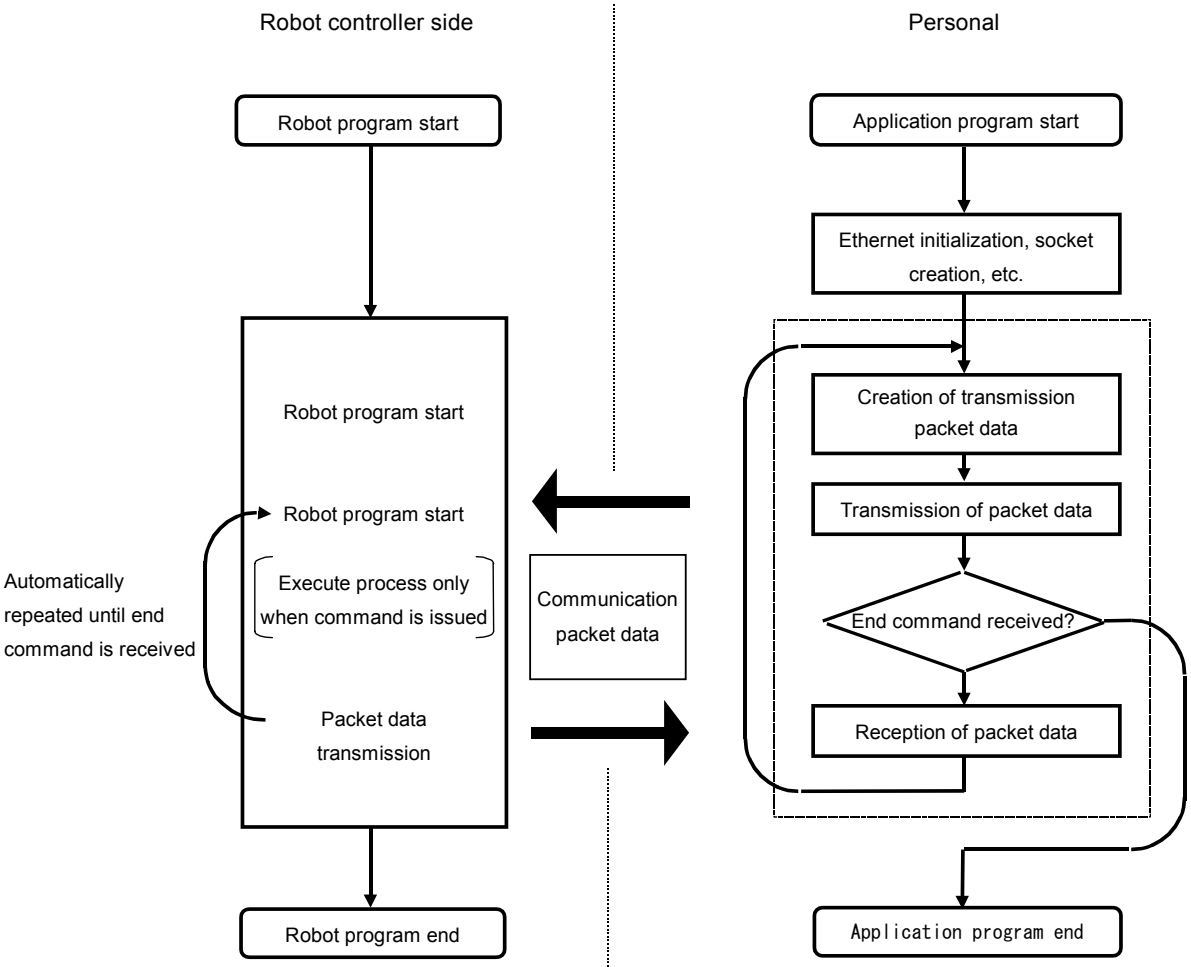
Windows personal
computer application



The following table lists the position command data for giving the target move position from the personal computer to the robot for each hour of the motion operation control cycle, and the monitor data types from the robot.
For more information about communication data, see Section 4.3.1, “Command Explanation” and Section 4.3.2, “Communication Data Packet Explanation” in this document.

Position command data type	Monitor data type
[1] Rectangular coordinate data [2] Joint coordinate data [3] Motor pulse coordinate data	[1] Rectangular coordinate data [2] Joint coordinate data [3] Motor pulse coordinate data [4] Rectangular coordinate data (command value after filter processing) [5] Joint coordinate data (command value after filter processing) [6] Motor pulse coordinate data (after filter processing) [7] Rectangular coordinate data (encoder feedback value) [8] Joint coordinate data (encoder feedback value) [9] Motor pulse coordinate data (encoder feedback value) [10] Current command (%) [11] Current feedback (%) Note: Items [7] through [11] are supported in software version H7 or later.

* Flow of real-time external control



4.3.1. Explanation of command

Either the MELFA-BASICIV or MOVEMASTER command languages can be used with the real-time external control function.

Note that the meanings of the arguments differ for the MELFA-BASICIV and MOVEMASTER commands. (Refer to following format and terminology.)

Refer to section "4.3.2 Explanation of communication data packet" for details on the structure of the communication data packet used with this function.

MXT (Move External)

[Function]

The absolute position data is retrieved from an external source at each controller control time (currently approx. 7.1msec), and the robot is directly moved.

[Format]

(1) For MELFA-BASICIV

MXT <File No.>, <Reply position data type> [, <Filter time constant>]

(2) For MOVEMASTER command

MXT <Communication destination No.>, <Reply position data type> [, <Filter time constant>]

[Terminology]

(1) For MELFA-BASICIV

<File No.> Describe a number between 1 and 8 assigned with the OPEN command.
If the communication destination is not designated with the OPEN command, an error will occur, and communication will not be possible.
In addition, data received from a source other than the communication destination will be ignored.

(2) For MOVEMASTER command

<Communication destination No.> Describe the communication destination as a number between 1 and 3 assigned with the parameters MXTCOM1 to MXTCOM3.
Designate the communication destination as an IP address in parameters MXTCOM1 to MXTCOM3. Communication will not be possible if not designated.
In addition, data received from a source other than the communication destination will be ignored.
For example, to assign the personal computer IP address 192.168.0.2 for the communication destination No. 1, set 192.168.0.2 in parameter MXTCOM1.

Settings common for MELFA-BASICIV and MOVEMASTER command

<Replay position data type> Designate the type of the position data to be received from the personal computer.
A XYZ/joint/motor pulse can be designated.
0: XYZ coordinate data
1: Joint coordinate data
2: Motor pulse coordinate data

<Filter time constant> Designate the filter time constant (msec). If 0 is designated, the filter will not be applied. (0 will be set when omitted.) A filter is applied on the reception position data, an obtuse command value is created and output to the servo.

[Reference Program]

(1) For MELFA-BASICIV

```
10 OPEN "ENET; 192.168.0.2" AS #1Set     'Ethernet communication destination IP address
20 MOV P1                                'Move to P1
30 MXT1,1,50                             'Move with real-time external control with filter time constant set to
                                          50msec
40 MOV P1                                'Move to P1
50 HLT                                   'Halt program
```

(2) For MOVEMASTER command

Set the Ethernet communication destination personal computer IP address as 192.168.0.2 in parameter MXTCOM1.

```
10 MO 1                                  'Move to position 1
20 MXT 1,1,50                            'Move with real-time external control with filter time constant set to
                                          50msec
30 MO 1                                  'Move to position 1
40 HLT                                   'Halt program
```

[Explanation]

* When the MXT command is executed, the position command for movement control can be retrieved from the personal computer connected on the network. (One-on-one communication)
* One position command can be retrieved and operated at the operation control time (currently 7.1msec).
* Operation of MXT command
  1) When this command is executed with the controller, the controller enters the command value reception enabled state.
  2) When the controller receives the command value from the personal computer, it will output the received command value to the servo within the next control process cycle.
  3) After the command value is sent to the servo, the controller information, such as the current position is sent from the controller to the personal computer.
  4) A reply is made from the controller to the personal computer only when the command value from the personal computer is sent to the controller.
  5) If the data is not received, the current position is maintained.
  6) When the real-time external command end command is received from the personal computer, the MXT command is ended.
  7) When the operation is stopped from the operating panel or external input, the MXT command will be halted, and the transmission/reception will also be halted until restart.
* The timeout is designated with the parameter MXTTOUT.
* One randomly designated (head bit, bit width) input/output signal can be transmitted and received simultaneously with the position data.
* A personal computer with sufficient processing speed must be used to command movement in the movement control time. A Windows NT or 2000/Pentium II 450MHz or higher console application is recommended.

