HITACHI

USER'S MANUAL

OPTION

ET.NET

(LQE720)

S10V

Programmable Controller

SVE-1-128(C)

USER'S MANUAL

OPTION

ET.NET

（LQE720）

S10V

Programmable Controller

First Edition, May 2005, SVE-1-128(A) (out of print)
Second Edition, October 2008, SVE-1-128(B) (out of print)
Third Edition, February 2011, SVE-1-128(C)

Printed in Japan.

BI<IC> (FL-MW2007, AI10)

SAFETY PRECAUTIONS

- Before installation, operation, maintenance, and/or inspection of this product, be sure to read through carefully this manual and other related manuals. Do not use this product until you are familiar with all the information concerning this product, safety information, and precautions provided in those manuals.
- Keep this manual in a readily accessible place so that users of this product may easily reach it.
- This manual contains information on potential hazards that is intended as a guide for safe use of this product. The potential hazards listed in the manual are divided into four hazard levels of danger, warning, caution, and notice, according to the level of their severity. The following are definitions of the safety labels containing the corresponding signal words DANGER, WARNING, CAUTION, and NOTICE.

⚠ **DANGER**	: This safety label identifies precautions that, if not heeded, will result in death or serious injury.
⚠ **WARNING**	: Identifies precautions that, if not heeded, could result in death or serious injury.
⚠ **CAUTION**	: Identifies precautions that, if not heeded, could result in minor or moderate injury.
NOTICE	: This safety label without a safety alert symbol identifies precautions that, if not heeded, could result in property damage or loss not related to personal injury.

Failure to observe any of the ⚠CAUTION and *NOTICE* statements used in this manual could also lead to a serious consequence, depending on the situation in which this product is used. Therefore, be sure to observe all of those statements without fail.

The following are definitions of the phrases “serious injury,” “minor or moderate injury,” and “property damage or loss not related to personal injury” used in the above definitions of the safety labels.

Serious injury: Is an injury that requires hospitalization for medical treatment, has aftereffects, and/or requires long-term follow-up care. Examples of serious injuries are as follows: vision loss, burn (caused by dry heat or extreme cold), electric-shock injury, broken bone, poisoning, etc.

Minor or moderate injury: Is an injury that does not require either hospitalization for medical treatment or long-term follow-up care. Examples of minor or moderate injuries are as follows: burn, electric-shock injury, etc.

Property damage or loss not related to personal injury: Is a damage to or loss of personal property. Examples of property damages or losses not related to personal injury are as follows: damage to this product or other equipment or their breakdown, loss of useful data, etc.

The safety precautions stated in this manual are based on the general rules of safety applicable to this product. These safety precautions are a necessary complement to the various safety measures included in this product. Although they have been planned carefully, the safety precautions posted on this product and in the manual do not cover every possible hazard. Common sense and caution must be used when operating this product. For safe operation and maintenance of this product, establish your own safety rules and regulations according to your unique needs. A variety of industry standards are available to establish such safety rules and regulations.

The following are the hazard warning statements contained in this manual.

(Page 3-5)

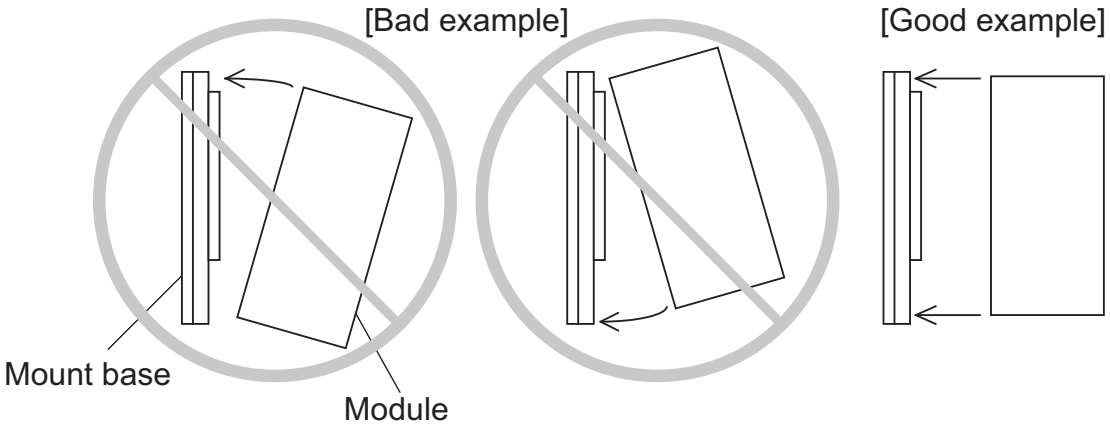
WARNING

- If the module emits smoke or foreign odor, immediately switch off the power supply and investigate the problem cause.
- Do not perform any installation, wiring, handling, or internal modification procedures other than stated in this manual. In no event will Hitachi be responsible for personal injury or death or any damage to Hitachi's product or peripheral equipment arising out of the use of such an unauthorized procedure.
- While the power is applied, never touch a terminal strip or connector pin. If you touch a terminal strip or connector pin while the power is applied, you may receive an electric shock.

(Page 3-5)

CAUTION

- Dust or other foreign matter might accumulate on the connector, resulting in poor contact. Immediately after the module is unpacked, perform the mounting and wiring procedures.
- To prevent the module from being damaged, observe the following precautions when you mount or demount the module:
  - Before mounting the module to the mount base connector, check that the connector pins are properly aligned and not bent, broken, or soiled with dirt or the like.
  - Ensure that the module is parallel to the mount base vertical surface as shown below when mounting. If you connect a module to or disconnect it from its connector while it is tilted, the connector pins may become damaged.
  - If the mount base is positioned overhead due to the employed enclosure structure, use a stepladder or the like and mount the module squarely. If you mount the module obliquely, the connector may become damaged.



(Page 3-3)

CAUTION

- At installation sites where there is a risk of a water leak, be sure to install the programmable controller in a drip-proof cubicle and use it. Disregarding this rule may result in failure of the product.
- Do not touch any of the modules in the programmable controller when they are in an energized state. Touching any of the modules in an energized state may lead to a discharge of static electricity from your body to the module, resulting in malfunction or breakage of the module. If you have no choice but to touch a module, be sure to discharge the static electricity by touching the metal frame of the cubicle and then touch the module. This is also true when you perform any of the following actions on a module in its non-energized state: 1) setting a switch on the module; 2) connecting or disconnecting the cable from the module; or 3) inserting or removing the connector from the module.

(Page 3-6)

CAUTION

- Observe the installation procedure stated in the manual.
  If the module is improperly installed, it may drop, become defective, or malfunction.
- Do not allow wire cuttings or other foreign matter to enter the module.
  The entry of foreign matter in the module may result in a fire or cause the module to become defective or malfunction.
- Static electricity may damage the module. Before starting the work, discharge all electrostatic charge from your body.
- Properly tighten the screws. If they are inadequately tightened, malfunction, smoke emission, or combustion may occur.

(Page 1-3)

NOTICE

- If the software supplied by Hitachi is modified for use, Hitachi cannot be responsible for accidents or losses resulting from such modification.
- Hitachi cannot be responsible for reliability if you use software other than supplied from Hitachi.
- Back up files on a daily basis. You might lose the contents of files due, for instance, to a file unit failure, power failure during a file access, or operating error. To provide against such contingencies, back up files according to an appropriate plan.
- Before scrapping the product, ask a professional waste disposal dealer in charge of scrapping work.
- Do not use a transceiver, cellular phone, or similar device near the module because module malfunction or system failure may occur due to noise.
- The contents of the memory may become damaged due, for instance, to a module failure. Be sure to make a backup of important data.
- Before constructing a system, creating a program, or performing a similar procedure, thoroughly read this manual to become familiar with the contained instructions and precautions. If you perform any incorrect procedure, the system may malfunction.
- Store this manual at a predetermined place where it can readily be referred to whenever it is needed.
- If you have any doubt or question about the contents of this manual, contact your local source.
- Hitachi cannot be responsible for accidents or losses resulting from a customer's misuse.
- If an emergency stop circuit, interlock circuit, or similar circuit is to be formulated, it must be positioned external to this module. If you do not observe this precaution, equipment damage or accident may occur when this module becomes defective.

(Page 2-2)

NOTICE
Switch off the power supply before operating the MAIN/SUB and the ST. No. setting switch. If you operate while the power supply is applied, it may result in a malfunction.

(Page 3-6)

NOTICE
Do not disassemble or modify the module. Failure to observe this precaution may result in a fire or cause the module to become defective or malfunction.

(Page 3-8)

NOTICE
● This hardware unit may malfunction if it is connected poorly or has a broken line. After connecting the connector, check whether the locking. ● Do not touch the connector during power on. Otherwise, the system may malfunction due to static electricity, etc.

(Page 4-5)

NOTICE
If Windows® opens a window during the uninstall process to display the question "Remove Shared File?," click the [No] button to retain shared files.

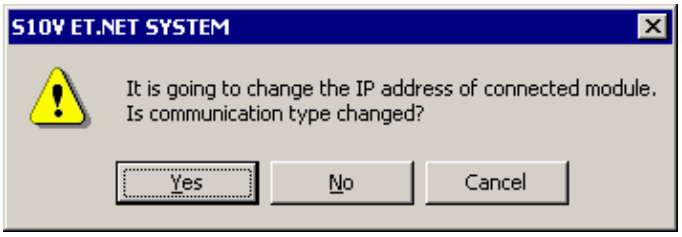
(Page 4-15)

NOTICE
The ET.NET module may have its IP address displayed along with an asterisk ("*") at its beginning. In this case, the asterisk indicates that, although the IP address has been changed, the ET.NET module is operating with its old IP address because the PCs has not been reset yet. To make the newly set IP address effective, be sure to reset the PCs.

(Page 4-18)

NOTICE

Any attempt to change the IP address of the ET.NET module connected by Ethernet cable wiring will have the following message displayed on screen.



If you want to reset the programmable controller and re-establish a connection with it by using the newly set IP address, click the [Yes] button. If not, click the [No] button. If you want to abort the IP address setting process, click the [Cancel] button.

(Page 4-22)

NOTICE

The routing information entered is not registered in the PCs or file until you click the [Write] button in the [Set IP Address] window. Thus, if you click the [Cancel] button in place of [OK] in that window, the routing information will not be registered in the PCs or file.

(Page 5-5)

NOTICE

- The maximum number of sockets that can be used simultaneously by one single module is 24 for TCP and UDP.
- The port numbers 0 to 9999 are reserved by the system; the user can use the port numbers 10000 to 65535 (except the ports 60015 and 60016 for TCP and 60013 and 60020 for UDP, which are used exclusively by the system).
- The length of data to be transmitted or received in each invocation of a function is 1 to 4096 bytes for TCP and 1 to 1472 bytes for UDP.
- The IP addresses and subnet masks are set in the operating system table in the LPU. When the LPU is replaced, these items need be set again.

- Forcible termination of task -
If a task using the socket handler is terminated forcibly, the socket remains in registered state (except when the task has executed tcp_close( ) or udp_close( ) for the socket used by that task). That is, the socket status at the time of task forcible termination remains undeleted although the task terminated. The socket in such a state is called a “floating socket”.
As a floating socket cannot be used by other tasks, it needs to be released by resetting the module or by turning off the power to the module and then back on again.

(Page 5-27)

NOTICE

Because the udp_receive( ) function receives data in units of packets, reserve a buffer area of 1472 bytes.

(Page 6-4)

NOTICE

The only port setting that is supported by the ET.NET module (model LQE720) is auto-negotiation. Do not use 100-Mbps full-duplex setting for the port of the switching hub. Disregarding this rule may result in a failure of data communication when the load on the communication line builds up.

(Page 7-2)

NOTICE
● Static electricity could cause damage to the module. Before handling the module allow static charges on the human body to discharge. ● Before replacing the module, switch it off to avoid electrical shock hazards and also to prevent it from being damaged or malfunctioning.

WARRANTY AND SERVICING

Unless a special warranty contract has been arranged, the following warranty is applicable to this product.

1. Warranty period and scope
   Warranty period
   The warranty period for this product is for one year after the product has been delivered to the specified delivery site.

   Scope
   If a malfunction should occur during the above warranty period while using this product under normal product specification conditions as described in this manual, please deliver the malfunctioning part of the product to the dealer or Hitachi Engineering & Services Co., Ltd. The malfunctioning part will be replaced or repaired free of charge. If the malfunctioning is shipped, however, the shipment charge and packaging expenses must be paid for by the customer.

   This warranty is not applicable if any of the following are true.

   - The malfunction was caused by handling or use of the product in a manner not specified in the product specifications.
   - The malfunction was caused by a unit other than that which was delivered.
   - The malfunction was caused by modifications or repairs made by a vendor other than the vendor that delivered the unit.
   - The malfunction was caused by a relay or other consumable which has passed the end of its service life.
   - The malfunction was caused by a disaster, natural or otherwise, for which the vendor is not responsible.

   The warranty mentioned here means the warranty for the individual product that is delivered. Therefore, we cannot be held responsible for any losses or lost profits that result from the operation of this product or from malfunctions of this product. This warranty is valid only in Japan and is not transferable.

2. Range of services
   The price of the delivered product does not include on-site servicing fees by engineers. Extra fees will be charged for the following:

   - Instruction for installation and adjustments, and witnessing trial operations.
   - Inspections, maintenance and adjustments.
   - Technical instruction, technical training and training schools.
   - Examinations and repairs after the warranty period is concluded.
   - Even if the warranty is valid, examination of malfunctions that are caused by reasons outside the above warranty scope.

This manual provides information for the following hardware and program product:

<Hardware product>
ET.NET (LQE720)

<Program product>
S-7895-29, S10V ET.NET SYSTEM, 02-03

Change Record (for SVE-1-128(B)): S10V ET.NET SYSTEM, 02-02

Description of added changes	Chapter/Section/Subsection
Replacing or adding on the module is newly added.	7.1.1

Change Record (for SVE-1-128(C)): S10V ET.NET SYSTEM, 02-03

Description of added changes	Chapter/Section/Subsection
Comparison, printing, and CSV-output of IP address setup information are newly added to the ET.NET system's functionality.	4.1.2
Offline mode is newly added to the ET.NET system's functionality.	4.1.2
A procedure for using the ET.NET system in offline mode is added.	4.3.1
A command for IP address setting in offline mode is added.	4.3.3
A command for loading IP address setup information in from a file is added.	4.3.10
A command for saving IP address setup information in a file is added.	4.3.11
A command for printing IP address setup information is added.	4.3.12
A command for outputting IP address setup information in CSV-format is added.	4.3.13
A command for comparing IP address setup information is added.	4.3.14

In addition to the above changes, all the unclear descriptions and typographical errors found are also corrected without prior notice.

Revision record

Revision No.	Revision Record (revision details and reason for revision)	Month, Year	Remarks
A	First Edition	May 2005	
B	Replacing or adding on the module is newly added.	October 2008	
C	Offline functionality is newly added and safety precautions are revised.	February 2011	

PREFACE

Thank you for purchasing the ET.NET module, which is an option for use with the S10V.
This manual, named "USER'S MANUAL OPTION ET.NET," describes how to use the ET.NET module. For proper use of the ET.NET module, it is requested that you thoroughly read this manual.

The S10V products are available in two types: standard model and environmentally resistant model. The environmentally resistant model has thicker platings and coatings than those for the standard model.
The model number of the environmentally resistant model is marked by adding the suffix "-Z" to the model number of the standard model.

(Example) Standard model: LQE720
Environmentally resistant model: LQE720-Z

This manual is applicable to both the standard model and environmentally resistant models. Although the descriptions contained in this manual are based on the standard model, follow the instructions set forth in this manual for proper use of the product even if you use the environmentally resistant model.

The Ethernet communication instructions and socket handlers of the Ladder Chart and HI-FLOW Systems are usable only in conjunction with LPU and CMU modules of the module revisions listed below. If these features are used with LPU and CMU modules of any other revisions, they will have no effect.

Module name	Module model	Module revision
LPU (basic module)	LQP510	H or later
CMU	LQP520	F or later

<Trademarks>
- Microsoft® Windows® operating system, Microsoft® Windows® 2000 operating system, Microsoft® Windows® XP operating system are registered trademarks of Microsoft Corporation in the United States and/or other countries.
- Ethernet® is a registered trademark of Xerox Corp.

<Note for storage capacity calculations>
- Memory capacities and requirements, file sizes and storage requirements, etc. must be calculated according to the formula 2^n. The following examples show the results of such calculations by 2^n (to the right of the equals signs).
  1 KB (kilobyte) = 1,024 bytes
  1 MB (megabyte) = 1,048,576 bytes
  1 GB (gigabyte) = 1,073,741,824 bytes
- As for disk capacities, they must be calculated using the formula 10^n. Listed below are the results of calculating the above example capacities using 10^n in place of 2^n.
  1 KB (kilobyte) = 1,000 bytes
  1 MB (megabyte) = $1{,}000^2$ bytes
  1 GB (gigabyte) = $1{,}000^3$ bytes

CONTENTS

FIGURES

TABLES

1 SPECIFICATIONS

1.1 Use

The ET.NET module (model: LQE720) may be used only in conjunction with an S10V LPU module to provide communication under TCP/IP or the UDP/IP protocols by way of a local area network conforming to the IEEE802.3i specifications (10BASE-T) or IEEE802.3u specifications (100BASE-TX). This module may be neither installed in the CPU unit of S10mini systems nor intermixed with a model LQE520 module among the same series of similar modules installed.

1.2 Specifications

1.2.1 General specifications

Item	Specifications
Model	LQE720
Maximum number of installable mount bases (*)	2 per LPU (S10V only)
Mass	180 g

(*) For the kinds of mount base in which the module can be mounted, see “3.2 Mount Base.”

The model LQE720 ET.NET module may not be mounted on the same mount base on which a model LQE520 ET.NET module is mounted. The former must be mounted on a separate mount base.

1.2.2 Communication specifications

Item	Specifications
Transmission method	Serial (bit serial) transmission
Electrical interface	Conforming to IEEE802.3 (conforming to CSMA/CD standard)
Coding system	Manchester
Protocol	TCP/IP or UDP/IP
Maximum number of connectable units	n per hub (The value of n depends on the hub.)
Maximum number of stations	1024 per network
Communication cable	Twisted-pair cable: Up to 100 m per segment (category 5 or higher)
Data transmission rate	10 Mbps, 100 Mbps

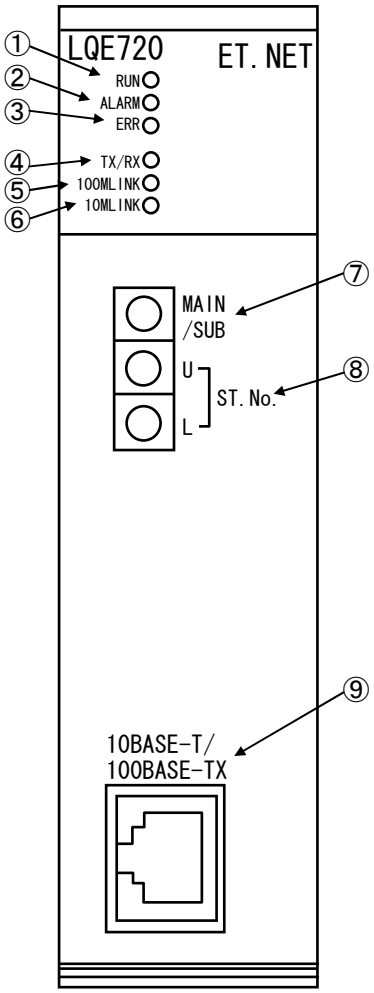
NOTICE

- If the software supplied by Hitachi is modified for use, Hitachi cannot be responsible for accidents or losses resulting from such modification.
- Hitachi cannot be responsible for reliability if you use software other than supplied from Hitachi.
- Back up files on a daily basis. You might lose the contents of files due, for instance, to a file unit failure, power failure during a file access, or operating error. To provide against such contingencies, back up files according to an appropriate plan.
- Before scrapping the product, ask a professional waste disposal dealer in charge of scrapping work.
- Do not use a transceiver, cellular phone, or similar device near the module because module malfunction or system failure may occur due to noise.
- The contents of the memory may become damaged due, for instance, to a module failure. Be sure to make a backup of important data.
- Before constructing a system, creating a program, or performing a similar procedure, thoroughly read this manual to become familiar with the contained instructions and precautions. If you perform any incorrect procedure, the system may malfunction.
- Store this manual at a predetermined place where it can readily be referred to whenever it is needed.
- If you have any doubt or question about the contents of this manual, contact your local source.
- Hitachi cannot be responsible for accidents or losses resulting from a customer's misuse.
- If an emergency stop circuit, interlock circuit, or similar circuit is to be formulated, it must be positioned external to this module. If you do not observe this precaution, equipment damage or accident may occur when this module becomes defective.

This Page Intentionally Left Blank

2 NAMES AND FUNCTIONS OF EACH PART

2.1 Names and Functions of Each Part



No.	Name	Function
①	RUN LED	Glows when the module is up and running normally.
②	ALARM LED	Glows when a setting error, such as an "IP address not set yet" condition, is detected.
③	ERR LED	Glows when a hardware fault is detected.
④	TX/RX LED	Glows when a data transmission or reception is in process.
⑤	100MLINK LED	Glows when the transmission line is connected for communication at 100 Mbps.
⑥	10MLINK LED	Glows when the transmission line is connected for communication at 10 Mbps.
⑦	MAIN/SUB setting switch	Used to define the module as the main or submodule. The allowable set value is one of the following: Switch setting / Explanation 0: Indicates that it is defined as the main module. 1: Indicates that it is defined as the submodule. 2 to B: Prohibited C to F: Prohibited (used for maintenance purposes)
⑧	ST. No. setting switches	Used to set an IP address. The value that can be set is one of the following: Switch setting (U / L) / Explanation U 0, L 0: Used when the module needs to be run with the IP address already set. U 0 to F, L 1 to E: Used when the module needs to be run by setting an IP address via the network. U F, L F: Used when the module needs to be connected with the Tool (i.e., a Windows PC used either as a maintenance PC, development PC, or a monitoring PC), and run with the fixed IP address ("192.192.192.1"). * For more information on how to use this module, see "4 OPERATION."
⑨	RJ45 connector	Used for data communication through the 10BASE-T/ 100BASE-TX interface.

NOTICE

Switch off the power supply before operating the MAIN/SUB and the ST. No. setting switch. If you operate while the power supply is applied, it may result in a malfunction.

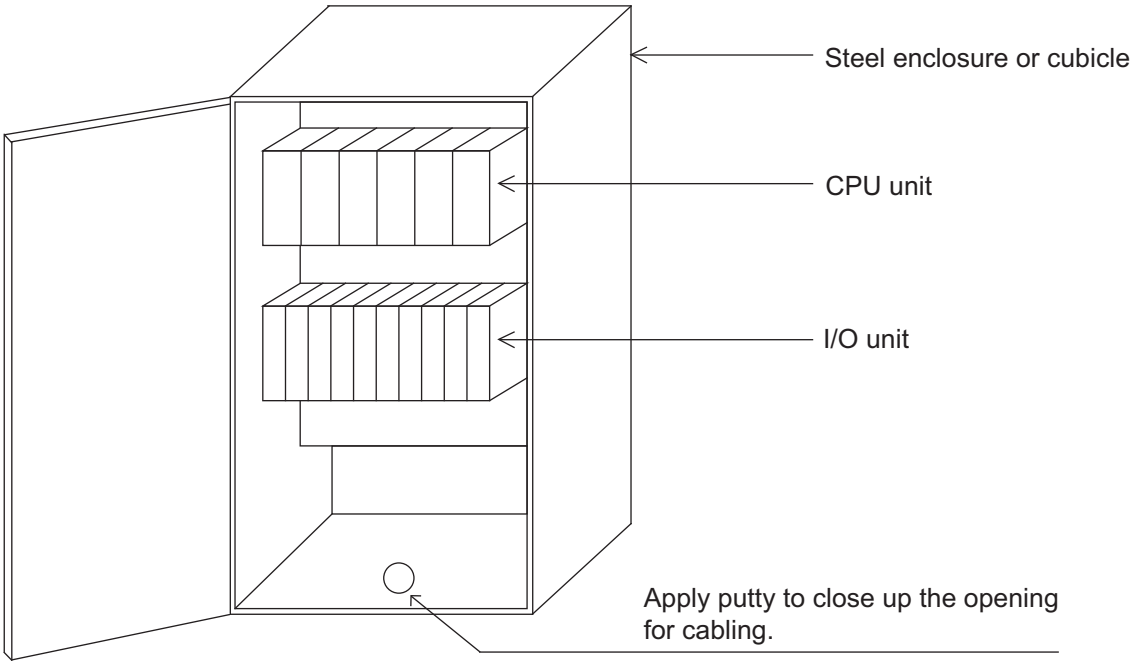
3 MOUNTING AND WIRING

3.1 Precautions for Using PCs

Hitachi's programmable controllers or PCs are a product of application of electronic circuit and processor technologies. The use of the product therefore requires special attention to be given to the following:

(1) The conditions to be met in system construction, such as maximum rated values, operating voltage ranges, heat dissipation characteristics, and mounting conditions, must all be within the warranty coverage stated in this manual. The manufacturer will not be held responsible for any damage that may be caused to the product and/or any physical injury that may be incurred as a result of using the product with conditions outside the warranty coverage.
In addition to the above requirement, fail-safe measures should also be provided in any user system by taking the expected failure rate and failure mode of the product into consideration. This is the case even when the product is used with every condition within the warranty coverage. The purpose of such fail-safe measures is to prevent the user system from suffering physical injuries, fire accidents, and/or other enlarged damages, due to the operation of the product.

(2) None of the PCs supplied to our customers is fireproof, dust-proof, and waterproof. So, please install your PCs in dust-proof and waterproof steel enclosures or cubicles as shown below.



CAUTION

- At installation sites where there is a risk of a water leak, be sure to install the programmable controller in a drip-proof cubicle and use it. Disregarding this rule may result in failure of the product.
- Do not touch any of the modules in the programmable controller when they are in an energized state. Touching any of the modules in an energized state may lead to a discharge of static electricity from your body to the module, resulting in malfunction or breakage of the module. If you have no choice but to touch a module, be sure to discharge the static electricity by touching the metal frame of the cubicle and then touch the module. This is also true when you perform any of the following actions on a module in its non-energized state: 1) setting a switch on the module; 2) connecting or disconnecting the cable from the module; or 3) inserting or removing the connector from the module.

3.2 Mount Base

The ET.NET module is mounted in the mount base for use. The table below lists the kinds of mount base in which the ET.NET module can be mounted.

Series	Name	Model
S10V	4-slot LPU mount base	HSC-1540
	8-slot LPU mount base	HSC-1580

3.3 Mounting the Module

Mount the option module in option slots (slot number 0 through 7) on the mount base as shown below.

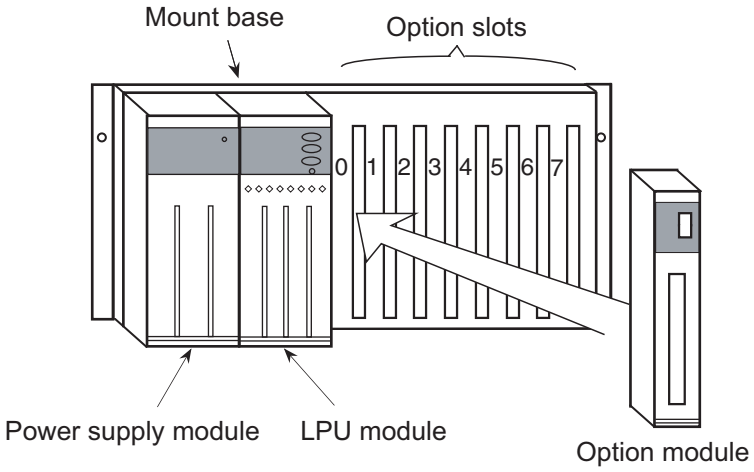


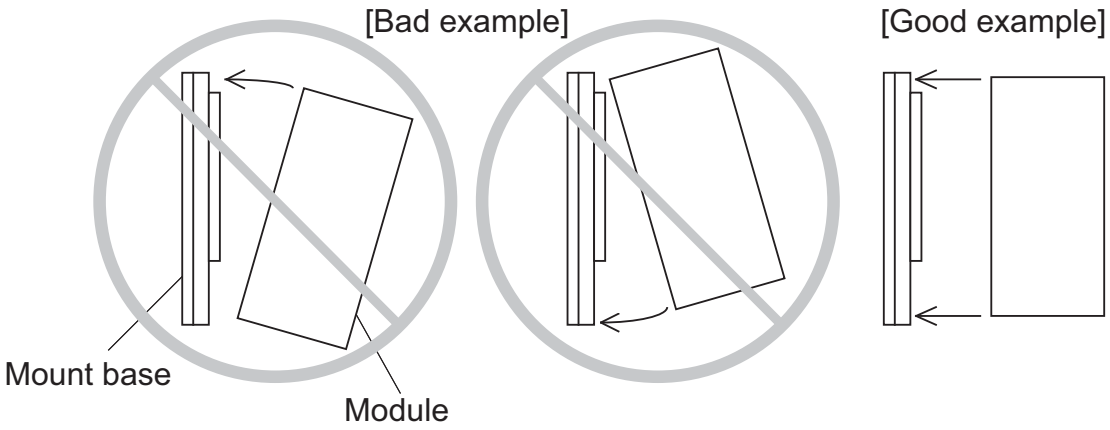
Figure 3-1 Mounting the Option Module

WARNING

- If the module emits smoke or foreign odor, immediately switch off the power supply and investigate the problem cause.
- Do not perform any installation, wiring, handling, or internal modification procedures other than stated in this manual. In no event will Hitachi be responsible for personal injury or death or any damage to Hitachi's product or peripheral equipment arising out of the use of such an unauthorized procedure.
- While the power is applied, never touch a terminal strip or connector pin. If you touch a terminal strip or connector pin while the power is applied, you may receive an electric shock.

CAUTION

- Dust or other foreign matter might accumulate on the connector, resulting in poor contact. Immediately after the module is unpacked, perform the mounting and wiring procedures.
- To prevent the module from being damaged, observe the following precautions when you mount or demount the module:
  - Before mounting the module to the mount base connector, check that the connector pins are properly aligned and not bent, broken, or soiled with dirt or the like.
  - Ensure that the module is parallel to the mount base vertical surface as shown below when mounting. If you connect a module to or disconnect it from its connector while it is tilted, the connector pins may become damaged.
  - If the mount base is positioned overhead due to the employed enclosure structure, use a stepladder or the like and mount the module squarely. If you mount the module obliquely, the connector may become damaged.



CAUTION

- Observe the installation procedure stated in the manual.
  If the module is improperly installed, it may drop, become defective, or malfunction.
- Do not allow wire cuttings or other foreign matter to enter the module.
  The entry of foreign matter in the module may result in a fire or cause the module to become defective or malfunction.
- Static electricity may damage the module. Before starting the work, discharge all electrostatic charge from your body.
- Properly tighten the screws. If they are inadequately tightened, malfunction, smoke emission, or combustion may occur.

NOTICE

Do not disassemble or modify the module. Failure to observe this precaution may result in a fire or cause the module to become defective or malfunction.

3.4 Ground Wiring

The ET.NET module requires no ground wiring.

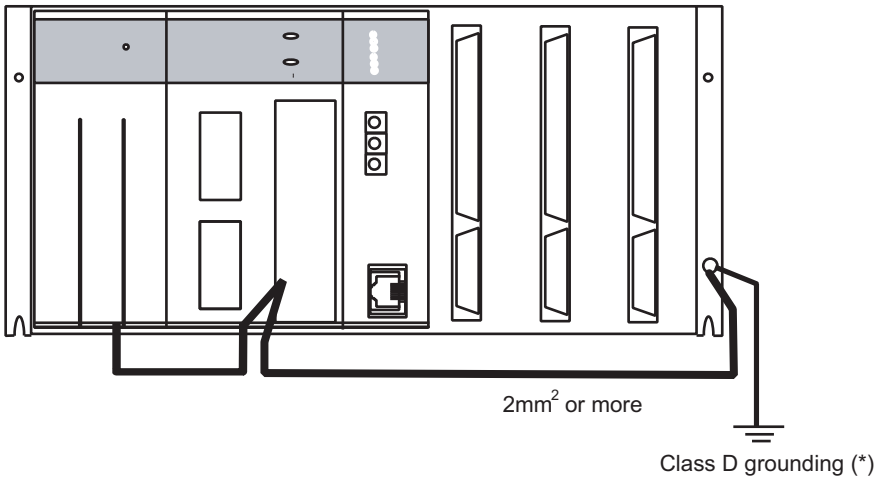


Figure 3-2 Ground Wiring

(*) Class D grounding is defined in the Technical Standard for Electrical Facilities of Japan. This standard states that the grounding resistance must be 100 ohms or less for equipment operating on 300 VAC or less, and 500 ohms or less for devices that shut down automatically within 0.5 seconds when shorting occurs in low tension lines.

3.5 Communication Cable Wiring

(1) 10BASE-T and 100BASE-TX communication cable wiring

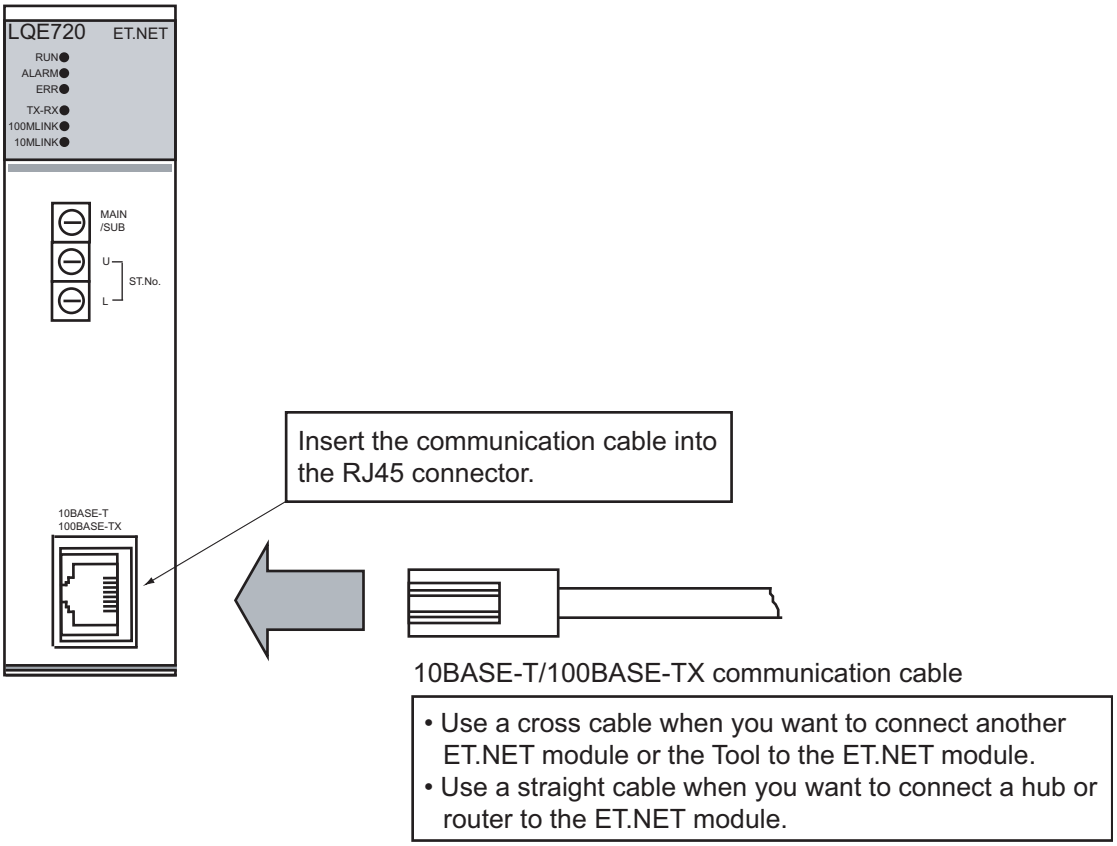


Figure 3-3 10BASE-T and 100BASE-TX Communication Cable Wiring

NOTICE
● This hardware unit may malfunction if it is connected poorly or has a broken line. After connecting the connector, check whether the locking. ● Do not touch the connector during power on. Otherwise, the system may malfunction due to static electricity, etc.

4 OPERATION

4.1 Startup Procedure

4.1.1 ET.NET system startup procedure

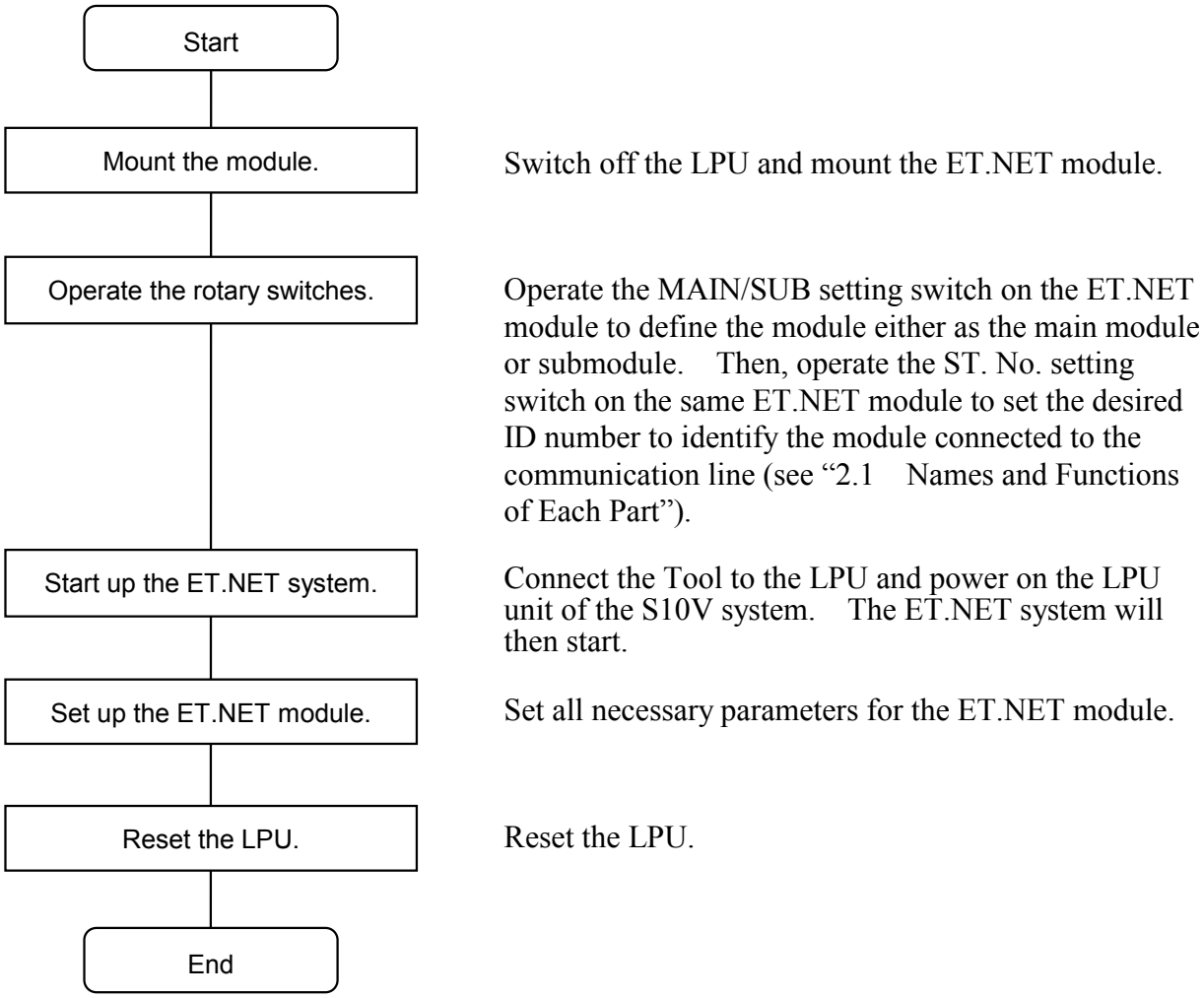


Figure 4-1 Startup Procedure

4.1.2 ET.NET system functions organizations

(1) Organization when a model LQE720 module is installed (online mode)

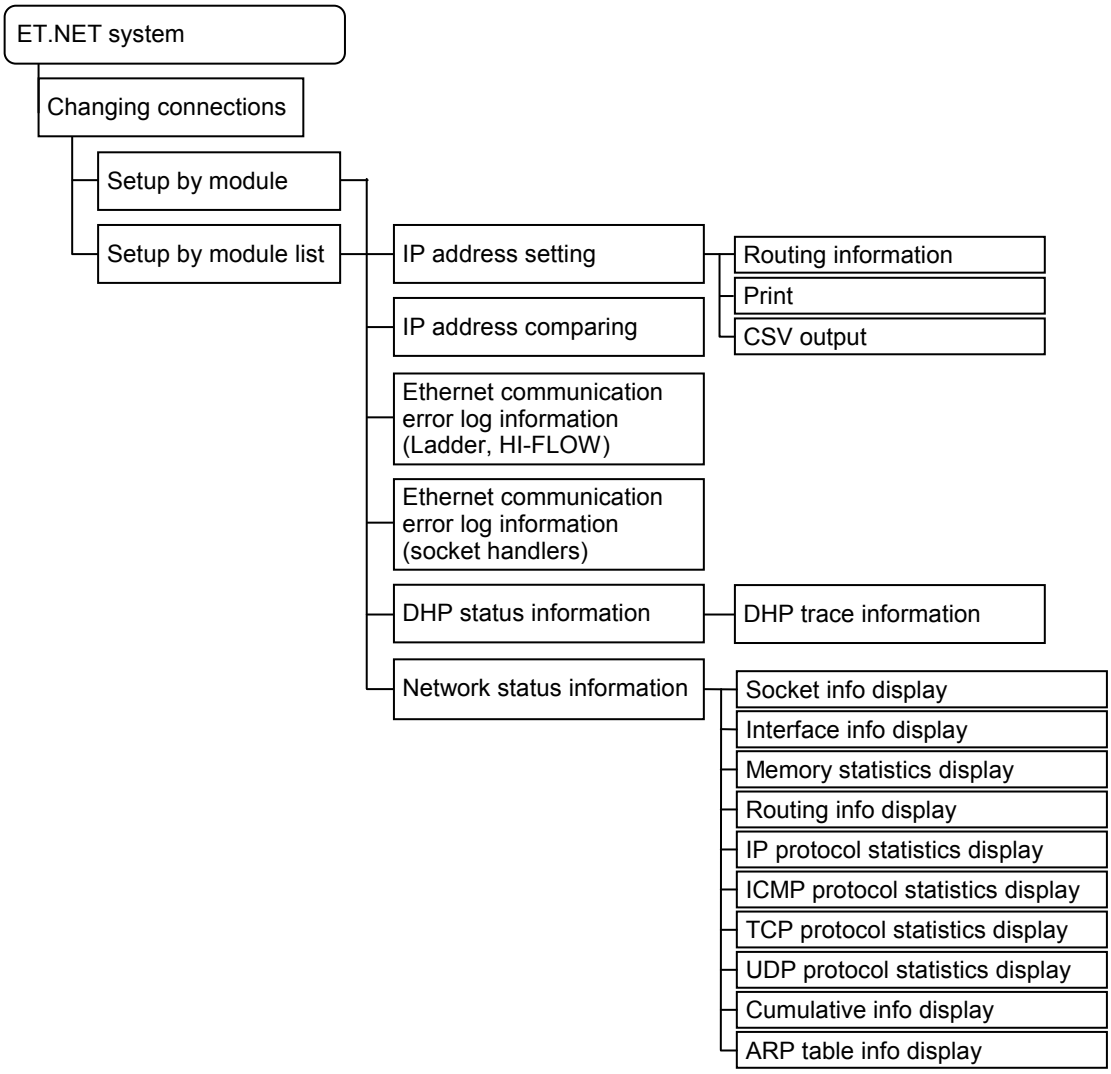


Figure 4-2 ET.NET System Functions Organization (When a Model LQE720 Module is Installed)

(2) Organization when a model LQE520 module is installed (online mode)

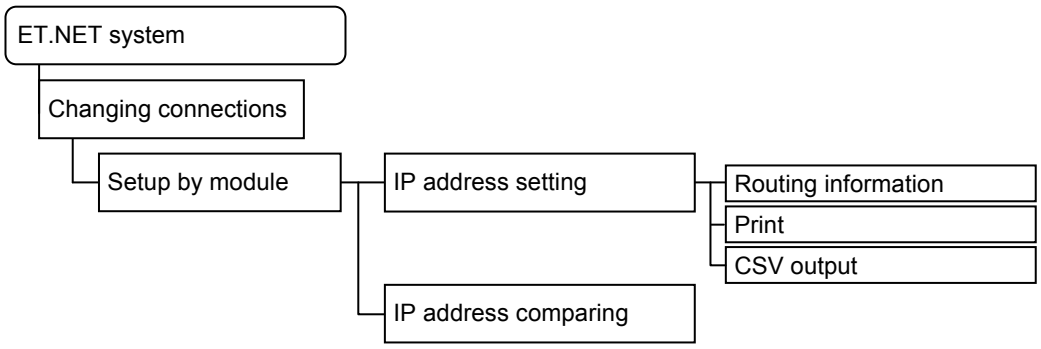


Figure 4-3 ET.NET System Functions Organization (When a Model LQE520 is Installed)

(3) Offline mode

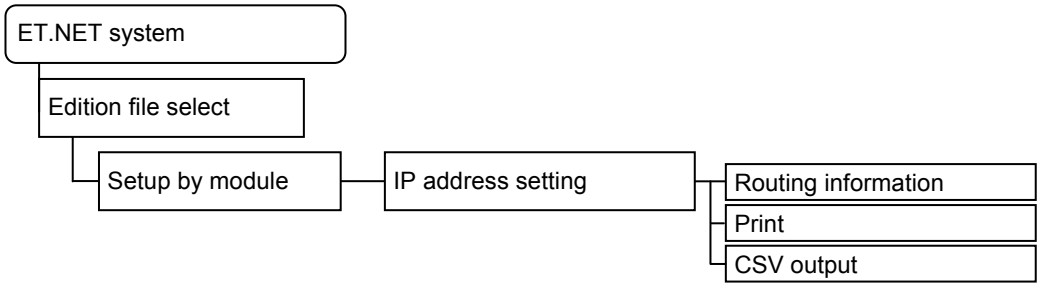


Figure 4-4 ET.NET System Functions Organization (Offline mode)

4.2 Installing and Starting Up the System

4.2.1 Installing

First check that the correct CD is on hand. The S10V ET.NET system runs on the Microsoft® Windows® 2000 operating system and Microsoft® Windows® XP operating system.
To install the S10V ET.NET system, you must execute the setup program that is stored in the S10V ET.NET system DISK1 folder on the CD.
Double-click [setup.exe] that is stored in the DISK1 folder on the S10V ET.NET system CD.
Since no window opens upon completion of installation, attach a shortcut to the desktop as needed.