4.3.2. Explanation of communication data packet

The structure of the communication data packet used with the real-time external control function is explained in this section. The same communication data packet for real-time external control is used for commanding the position and for monitoring. The contents differ when transmitting (commanding) from the personal computer to the controller and when receiving (monitoring) from the controller to the personal computer.

Refer to the following communication data packet structure and section "5.2.2 Sample program for real-time external control function", and create the application. The C language data type is used in the following table. In addition, there are the communication data packet 1 and the communication data packet 2 by the software version of the controller. Choose according to the software version of the controller of use. Refer to "1.5 Checking the robot controller software version" for check method of the version.

(1) Communication data packet 1. When the software version is H6 or earlier.

Name	Data type	Explanation
Command	unsigned short (2-byte)	Designate the validity of the real-time external command, and the end. 0 // Real-time external command invalid 1 // Real-time external command valid 255 // Real-time external command end
Transmission data type designation SendType	unsigned short (2-byte)	1) When transmitting (commanding) from the personal computer to the controller, designate the type of position data transmitted from the personal computer. There is no data at the first transmission. 0 // No data 1 // XYZ data 2 // Joint data 3 // Motor pulse data 2) When receiving (monitoring) from the controller to the personal computer, indicate the type of position data replied from the controller. 0 // No data 1 // XYZ data 2 // Joint data 3 // Motor pulse data 4 // XYZ data (Position after filter process) 5 // Joint data (Position after filter process) 6 // Motor pulse data (Position after filter process)
Reply data type designation RecvType	unsigned short (2-byte)	1) When transmitting (commanding) from the personal computer to the controller, designate the type of data replied from the controller. 0 // No data 1 // XYZ data 2 // Joint data 3 // pulse data 4 // XYZ data (Position after filter process) 5 // Joint data (Position after filter process) 6 // Motor pulse data (Position after filter process) 2) When receiving (monitoring) from the controller to the personal computer, this has no significant meaning.
Reservation reserve	unsigned short (2byte)	Not used.

Name	Data type	Explanation
Position data Pos / jnt / pls	POSE, JOINT or PULSE (40-byte) * Refer to strdef.h in the sample program for details on each data structure.	1) When transmitting (commanding) from the personal computer to the controller, designate the command position data transmitted from the personal computer. Set this to the same data type as that designated for the transmission data type designation. 2) When receiving (monitoring) from the controller to the personal computer, this indicates the position data replied from the controller. The contents of the data are common. POSE // XYZ type [mm/rad] JOINT // Joint type [rad] PULSE // Motor pulse type [pulse]
Transmission input/output signal data designation SendIOType	unsigned short (2-byte)	1) When transmitting (commanding) from the personal computer to the controller, designate the data type of the input/output signal transmitted from the personal computer. Designate "No data" when not using this function. 2) When receiving (monitoring) from the controller to the personal computer, this indicates the data type of the input/output signal replied from the controller. The contents of the data are common. 0 // No data 1 // Output signal 2 // Input signal
Reply input/output signal data designation RecvIOType	unsigned short (2-byte)	1) When transmitting (commanding) from the personal computer to the controller, designate the data type of the input/output signal replied from the controller. Designate "No data" when not using this function. 0 // No data 1 // Output signal 2 // Input signal 2) When receiving (monitoring) from the controller to the personal computer, this has no significant meaning.

Input/output signal data
BitTop
BitMask
IoData
	unsigned short
	unsigned short
	unsigned short
	(2-byte x 3)
		1) When transmitting (commanding) from the personal computer to the controller, designate the output signal data transmitted from the personal computer.
		
		2) When receiving (monitoring) from the controller to the personal computer, this indicates the input/output signal data replied from the controller.
		
		The contents of the data are common.
		BitTop; // Head bit No. of input or output signal
		BitMask; // Bit mask pattern designation (valid only for commanding)
		IoData; // Input or output signal data value (for monitoring)
		Output signal data value (for commanding)
		* Data is 16-bit data

Timeout time counter value		1) When transmitting (commanding) from the personal computer to the controller, this has no significant meaning.

Name	Data type	Explanation
Tcount	unsigned short (2-byte)	2) When receiving (monitoring) from controller to personal computer, if the timeout time parameter MXTTOUT is a value other than -1, this indicates the No. of times communication with the controller was not possible. When the No. of times is counted and reaches the maximum value, the value will return to the minimum value 0, and the count will be repeated. This is set to 0 when the MXT command is started.
Counter value for communication data Ccount	unsigned long (4-byte)	1) When transmitting (commanding) from the personal computer to the controller, this has no significant meaning. 2) When receiving (monitoring) from controller to personal computer, this indicates the No. of communication times. When the No. of times is counted and reaches the maximum value, the value will return to the minimum value 0, and the count will be repeated. This is set to 0 when the MXT command is started.

(2) Communication data packet 2. When the software version is H7 or later.

Command	unsigned short (2-byte)	Designate the validity of the real-time external command, and the end. 0 // Real-time external command invalid 1 // Real-time external command valid 255 // Real-time external command end
Transmission data type designation SendType	unsigned short (2-byte)	1) When transmitting (commanding) from the personal computer to the controller, designate the type of position data transmitted from the personal computer. There is no data at the first transmission. 0 // No data 1 // XYZ data 2 // Joint data 3 // Motor pulse data 2) When receiving (monitoring) from the controller to the personal computer, indicate the type of position data replied from the controller. 0 // No data 1 // XYZ data 2 // Joint data 3 // Motor pulse data 4 // XYZ data (Position after filter process) 5 // Joint data (Position after filter process) 6 // Motor pulse data (Position after filter process) **7 // XYZ data (Encoder feedback value)** **8 // Joint data (Encoder feedback value)** **9 // Motor pulse data (Encoder feedback value)** **10 // Current command [%]** **11 // Current feedback [%]** * It is the same as RecvType. You may use whichever.
Reply data type designation RecvType	unsigned short (2-byte)	□1) When transmitting (commanding) from the personal computer to the controller, designate the type of data replied from the controller. 0 // No data 1 // XYZ data 2 // Joint data 3 // pulse data