- The S10V BASE SYSTEM is required for operating the S10V ET.NET system. If it is not installed, you cannot install the S10V ET.NET system.
- Before installing the S10V ET.NET system, be sure to exit all the currently open Windows®-based programs. Do not forget to exit anti-virus software and other memory-resident programs. If you install the ET.NET system without exiting such programs, an error may occur during installation. If such an error occurs, first uninstall the S10V ET.NET system as directed in "4.2.2 Uninstalling," exit all the Windows®-based programs, and then install the S10V ET.NET system again.

4.2.2 Uninstalling

From the [Start] button, select [Settings] and then click [Control]. When the Control Panel opens, double-click [Add/Remove Programs], select [S10V ET.NET SYSTEM] from the [Install/Uninstall] tab, and then click the [Remove] button. When the [Confirm File Deletion] window opens, click the [Yes] button.

NOTICE

If Windows® opens a window during the uninstall process to display the question "Remove Shared File?," click the [No] button to retain shared files.

4.2.3 Starting up the system

To start up the ET.NET system, perform the following procedure:

(1) Select [Program] - [Hitachi S10V] - [S10V ET.NET SYSTEM] - [S10V ET.NET SYSTEM] from the Start button. Then, the ET.NET system will come into operation and display the following [[S10V] ET.NET] window:

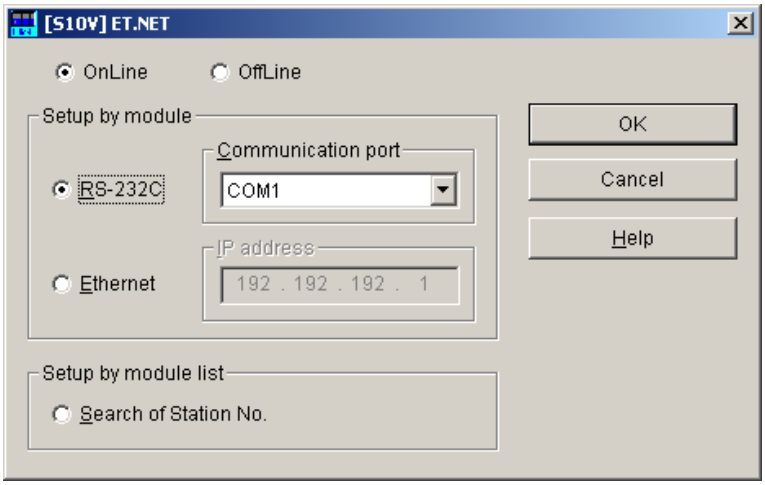


Figure 4-5 [[S10V] ET.NET] Window

4.2.4 Changing connections

Function: Set the communication type between the PCs and the PC.
Operation: The procedure is shown below.

① Choose in the ET.NET system's [[S10V] ET.NET] window the desired type of connection that you want to establish between your personal computer (tool) and the programmable controller.

- If you want to use an RS-232C connection:
  Check the [RS-232C] radio button and select the [Communication port] you want to use from the pulldown menu. The communication ports available are "COM1" through "COM4". The default port is "COM1".

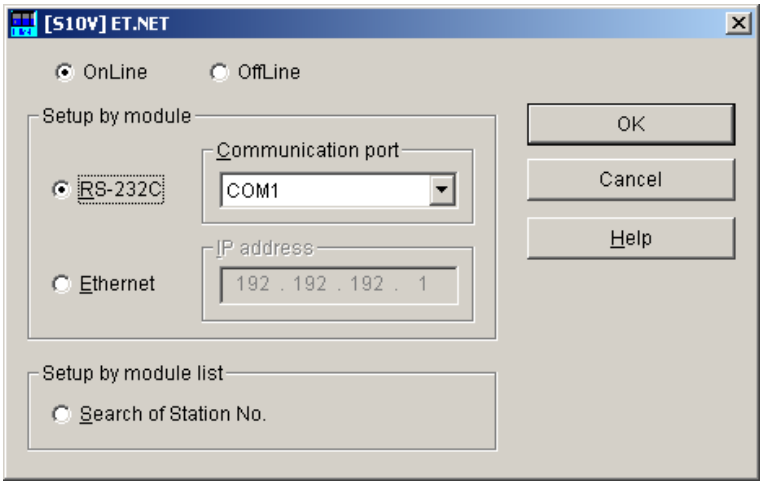


Figure 4-6 An [[S10V] ET.NET] Window in which [RS-232C] Is Selected

- If you want to use an Ethernet connection:
  Check the [Ethernet] radio button and enter the IP address of the PCs with which you want to establish a connection.

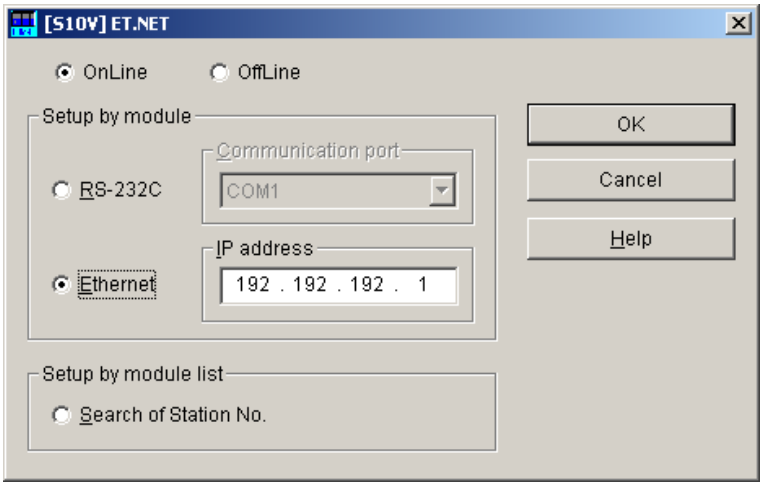


Figure 4-7 An [[S10V] ET.NET] Window in which [Ethernet] Is Selected

If you are not sure about the IP address of the PCs, check the [Search of Station No.] radio button in place of [Ethernet].

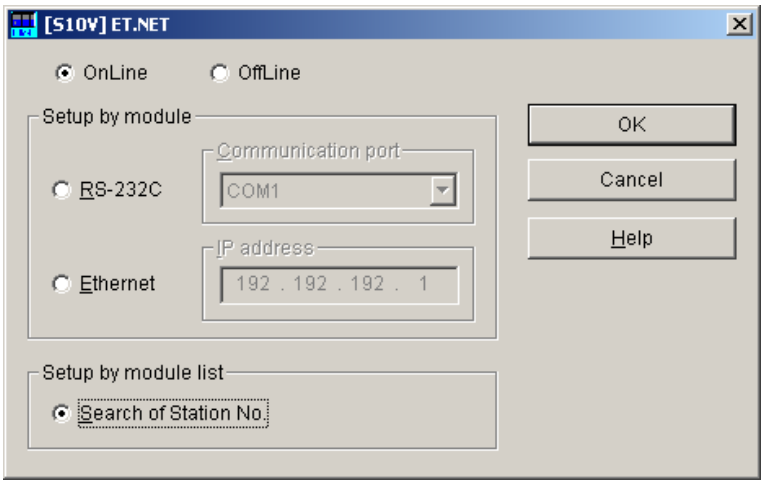


Figure 4-8 An [[S10V] ET.NET] Window in which [Search of Station No.] Is Selected

② When the above step is completed, click on the OK button. Then, if you have checked the [RS-232C] or [Ethernet] radio button in the above step, the [[Online] Setup by module] window will appear on screen. If you have checked the [Search of Station No.] radio button instead, the [Setup by module list] window will appear. In either case, click the desired command button.

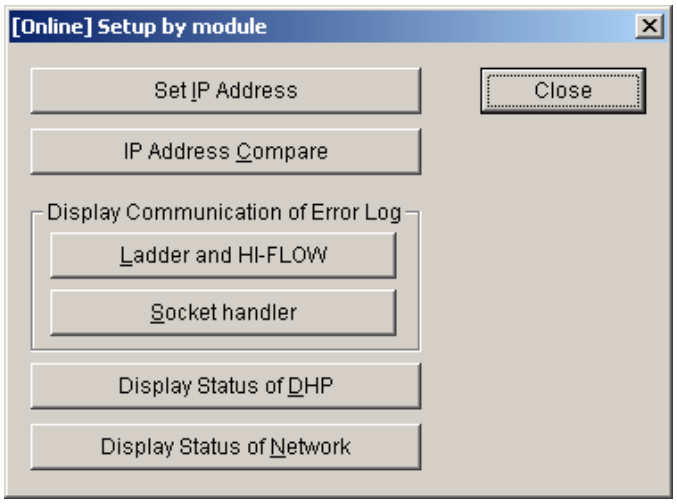


Figure 4-9 [[Online] Setup by Module] Window

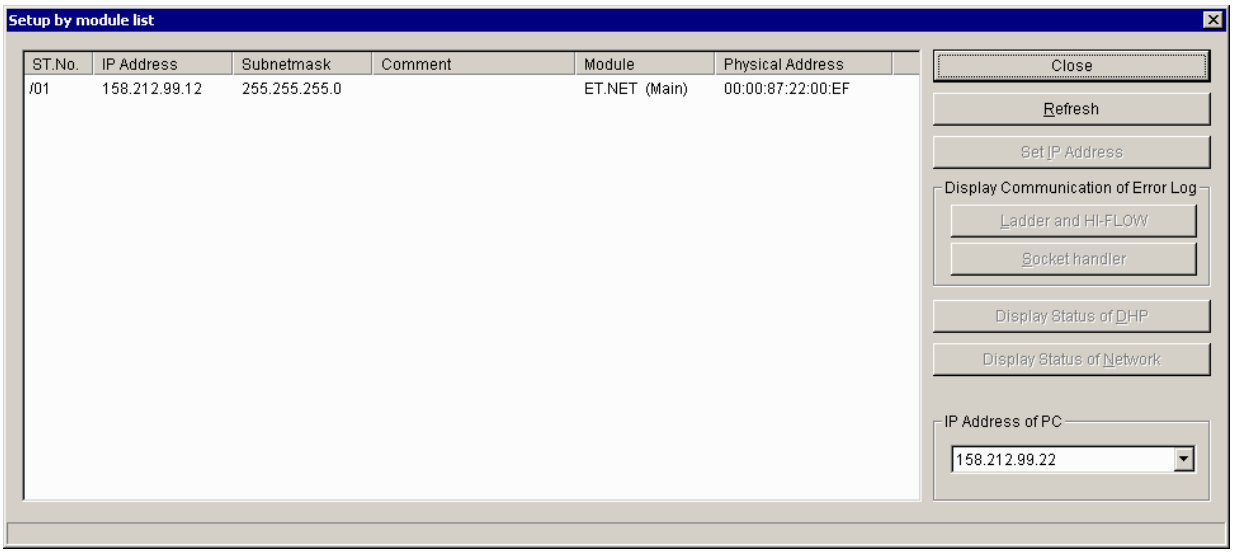


Figure 4-10 [Setup by Module List] Window

4.2.5 Closing the system

In the [[S10V] ET.NET] window (see Figure 4-5), click the [×] or [Cancel] button.

4.3 Commands

4.3.1 Setup by module

Function: Display the requested window. The functions available thereafter differ between online mode and offline mode.

Operation: The procedures provided for use in online and offline modes are described separately below.

(1) Procedure for use in online mode

① Check the [RS-232C] or [Ethernet] radio button in the [[S10V] ET.NET] window and click the OK button.

② The [[Online] Setup by module] window will then appear.

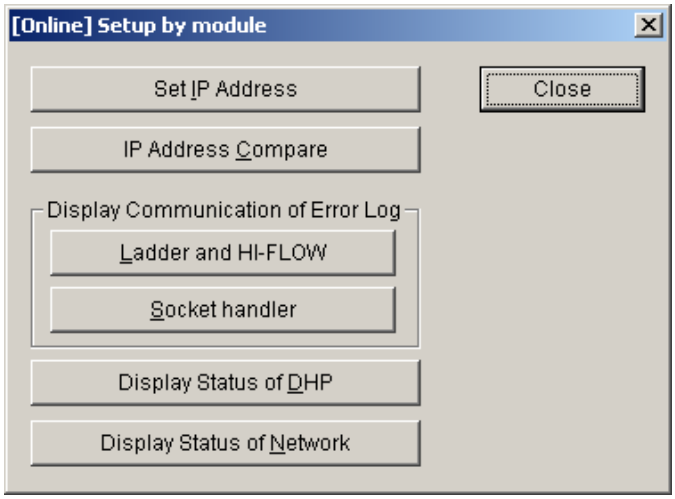


Figure 4-11 [[Online] Setup by module] Window

③ If you want to set IP address and other information for the ET.NET module, click the Set IP Address button. The [[Online] Set IP Address] window will then appear. For information on how to set an IP address, see "4.3.3 IP address setting."
If you want to compare IP address setup information between a selected file and the programmable controller, click the IP Address Compare button in the [[Online] Setup by module] window. The [Open] window will then appear. Select the desired IP address setup info file. For more information, see "4.3.14 IP address compare."

④ If you want to display the error log in which errors detected during Ethernet communications by the Ladder and HI-FLOW systems are recorded, click the [Ladder and HI-FLOW] button. The [Display Ethernet communication of Error Log (Ladder and HI-FLOW)] window will then appear. For more information, see "4.3.5 Ethernet communication error log information (Ladder, HI-FLOW)." If you want to display the error log in which errors detected during Ethernet communications by socket handlers are recorded, click the [Socket handler] button. The [Display Ethernet communication of Error Log (Socket Handler)] window will then appear. For more information, see "4.3.6 Ethernet communication error log information (socket handlers)."

⑤ If you want to display the current logging mode status of DHP or its trace information, click the [Display Status of DHP] button. The [Display Status of DHP] window will then appear. For more information, see "4.3.7 DHP status information."

⑥ If you want to display network status and cumulative information for the CMU and ET.NET modules, click the [Display Status of Network] button. The [Display Status of Network] window will then appear. For more information, see "4.3.9 Network status information."

⑦ If you want to exit the [[Online] Setup by module] window, click the [Close] button. The [[S10V] ET.NET] window will then become active again.

(2) Procedure for use in offline mode

① In the [[S10V] ET.NET] window displayed, select the [Offline] radio button. The [OK] button in the window will then change into the [Edition file select] button, as shown below.

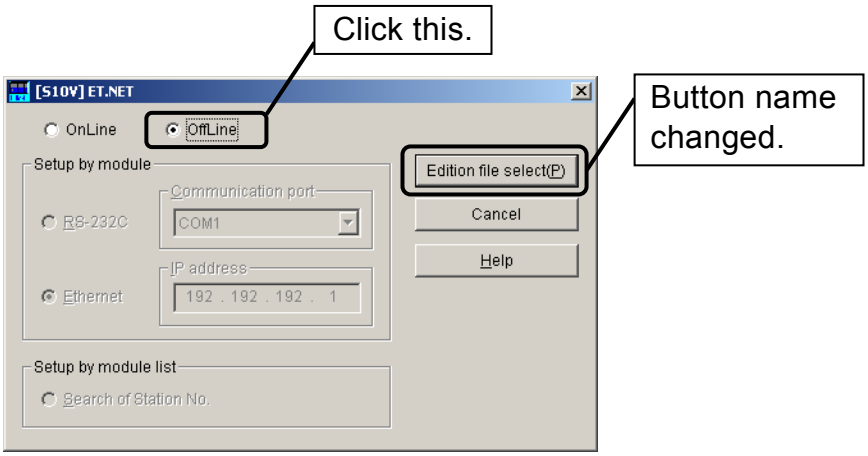


Figure 4-12 An [[S10V] ET.NET] Window in which the [Offline] Radio Button Is Clicked

② Click the [Edition file select] button. The [Open] window as shown below will then appear.

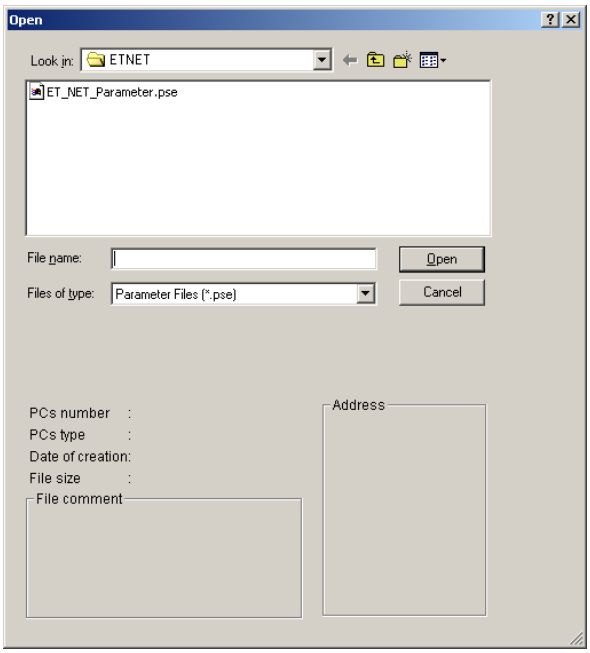


Figure 4-13 [Open] Window

If you want to edit an existing ET.NET parameter file, select it. Then, the [Open] window will close and the [[Offline] Setup by module] window will open.
If you want to create a new ET.NET parameter file, enter a unique file name that is not duplicated in the folder. Then, the [Open] window will close and the following confirmation dialog box will appear:

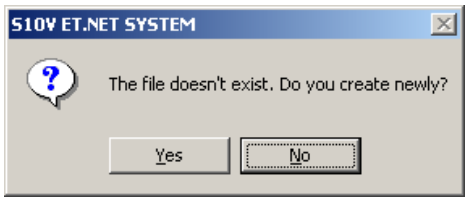


Figure 4-14 A Confirmation Dialog Box for Creating a File

If you click the Yes button, the confirmation dialog box will close and the [[Offline] Setup by module] window as shown below will open. If you click the No button instead, the confirmation dialog box will close and the [Open] window will become active again.

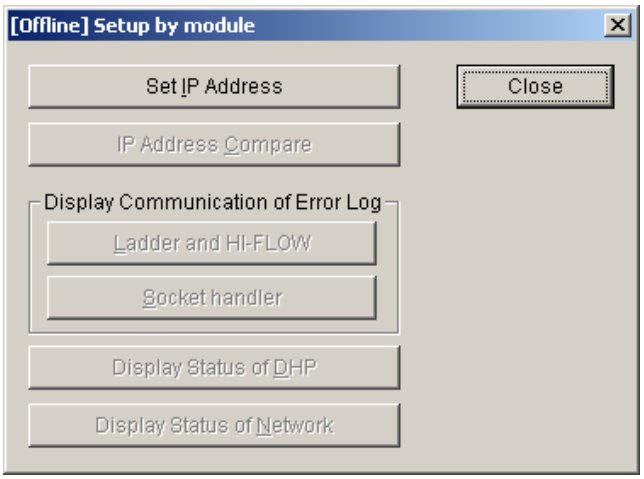


Figure 4-15 [[Offline] Setup by module] Window

③ If you want to set an IP address and other information, click the Set IP Address button in the [[Offline] Setup by module] window. The [[Offline] Set IP Address] window will then appear. For more information, see “4.3.3 IP address setting.”

④ If you want to exit the [[Offline] Setup by module] window, click the Close button in the window. The [[S10V] ET.NET] window will then become active again.

4.3.2 Setup by module list

Function: Display a list of the ET.NET module(s) connected to the same network to which the Tool is connected.

Operation: The procedure used is shown below.

① Check the [Search of Station No.] radio button in the Changing connections window and click the [OK] button.

② The [Setup by module list] window will then appear.

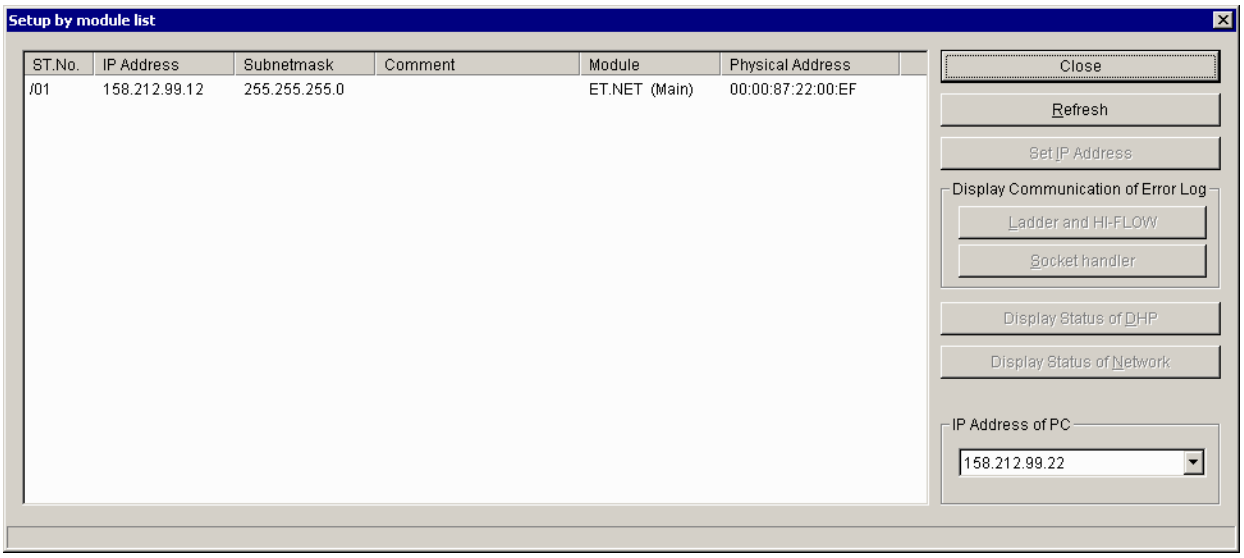


Figure 4-16 [Setup by module list] Window

This window presents a list of the ET.NET module(s) connected to the same network to which the Tool is connected. Select from this list the ET.NET module with which you want to establish a connection.

③ If the Tool (PC) has more than one LAN card installed in it, and you want to change the IP address with which a connection is to be established, then select the desired IP address in the [IP address of PC] box and click the [Refresh] button. The content of the list will then be renewed.

④ If you want to set an IP address and other information for the ET.NET module, click the [Set IP Address] button.

⑤ If you want to display the error log in which errors detected during Ethernet communications by the Ladder and HI-FLOW systems are recorded, click the Ladder and HI-FLOW button. If you want to display the error log in which errors detected during Ethernet communications by socket handlers are recorded, click the Socket handler button.

⑥ If you want to display the current logging mode status of DHP or its trace information, click the Display Status of DHP button.

⑦ If you want to display network status and cumulative information for the CMU and ET.NET modules, click the Display Status of Network button.

⑧ If you want to exit the [Setup by module list] window, click the Close button.

NOTICE
The ET.NET module may have its IP address displayed along with an asterisk (“*”) at its beginning. In this case, the asterisk indicates that, although the IP address has been changed, the ET.NET module is operating with its old IP address because the PCs has not been reset yet. To make the newly set IP address effective, be sure to reset the PCs.

4.3.3 IP address setting

Function: Set an IP address and other information for the ET.NET module. The destination of writing the specified IP address and other information differs between online mode and offline mode:

In online mode: It is the programmable controller with which your personal computer has a connection established.

In offline mode: It is the file that is selected via the [[S10V] ET.NET] window.

Operation: The procedure used is shown below.

(1) Procedure for use in online mode

① Click the Set IP Address button in the [[Online] Setup by module] or [Setup by module list] window.

② The [[Online] Set IP Address] window as shown below will then appear. Set the desired IP address and other information.

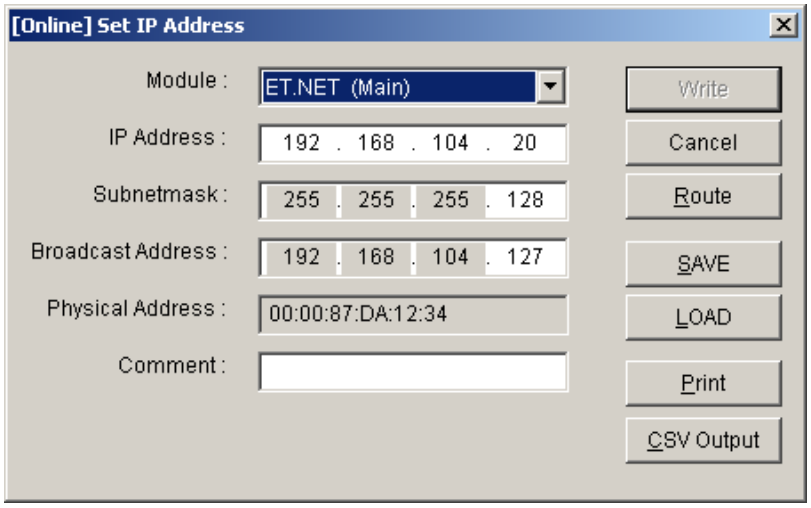


Figure 4-17 [[Online] Set IP Address] Window

- Module
  Select the ET.NET module you want to set up.

Possible choice	Remarks
ET.NET (main module)	Default
ET.NET (sub module)	

- IP Address/Subnetmask/Broadcast Address
  Set an IP address, subnet mask, and broadcast address for the ET.NET module. For details, see "6.3 System Definition Information."
- Physical Address
  The 48-bit address to which the ET.NET module is assigned is displayed in this box. If no ET.NET module is installed for this physical address, either of the values "00:00:00:00:00:00" and "FF:FF:FF:FF:FF:FF" is displayed instead.
- Comment
  A user comment of up to 16 single-byte alphanumeric characters and hyphens may be entered in this box, as necessary.

③ If you want to make entries into the routing table, click the Route button. The [[Online] Route] window will then appear. Enter remote-station (communication point) addresses and gateway addresses for all necessary routes.

④ When the above step is completed, click the Write button if you want to save all the settings and entries thus far made.
If not, click the Cancel button.

⑤ Click the Write button. The following reset confirmation message will then appear:

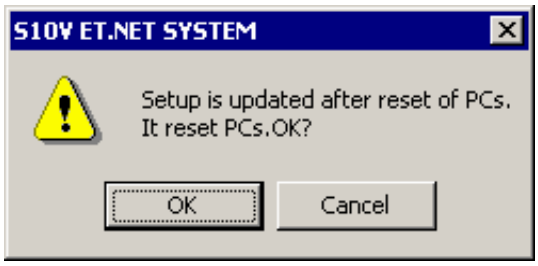
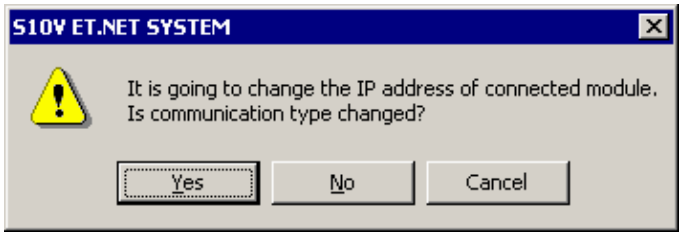


Figure 4-18 A Confirmation Message for Reset

To reset the programmable controller, click the OK button. Then, the newly set IP address and route information will become effective. In this case, it is effective only for the selected module, which is either the main module or the submodule, depending on the module's switch setting. In this procedure, the main module is shown as an example, so the new IP address and route information is written to the programmable controller only in behalf of the main module. If you also want to write IP address and routing information for the submodule to it, select "ET.NET (SUB)", specify the IP address and routing information for the submodule, and click the Write button.

NOTICE

Any attempt to change the IP address of the ET.NET module connected by Ethernet cable wiring will have the following message displayed on screen.



If you want to reset the programmable controller and re-establish a connection with it by using the newly set IP address, click the Yes button. If not, click the No button. If you want to abort the IP address setting process, click the Cancel button.

(2) Procedure for use in offline mode

① In the [[Offline] Setup by module] window displayed, click the Set IP Address button. The [[Offline] Set IP Address] window will then appear.

② Specify the desired IP address and other information (see the description of Step ② in "(1) Procedure for use in online mode" for examples, where the displayed physical address should be replaced with "00:00:00:00:00:00" in offline mode).

③ If you want to set up a routing table for the ET.NET module, click the Route button. The [[Offline] Route] window will then appear. Specify the network address or IP address of the remote station in each route along with the gateway address.

④ When all the necessary parameters have been entered, click the Write button if they really need to be set. Otherwise, click the Cancel button instead.

⑤ If the Write button is clicked, a confirmation message as shown below appears, asking if you really want to write the specified parameters to the file.

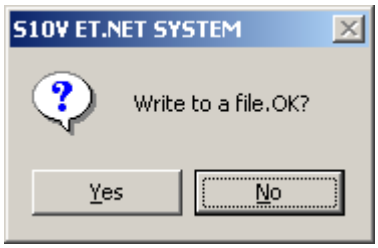


Figure 4-19 A Confirmation Message for Writing to the File

If you really want to, click the Yes button in the dialog box. The confirmation message dialog box will then close and the specified IP address and other information (for the main module or submodule) will be written to the file that has been selected via the [[S10V] ET.NET] window. When the writing is completed, a message to that effect will appear as shown below. Click the OK button.

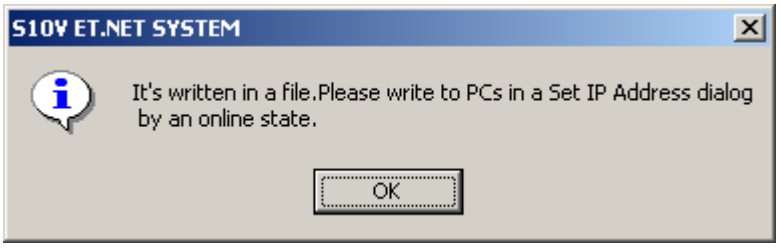


Figure 4-20 An End-of-Writing Message

If you do not want to, click the No button in the confirmation message dialog box. The dialog box will then close and the [[Offline] Set IP Address] window become active again.

You might click the Cancel button in the [[Offline] Set IP Address] window while the Set IP Address command is in an editing state(*). In this case, a confirmation message as shown below will appear, asking if you want to end the write operation without writing the parameters to the file.

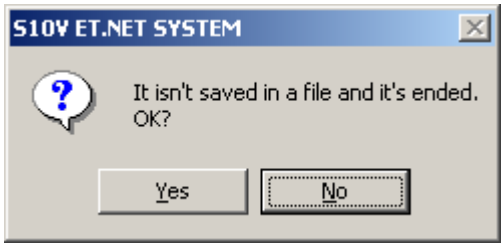


Figure 4-21 A Confirmation Message for Ending without Writing to the File

If you click the Yes button in the above dialog box, the result of the editing, including the routing information, will all be discarded and the [[Offline] Setup by module] window will become active again.
If you click the No button instead, the [[Offline] Set IP Address] window will become again.

(*) The fact that the Set IP Address command is in an editing state is indicated by an asterisk ("*") displayed to the right of the [[Offline] Set IP Address] window's title in the title bar.

If the Set IP Address command is not in an editing state, clicking the Cancel button will not present the confirmation message shown in Figure 4-21. It will simply cause the [[Offline] Setup by module] window to become active again.

4.3.4 Routing information

Function: Set routing information for the ET.NET module.

Operation: The procedure used is shown below. This procedure is the same in both online mode and offline mode.

① Click the Route button on the [[Online] Set IP Address] window.

② The [[Online] Route] window will then appear. Enter all necessary routing information in the table.

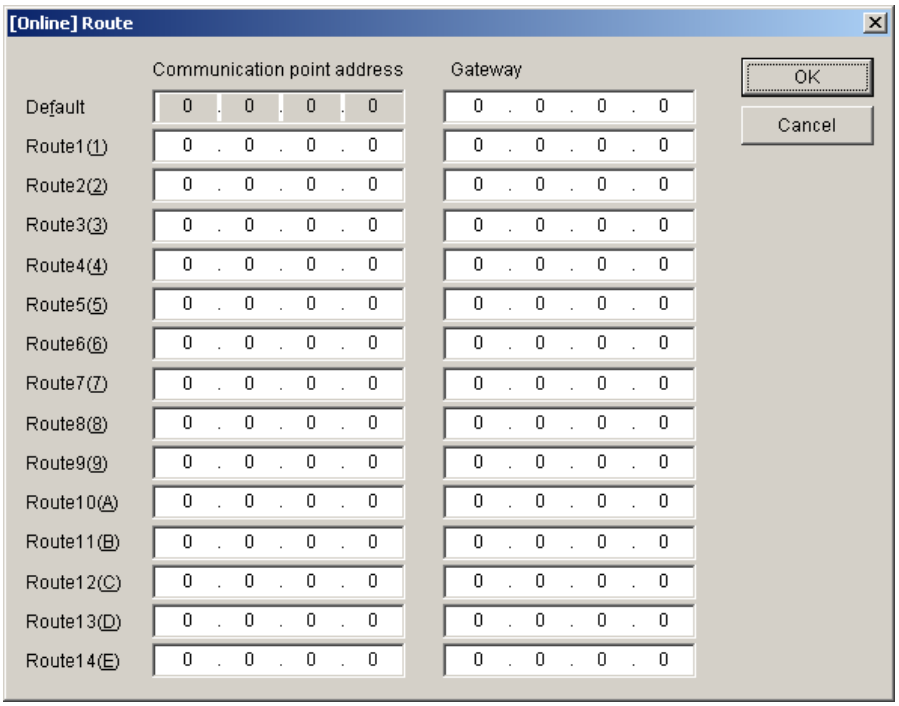


Figure 4-22 [[Online] Route] Window

- Communication point address
  Enter the network address or IP address of each remote station.
- Gateway IP address (If a communication point address is specified first for a route, the network address will be automatically displayed.)
  Enter the IP address of the gateway for the route.

③ When the above step is completed, click the OK button if you want to set the routing information entered.
If not, click the Cancel button.

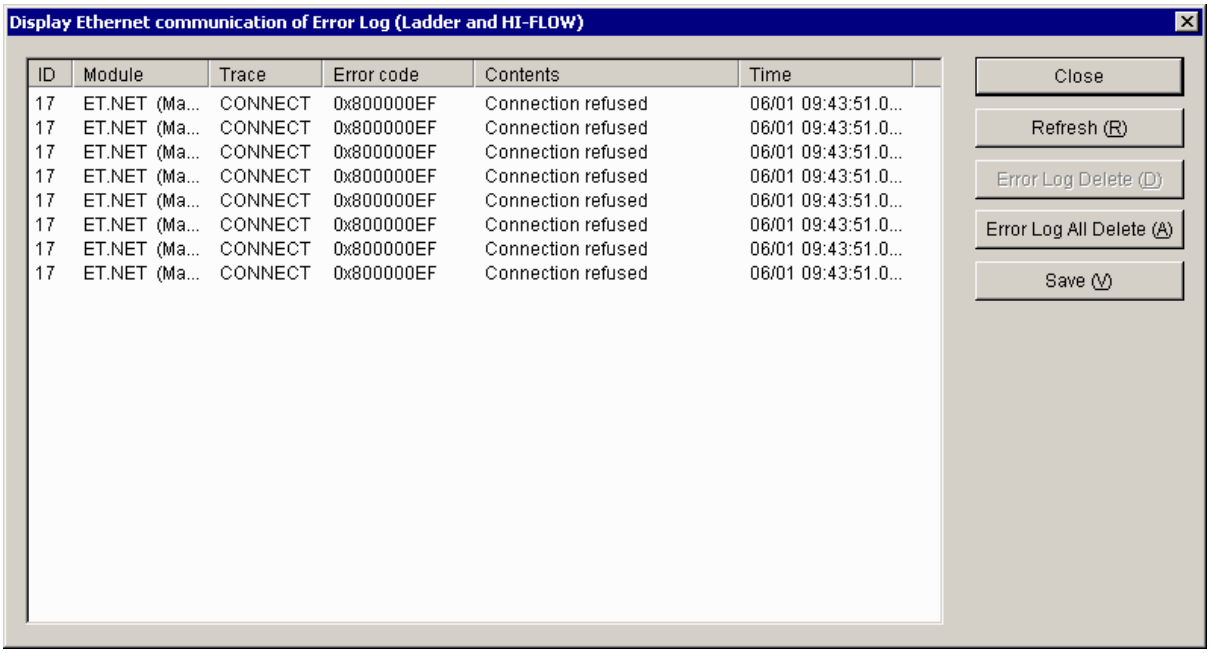
NOTICE
The routing information entered is not registered in the PCs or file until you click the Write button in the [Set IP Address] window. Thus, if you click the Cancel button in place of OK in that window, the routing information will not be registered in the PCs or file.

4.3.5 Ethernet communication error log information (Ladder, HI-FLOW)

Function: Display the error log in which errors detected during Ethernet communications by the Ladder and HI-FLOW systems are recorded.
Operation: The procedure used is shown below.

① Click the Ladder and HI-FLOW button in the [Setup by module] or the [Setup by module list] window.

② The [Display Ethernet communication of Error Log (Ladder and HI-FLOW)] window will then appear.



ID	Module	Trace	Error code	Contents	Time
17	ET.NET (Ma...	CONNECT	0x800000EF	Connection refused	06/01 09:43:51.0...
17	ET.NET (Ma...	CONNECT	0x800000EF	Connection refused	06/01 09:43:51.0...
17	ET.NET (Ma...	CONNECT	0x800000EF	Connection refused	06/01 09:43:51.0...
17	ET.NET (Ma...	CONNECT	0x800000EF	Connection refused	06/01 09:43:51.0...
17	ET.NET (Ma...	CONNECT	0x800000EF	Connection refused	06/01 09:43:51.0...
17	ET.NET (Ma...	CONNECT	0x800000EF	Connection refused	06/01 09:43:51.0...
17	ET.NET (Ma...	CONNECT	0x800000EF	Connection refused	06/01 09:43:51.0...
17	ET.NET (Ma...	CONNECT	0x800000EF	Connection refused	06/01 09:43:51.0...

Figure 4-23 [Display Ethernet communication of Error Log (Ladder and HI-FLOW)] Window

③ The content of the error log displayed is as follows:

Item	Content
ID	Table number of Ladder and HI-FLOW Ethernet communications management table
Module	Module name
Trace	Meaning of trace code in trace information
Error code	Error code for the error detected
Contents	Meaning of error code for the error detected
Time	Time the error occurred

④ If you want to display the latest error information on screen, click the Refresh button.

⑤ If you want to delete the error log information for a specified ID, click the Error Log Delete button. If you want to delete all the displayed error log information at once, click the Error Log All Delete button.

⑥ If you want to save the displayed error log information in a text file, click the Save button. When the [Save As] window appears, choose the folder in which to store the text file, and enter its file name.

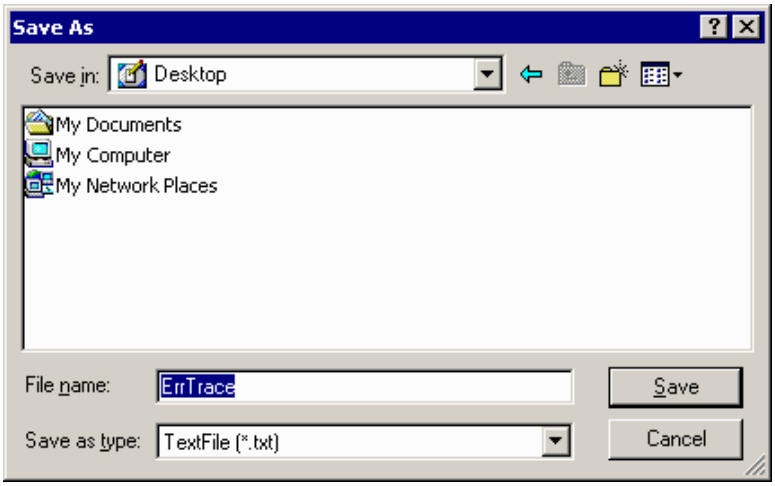


Figure 4-24 [Save As] Window (for saving Ethernet communication error log info [ladder, HI-FLOW])

Then, click the Save button. The error log information will then be saved in the text file.

⑦ If you want to exit the [Display Ethernet communication of Error Log (Ladder and HI-FLOW)] window, click the Close button.

4.3.6 Ethernet communication error log information (socket handlers)

Function: Display the error log in which errors detected during Ethernet communications by socket handlers are recorded.

Operation: The procedure used is shown below.

① Click the [Socket handler] button in the [Setup by module] or the [Setup by module list] window.

② The [Display Ethernet communication of Error Log (Socket handler)] window will then appear.

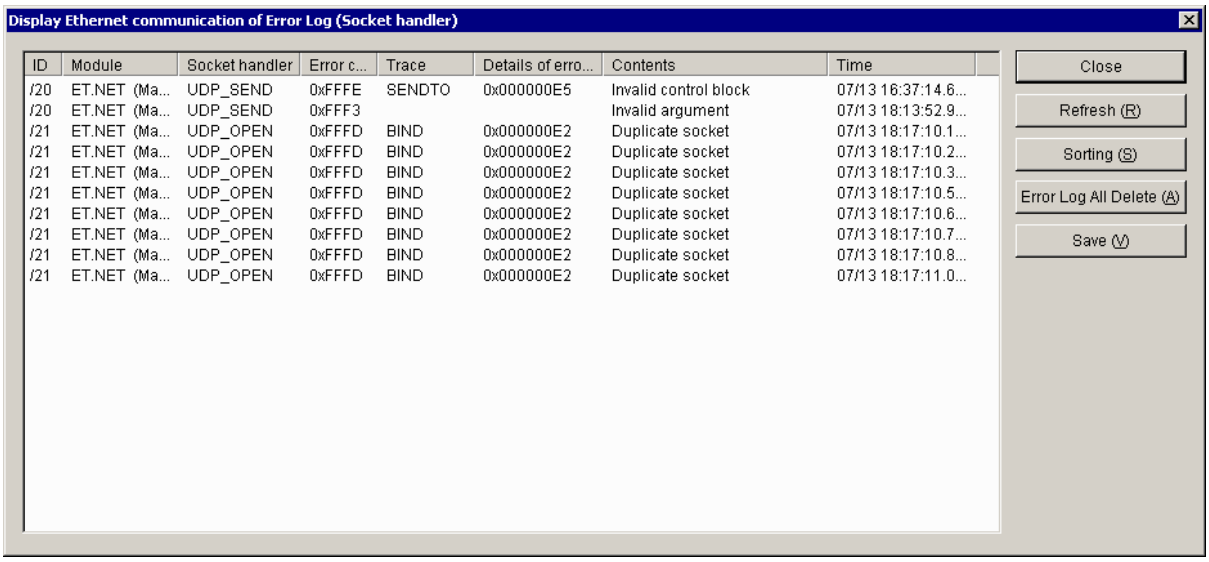


Figure 4-25 [Display Ethernet communication of Error Log (Socket handler)] Window

③ The content of the error log displayed is as follows:

Item	Content
ID	Socket ID of socket handler
Module	Module name
Socket handler	Name of socket handler
Error code	Error code from socket handler
Trace	Place at which the error was detected.
Details of error code	Details of error code when the error was detected.
Contents	Meaning of error code for the error detected
Time	Time the error occurred

④ If you want to display the latest error information on screen, click the Refresh button.

⑤ If you want to rearrange the displayed error log information in the order in which the errors occurred, click the Sorting button. Each time you click the Sorting button, the displayed information will be rearranged in ascending and descending orders alternately.

⑥ If you want to delete all the displayed error log information at once, click the Error Log All Delete button.

⑦ If you want to save the displayed error log information in a text file, click the Save button. When the [Save As] window appears, choose the folder in which to store the text file, and enter its file name.

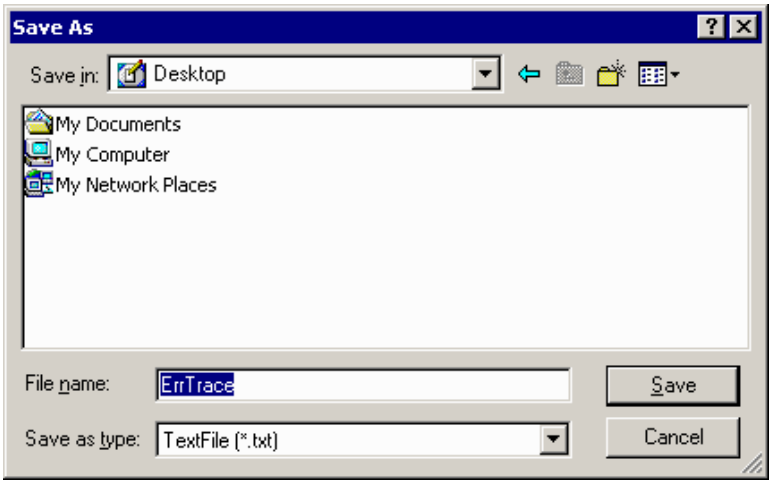


Figure 4-26 [Save As] Window (for saving Ethernet communication error log info [socket handlers])

Then, click the Save button. The error log information will then be saved in the text file.

⑧ If you want to exit the [Display Ethernet communication of Error Log (Socket handler)] window, click the Close button.

4.3.7 DHP status information

Function: Enable or disable the logging mode of DHP (debugging helper).
Operation: The procedure used is shown below.

① Click the Display Status of DHP button in the [Setup by module] or the [Setup by module list] window.

② The [Display Status of DHP] window will then appear.

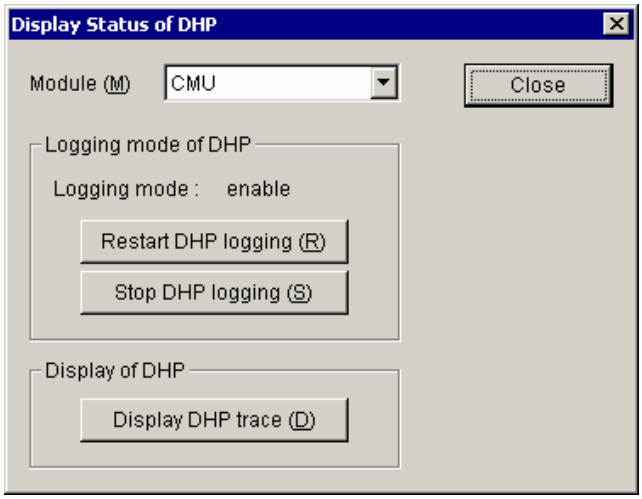


Figure 4-27 [Display Status of DHP] Window

③ The [Module] box in the window presents the names of the CMU and ET.NET modules installed in the PCs. Choose in the [Module] box the module for which you want to display DHP information or make a necessary setting.

④ The current logging mode status of DHP is displayed to the right of the label "Logging mode". To enable the logging mode, click the Restart DHP logging button. To disable it, click the Stop DHP logging button.

⑤ If you want to display DHP's trace information, click the Display DHP trace button.