		4 // XYZ data (Position after filter process) 5 // Joint data (Position after filter process) 6 // Motor pulse data (Position after filter process) **7 // XYZ data (Encoder feedback value)** **8 // Joint data (Encoder feedback value)** **9 // Motor pulse data (Encoder feedback value)** **10 // Current command [%]** **11 // Current feedback [%]** 2) When receiving (monitoring) from the controller to the personal computer, indicate the type of position data replied from the controller. 0 // No data 1 // XYZ data 2 // Joint data 3 // Motor pulse data 4 // XYZ data (Position after filter process) 5 // Joint data (Position after filter process) 6 // Motor pulse data (Position after filter process) **7 // XYZ data (Encoder feedback value)** **8 // Joint data (Encoder feedback value)** **9 // Motor pulse data (Encoder feedback value)** **10 // Current command [%]** **11 // Current feedback [%]** * It is the same as RecvType. You may use whichever.
Reservation reserve	unsigned short (2byte)	Not used.
Position data Pos / jnt / pls	POSE, JOINT or PULSE (40-byte) * Refer to strdef.h in the sample program for details on each data structure.	1) When transmitting (commanding) from the personal computer to the controller, designate the command position data transmitted from the personal computer. Set this to the same data type as that designated for the transmission data type designation. 2) When receiving (monitoring) from the controller to the personal computer, this indicates the position data replied from the controller. The data type is shown in SendType (= RecvType) . The contents of data are common to command/monitor. POSE // XYZ type [mm/rad] JOINT // Joint type [rad] PULSE // Motor pulse type [the pulse] or Current type [%].
Transmission input/output signal data designation SendIOType	unsigned short (2-byte)	1) When transmitting (commanding) from the personal computer to the controller, designate the data type of the input/output signal transmitted from the personal computer. Designate "No data" when not using this function. 2) When receiving (monitoring) from the controller to the personal computer, this indicates the data type of the input/output signal replied from the controller. The contents of the data are common. 0 // No data 1 // Output signal 2 // Input signal
Reply input/output signal data designation		1) When transmitting (commanding) from the personal computer to the controller, designate the data type of the input/output signal replied from

RecvIOType	unsigned short (2-byte)	the controller. Designate "No data" when not using this function. 0 // No data 1 // Output signal 2 // Input signal 2) When receiving (monitoring) from the controller to the personal computer, Not used.

Input/output signal data
BitTop
BitMask
IoData
	unsigned short
	unsigned short
	unsigned short
	(2-byte x 3)
		1) When transmitting (commanding) from the personal computer to the controller, designate the output signal data transmitted from the personal computer.
		
		2) When receiving (monitoring) from the controller to the personal computer, this indicates the input/output signal data replied from the controller.
		
		The contents of the data are common.
		BitTop; // Head bit No. of input or output signal
		BitMask; // Bit mask pattern designation (valid only for commanding)
		IoData; // Input or output signal data value (for monitoring)
		Output signal data value (for commanding)
		* Data is 16-bit data

Timeout time counter value Tcount	unsigned short (2-byte)	1) When transmitting (commanding) from the personal computer to the controller, Not used. 2) When receiving (monitoring) from controller to personal computer, if the timeout time parameter MXTTOUT is a value other than -1, this indicates the No. of times communication with the controller was not possible. When the No. of times is counted and reaches the maximum value, the value will return to the minimum value 0, and the count will be repeated. This is set to 0 when the MXT command is started.
Counter value for communication data Ccount	unsigned long (4-byte)	1) When transmitting (commanding) from the personal computer to the controller, Not used. 2) When receiving (monitoring) from controller to personal computer, this indicates the No. of communication times. When the No. of times is counted and reaches the maximum value, the value will return to the minimum value 0, and the count will be repeated. This is set to 0 when the MXT command is started.
Reply data-type specification addition 1 RecvType1	**unsigned short (2-byte)**	**It is the same as reply data-type specification (RecvType). Don't use it for instructions.**
Reservation 1 reserve1	**unsigned short (2-byte)**	**Not used.**
Data addition 1 pos / jnt / pls	**Any of POSE/JOINT/PULSE. (40-byte)**	**It is the same as data of pos/jnt/pls. Don't use it for instructions.**
Reply data-type specification addition 2 RecvType2	**unsigned short (2-byte)**	**It is the same as reply data-type specification (RecvType). Don't use it for instructions.**
Reservation 2	**unsigned short**	**Not used.**

Reserve2	(2-byte)	
Data addition 2 pos / jnt / pls	Any of POSE/JOINT/PULSE. (40-byte)	It is the same as data of pos/jnt/pls. Don't use it for instructions.
Reply data-type specification addition 3 RecvType3	unsigned short (2-byte)	It is the same as reply data-type specification (RecvType). Don't use it for instructions.
Reservation 3 Reserve3	unsigned short (2-byte)	Not used.
Data addition 3 pos / jnt / pls	Any of POSE/JOINT/PULSE. (40-byte)	It is the same as data of pos/jnt/pls. Don't use it for instructions.

5. Appendix

5.1. Error list

The errors which occur only when the Ethernet interface is used are listed as follows.

Error No.	Error causes and remedies
7800	■ Two Ethernet interfaces are installed. Cause) One Ethernet interface alone is allowed to install. Measures) Install one Ethernet interface. ■ Initialization error of Ethernet interface. Cause) The card is faulted. Measures) Replace the card.
7810	■ Parameter ***** setting error of Ethernet interface parameter. Cause) ***** parameter is wrongly set. (The parameter name is input in *****.) Measures) Check the setting content of the parameter.
7820	■ MXT Command time out. Cause) The time set in parameter MXTTOUT was exceeded. Measures) Check parameter MXTTOUT.
7830	■ Ethernet interface not installed. Cause) The Ethernet interface is not installed. Measures) Install the Ethernet interface.
7840	■ Received MXT command data illegal. Cause) The command argument and data type do not match. Measures) Check the contents of the command and the communication data packet to be transmitted.

For the other errors except these, refer to the errors list of the instruction manual of the controller.

5.2. Sample program

This is the sample program of the Ethernet interface.

5.2.1. Sample program of data link

The sample program to do the data link with Microsoft Visual Basic 5.0/6.0 (hereafter written as VB) is herein described.

The program creation is briefly introduced with the following procedure.

For details of VB operation and application producing method, refer to the instruction manual of this software.

(1) Preparation of Winsock control

(2) Production of form screen

(3) Program (Form1.frm)

There is the program following 2 passages. Use either according to the customer's system.

1) Program for the clients (when using the personal computer as the client and using the controller as the server).

2) Program for the server (when using the personal computer as the server and using the controller as the client).

* About the work of 1) 2), the client and the server are the same.

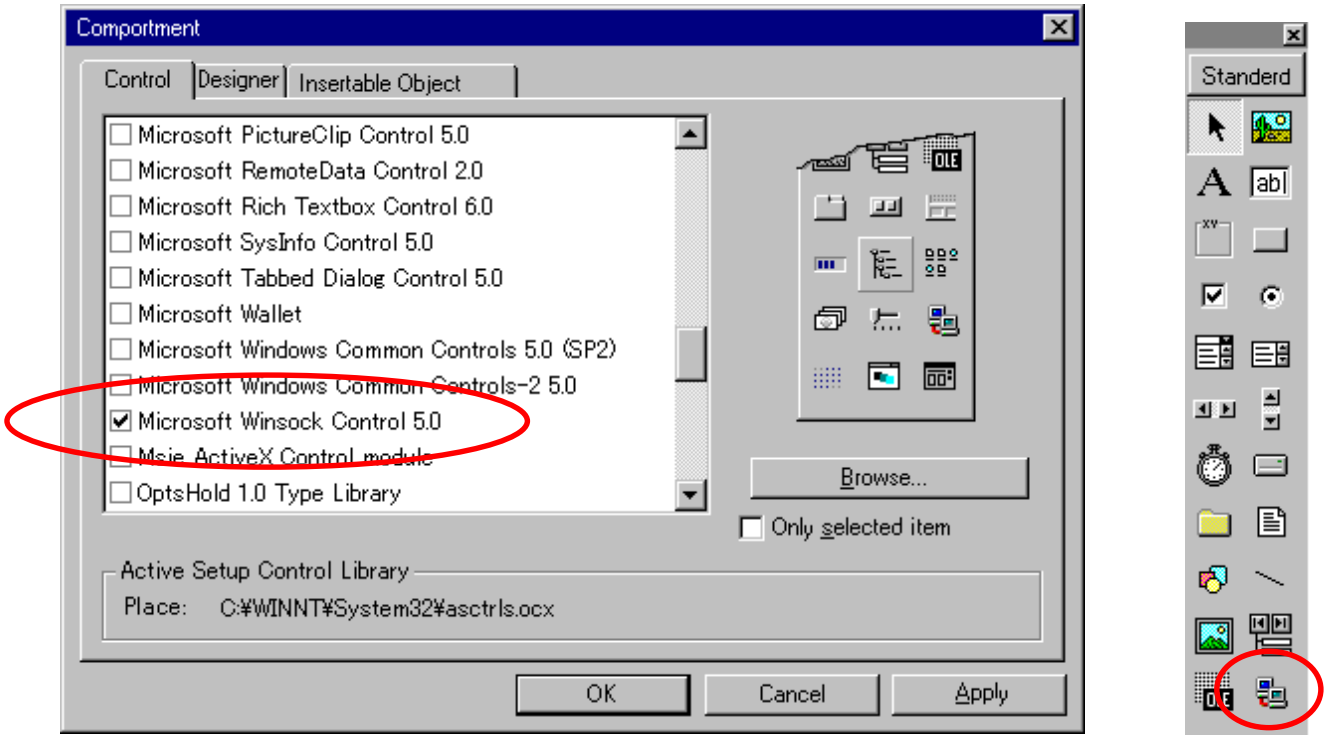
Here, VB requires either Professional Edition or Enterprise Edition. Learning Edition can not be used since Winsock (Windows Socket) control is not appended.

(1) Preparation of Winsock control

Winsock control is added to the project.

Start-up VB, newly open standard EXE and click "component" of "project" menu, and the window will be displayed as follows. And, check "Microsoft Winsock Control **". (Lower left drawing ** represents the version)

"Winsock" is added to the tool box. (Lower right drawing)

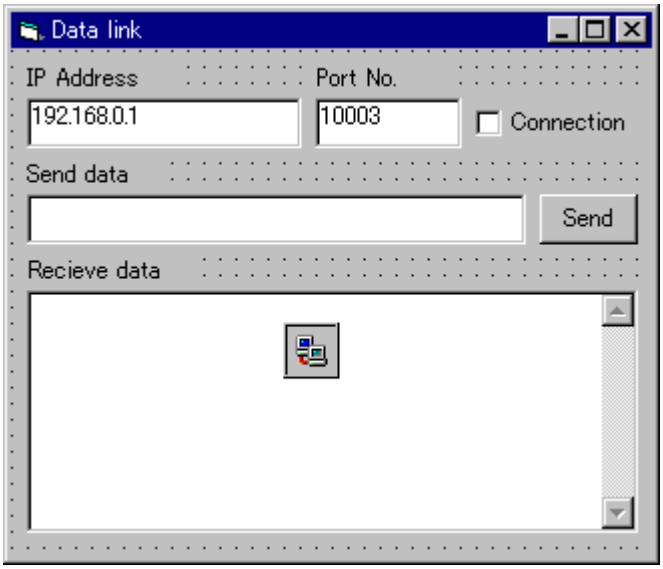


(2) Production of form screen

On the form, 4 test boxes, 1 command button, 1 check box and 1 Winsock control are arranged.

The major change points of the properties are shown below.

Major changed points of properties		
Object name	Property	Setting value
Form1	Caption	Data link
Command1	Caption	Send
	Enabled	False
Text1	Text	192.168.0.1
Text2	Text	10003
Text3	Text	
Text4	MultiLine	True
	ScrollBars	2-Vertical
Check1	Caption	Connection



(3) Program (Form1.frm)

```
VERSION 5.00
Object = "{248DD890-BB45-11CF-9ABC-0080C7E7B78D}#1.0#0"; "MSWINSCK.OCX"

Begin VB.Form Form1                    'Screen setting      From here ↓
   Caption         =   "Data link"
   ClientHeight    =   3795
   ClientLeft      =   60
   ClientTop       =   345
   ClientWidth     =   4800
   LinkTopic       =   "Form1"
   ScaleHeight     =   3795
   ScaleWidth      =   4800
   StartUpPosition =   3  Predefined value of Windows
   Begin MSWinsockLib.Winsock Winsock1
      Left         =   2040
      Top          =   2040
      _ExtentX     =   741
      _ExtentY     =   741
   End
   Begin VB.CommandButton Command1
      Caption      =   "Send"
      Enabled      =   0   'False
      Height       =   375
      Left         =   3960
      TabIndex     =   6
      Top          =   1080
      Width        =   735
   End
   Begin VB.CheckBox Check1
      Caption      =   "Connection"
      Height       =   375
      Left         =   3960
      TabIndex     =   4
      Top          =   360
      Width        =   735
   End
   Begin VB.TextBox Text4
      Height       =   1815
      Left         =   120
      MultiLine    =   -1  'True
      ScrollBars   =   2  'Vertical
      TabIndex     =   7
      Top          =   1800
      Width        =   4575
   End
   Begin VB.TextBox Text3
      Height       =   375
      Left         =   120
      TabIndex     =   5
      Top          =   1080
      Width        =   3735
   End
   Begin VB.TextBox Text2
      Height       =   375
      Left         =   2280
      TabIndex     =   3
      Text         =   "10003"
```

```
        Top             =   360
        Width           =   1575
    End
    Begin VB.TextBox Text1
        Height          =   375
        Left            =   120
        TabIndex        =   2
        Text            =   "192.168.0.1"
        Top             =   360
        Width           =   2055
    End
    Begin VB.Label Label4
        Caption         =   "Receive data"
        Height          =   195
        Left            =   120
        TabIndex        =   9
        Top             =   1560
        Width           =   975
    End
    Begin VB.Label Label3
        Caption         =   "Send data"
        Height          =   195
        Left            =   120
        TabIndex        =   8
        Top             =   840
        Width           =   975
    End
    Begin VB.Label Label2
        Caption         =   "Port No."
        Height          =   195
        Left            =   2280
        TabIndex        =   1
        Top             =   120
        Width           =   975
    End
    Begin VB.Label Label1
        Caption         =   "IP address"
        Height          =   255
        Left            =   120
        TabIndex        =   0
        Top             =   120
        Width           =   1095
    End
End                     'Screen setting     To here  ↑

Attribute VB_Name          = "Form1"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable   = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed     = False
```

1) Program for the clients (when using the personal computer as the client and using the controller as the server).

```
Option Explicit
Dim RecvData() As Byte

Private Sub Check1_Click()                ' Process when the connection check button is clicked
If Check1.Value Then
    Winsock1.RemoteHost   = Text1.Text
    Winsock1.RemotePort   = Text2.Text
    Winsock1.Connect
Else
    Winsock1.Close
End If
End Sub

Private Sub Winsock1_Connect()          ' Process when the network can be connected
    Command1.Enabled = True
End Sub

Private Sub Winsock1_Close()              ' Process when the network is closed
    Check1.Value = False
End Sub

Private Sub Command1_Click()             ' Process when "Transmission" command button is clicked
    Winsock1.SendData (Text3.Text)
End Sub

Private Sub Winsock1_DataArrival(ByVal bytesTotal As Long)      ' Process when the received data arrives
    If bytesTotal > 0 Then
        ReDim RecvData(bytesTotal - 1)
        Call Winsock1.GetData(RecvData, , bytesTotal)
        Text4.SelStart = Len(Text4.Text)
        Text4.SelText = StrConv(RecvData, vbUnicode)
    End If
End Sub

Private Sub Winsock1_Error(ByVal Number As Integer, _
    Description As String, ByVal Scode As Long, _
    ByVal Source As String, ByVal HelpFile As String, _
    ByVal HelpContext As Long, CancelDisplay As Boolean)       ' Process when an error occurs in Window Socket
    Check1.Value = False
    Command1.Enabled = False
    Winsock1.Close
    MsgBox " Error:" & Number & "(" & Description & ")"
End Sub
```