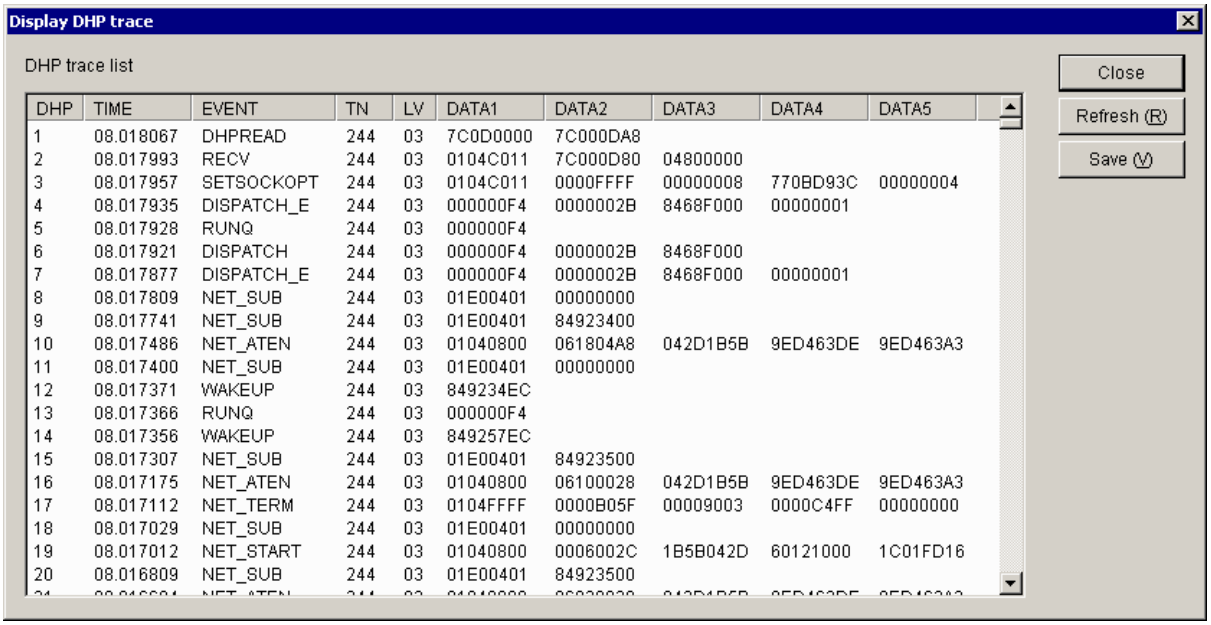
⑥ If you want to exit the [Display Status of DHP] window, click the Close button.

4.3.8 DHP trace information

Function: Display DHP's trace information.
Operation: The procedure used is shown below.

① Click the [Display DHP trace] button in the [Display Status of DHP] window.

② The [Display DHP trace] window will then appear. For details on the displayed information, see "7 MAINTENANCE."



DHP	TIME	EVENT	TN	LV	DATA1	DATA2	DATA3	DATA4	DATA5
1	08.018067	DHPREAD	244	03	7C0D0000	7C000DA8			
2	08.017993	RECV	244	03	0104C011	7C000D80	04800000		
3	08.017957	SETSOCKOPT	244	03	0104C011	0000FFFF	00000008	770BD93C	00000004
4	08.017935	DISPATCH_E	244	03	000000F4	0000002B	8468F000	00000001	
5	08.017928	RUNQ	244	03	000000F4				
6	08.017921	DISPATCH	244	03	000000F4	0000002B	8468F000		
7	08.017877	DISPATCH_E	244	03	000000F4	0000002B	8468F000	00000001	
8	08.017809	NET_SUB	244	03	01E00401	00000000			
9	08.017741	NET_SUB	244	03	01E00401	84923400			
10	08.017486	NET_ATEN	244	03	01040800	061804A8	042D1B5B	9ED463DE	9ED463A3
11	08.017400	NET_SUB	244	03	01E00401	00000000			
12	08.017371	WAKEUP	244	03	849234EC				
13	08.017366	RUNQ	244	03	000000F4				
14	08.017356	WAKEUP	244	03	849257EC				
15	08.017307	NET_SUB	244	03	01E00401	84923500			
16	08.017175	NET_ATEN	244	03	01040800	06100028	042D1B5B	9ED463DE	9ED463A3
17	08.017112	NET_TERM	244	03	0104FFFF	0000B05F	00009003	0000C4FF	00000000
18	08.017029	NET_SUB	244	03	01E00401	00000000			
19	08.017012	NET_START	244	03	01040800	0006002C	1B5B042D	60121000	1C01FD16
20	08.016809	NET_SUB	244	03	01E00401	84923500			

Figure 4-28 [Display DHP trace] Window

③ The content of the displayed list is as follows:

Item	Content
DHP	DHP trace number displayed
TIME	Time the tracing was made: tt.tttttt where tt is seconds and tttttt microseconds.
EVENT	Type of trace point
TN	Task number
LV	Priority level
DATA1 to DATA5	Each is a piece of trace data (output in hexadecimal format).

④ If you want to display the latest DHP trace information on screen, click the Refresh button.

⑤ If you want to save the displayed trace information in a text file, click the Save button. When the [Save As] window appears, choose the folder in which to store the text file, and enter its file name.

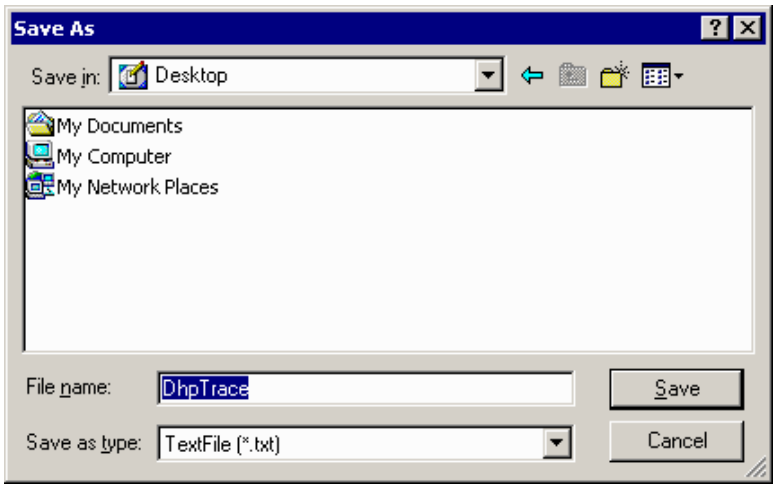


Figure 4-29 [Save As] Window (for saving DHP trace information)

Then, click the Save button. The trace information will then be saved in the text file.

⑥ If you want to exit the [Display DHP trace] window, click the Close button.

4.3.9 Network status information

Function: Display network status information for the CMU and ET.NET modules.
Operation: The procedure used is shown below.

① Click the Display Status of Network button in the [Setup by module] or the [Setup by module list] window.

② The [Display Status of Network] window will then appear.

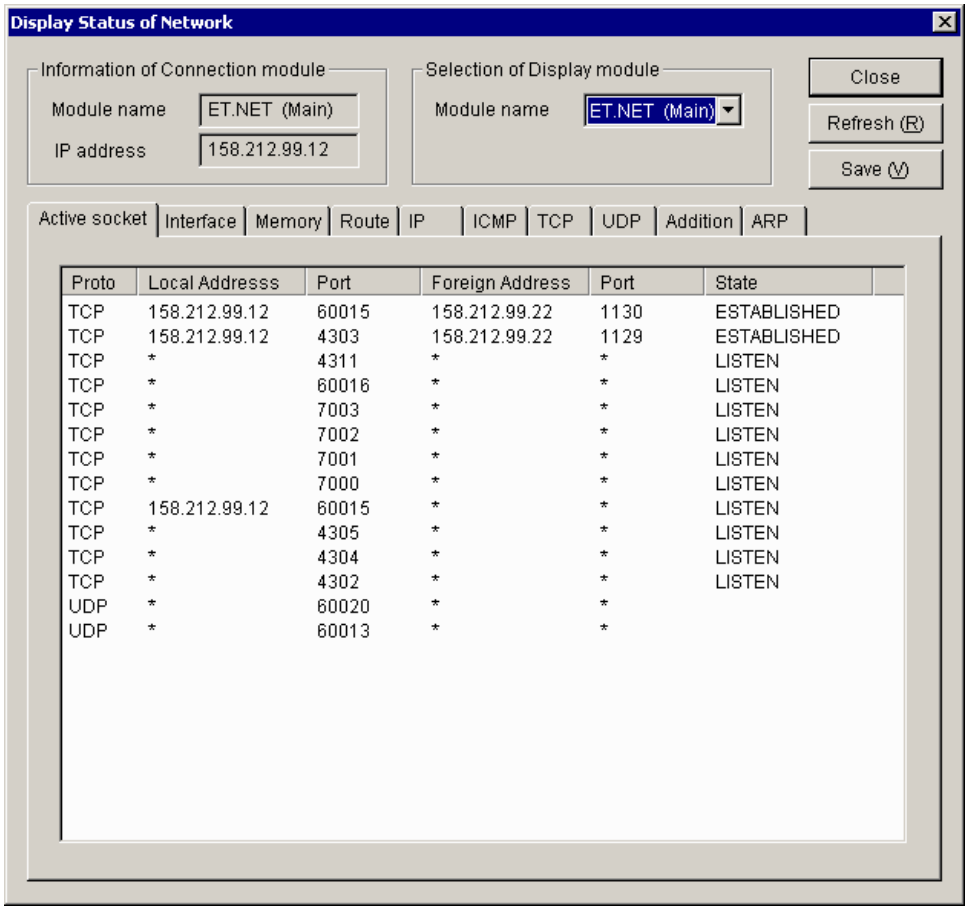


Figure 4-30 [Display Status of Network] Window

③ Choose in the [Selection of Display module/Module name] box the module for which you want to display network status information.

④ Click the desired tab to display the associated network status information. The types of network status information that can be displayed are as follows:

Item	Type of information displayed
Active socket	Socket information
Interface	Currently running network interfaces information
Memory	Send/receive buffer management information
Route	Routing information
IP	IP protocol statistics
ICMP	ICMP protocol statistics
TCP	TCP protocol statistics
UDP	UDP protocol statistics
Addition	Interface cumulative information
ARP	ARP table information

⑤ Click the Refresh button to display the selected set of network status information. For details on the displayed information, see "7.3.3 Meanings of network status information items."

⑥ If you want to save the displayed network status information in a text file, click the Save button. When the [Save As] window appears, choose the folder in which to store the text file, and enter its file name.

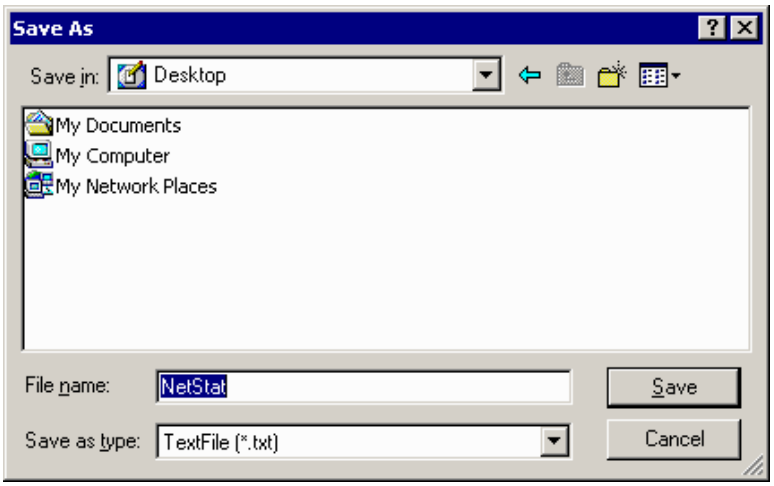


Figure 4-31 [Save As] Window (for saving network status information)

Then, click the Save button. The network status information will then be saved in the text file.

⑦ If you want to exit the [Display Status of Network] window, click the Close button.

4.3.10 Load IP address setup information in from file

Function: Loads the contents of a selected IP address setup information file into the [[Online] Set IP Address] window and subsequently into the [[Online] Route] window if requested. (This command is supported only by Ver-Rev 02-03 or later of the ET.NET system and can be used only in online mode.)

Operation: The procedure used is described below.

① Establish a connection with the programmable controller in online mode. (For details, see "4.2.4 Changing connections.")

② Display the [[Online] Setup by module] window and click the Set IP Address button.

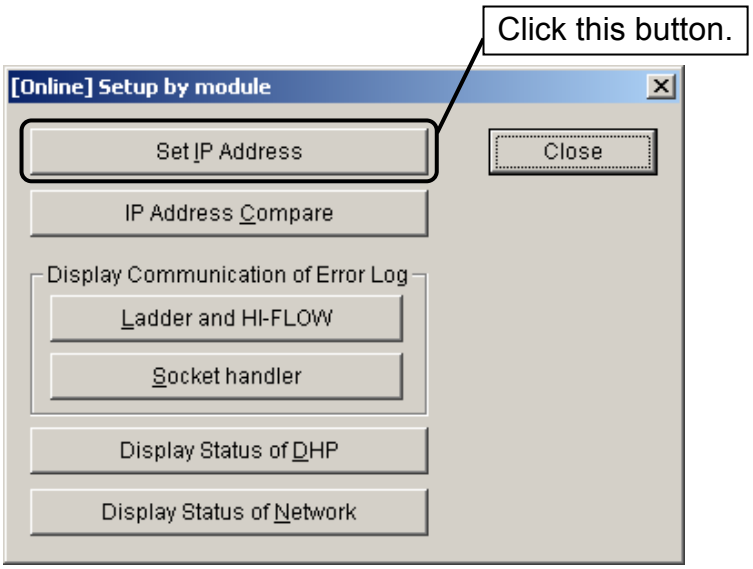


Figure 4-32 The [[Online] Setup by module] Window and Clicking the Set IP Address Button

③ The [[Online] Set IP Address] window as shown below is displayed. In this window, click the LOAD button.

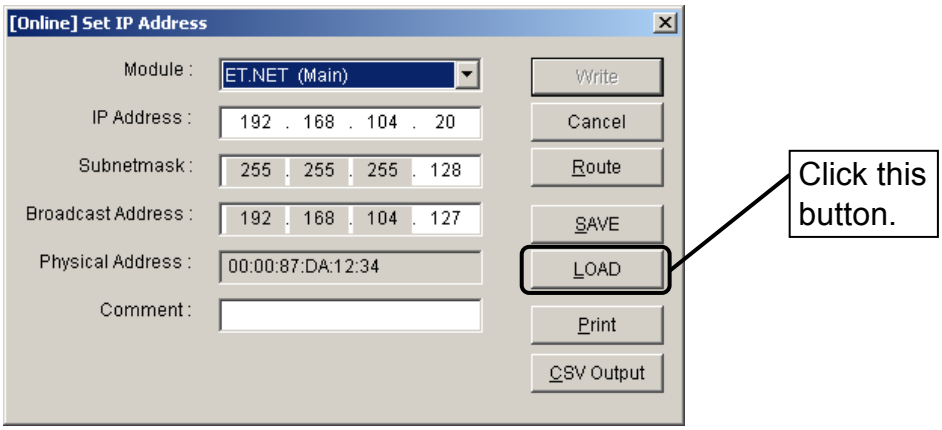


Figure 4-33 The [[Online] Set IP Address] Window and Clicking the LOAD Button

④ The [Open] window as shown below appears. Select the desired IP address setup information file and click the Open button.

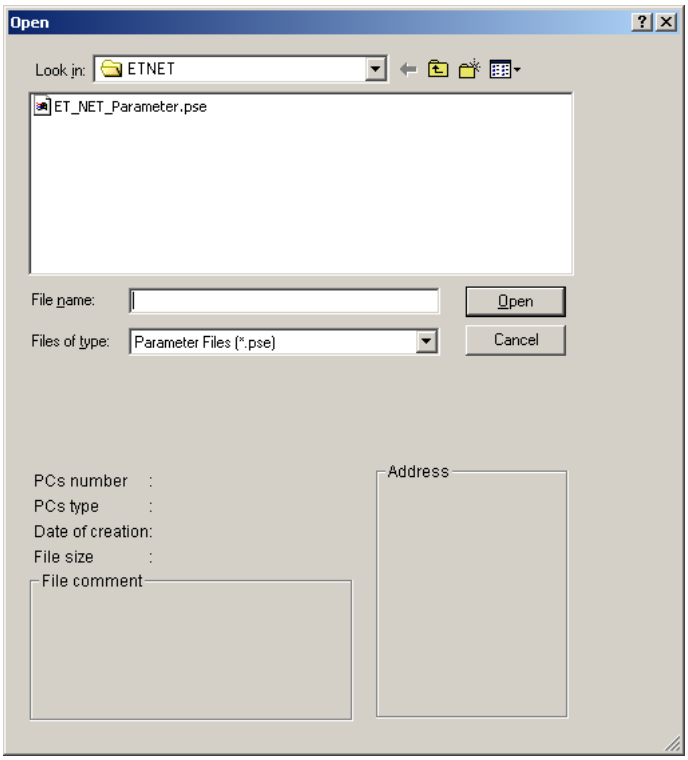


Figure 4-34 The [Open] Window

⑤ The [Open] window closes and the contents of the selected IP address setup info file are displayed in the [[Online] Set IP Address] window, as shown below.

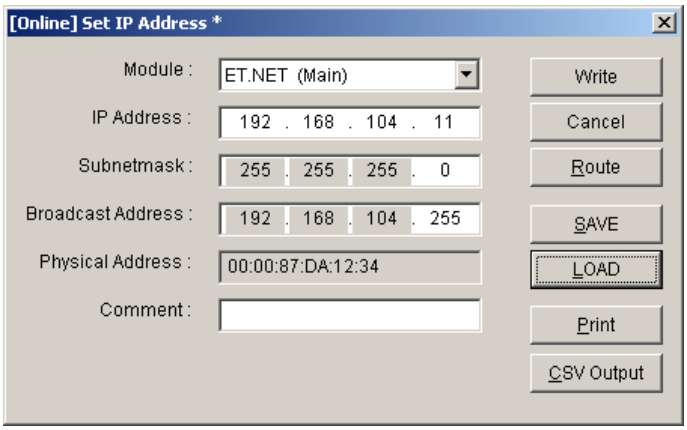


Figure 4-35 The [[Online] Set IP Address] Window Showing the Contents of the Selected IP Address Setup Info File

At the same time, an asterisk (“*”) is appended to the end of the title of the [[Online] Set IP Address] window.

4.3.11 Save IP address setup information in file

Function: Saves the displayed IP address setup information (including the routing information) in a specified file. (This command is supported only by Ver-Rev 02-03 or later of the ET.NET system and can be used only in online mode.)

Operation: The procedure used is described below.

① Establish a connection with the programmable controller in online mode. (For details, see "4.2.4 Changing connections.")

② Display the [[Online] Setup by module] window and click the Set IP Address button.

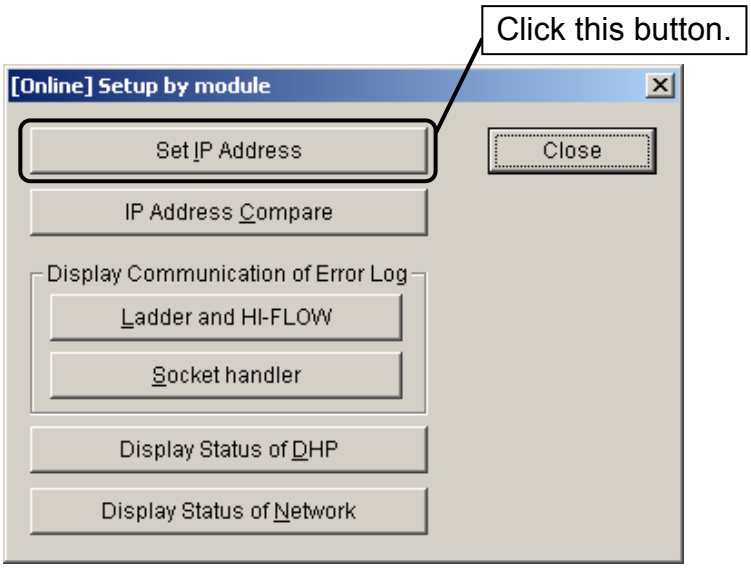


Figure 4-36 The [[Online] Setup by module] Window and Clicking the Set IP Address Button

③ The [[Online] Set IP Address] window as shown below is displayed. In this window, click the SAVE button.

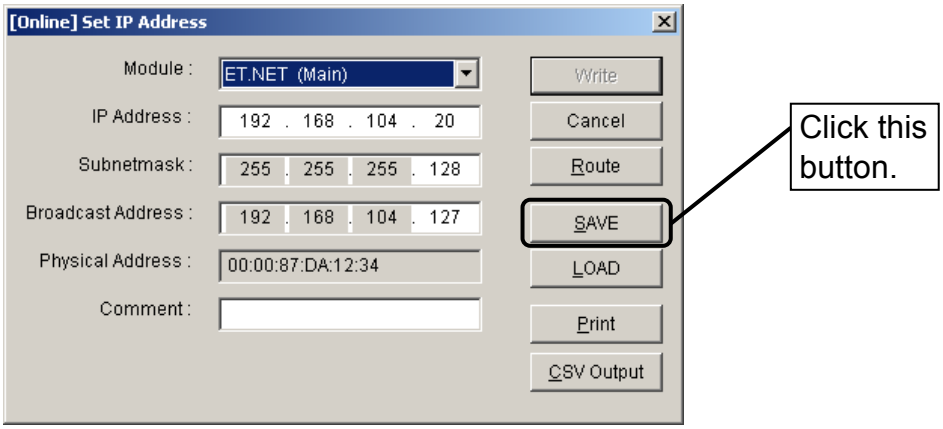


Figure 4-37 The [[Online] Set IP Address] Window and Clicking the SAVE Button

④ The [Save As] window as shown below appears. In this window, select the desired folder ("Save in" folder) and enter a unique file name ("File name"). Then, click the Save button.

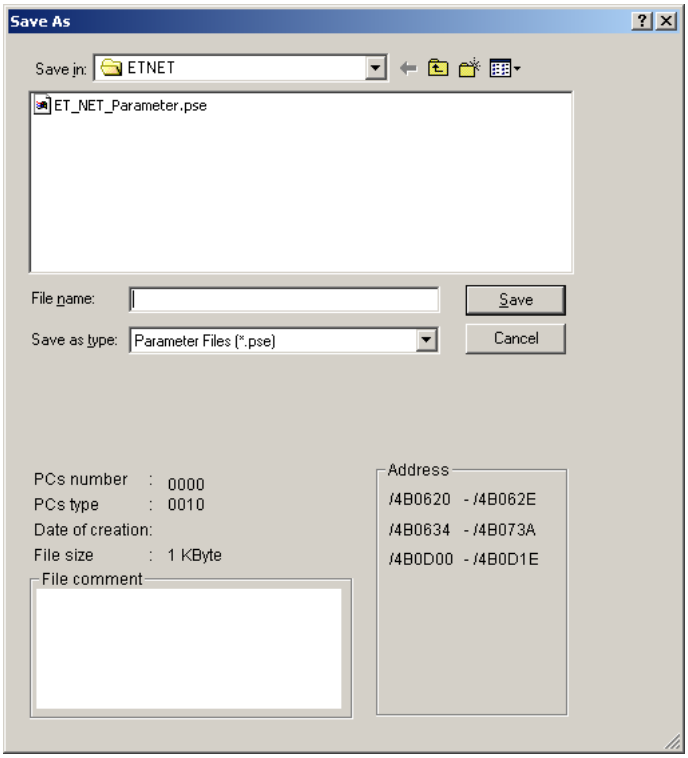


Figure 4-38 The [Save As] Window

⑤ The [Save As] window closes and the displayed IP address setup information (including the routing information) is saved in the specified file.