2) Program for the server (when using the personal computer as the server and using the controller as the client).

```
Option Explicit
Dim RecvData() As Byte

Private Sub Form_Load()
    Text1.Enabled = False          ' Make edit of the IP address impossible.
End Sub

Private Sub Check1_Click()         ' Process when the connection check button is clicked
    If Check1.Value Then
        Text1.Text = Winsock1.LocalIP
        Winsock1.LocalPort = Text2.Text
        Winsock1.Listen
    Else
        Command1.Enabled = False
        Winsock1.Close
    End If
End Sub

Private Sub Winsock1_Connect()    ' Process when the network can be connected
    Command1.Enabled = True
End Sub

Private Sub Winsock1_Close()      ' Process when the network is closed
    Check1.Value = False
End Sub

Private Sub Command1_Click()      ' Process when "Transmission" command button is clicked
    Winsock1.SendData (Text3.Text)
End Sub

Private Sub Winsock1_ConnectionRequest(ByVal requestID As Long) ' Process when the connection demand comes
    If Winsock1.State <> sckClosed Then Winsock1.Close
    Winsock1.Accept requestID
    Command1.Enabled = True
End Sub

Private Sub Winsock1_DataArrival(ByVal bytesTotal As Long)          ' Process when the received data arrives
    If bytesTotal > 0 Then
        ReDim RecvData(bytesTotal - 1)
        Call Winsock1.GetData(RecvData, , bytesTotal)
        Text4.SelStart = Len(Text4.Text)
        Text4.SelText = StrConv(RecvData, vbUnicode)
        Text4.Text = Text4.Text & vbCrLf
    End If
End Sub

Private Sub Winsock1_Error(ByVal Number As Integer, _
    Description As String, ByVal Scode As Long, _
    ByVal Source As String, ByVal HelpFile As String, _
    ByVal HelpContext As Long, CancelDisplay As Boolean)        ' Process when an error occurs in Window Socket
    Check1.Value = False
    Command1.Enabled = False
    Winsock1.Close
    MsgBox "Error:" & Number & "(" & Description & ")"
End Sub
```

■ Relation of OPEN command communication line file name COMn: and parameter COMDEV

COMDEV (1), (2), (3), (4), (5), (6), (7), (8)

Communication line file name	COMDEV
COM1:	(1)
COM2:	(2)
COM3:	(3)
COM4:	(4)
COM5:	(5)
COM6:	(6)
COM7:	(7)
COM8:	(8)

■ Channel name assigned to parameter COMDEV and protocol setting parameter name

OPT11 to OPT19 are assigned to (1) to (8).

The protocol is set in 2 (data link).

Port No.*1	Channel name	Protocol	
	COMDEV setting value	CPRCE**	Setting value
10001	OPT11	CPRCE11	2
10002	OPT12	CPRCE12	2
10003	OPT13	CPRCE13	2
10004	OPT14	CPRCE14	2
10005	OPT15	CPRCE15	2
10006	OPT16	CPRCE16	2
10007	OPT17	CPRCE17	2
10008	OPT18	CPRCE18	2
10009	OPT19	CPRCE19	2

*1... The port No. can be changed with parameter NETPORT.

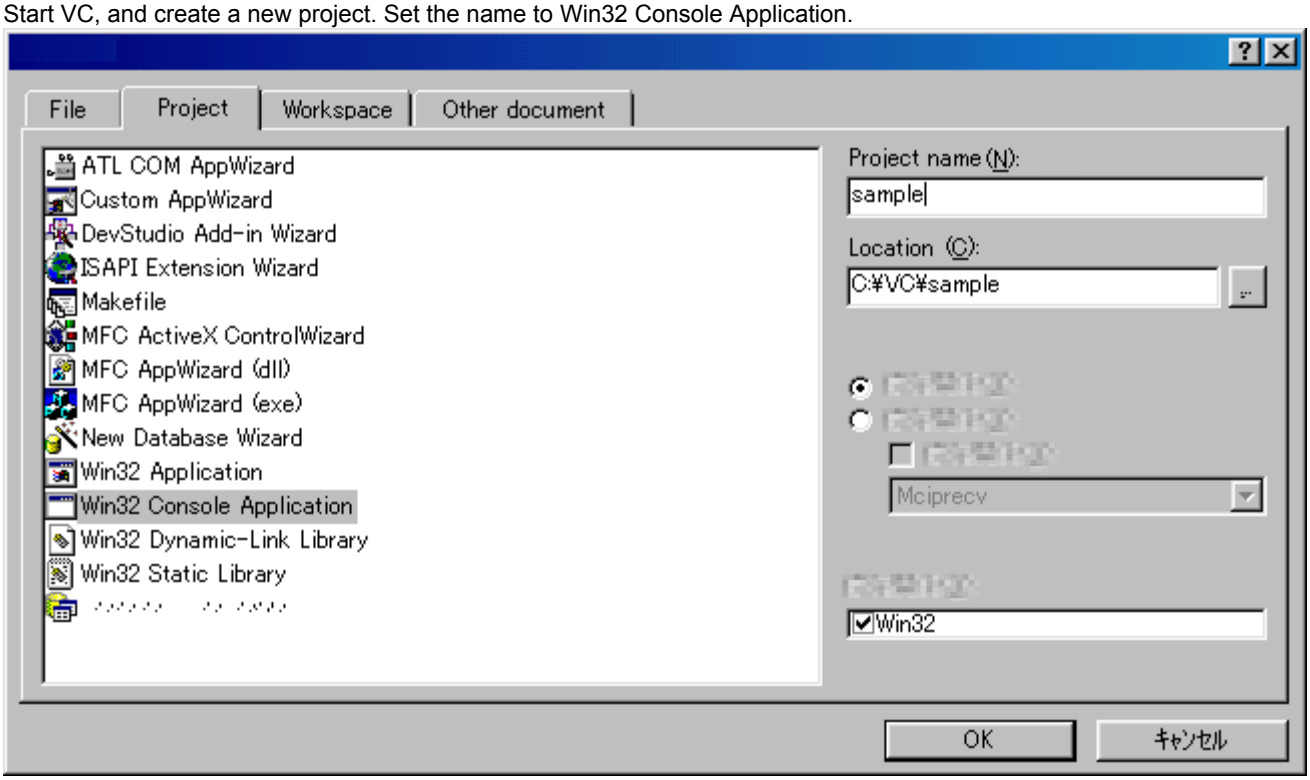
5.2.2. Sample program for real-time external control function

A sample program that establishes a data link using Microsoft Visual C++5.0/6.0 (hereinafter VC) is shown below.
The procedures for creating the program are briefly explained below.
Refer to the software manuals for details on operating VC and creating the application.

(1) Create new project
(2) Create program sample.cpp/strdef.h

(1) Create new project

Start VC, and create a new project. Set the name to Win32 Console Application.



Using the project setting, add wsock32.lib to the object/library module.

(2) Create program sample.cpp/strdef.h

Newly create the header file strdef.h and source file sample.cpp.

Make the comment a part of header file strdef.h according to the software version of the controller of use. (Refer to the list of strdef.h shown below for detail).

Refer to "1.5 Checking the robot controller software version" for check method of the version.

<Notes at compiling>

Use the setup of the alignment compiler option of the structure member with the 8 bytes of initial value. After new creation of the project of Visual C++, if the setup is used with initial value, there is no problem. Refer to the help of Visual C++ for details.