4.3.12 Print IP address setup information

Function: Prints the displayed IP address setup information on a specified printer. (This command is supported only by Ver-Rev 02-03 or later of the ET.NET system and can be used only in online mode.)

Operation: The procedure used is described below.

① Establish a connection with the programmable controller in online mode. (For details, see "4.2.4 Changing connections.")

② Display the [[Online] Setup by module] window and click the Set IP Address button.

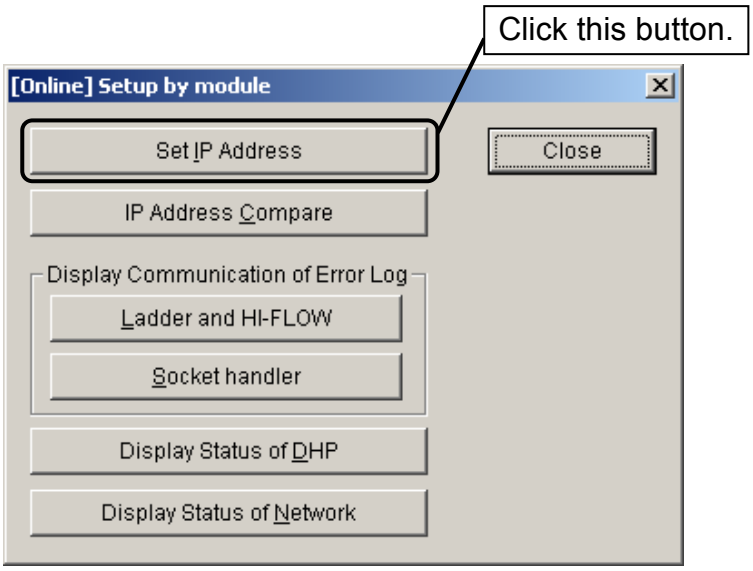


Figure 4-39 The [[Online] Setup by module] Window and Clicking the Set IP Address Button

③ The [[Online] Set IP Address] window as shown below is displayed. In this window, click the Print button.

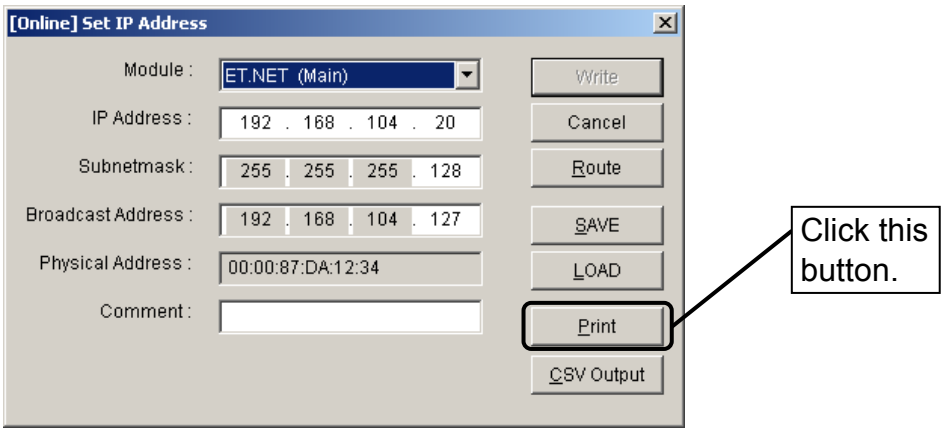


Figure 4-40 The [[Online] Set IP Address] Window and Clicking the Print Button

④ The [Print] dialog box as shown below is displayed. Specify the desired printer and properties and click the OK button. The displayed IP address setup information will then be printed on the printer.

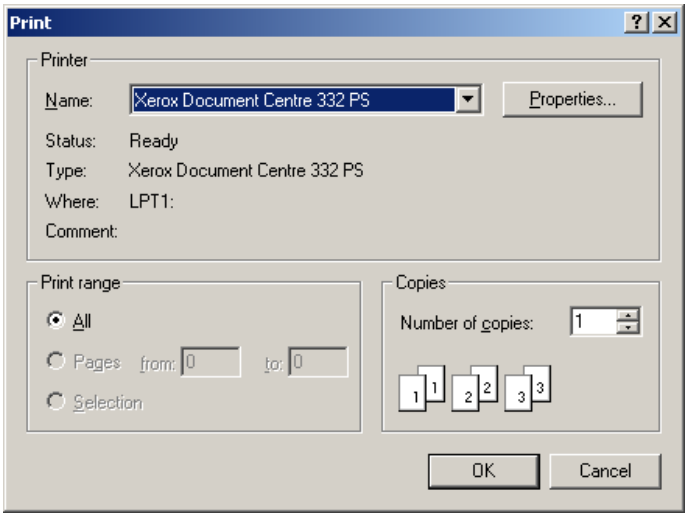


Figure 4-41 The [Print] Dialog Box

```
ET.NET     2010/06/11    21:01:28
Ethrer Net(192.192.192.1)

ET.NET(MAIN)
  IP Address              192.168.104. 20
  Subnetmask              255.255.255.128
  Broadcast Address       192.168.104.127
  Physical Address        00:00:87:DA:12:34
  Comment

Route
                          Communication point address Gateway
  Default                   0. 0. 0. 0                Not yet setup
  Route1                  Not yet setup               Not yet setup
  Route2                  Not yet setup               Not yet setup
  Route3                  Not yet setup               Not yet setup
  Route4                  Not yet setup               Not yet setup
  Route5                  Not yet setup               Not yet setup
  Route6                  Not yet setup               Not yet setup
  Route7                  Not yet setup               Not yet setup
  Route8                  Not yet setup               Not yet setup
  Route9                  Not yet setup               Not yet setup
  Route10                 Not yet setup               Not yet setup
  Route11                 Not yet setup               Not yet setup
  Route12                 Not yet setup               Not yet setup
  Route13                 Not yet setup               Not yet setup
  Route14                 Not yet setup               Not yet setup

ET.NET(SUB)
  IP Address                1. 0. 0.100
  Subnetmask              255.128. 0. 0
  Broadcast Address         1.127.255.255
  Physical Address        00:00:00:00:00:00
  Comment

Route
                          Communication point address Gateway
  Default                   0. 0. 0. 0                Not yet setup
  Route1                  Not yet setup               Not yet setup
  Route2                  Not yet setup               Not yet setup
  Route3                  Not yet setup               Not yet setup
  Route4                  Not yet setup               Not yet setup
  Route5                  Not yet setup               Not yet setup
  Route6                  Not yet setup               Not yet setup
  Route7                  Not yet setup               Not yet setup
  Route8                  Not yet setup               Not yet setup
  Route9                  Not yet setup               Not yet setup
  Route10                 Not yet setup               Not yet setup
  Route11                 Not yet setup               Not yet setup
  Route12                 Not yet setup               Not yet setup
  Route13                 Not yet setup               Not yet setup
  Route14                 Not yet setup               Not yet setup
```

Figure 4-42 A Printout of the Displayed IP Address Setup Information

4.3.13 Output IP address setup information in CSV format

Function: Outputs the displayed IP address setup information to a specified file in CSV (comma-separated values) format. (This command is supported only by Ver-Rev 02-03 or later of the ET.NET system and can be used only in online mode.)

Operation: The procedure used is described below.

① Establish a connection with the programmable controller in online mode. (For details, see "4.2.4 Changing connections.")

② Display the [[Online] Setup by module] window and click the Set IP Address button.

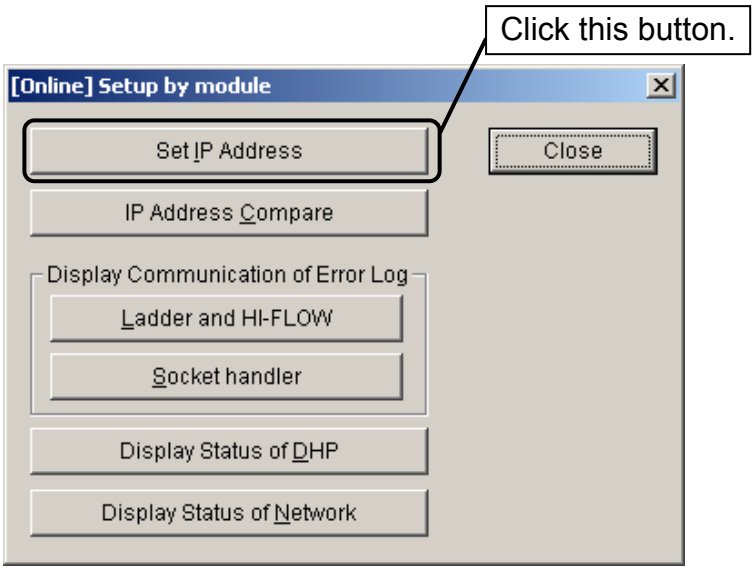


Figure 4-43 The [[Online] Setup by module] Window and Clicking the Set IP Address Button

③ The [[Online] Set IP Address] window as shown below is displayed. In this window, click the CSV Output button.

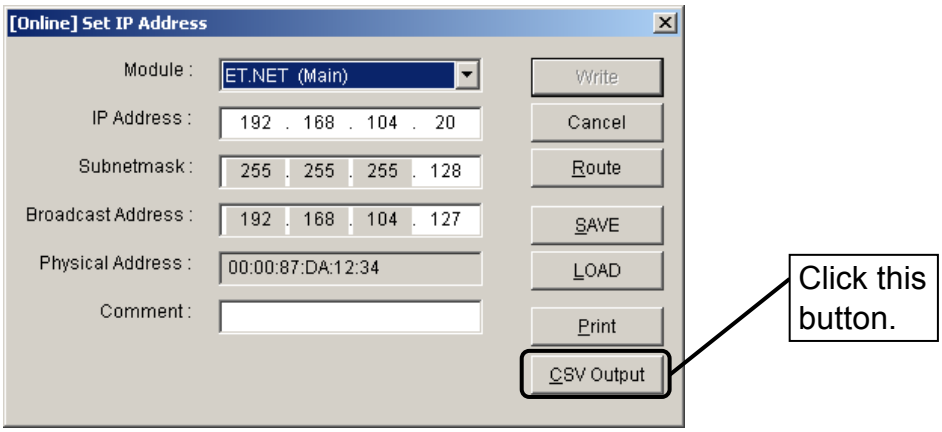


Figure 4-44 The [[Online] Set IP Address] Window and Clicking the CSV Output Button

④ The [Save As] window as shown below appears. In this window, select the desired folder and enter a unique file name. Then, click the Save button. The displayed IP address setup information will then be output to the specified file in CSV format.

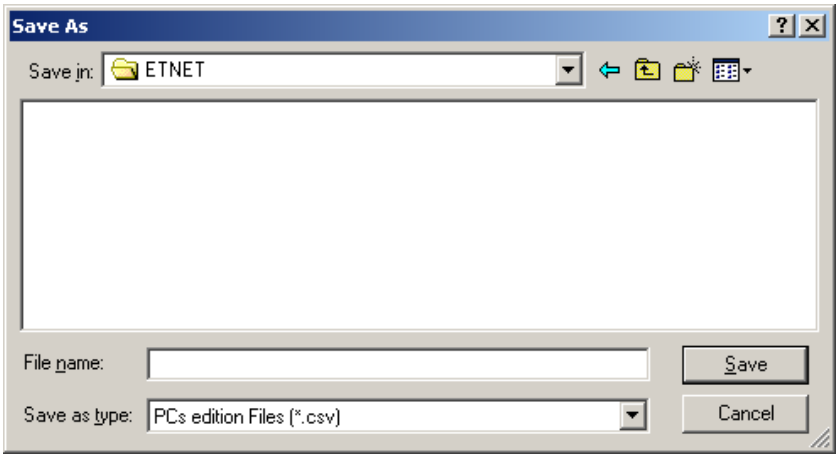


Figure 4-45 The [Save As] Window for Output in CSV Format

4.3.14 IP address compare

Function: Compares the IP address setup information defined in the connected programmable controller with the IP address setup information stored in a selected file and displays the result. (This command is supported only by Ver-Rev 02-03 or later of the ET.NET system and can be used only in online mode.)

Operation: The procedure used is described below.

① Establish a connection with the programmable controller in online mode. (For details, see "4.2.4 Changing connections.")

② Display the [[Online] Setup by module] window and click the IP Address Compare button.

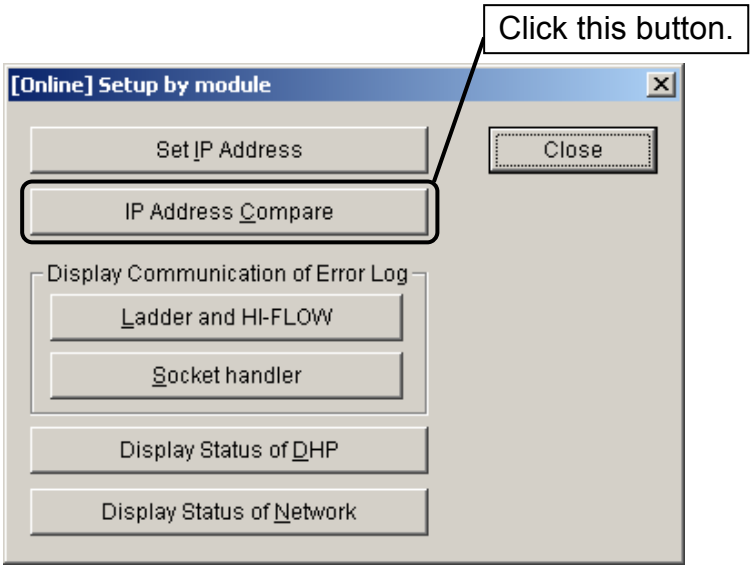


Figure 4-46 The [[Online] Setup by module] Window and Clicking the IP Address Compare Button

③ The [Open] window as shown below is displayed. Choose the desired IP address setup info file and click the Open button.

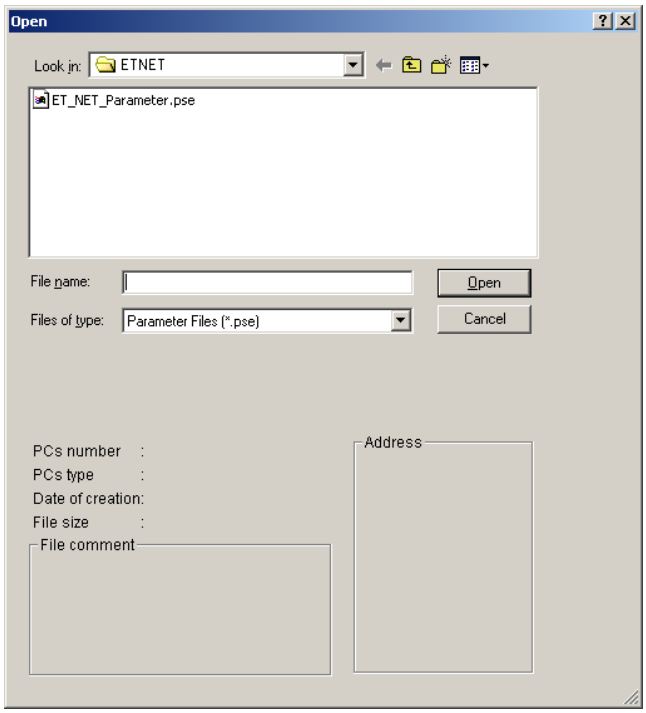


Figure 4-47 The [Open] Window

④ Comparison is performed. If no differences are found between the two sets of IP address setup information(*), a message to that effect is displayed as shown below. Click the OK button.

Figure 4-48 The No Differences Found Message

(*) The IP address setup information consists of the following items:

- IP address (MAIN/SUB)
- Subnet mask (MAIN/SUB)
- Broadcast address (MAIN/SUB)
- Comment (MAIN/SUB)
- Routing information (MAIN/SUB)

⑤ If differences are found between them with regard to the above items of IP address setup information, they will be displayed in a list, as shown below. Check the differences and correct them if necessary.

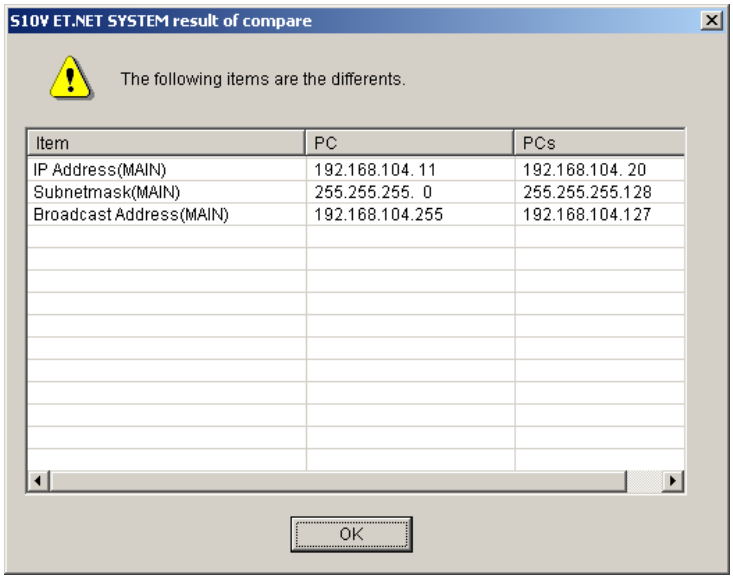


Figure 4-49 An Example of the Differences Found Message

5 PROGRAMMING

5.1 Software Configuration of ET.NET

The ET.NET software consists of system programs stored in the ET.NET module and user programs, which are created by users themselves.

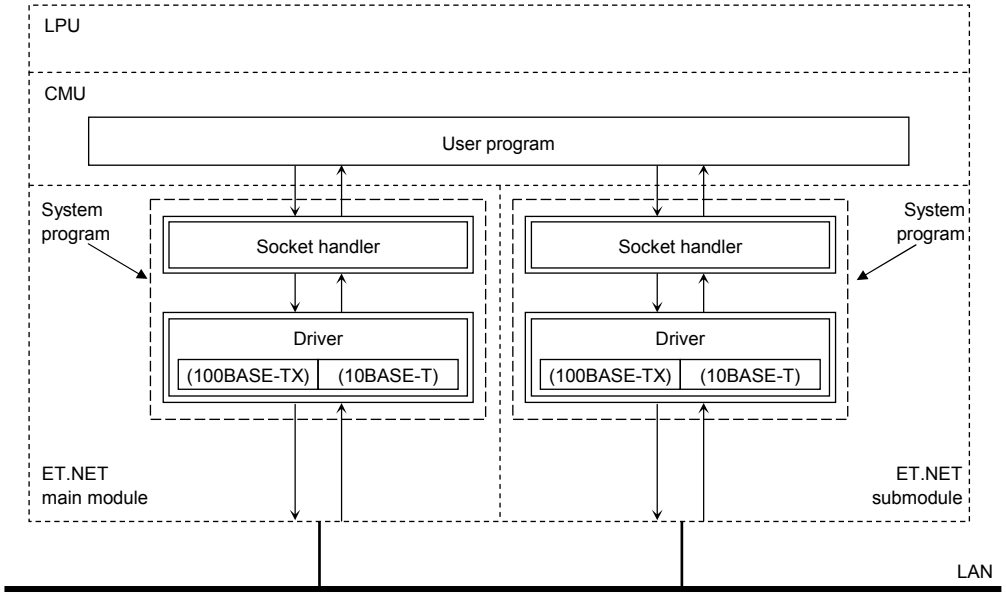


Figure 5-1 ET.NET Software Map

5.2 User Program

The user program starts the socket handler, and sends or receives data.

5.3 Socket Handler

The socket handler, invoked as a function in C, controls the ET.NET module for user program, and carries out data transmission and reception. The socket handler consists of 20 functions.
Call the socket handler by specifying its entry addresses. A user program cannot be created (linked) in a form including the socket handler.

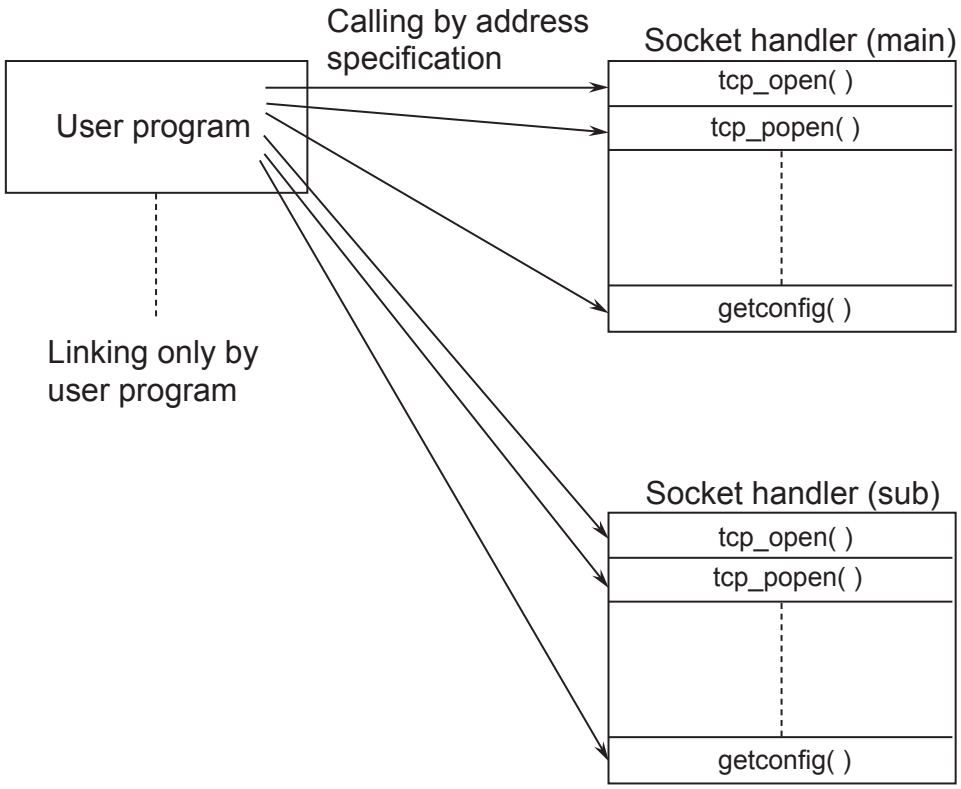


Figure 5-2 Calling a Socket Handler from a User Program

5.3.1 Socket handler function list

The table below lists the names of the socket handlers available, with a summary description of the function of each.
Subroutine call address of each socket handler is same as address of LQE520 and LQE720.