■ Header file strdef.h

```
//***************************************************************************************
// Real-time control sample program
// Communication packet data structure definition header file
//***************************************************************************************
// strdef.h

// If the software version of the controller is H7 or later, validate the following define line.
// If the version is H6 or earlier, make the following line the comment. (invalid).
#define VER_H7

/************************************************************************/
/*      Joint coordinate system (Set unused axis to 0)                  */
/*      Refer to the instruction manual enclosed                        */
/*      with each robot for details on each element.                    */
/************************************************************************/
typedef struct{
    float       j1;             //  J1 axis angle (radian)
    float       j2;             //  J2 axis angle (radian)
    float       j3;             //  J3 axis angle (radian)
    float       j4;             //  J4 axis angle (radian)
    float       j5;             //  J5 axis angle (radian)
    float       j6;             //  J6 axis angle (radian)
    float       j7;             //  Additional axis 1 (J7 axis angle) (radian)
    float       j8;             //  Additional axis 2 (J8 axis angle) (radian)
} JOINT;

/************************************************************************/
/*      XYZ coordinate system (Set unused axis to 0)                    */
/*      Refer to the instruction manual enclosed                        */
/*      with each robot for details on each element.                    */
/************************************************************************/
typedef struct{
    float       x;              //  X axis coordinate value (mm)
    float       y;              //  Y axis coordinate value (mm)
    float       z;              //  Z axis coordinate value (mm)
    float       a;              //  A axis coordinate value (radian)
    float       b;              //  B axis coordinate value (radian)
    float       c;              //  C axis coordinate value (radian)
    float       l1;             //  Additional axis 1 (mm or radian)
    float       l2;             //  Additional axis 2 (mm or radian)
} WORLD;

typedef struct{
    WORLD       w;
    unsigned int    sflg1;      //  Structural flag 1
    unsigned int    sflg2;      //  Structural flag 2
} POSE;

/************************************************************************/
/*      Pulse coordinate system (Set unused axis to 0)                  */
/*      These coordinates express each joint                            */
/*      with a motor pulse value.                                       */
/************************************************************************/
typedef struct{
    long        p1;             //  Motor 1 axis
    long        p2;             //  Motor 2 axis
```

```
    long        p3;             //  Motor 3 axis
    long        p4;             //  Motor 4 axis
    long        p5;             //  Motor 5 axis
    long        p6;             //  Motor 6 axis
    long        p7;             //  Additional axis 1 (Motor 7 axis)
    long        p8;             //  Additional axis 2 (Motor 8 axis)
} PULSE;


/************************************************************/
/*  Real-time function communication data packet    */
/************************************************************/
typedef struct enet_rtcmd_str {
    unsigned short  Command;    //  Command
#define  MXT_CMD_NULL       0       //  Real-time external command invalid
#define  MXT_CMD_MOVE       1       //  Real-time external command valid
#define  MXT_CMD_END        255     //  Real-time external command end

    unsigned short  SendType;   //  Command data type designation
    unsigned short  RecvType;   //  Monitor data type designation
                                    ////////////// Command or monitor data type ///
#define  MXT_TYP_NULL       0       //  No data
                                    //  For the command and monitor ////////////////////
#define  MXT_TYP_POSE       1       //  XYZ data
#define  MXT_TYP_JOINT      2       //  Joint data
#define  MXT_TYP_PULSE      3       //  pulse data
                                    ////////////// For position related monitor ///
#define  MXT_TYP_FPOSE      4       //  XYZ data (after filter process)
#define  MXT_TYP_FJOINT     5       //  Joint data (after filter process)
#define  MXT_TYP_FPULSE     6       //  Pulse data (after filter process)
#define  MXT_TYP_FB_POSE    7       //  XYZ data (Encoder feedback value)          <H7A>
#define  MXT_TYP_FB_JOINT   8       //  Joint data (Encoder feedback value)        <H7A>
#define  MXT_TYP_FB_PULSE   9       //  Pulse data (Encoder feedback value)        <H7A>
                                    //  For current related monitors ////////////////////  <H7A>
#define  MXT_TYP_CMDCUR     10      //  Electric current command                   <H7A>
#define  MXT_TYP_FBKCUR     11      //  Electric current feedback                  <H7A>

    unsigned short  reserve;            //  Reserved
    union rtdata {                  //  Command data
        POSE    pos;                    //  XYZ type [mm/rad]
        JOINT   jnt;                    //  Joint type [rad]
        PULSE pls;                      //  Pulse type [pls]
        long    lng1[8];                //  Integer type [% / non-unit]
    } dat;

    unsigned short  SendIOType;  // Send input/output signal data designation
    unsigned short  RecvIOType;  // Return input/output signal data designation
#define  MXT_IO_NULL        0       //  No data
#define  MXT_IO_OUT         1       //  Output signal
#define  MXT_IO_IN          2       //  Input signal

    unsigned short  BitTop;         // Head bit No.
    unsigned short  BitMask;        // Transmission bit mask pattern designation (0x0001-0xffff)
    unsigned short  IoData;         // Input/output signal data (0x0000-0xffff)

    unsigned short  TCount;         // Timeout time counter value
    unsigned long   CCount;         // Transmission data counter value
```

```
#ifdef   VER_H7
    unsigned short   RecvType1;   // Reply data-type specification 1 .
    unsigned short   reserve1;    // Reserved 1
    union rtdata1 {               // Monitor data 1 .
        POSE   pos1;              // XYZ type [mm/rad] .
        JOINT  jnt1;              // JOINT type [mm/rad] .
        PULSE pls1;               // PULSE type [mm/rad] .
        long    lng1[8];          // Integer type [% / non-unit] .
    } dat1;
    unsigned short   RecvType2;   // Reply data-type specification 2 .
    unsigned short   reserve2;    // Reserved 2
    union rtdata2 {               // Monitor data 2 .
        POSE   pos2;              // XYZ type [mm/rad] .
        JOINT  jnt2;              // JOINT type [mm/rad] .
        PULSE pls2;               // PULSE type [mm/rad] or Integer type [% / non-unit].
        long    lng2[8];          // Integer type [% / non-unit] .
    } dat2;
    unsigned short   RecvType3;   // Reply data-type specification 3 .
    unsigned short   reserve3;    // Reserved 3
    union rtdata3 {               // Monitor data 3 .
        POSE   pos3;              // XYZ type [mm/rad] .
        JOINT  jnt3;              // JOINT type [mm/rad] .
        PULSE pls3;               // PULSE type [mm/rad] or Integer type [% / non-unit].
        long    lng3[8];          // Integer type [% / non-unit] .
    } dat3;
#endif

} MXTCMD;
```