Table 5-1 Socket Handler Function List

Name	Subroutine call address (S10mini and S10V are in common)		Function	Corresponding program
	Main	Sub		
tcp_open( )	/874100	/8F4100	Actively opens TCP.	TCP/IP
tcp_popen( )	/874106	/8F4106	Passively opens TCP.	TCP/IP
tcp_accept( )	/87410C	/8F410C	Accepts a TCP connection request.	TCP/IP
tcp_close( )	/874112	/8F4112	Terminates a TCP connection.	TCP/IP
tcp_abort( )	/87411E	/8F411E	Kills a TCP connection.	TCP/IP
tcp_getaddr( )	/874124	/8F4124	Reads TCP socket information.	TCP/IP
tcp_stat( )	/87412A	/8F412A	Reads TCP connection status.	TCP/IP
tcp_send( )	/874130	/8F4130	Sends TCP data.	TCP/IP
tcp_receive( )	/874136	/8F4136	Receives TCP data.	TCP/IP
udp_open( )	/874160	/8F4160	Opens UDP.	UDP/IP
udp_close( )	/874166	/8F4166	Closes UDP.	UDP/IP
udp_send( )	/87416C	/8F416C	Sends UDP data.	UDP/IP
udp_receive( )	/874172	/8F4172	Receives UDP data.	UDP/IP
route_list( )	/874178	/8F4178	Reads routing information.	TCP/IP and UDP/IP
route_del( )	/87417E	/8F417E	Deletes routing information.	TCP/IP and UDP/IP
route_add( )	/874184	/8F4184	Registers routing information.	TCP/IP and UDP/IP
arp_list( )	/87418A	/8F418A	Reads ARP information.	TCP/IP and UDP/IP
arp_del( )	/874190	/8F4190	Deletes ARP information.	TCP/IP and UDP/IP
arp_add( )	/874196	/8F4196	Registers ARP information.	TCP/IP and UDP/IP
getconfig( )	/87419C	/8F419C	Reads configuration information.	TCP/IP and UDP/IP

NOTICE

- The maximum number of sockets that can be used simultaneously by one single module is 24 for TCP and UDP.
- The port numbers 0 to 9999 are reserved by the system; the user can use the port numbers 10000 to 65535 (except the ports 60015 and 60016 for TCP and 60013 and 60020 for UDP, which are used exclusively by the system).
- The length of data to be transmitted or received in each invocation of a function is 1 to 4096 bytes for TCP and 1 to 1472 bytes for UDP.
- The IP addresses and subnet masks are set in the operating system table in the LPU. When the LPU is replaced, these items need be set again.

- Forcible termination of task -
If a task using the socket handler is terminated forcibly, the socket remains in registered state (except when the task has executed tcp_close( ) or udp_close( ) for the socket used by that task). That is, the socket status at the time of task forcible termination remains undeleted although the task terminated. The socket in such a state is called a “floating socket”.
As a floating socket cannot be used by other tasks, it needs to be released by resetting the module or by turning off the power to the module and then back on again.

tcp_open( )

Function This function registers a socket of the TCP/IP program, reserves a port, and issues a connection request for a remote station. The registered socket ID or an error code is returned as the return value. This function transmits SYN and waits for connection establishment (SYN reception from remote station). If there is no response from the remote station within 45 seconds, this function ends up with a port release error (error code: /F0FF). In this case, reissue tcp_open( ).

Linking procedure

Main	Sub

struct open_p {
long dst_ip;
short dst_port;
short src_port;
char notuse;
char ttl;
};
~
short (*tcp_open)( );
short rtn;
struct open_p *padr;
~
tcp_open =(short (*) ( )) 0x874100;
~
rtn = (*tcp_open)(padr);
~
	struct open_p {
	long dst_ip;
	short dst_port;
	short src_port;
	char notuse;
	char ttl;
	};
	~
	short (*tcp_open)( );
	short rtn;
	struct open_p *padr;
	~
	tcp_open =(short (*) ( )) 0x8F4100;
	~
	rtn = (*tcp_open)(padr);
	~

Parameters

Input parameters:

padr: Starting address of input parameters (Specify an even address.)

padr -> dst_ip: IP address of remote station

padr -> dst_port: Port number of remote station

padr ->src_port: Port number of local station

(If ttl is set to 0, the port number automatically assigned is in the range 1024 to 2047.)

padr ->notuse: Fixed at 0 (unused)

padr ->ttl: Time to live (If ttl is set to 0, the default value (30) is assumed.)

Output parameters:

Return value: The registered socket ID or an error code is returned.

(/0001 to /0018): Registered socket ID

(/F000 to /FFFF): Error occurred.

For a definition of the error code, see “7.3.4 Error codes for socket handler-reported errors.”

tcp_popen( )

Function This function registers a socket for the TCP/IP program, and puts the socket into passive state. The registered socket ID or an error code is returned as the return value. This function is equivalent to socket+bind+listen in UNIX. If dst_ip and dst_port are set to 0, a connection request from any remote station can be accepted. If src_port is set to 0, optional port from 1024 to 2047 is reserved.

Linking procedure

Main	Sub

struct popen_p {
long dst_ip;
short dst_port;
short src_port;
char listennum;
char ttl;
};
~
short (*tcp_popen)( );
short rtn;
struct popen_p *padr;
~
tcp_popen = (short (*)( ))0x874106;
~
rtn = (*tcp_popen)(padr);
~
	struct popen_p {
	long dst_ip;
	short dst_port;
	short src_port;
	char listennum;
	char ttl;
	};
	~
	short (*tcp_popen)( );
	short rtn;
	struct popen_p *padr;
	~
	tcp_popen = (short (*)( ))0x8F4106;
	~
	rtn = (*tcp_popen)(padr);
	~

Parameters

Input parameters:

padr: Starting address of input parameters (Specify an even address.)

padr -> dst_ip: IP address of remote station (If remote station is not specified, dst_ip is set to 0.)

padr -> dst_port: Port number of remote station (If remote station is not specified, dst_port is set to 0.)

padr -> src_port: Port number of local station

padr -> listennum: Fixed at 0

padr -> ttl: Time to live (If ttl is set to 0, the default value (30) is assumed.)

Output parameters:

Return value: The registered socket ID or an error code is returned.

(/0001 to /0018): Registered socket ID

(/F000 to /FFFF): Error occurred.

For a definition of the error code, see “7.3.4 Error codes for socket handler-reported errors.”

tcp_accept( )

Function This function waits for a connection request (SYN reception) for the socket ID that was placed in passive state by the tcp_popen( ) function in the TCP/IP program, and accepts connection establishment. The socket ID registered after connection establishment or an error code is returned as the return value. The socket ID in an input parameter and that registered after connection establishment have the same value. This function continues waiting until the remote station is connected.

Linking procedure

Main	Sub

struct accept_p {
short s_id;
};
~
short (*tcp_accept)( );
short rtn;
struct accept_p *padr;
~
tcp_accept =(short (*) ( )) 0x87410C;
~
rtn = (*tcp_accept)(padr);
~
	struct accept_p {
	short s_id;
	};
	~
	short (*tcp_accept)( );
	short rtn;
	struct accept_p *padr;
	~
	tcp_accept =(short (*) ( )) 0x8F410C;
	~
	rtn = (*tcp_accept)(padr);
	~

Parameters

Input parameters:

padr: Starting address of input parameters (Specify an even address.)

padr -> s_id: Socket ID

Output parameters:

Return value: The registered socket ID or an error code is returned.

(/0001 to /0018): Registered socket ID

(/F000 to /FFFF): Error occurred.

For a definition of the error code, see “7.3.4 Error codes for socket handler-reported errors.”

tcp_close( )

Function This function terminates the connection corresponding to a socket ID, and deletes the socket. The processing result is returned as the return value. This function transmits FIN characters and waits for connection termination (FIN reception from remote station). If there is no response from the remote station within 30 seconds, this function ends up with a socket driver timeout error (error code: /F012). In this case, issue tcp_abort( ).

Linking procedure

Main	Sub

struct close_p {
short s_id;
};
~
short (*tcp_close)( );
short rtn;
struct close_p *padr;
~
tcp_close = (short (*) ( ))0x874112;
~
rtn = (*tcp_close)(padr);
~
	struct close_p {
	short s_id;
	};
	~
	short (*tcp_close)( );
	short rtn;
	struct close_p *padr;
	~
	tcp_close = (short (*) ( ))0x8F4112;
	~
	rtn = (*tcp_close)(padr);
	~

Parameters

Input parameters:

padr: Starting address of input parameters (Specify an even address.)

padr -> s_id: Socket ID

Output parameters:

Return value: The processing result is returned.

(0): Normal termination

(/F000 to /FFFF): Error occurred.

For a definition of the error code, see “7.3.4 Error codes for socket handler-reported errors.”

tcp_abort( )

Function This function kills (by sending RST characters) the connection corresponding to a socket ID, and deletes the socket. The processing result is returned as the return value.

Linking procedure

Main	Sub

struct sid_p {
short s_id;
};
~
short (*tcp_abort)( );
short rtn;
struct sid_p *padr;
~
tcp_abort = (short(*)( ))0x87411E;
~
rtn = (*tcp_abort)(padr);
~
	struct sid_p {
	short s_id;
	};
	~
	short (*tcp_abort)( );
	short rtn;
	struct sid_p *padr;
	~
	tcp_abort = (short(*)( ))0x8F411E;
	~
	rtn = (*tcp_abort)(padr);
	~

Parameters

Input parameters:

padr: Starting address of input parameters (Specify an even address.)

padr -> s_id: Socket ID

Output parameters:

Return value: The processing result is returned.

(0): Normal termination

(/F000 to /FFFF): Error occurred.

For a definition of the error code, see “7.3.4 Error codes for socket handler-reported errors.”

tcp_getaddr( )

Function This function obtains the IP address of the remote station to be connected corresponding to a socket ID and the port numbers of the local and remote stations. The processing result is returned as the return value. When the result is normal termination, the obtained information at outinf is validated.

Linking procedure

Main	Sub

struct sid_p {
short s_id;
};
struct getaddr_p {
long ipaddr;
short src_port;
short dst_port;
};
〜
short (*tcp_getaddr)( );
short rtn;
struct sid_p *padr;
struct getaddr_p *outinf;
〜
tcp_getaddr = (short(*)( ))0x874124;
〜
rtn = (*tcp_getaddr)(padr, outinf);
〜
	struct sid_p {
	short s_id;
	};
	struct getaddr_p {
	long ipaddr;
	short src_port;
	short dst_port;
	};
	〜
	short (*tcp_getaddr)( );
	short rtn;
	struct sid_p *padr;
	struct getaddr_p *outinf;
	〜
	tcp_getaddr = (short(*)( ))0x8F4124;
	〜
	rtn = (*tcp_getaddr)(padr, outinf);
	〜

Parameters

Input parameters:

padr: Starting address of input parameters (Specify an even address.)

padr -> s_id: Socket ID

Output parameters:

outinf: Starting address of output parameters (Specify an even address.)

outinf -> ipaddr: IP address of remote station

outinf -> src_port: Port number of local station

outinf -> dst_port: Port number of remote station

Return value: The processing result is returned.

(0): Normal termination

(/F000 to /FFFF): Error occurred.

For a definition of the error code, see “7.3.4 Error codes for socket handler-reported errors.”

tcp_stat( )

Function This function obtains the status of the connection corresponding to a socket ID. The processing result is returned as the return value. When the result is normal termination, the obtained information at outinf is validated.

Linking procedure

Main	Sub

struct sid_p {
short s_id;
};
struct stat_p {
unsigned short stat;
unsigned short urg;
unsigned short sendwin;
unsigned short recvwin;
};
〜
short (*tcp_stat)( );
short rtn;
struct sid_p *padr;
struct stat_p *outinf;
〜
tcp_stat =(short(*)( )) 0x87412A;
〜
rtn = (*tcp_stat)(padr, outinf);
〜
	struct sid_p {
	short s_id;
	};
	struct stat_p {
	unsigned short stat;
	unsigned short urg;
	unsigned short sendwin;
	unsigned short recvwin;
	};
	〜
	short (*tcp_stat)( );
	short rtn;
	struct sid_p *padr;
	struct stat_p *outinf;
	〜
	tcp_stat =(short(*)( )) 0x8F412A;
	〜
	rtn = (*tcp_stat)(padr, outinf);
	〜

Parameters

Input parameters:

padr: Starting address of input parameters (Specify an even address.)

padr -> s_id: Socket ID

Output parameters:

outinf: Starting address of output parameters (Specify an even address.)

outinf -> stat: Connection status

0: CLOSED
1: LISTEN
2: SYN_SENT
3: SYN_RECEIVED
4: ESTABLISHED
5: CLOSE_WAIT
6: FIN_WAIT_1
7: CLOSING
8: LAST_ACK
9: FIN_WAIT_2
10: TIME_WAIT

outinf -> urg: Whether there is urgent data

0: There is no urgent data.
Other than 0: Number of urgent data items

outinf -> sendwin: Remaining quantity of send data of send window

outinf -> recvwin: Amount of receive data that has arrived

Return value: The processing result is returned.

(0): Normal termination

(/F000 to /FFFF): Error occurred.

For a definition of the error code, see “7.3.4 Error codes for socket handler-reported errors.”

tcp_send( )

Function This function sends data to the connection corresponding to a socket ID. The starting address and the length of the sent data are indicated by parameters buf and len, respectively. The processing result is returned as the return value. If the value /F012 is returned as the processing result, confirm that transmission is being retried, by checking the connection status and the residual quantity of the send window obtained by the tcp_stat( ) function. The tcp_send( ) function makes a return when the data is stored in the send window. Confirm the data transmission status by the remaining quantity of send data of the send window obtained by tcp_stat( ).

Linking procedure

Main	Sub

struct send_p {
short s_id;
short len;
char *buf;
};
~
short (*tcp_send)( );
short rtn;
struct send_p *padr;
~
tcp_send = (short(*) ( ))0x874130;
~
rtn = (*tcp_send)(padr);
~
	struct send_p {
	short s_id;
	short len;
	char *buf;
	};
	~
	short (*tcp_send)( );
	short rtn;
	struct send_p *padr;
	~
	tcp_send = (short(*) ( ))0x8F4130;
	~
	rtn = (*tcp_send)(padr);
	~

Parameters

Input parameters:

padr: Starting address of input parameters (Specify an even address.)

padr -> s_id: Socket ID

padr -> len: Length of sent data (1 to 4096 bytes)

padr -> buf: Starting address of sent data (Specify an even address.)

Output parameters:

Return value: The processing result is returned.

(0): Normal termination

(/F000 to /FFFF): Error occurred.

For a definition of the error code, see “7.3.4 Error codes for socket handler-reported errors.”

tcp_receive( )

Function This function receives data from the connection corresponding to a socket ID. The received data is stored in the receive buffer whose the starting address is indicated by parameter buf. The data length is specified by parameter len. The processing result is returned as the return value. In this function, receive wait time can be specified for parameter tim. However, this function makes a return when the data is received, even if the wait time has not elapsed.

Linking procedure

Main	Sub

struct receive_p {
short s_id;
short len;
char *buf;
long tim;
};
~
short (*tcp_receive)( );
short rtn;
struct receive_p *padr;
~
tcp_receive =(short(*) ( )) 0x874136;
~
rtn = (*tcp_receive)(padr);
~
	struct receive_p {
	short s_id;
	short len;
	char *buf;
	long tim;
	};
	~
	short (*tcp_receive)( );
	short rtn;
	struct receive_p *padr;
	~
	tcp_receive =(short(*) ( )) 0x8F4136;
	~
	rtn = (*tcp_receive)(padr);
	~

Parameters

Input parameters:

padr: Starting address of input parameters (Specify an even address.)

padr -> s_id: Socket ID

padr -> len: Receive buffer length (1 to 4096 bytes)

padr -> buf: Starting address of receive buffer (Specify an even address.)

padr -> tim: Receive wait time (0 to 86,400,000 ms [24 hours])

Output parameters:

Return value: The processing result is returned.

(0): Normal termination (no receive data)

(/0001 to /1000): Normal termination (number of received bytes)

(/F000 to /FFFF): Error occurred.

For a definition of the error code, see “7.3.4 Error codes for socket handler-reported errors.”

udp_open( )

Function This function registers a socket for the UDP/IP program, and reserves a port. The registered socket ID or an error code is returned as the return value.
If a 0 is specified for parameter dst_ip, packets can be received from an arbitrary host.
If a 0 is specified in the parameter dst_port, data can be received from an arbitrary port.
If a 0 is specified in the parameter src_port, unused ports from 1024 to 2047 are reserved.

Linking procedure

Main	Sub

struct uopen_p {
long dst_ip;
short dst_port;
short src_port;
char pktmode;
char ttl;
};
~
short (*udp_open)( );
short rtn;
struct uopen_p *padr;
~
udp_open =(short(*) ( )) 0x874160;
~
rtn = (*udp_open)(padr);
~
	struct uopen_p {
	long dst_ip;
	short dst_port;
	short src_port;
	char pktmode;
	char ttl;
	};
	~
	short (*udp_open)( );
	short rtn;
	struct uopen_p *padr;
	~
	udp_open =(short(*) ( )) 0x8F4160;
	~
	rtn = (*udp_open)(padr);
	~

Parameters

Input parameters:

padr: Starting address of input parameters (Specify an even address.)

padr -> dst_ip: IP address of remote station

padr -> dst_port: Port number of remote station

padr -> src_port: Port number of local station

padr -> pktmode: Packet mode (fixed to 0)

padr -> ttl: Time to live (If ttl is set to 0, the default value (30) is assumed.)

Output parameters:

Return value: The registered socket ID or an error code is returned.

(/0020 to /0027): Registered socket ID

(/F000 to /FFFF): Error occurred.

For a definition of the error code, see “7.3.4 Error codes for socket handler-reported errors.”

udp_close( )

Function This function deletes the socket identified by a given socket ID. The processing result is returned as the return value.

Linking procedure

Main	Sub

struct uclose_p {
short s_id;
};
~
short (*udp_close)( );
short rtn;
struct uclose_p *padr;
~
udp_close =(short(*) ( )) 0x874166;
~
rtn = (*udp_close)(padr);
~
	struct uclose_p {
	short s_id;
	};
	~
	short (*udp_close)( );
	short rtn;
	struct uclose_p *padr;
	~
	udp_close =(short(*) ( )) 0x8F4166;
	~
	rtn = (*udp_close)(padr);
	~

Parameters

Input parameters:

padr: Starting address of input parameters (Specify an even address.)

padr -> s_id: Socket ID

Output parameters:

Return value: The processing result is returned.

(0): Normal termination

(/F000 to /FFFF): Error occurred.

For a definition of the error code, see “7.3.4 Error codes for socket handler-reported errors.”

udp_send( )

Function This function sends data to the socket identified by a given socket ID. The starting address and the length of the sent data are indicated by the parameters buf and len, respectively. The processing result is returned as the return value. As for specifications of dst_ip and dst_port, those specified in udp_open( ) have priority.

Linking procedure

Main	Sub

struct usend_p {
short s_id;
short notuse;
long dst_ip;
short dst_port;
short len;
char *buf;
};
~
short (*udp_send)( );
short rtn;
struct usend_p *padr;
~
udp_send =(short(*) ( )) 0x87416C;
~
rtn = (*udp_send)(padr);
~
	struct usend_p {
	short s_id;
	short notuse;
	long dst_ip;
	short dst_port;
	short len;
	char *buf;
	};
	~
	short (*udp_send)( );
	short rtn;
	struct usend_p *padr;
	~
	udp_send =(short(*) ( )) 0x8F416C;
	~
	rtn = (*udp_send)(padr);
	~

Parameters

Input parameters:

padr: Starting address of input parameters (Specify an even address.)

padr -> s_id: Socket ID

padr -> notuse: Fixed at 0 (unused)

padr -> dst_ip: IP address of remote station

padr -> dst_port: Port number of remote station

padr -> len: Length of sent data (1 to 1472 bytes)

padr -> buf: Starting address of send data
(Specify an even address.)

If a value other than 0 is specified in udp_open( ), dst_ip and dst_port specifications in udp_open( ) are used.

Output parameters:

Return value: The processing result is returned.

(0): Normal termination

(/F000 to /FFFF): Error occurred.

For a definition of the error code, see “7.3.4 Error codes for socket handler-reported errors.”

■ Specifications of dst_ip and dst_port

- If a value other than 0 is specified in udp_open( ), the parameters specified in udp_open( ) are used.
- If a 0 is specified in udp_open( ), the parameters specified in udp_send( ) are used.
- If a 0 is specified in both udp_open( ) and udp_send( ), the function returns with an invalid address error (error code: /FFF0). In this case, correct the user program.

udp_receive( )

Function This function receives data from the socket identified by a given socket ID. The received data is stored in the receive buffer whose starting address is indicated by the parameter buf.
The processing result is returned as the return value. In this function, receive wait time can be specified in the parameter tim. However, this function makes a return when the data is received, even if the wait time has not elapsed.

Linking procedure

Main	Sub

struct ureceive_p {
short s_id;
short notuse;
char *buf;
long tim;
};
~
short (*udp_receive)( );
short rtn;
struct ureceive_p *padr;
~
udp_receive =(short(*) ( )) 0x874172;
~
rtn = (*udp_receive)(padr);
~
	struct ureceive_p {
	short s_id;
	short notuse;
	char *buf;
	long tim;
	};
	~
	short (*udp_receive)( );
	short rtn;
	struct ureceive_p *padr;
	~
	udp_receive =(short(*) ( )) 0x8F4172;
	~
	rtn = (*udp_receive)(padr);
	~

Parameters

Input parameters:

padr: Starting address of input parameters (Specify an even address.)

padr -> s_id: Socket ID

padr -> notuse: Fixed at 0 (unused)

padr -> buf: Starting address of receive buffer
(Specify an even address.)

padr -> tim: Receive wait time (0 to 86,400,000 ms [24 hours])

Output parameters:

Return value: The processing result is returned.

(0): Normal termination (no receive data)

(/0001 to /05C0): Normal termination (number of received bytes)

(/F000 to /FFFF): Error occurred.

For a definition of the error code, see “7.3.4 Error codes for socket handler-reported errors.”

NOTICE
Because the udp_receive( ) function receives data in units of packets, reserve a buffer area of 1472 bytes.

route_list( )

Function This function obtains routing information. (The maximum size in of the routing information table is 17 [routes].) The number of obtained entries is returned as the return value. If a 0 is specified for the parameter len, only the number of registered entries is returned. For len, specify a multiple of 16 (bytes).

Linking procedure

Main	Sub

struct lstrt_p {
short len;
short notues;
void *buf;
};
~
short (*route_list)( );
short rtn;
struct lstrt_p *padr;
~
route_list = (short(*) ( ))0x874178;
~
rtn = (*route_list)(padr);
~
	struct lstrt_p {
	short len;
	short notues;
	void *buf;
	};
	~
	short (*route_list)( );
	short rtn;
	struct lstrt_p *padr;
	~
	route_list = (short(*) ( ))0x8F4178;
	~
	rtn = (*route_list)(padr);
	~

Parameters

Input parameters:

padr: Starting address of input parameters (Specify an even address.)

padr -> len: Data length (number of bytes; multiple of 16)

padr -> notes: Fixed at 0 (unused)

padr -> buf: Starting address of data
(Specify an even address.)

Output parameters:

Return value: The number of obtained entries is returned.

(0): No entry

(/0001 to /0010): Number of obtained entries

Structure of obtained data (contents of buf):

```
typedef struct{
        unsigned long dstaddr: IP address of remote station
        unsigned long getwayadder: IP address of gateway
        unsigned short metric: Metric (number of gateways passed)
        unsigned short rt_types: Type
        unsigned short refcnt: Reference counter
        unsigned short notuse: (Unused)
}routeentry
```

■ About data length (len) specifications

If a specified data length (len) is smaller than the registered data length (i.e., the number of currently registered entries multiplied by the size [16 bytes] of each entry), the routing information obtained will be as long as the specified data length, the return value returned indicating the number of entries obtained. However, if it is smaller than the size of each entry, a return value of 0 will be returned.

If you want to obtain all the stored routing information, perform the following steps: 1) specify a value of 0 as the data length (len) to route_list() and issue the function; 2) the function will then return a count of the number of currently registered entries; and 3) specify the number of currently registered entries multiplied by the size (16 bytes) of each entry as the data length to route_list() and issue the function.