■ Source file sample.cpp

```
// sample.cpp

// Change the definition in the "strdef.h" file by the S/W version of the controller.
// Refer to the "strdef.h" file for details.

#include <windows.h>
#include <iostream.h>
#include <winsock.h>
#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <math.h>
#include "strdef.h"
#define NO_FLAGS_SET 0
#define MAXBUFLEN 512

INT main(VOID)
{
    WSADATA Data;
    SOCKADDR_IN destSockAddr;
    SOCKET destSocket;
    unsigned long destAddr;
    int status;
    int numsnt;
    int numrcv;
    char sendText[MAXBUFLEN];
    char recvText[MAXBUFLEN];
```

```
char dst_ip_address[MAXBUFLEN];
unsigned short   port;
char msg[MAXBUFLEN];
char buf[MAXBUFLEN];
char type, type_mon[4];
unsigned short IOSendType;    // Send input/output signal data designation
unsigned short IORecvType;    // Reply input/output signal data designation
unsigned short IOBitTop=0;
unsigned short IOBitMask=0xffff;
unsigned short IOBitData=0;

cout << " Input connection destination IP address (192.168.0.1) ->";
cin.getline(dst_ip_address, MAXBUFLEN);
if(dst_ip_address[0]==0)      strcpy(dst_ip_address, "192.168.0.1");

cout << " Input connection destination port No. (10000) -> ";
cin.getline(msg, MAXBUFLEN);
if(msg[0]!=0)    port=atoi(msg);
else                port=10000;

cout << " Use input/output signal?（[Y] / [N]） -> ";
cin.getline(msg, MAXBUFLEN);
if(msg[0]!=0 && (msg[0]=='Y' || msg[0]=='y')) {
    cout << "   What is target? Input signal/output signal（[I]nput / [O]utput） -> ";
    cin.getline(msg, MAXBUFLEN);
    switch(msg[0]) {
        case 'O':                // Set target to output signal
        case 'o':
            IOSendType = MXT_IO_OUT;
            IORecvType = MXT_IO_OUT;
            break;
        case 'I':                // Set target to input signal
        case 'i':
        default:
            IOSendType = MXT_IO_NULL;
            IORecvType = MXT_IO_IN;
            break;
    }

    cout << " Input head bit No. （0～32767） -> ";
    cin.getline(msg, MAXBUFLEN);
    if(msg[0]!=0)    IOBitTop = atoi(msg);
    else               IOBitTop = 0;

    if(IOSendType==MXT_IO_OUT) {      // Only for output signal
        cout << "   Input bit mask pattern for output as hexadecimal （0000～FFFF） -> ";
        cin.getline(msg, MAXBUFLEN);
        if(msg[0]!=0)    sscanf(msg,"%4x",&IOBitMask);
        else               IOBitMask = 0;
        cout << "   Input bit data for output as hexadecimal （0000～FFFF） -> ";
        cin.getline(msg, MAXBUFLEN);
        if(msg[0]!=0)    sscanf(msg,"%4x",&IOBitData);
        else               IOBitData = 0;
    }
}
cout <<" --- Input the data type of command.     --- ¥n";
cout <<"[0: None / 1: XYZ / 2:JOINT / 3: PULSE]¥n".;
cout <<" -- please input the number --   [0] - [3]->";
```

```
        cin.getline(msg, MAXBUFLEN);
        type = atoi(msg);

    #ifdef   VER_H7
        for(int k=0; k<4; k++) {
            sprintf (msg," --- input the data type of monitor     ( %d-th ) --- ¥n", k); .
            cout << msg;
            cout << "[0: None]¥n";
            cout << "[1: XYZ / 2:JOINT / 3: PULSE]   .....  Command value ¥n";
            cout << "[4: XYZ/ 5: JOINT/ 6: PULSE]   .....  Command value after the filter process ¥n";
            cout << "[7: XYZ/ 5:JOINT/ 6:PULSE]   .....  Feedback value. ¥n";
            cout << "[10: Electric current value / 11: Electric current feedback] ... Electric current value. ¥n";
            cout << "Input the numeral [0]〜[11] -> ";
            cin.getline(msg, MAXBUFLEN);
            type_mon[k] = atoi(msg);
        }
    #else
        type_mon[0]=type;
        type_mon[1]=type_mon[2]=type_mon[3]=0;
    #endif
        sprintf(msg, "IP=%s / PORT=%d / Send Type=%d / Monitor Type0/1/2/3=%d/%d/%d/%d"
                        , dst_ip_address, port , type
    , type_mon[0], type_mon[1], type_mon[2], type_mon[3]);
        cout << msg << endl;

        cout << "[Enter]= End ／ [d]= Monitor data display";
        cout << "[z/x]= Increment/decrement first command data transmitted by the delta amount. ";

        cout << " Is it all right? [Enter] / [Ctrl+C] ";
        cin.getline(msg, MAXBUFLEN);

        // Windows Socket DLL initialization
        status=WSAStartup(MAKEWORD(1, 1), &Data);
        if (status != 0)
        cerr << "ERROR: WSAStartup unsuccessful" << endl;

        // IP address, port, etc., setting
        memset(&destSockAddr, 0, sizeof(destSockAddr));
        destAddr=inet_addr(dst_ip_address);
        memcpy(&destSockAddr.sin_addr, &destAddr, sizeof(destAddr));
        destSockAddr.sin_port=htons(port);
        destSockAddr.sin_family=AF_INET;

        // Socket creation
        destSocket=socket(AF_INET, SOCK_DGRAM, 0);
        if (destSocket == INVALID_SOCKET) {
            cerr << "ERROR: socket unsuccessful" << endl;
            status=WSACleanup();
            if (status == SOCKET_ERROR)
                cerr << "ERROR: WSACleanup unsuccessful" << endl;
            return(1);
        }

        MXTCMD    MXTsend;
        MXTCMD    MXTrecv;
        JOINT     jnt_now;
        POSE      pos_now;
        PULSE     pls_now;
```

```
    unsigned long     counter = 0;
    int loop = 1;
    int disp = 0;
    int disp_data = 0;
    int ch;
    float delta=(float)0.0;
    long ratio=1;
    int retry;
    fd_set        SockSet;                    // Socket group used with select
    timeval       sTimeOut;                   // For timeout setting

    memset(&MXTsend, 0, sizeof(MXTsend));
    memset(&jnt_now, 0, sizeof(JOINT));
    memset(&pos_now, 0, sizeof(POSE));
    memset(&pls_now, 0, sizeof(PULSE));

    while(loop) {

        memset(&MXTsend, 0, sizeof(MXTsend));
        memset(&MXTrecv, 0, sizeof(MXTrecv));

        // Transmission data creation
        if(loop==1) {     // Only first time
            MXTsend.Command = MXT_CMD_NULL;
            MXTsend.SendType = MXT_TYP_NULL;
            MXTsend.RecvType = type;
            MXTsend.SendIOType = MXT_IO_NULL;
            MXTsend.RecvIOType = IOSendType;
            MXTsend.CCount = counter = 0;
        }
        else {              // Second and following times
            MXTsend.Command = MXT_CMD_MOVE;
            MXTsend.SendType = type;
            MXTsend.RecvType = type*_mon[0];
#ifdef   VER_H7
            MXTsend.RecvType1= type_mon[1];
            MXTsend.RecvType2= type_mon[2];
            MXTsend.RecvType3= type_mon[3];
#endif
            switch(type) {
                case MXT_TYP_JOINT:
                    memcpy(&MXTsend.dat.jnt, &jnt_now, sizeof(JOINT));
                    MXTsend.dat.jnt.j1 += (float)(delta*ratio*3.141592/180.0);
                    break;
                case MXT_TYP_POSE:
                    memcpy(&MXTsend.dat.pos, &pos_now, sizeof(POSE));
                    MXTsend.dat.pos.w.x += (delta*ratio);
                    break;
                case MXT_TYP_PULSE:
                    memcpy(&MXTsend.dat.pls, &pls_now, sizeof(PULSE));
                    MXTsend.dat.pls.p1 += (long)((delta*ratio)*10);
                    break;
                default:
                    break;
            }
            MXTsend.SendIOType = IOSendType;
            MXTsend.RecvIOType = IORecvType;
```

```
        MXTsend.BitTop = IOBitTop;
        MXTsend.BitMask =IOBitMask;
        MXTsend.IoData = IOBitData;
        MXTsend.CCount = counter;
    }

    // Keyboard input
    // [Enter]=End / [d]= Display the monitor data, or none / [0/1/2/3]= Change of monitor data display
    // [z/x]=Increment/decrement first command data transmitted by the delta amount
    while(kbhit()!=0) {
        ch=getch();
        switch(ch) {
        case 0x0d:
            MXTsend.Command = MXT_CMD_END;
            loop = 0;
            break;
        case 'Z':
        case 'z':
            delta += (float)0.1;
            break;
        case 'X':
        case 'x':
            delta -= (float)0.1;
            break;
        case 'C':
        case 'c':
            delta = (float)0.0;
            break;
        case 'd':
            disp = ~disp;
            break;
        case '0': case '1': case '2': case '3':
            disp_data = ch - '0';
            break;
        }
    }

    memset(sendText, 0, MAXBUFLEN);
    memcpy(sendText, &MXTsend, sizeof(MXTsend));
    if(disp) {
        sprintf(buf, "Send       (%ld):",counter);
        cout << buf << endl;
    }
    numsnt=sendto(destSocket, sendText, sizeof(MXTCMD), NO_FLAGS_SET
                                , (LPSOCKADDR) &destSockAddr, sizeof(destSockAddr));
    if (numsnt != sizeof(MXTCMD)) {
        cerr << "ERROR: sendto unsuccessful" << endl;
        status=closesocket(destSocket);
        if (status == SOCKET_ERROR)
            cerr << "ERROR: closesocket unsuccessful" << endl;
        status=WSACleanup();
        if (status == SOCKET_ERROR)
            cerr << "ERROR: WSACleanup unsuccessful" << endl;
            return(1);
    }

    memset(recvText, 0, MAXBUFLEN);
```

```
        retry = 1;                                          // No. of reception retries
        while(retry) {
            FD_ZERO(&SockSet);                              // SockSet initialization
            FD_SET(destSocket, &SockSet);                   // Socket registration
            sTimeOut.tv_sec = 1;                            // Transmission timeout setting (sec)
            sTimeOut.tv_usec = 0;                           //                              (μsec)
            status = select(0,   &SockSet, (fd_set *)NULL, (fd_set *)NULL,   &sTimeOut);
            if(status == SOCKET_ERROR) {
                return(1);
            }
                                                            // If it receives by the time-out
            if((status > 0) && (FD_ISSET(destSocket, &SockSet) != 0)) {
                numrcv=recvfrom(destSocket, recvText, MAXBUFLEN, NO_FLAGS_SET, NULL, NULL);
                if (numrcv == SOCKET_ERROR) {
                    cerr << "ERROR: recvfrom unsuccessful" << endl;
                    status=closesocket(destSocket);
                    if (status == SOCKET_ERROR)
                        cerr << "ERROR: closesocket unsuccessful" << endl;
                    status=WSACleanup();
                    if (status == SOCKET_ERROR)
                        cerr << "ERROR: WSACleanup unsuccessful" << endl;
                      return(1);
                }
                memcpy(&MXTrecv, recvText, sizeof(MXTrecv));
                char    str[10];
                if(MXTrecv.SendIOType==MXT_IO_IN)
                        sprintf(str,"IN%04x", MXTrecv.IoData);
                else if(MXTrecv.SendIOType==MXT_IO_OUT)
                        sprintf(str,"OT%04x", MXTrecv.IoData);
                else    sprintf(str,"------");

                int     DispType;
                void    *DispData;
#ifdef  VER_H7
                switch(disp_data) {
                    case 0:
                        DispType = MXTrecv.RecvType;
                        DispData = &MXTrecv.dat;
                        break;
                    case 1:
                        DispType = MXTrecv.RecvType1;
                        DispData = &MXTrecv.dat1;
                        break;
                    case 2:
                        DispType = MXTrecv.RecvType2;
                        DispData = &MXTrecv.dat2;
                        break;
                    case 3:
                        DispType = MXTrecv.RecvType3;
                        DispData = &MXTrecv.dat3;
                        break;
                    default:
                        break;
                }
#else
                DispType = MXTrecv.SendType;
                DispData = &MXTrecv.dat;
#endif
```

```
switch(DispType) {
    case MXT_TYP_JOINT:
    case MXT_TYP_FJOINT:
    case MXT_TYP_FB_JOINT:
        if(loop==1) {
            memcpy(&jnt_now, DispData, sizeof(JOINT));
            loop = 2;
        }
        if(disp) {
            JOINT     *j=(JOINT*)DispData;
            sprintf(buf, "Receive (%ld): TCount=%d Type(JOINT)=%d¥n
                    %7.2f,%7.2f,%7.2f,%7.2f,%7.2f,%7.2f,%7.2f,%7.2f (%s)"
                     ,MXTrecv.CCount,MXTrecv.TCount,DispType
                     ,j->j1, j->j2, j->j3 ,j->j4, j->j5, j->j6, j->j7, j->j8, str);
            cout << buf << endl;
        }
        break;
    case MXT_TYP_POSE:
    case MXT_TYP_FPOSE:
    case MXT_TYP_FB_POSE:
        if(loop==1) {
            memcpy(&pos_now, &MXTrecv.dat.pos, sizeof(POSE));
            loop = 2;
        }
        if(disp) {
            POSE       *p=(POSE*)DispData;
            sprintf(buf, "Receive (%ld): TCount=%d    Type(POSE)=%d¥n
                     %7.2f,%7.2f,%7.2f,%7.2f,%7.2f,%7.2f, %04x,%04x (%s)"
                     ,MXTrecv.CCount,MXTrecv.TCount,DispType
                     ,p->w.x, p->w.y, p->w.z, p->w.a, p->w.b, p->w.c
                      , p->sflg1, p->sflg2, str);
            cout << buf << endl;
        }
        break;
    case MXT_TYP_PULSE:
    case MXT_TYP_FPULSE:
    case MXT_TYP_FB_PULSE:
    case MXT_TYP_CMDCUR:
    case MXT_TYP_FBKCUR:
        if(loop==1) {
            memcpy(&pls_now, &MXTrecv.dat.pls, sizeof(PULSE));
            loop = 2;
        }
        if(disp) {
            PULSE *l=(PULSE*)DispData;
            sprintf(buf, "Receive (%ld): TCount=%d    Type(PULSE/OTHER)=%d¥n
                    %ld,%ld,%ld,%ld,%ld,%ld,%ld,%ld (%s)"
                     ,MXTrecv.CCount,MXTrecv.TCount,DispType
                     ,l->p1, l->p2, l->p3, l->p4, l->p5, l->p6, l->p7, l->p8, str);
            cout << buf << endl;
        }
        break;
    case MXT_TYP_NULL:
        if(loop==1) {
            loop = 2;
        }
        if(disp) {
            sprintf(buf, "Receive (%ld): TCount=%d    Type(NULL)=%d¥n (%s)"
```

```
                                        ,MXTrecv.CCount,MXTrecv.TCount, DispType, str);
                                cout << buf << endl;
                            }
                            break;
                        default:
                            cout << "Bad data type.¥n" << endl;
                            break;
                    }
                    counter++;                  // Count up only when communication is successful
                    retry=0;                    // Leave reception loop
                }
                else {   // Reception timeout
                    cout << "... Receive Timeout!    <Push [Enter] to stop the program>" << endl;
                    retry--;                    // No. of retries subtraction
                    if(retry==0)      loop=0;   // End program if No. of retries is 0
                }
            } /* while(retry) */
        } /* while(loop) */

        // End
        cout << "/// End /// ";
        sprintf(buf, "counter = %ld", counter);
        cout << buf << endl;

        //   Close socket
        status=closesocket(destSocket);
        if (status == SOCKET_ERROR)
            cerr << "ERROR: closesocket unsuccessful" << endl;
        status=WSACleanup();
        if (status == SOCKET_ERROR)
            cerr << "ERROR: WSACleanup unsuccessful" << endl;

        return 0;
}
```

MITSUBISHI ELECTRIC CORPORATION
HEAD OFFICE: TOKYO BUILDING, 2-7-3, MARUNOUCHI, CHIYODA-KU, TOKYO 100-8310, JAPAN
NAGOYA WORKS: 5-1-14, YADA-MINAMI, HIGASHI-KU, NAGOYA 461-8670, JAPAN

Authorised representative:
MITSUBISHI ELECTRIC EUROPE B.V. GERMANY
Gothaer Str. 8, 40880 Ratingen / P.O. Box 1548, 40835 Ratingen, Germany

Oct..2009 MEE Printed in Japan on recycled paper.