Alternatively, in Step 3, you can specify the maximum number of registerable entries (17) multiplied by the size of each entry as the data length to route_list() and issue the function.

route_del( )

Function This function deletes routing information from the routing information table. The processing result is returned as the return value.

Linking procedure

Main	Sub

struct delrt_p {
long dstaddr;
long gtwayaddr;
};
〜
short (*route_del)( );
short rtn;
struct delrt_p *padr;
〜
route_del =(short(*) ( )) 0x87417E;
〜
rtn = (*route_del)(padr);
〜
	struct delrt_p {
	long dstaddr;
	long gtwayaddr;
	};
	〜
	short (*route_del)( );
	short rtn;
	struct delrt_p *padr;
	〜
	route_del =(short(*) ( )) 0x8F417E;
	〜
	rtn = (*route_del)(padr);
	〜

Parameters

Input parameters:

padr: Starting address of input parameters (Specify an even address.)

padr -> dstaddr: IP address of remote station

padr -> gtwayaddr: IP address of gateway

Output parameters:

Return value: The processing result is returned.

(0): Normal termination

(/F000 to /FFFF): Error occurred.

For a definition of the error code, see “7.3.4 Error codes for socket handler-reported errors.”

route_add( )

Function This function adds routing information to the routing information table. The processing result is returned as the return value.
If this function is unable to add routing information to the table because the table is full, the function will end up with an "insufficient internal buffer space" error (error code = /FFFF). In this case, delete all unnecessary routing information from the table by issuing route_del(), and then issue route_add() again.

Linking procedure

Main	Sub

struct addrt_p {
long dstaddr;
long gtwayaddr;
short metric;
};
≀
short (*route_add)( );
short rtn;
struct addrt_p *padr;
≀
route_add = (short(*) ( ))0x874184;
≀
rtn = (*route_add)(padr);
≀
	struct addrt_p {
	long dstaddr;
	long gtwayaddr;
	short metric;
	};
	≀
	short (*route_add)( );
	short rtn;
	struct addrt_p *padr;
	≀
	route_add = (short(*) ( ))0x8F4184;
	≀
	rtn = (*route_add)(padr);
	≀

Parameters

Input parameters:

padr: Starting address of input parameters (Specify an even address.)

padr -> dstaddr: IP address of remote station

padr -> gtwayaddr: IP address of gateway

padr -> metric: Metric (number of gateways passed)

Output parameters:

Return value: The processing result is returned.

(0): Normal termination

(/F000 to /FFFF): Error occurred.

For a definition of the error code, see "7.3.4 Error codes for socket handler-reported errors."

arp_list( )

Function This function obtains ARP information. (The maximum size in of the ARP information table is 32 [ARPs].) The number of obtained entries is returned as the return value. If a 0 is specified for parameter len, this function will return a count of the number of entries currently registered. For len, specify a multiple of 12 (bytes).

Linking procedure

Main	Sub

struct lstarp_p {
short len;
short notuse;
void *buf;
};
~
short (*arp_list)( );
short rtn;
struct lstarp_p *padr;
~
arp_list =(short (*)( ))0x87418A;
~
rtn = (*arp_list)(padr);
~
	struct lstarp_p {
	short len;
	short notuse;
	void *buf;
	};
	~
	short (*arp_list)( );
	short rtn;
	struct lstarp_p *padr;
	~
	arp_list =(short (*)( ))0x8F418A;
	~
	rtn = (*arp_list)(padr);
	~

Parameters

Input parameters:

padr: Starting address of input parameters (Specify an even address.)

padr -> len: Data length (number of bytes; multiple of 12)

padr -> notuse: Fixed at 0 (unused)

padr -> buf: Starting address of data

(Specify an even address.)

Output parameters:

Return value: The number of obtained entries is returned.

(0): No entry

(/0001 to /0020): Number of obtained entries

Structure of obtained data (contents of buf):

```
typedef struct{
        unsigned long ip_addr: IP address of remote station
        unsigned char et_addr[6]: Physical address of remote station
        unsigned char ar_timer: Timer
        unsigned char ar_flags: Flag
}arpt-t
```

■ About data length (len) specifications

If a specified data length (len) is smaller than the registered data length (i.e., the number of currently registered entries multiplied by the size [12 bytes] of each entry), the ARP information obtained will be as long as the specified data length, the return value returned indicating the number of entries obtained. However, if it is smaller than the size of each entry, a return value of 0 will be returned.

If you want to obtain all the stored ARP information, perform the following steps: 1) specify a value of 0 as the data length (len) to route_list() and issue the function; 2) the function will then return a count of the number of currently registered entries; and 3) specify the number of currently registered entries multiplied by the size (12 bytes) of each entry as the data length to route_list() and issue the function.

Alternatively, in Step 3, you can specify the maximum number of registerable entries (32) multiplied by the size of each entry as the data length to route_list() and issue the function.

arp_del( )

Function This function deletes ARP information from the ARP information table. The processing result is returned as the return value.

Linking procedure

Main	Sub

struct delarp_p {
unsigned long ipaddr;
unsigned char etaddr[6];
};
~
short (*arp_del)( );
short rtn;
struct delarp_p *padr;
~
arp_del =(short(*) ( )) 0x874190;
~
rtn = (*arp_del)(padr);
~
	struct delarp_p {
	unsigned long ipaddr;
	unsigned char etaddr[6];
	};
	~
	short (*arp_del)( );
	short rtn;
	struct delarp_p *padr;
	~
	arp_del =(short(*) ( )) 0x8F4190;
	~
	rtn = (*arp_del)(padr);
	~

Parameters

Input parameters:

padr: Starting address of input parameters (Specify an even address.)

padr -> ipaddr: IP address of remote station

padr -> etaddr[6]: Physical address of remote station

Output parameters:

Return value: The processing result is returned.

(0): Normal termination

(/F000 to /FFFF): Error occurred.

For a definition of the error code, see “7.3.4 Error codes for socket handler-reported errors.”

arp_add( )

Function This function adds ARP information to the ARP information table. The processing result is returned as the return value.
If this function is unable to add ARP information to the table because the table is full, the function will end up with an "insufficient internal buffer space" error (error code = /FFFF). In this case, delete all unnecessary ARP information from the table by issuing arp_del(), and then issue arp_add() again.

Linking procedure

Main	Sub

struct addarp_p {
long ipaddr;
char etaddr[6];
short flag;
};
≀
short (*arp_add)( );
short rtn;
struct addarp_p *padr;
≀
arp_add =(short(*) ( )) 0x874196;
≀
rtn = (*arp_add)(padr);
≀
	struct addarp_p {
	long ipaddr;
	char etaddr[6];
	short flag;
	};
	≀
	short (*arp_add)( );
	short rtn;
	struct addarp_p *padr;
	≀
	arp_add =(short(*) ( )) 0x8F4196;
	≀
	rtn = (*arp_add)(padr);
	≀

Parameters

Input parameters:

padr: Starting address of input parameters (Specify an even address.)

padr -> ipaddr: IP address of remote station

padr -> etaddr[6]: Physical address of remote station

padr -> flag: Flag (fixed at 0)

Output parameters:

Return value: The processing result is returned.

(0): Normal termination

(/F000 to /FFFF): Error occurred.

For a definition of the error code, see "7.3.4 Error codes for socket handler-reported errors."

getconfig( )

Function This function obtains the configuration blocks. The processing result is returned as the return value.

Linking procedure

Main	Sub

struct config_p {
void *config_ptr;
};
〜
short (*getconfig)( );
short rtn;
struct config_p *padr;
〜
getconfig = (short(*) ( ))0x87419C;
〜
rtn = (*getconfig)(padr);
〜
	struct config_p {
	void *config_ptr;
	};
	〜
	short (*getconfig)( );
	short rtn;
	struct config_p *padr;
	〜
	getconfig = (short(*) ( ))0x8F419C;
	〜
	rtn = (*getconfig)(padr);
	〜

Parameters

Input parameters:

padr: Starting address of input parameters (Specify an even address.)

padr -> config_ptr: Starting address of configuration block

Output parameters:

Return value: The processing result is returned.

(0): Normal termination

Configuration block:

Data structure of configuration block

```
struct config_ptr{
        long ip_addr: IP address (network order) of local station (option)
        long netmask: Subnetwork mask (option)
        long broadcast: Broadcast address (option)
        char tcp_num: Maximum number of TCP sockets (24)
        char udp_num: Maximum number of UDP sockets (24)
        char rt_num: Size of routing information table (17)
        char arp_num: Size of ARP information table (32)
        short tcp_win: Size of TCP send/receive window (4096)
};
```

5.4 Examples of Socket Handler Issuance Procedure

5.4.1 Example of using TCP/IP program

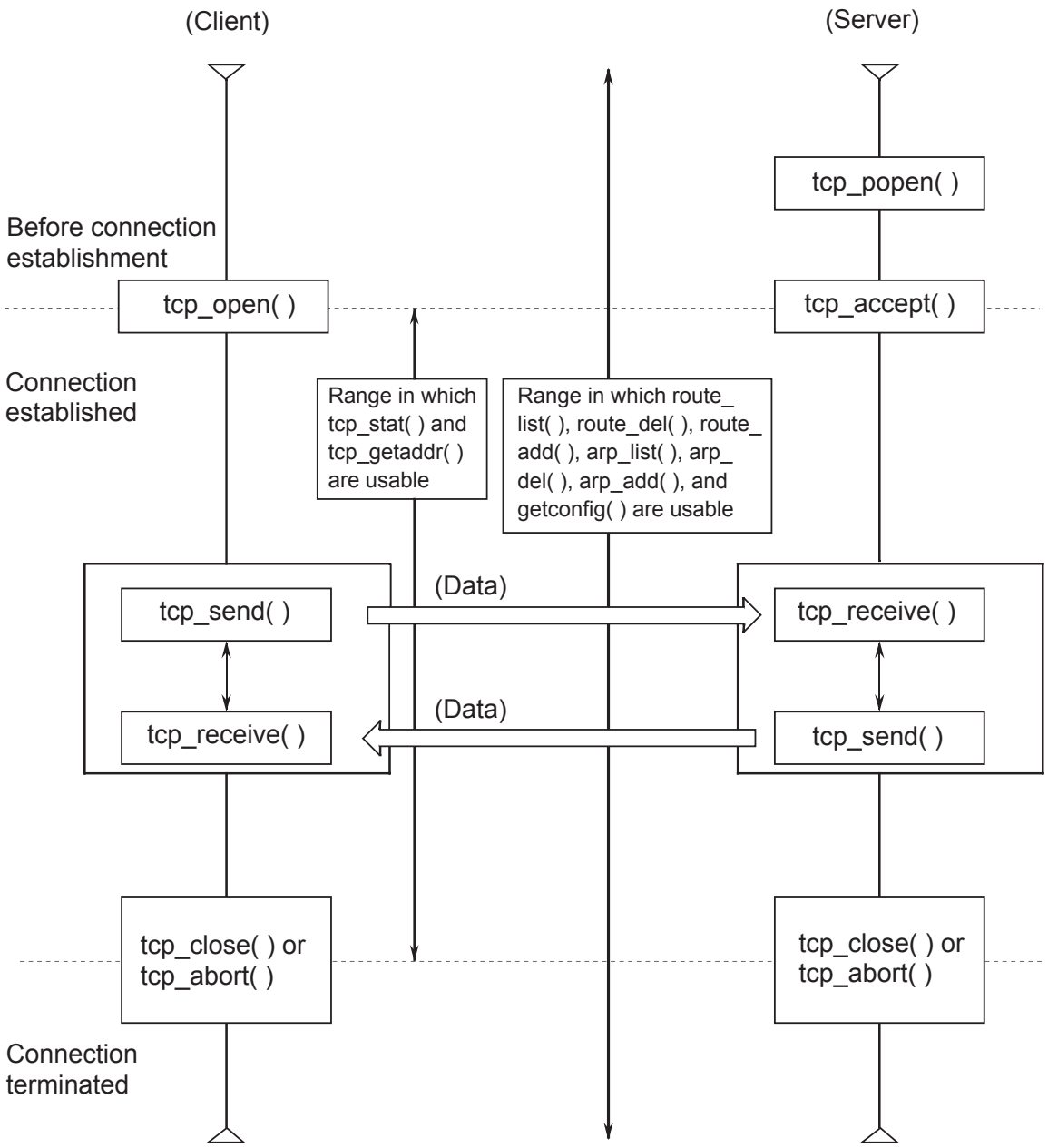


Figure 5-3 Example of Releasing a Socket Handler When Running a TCP/IP Program

5.4.2 Example of using UDP/IP program

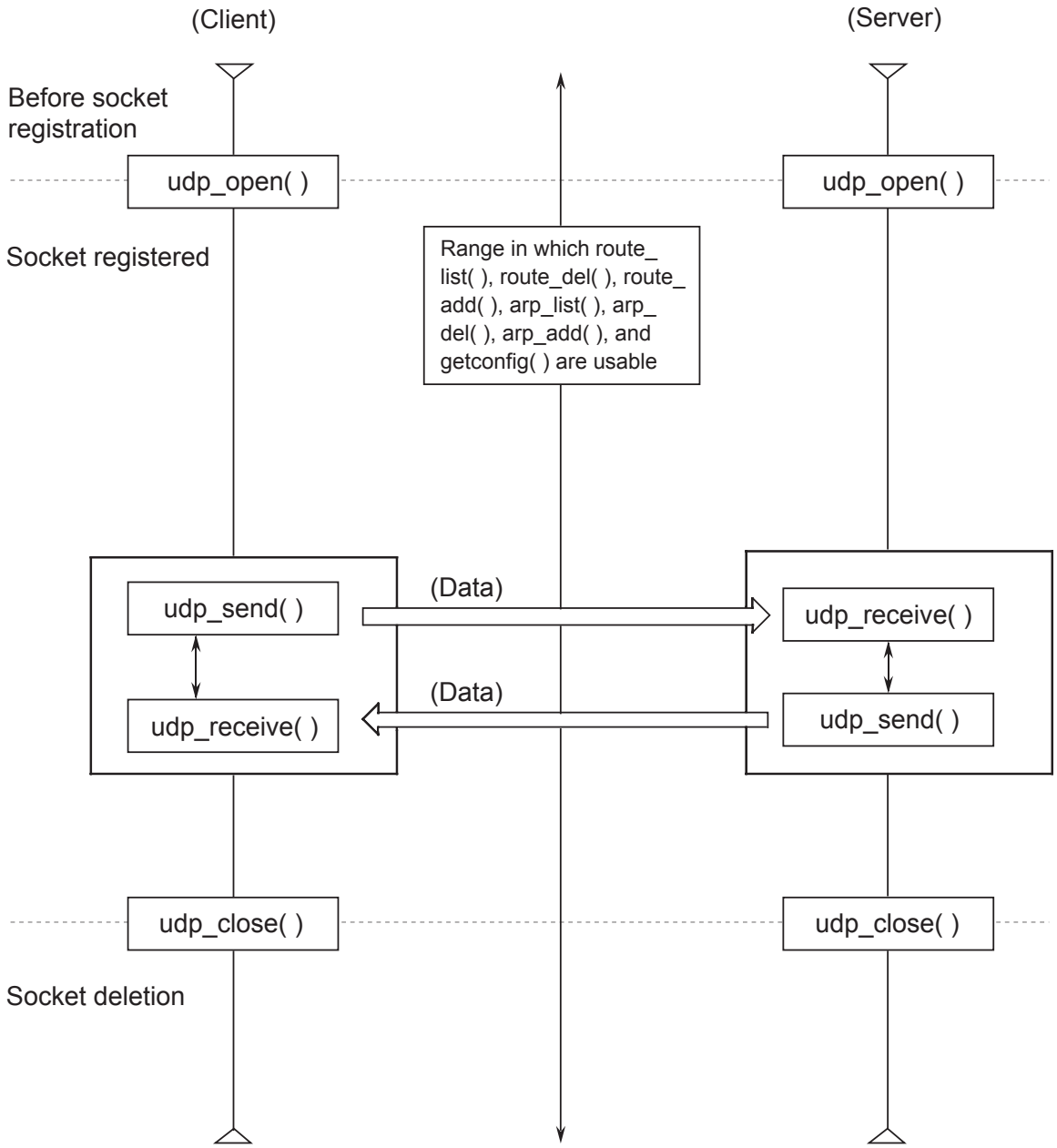
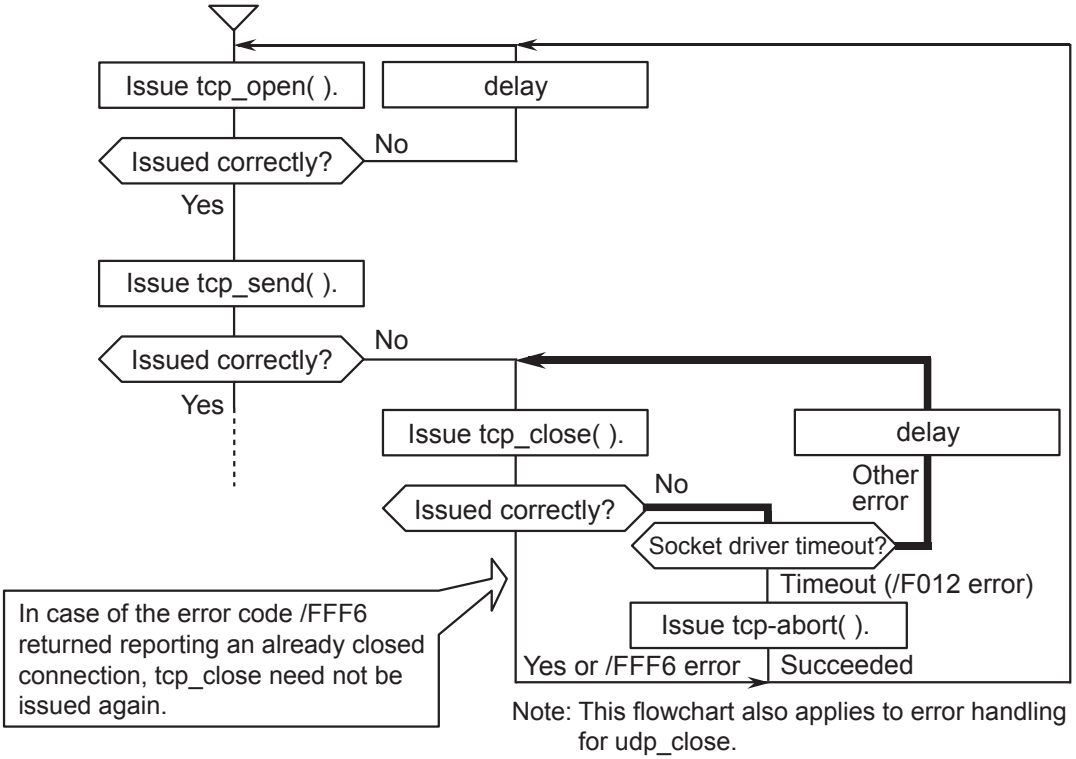


Figure 5-4 Example of Releasing a Socket Handler When Running a UDP/IP Program

- Error handling for tcp_close

You may issue tcp_close when the return code from a socket handler function indicates an error. If you have issued it, also check the return code from tcp_close. If the code indicates an error, issue tcp_close again as indicated in the table, which lists codes associated with errors detected by the socket handler, in order to eliminate the cause of the error. Otherwise, a connection may not be established again or a floating socket may be generated. An example of programming (flowchart) showing how the socket handler issues socket library functions is given below.



- Inhibited asynchronous access to the same socket

Multiple socket library functions asynchronously issued to a single socket may result in incorrect execution results of functions. This problem is likely to occur when multiple tasks issue socket library functions to the same socket. Make sure that one task handles one socket.

- Transmission timeout detection time

When the LQE720 issues a socket library function, an ACK packet may cause a timeout due to a communication error or a failure in the remote device. It takes time to detect a timeout as indicated in the table below. Therefore, at least the time in the table is required after a timeout of the socket handler is detected before the socket library function is issued again or a connection is established again. Assuming that communication errors are inevitable, confirm at the design stage that the detection time in the table do not cause a problem.

Item		Detection time	Description
tcp_open( ) timeout (SYN retry interval)		75 s	When receiving no response from the remote device, the socket handler retries SYN at the following intervals: 6 s, 12 s, 24 s, and 33 s
tcp_send( ) timeout (SEND retry interval)		30 s	When receiving no response from the remote device, the socket handler retransmits at the following intervals: 1 s, 2 s, 4 s, 8 s, and 16 s If 30 seconds pass after the socket handler has issued tcp_send( ), the socket handler detects a socket driver timeout (return code: /F012).
tcp_close( ) timeout (FIN retry interval)		30 s	When receiving FIN from the remote device and detecting the normal line disconnection, the socket handler ends immediately. When the module (LQE720) sends FIN to disconnect the line, the socket handler also ends immediately. When receiving no response from the remote device, the socket handler retries FIN at the following intervals: 1 s, 2 s, 4 s, 8 s, and 16 s If 30 seconds pass after the socket handler has issued tcp_close( ), the socket handler detects a socket driver timeout (return code: /F012). Issue tcp_abort to disconnect the line.
Response timeout	tcp_close( ), tcp_send( ), udp_close( )	30 s	Time from when the socket handler issues a command to a microprogram until it is judged that there is no response.
	tcp_abort( ), route_list( ), route_del( ), route_add( ), arp_list( ), arp_del( ), arp_add( ), getconfig( ), udp_send( ), tcp_getaddr( ), tcp_stat( )	10 s	

5.5 Inter-CPU Communication Sample Program

5.5.1 System configuration and program configuration

Figure 5-5 shows the system configuration. The program has the ET.NET module of CPU01 and that of CPU02 connected with each other by a logical line and lets CPU02 send 1024 bytes of data and CPU01 receive it.
To run this program, be sure to invoke a user program from CPU01.

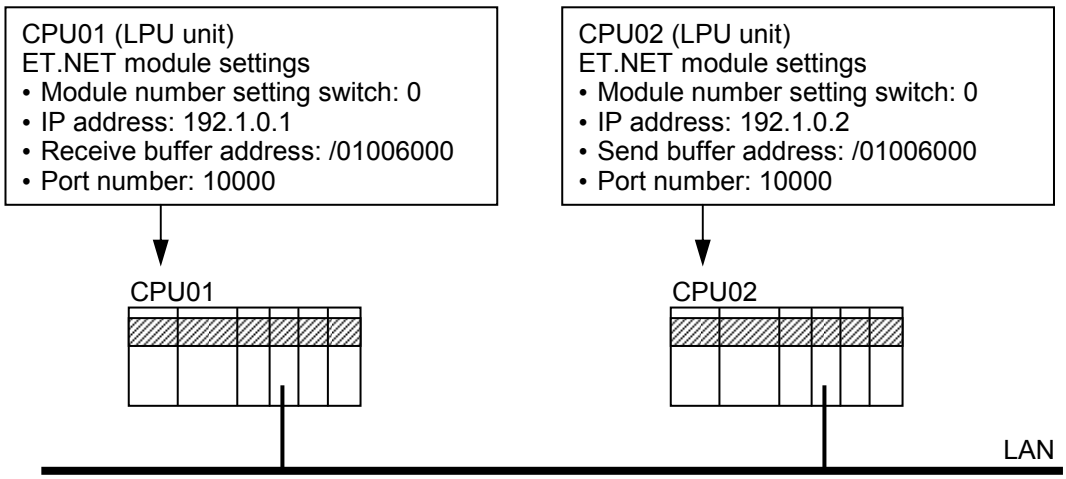


Figure 5-5 A Sample System Configuration for the Inter-CPU Communication Program

5.5.2 CPU01 program flowchart and sample program

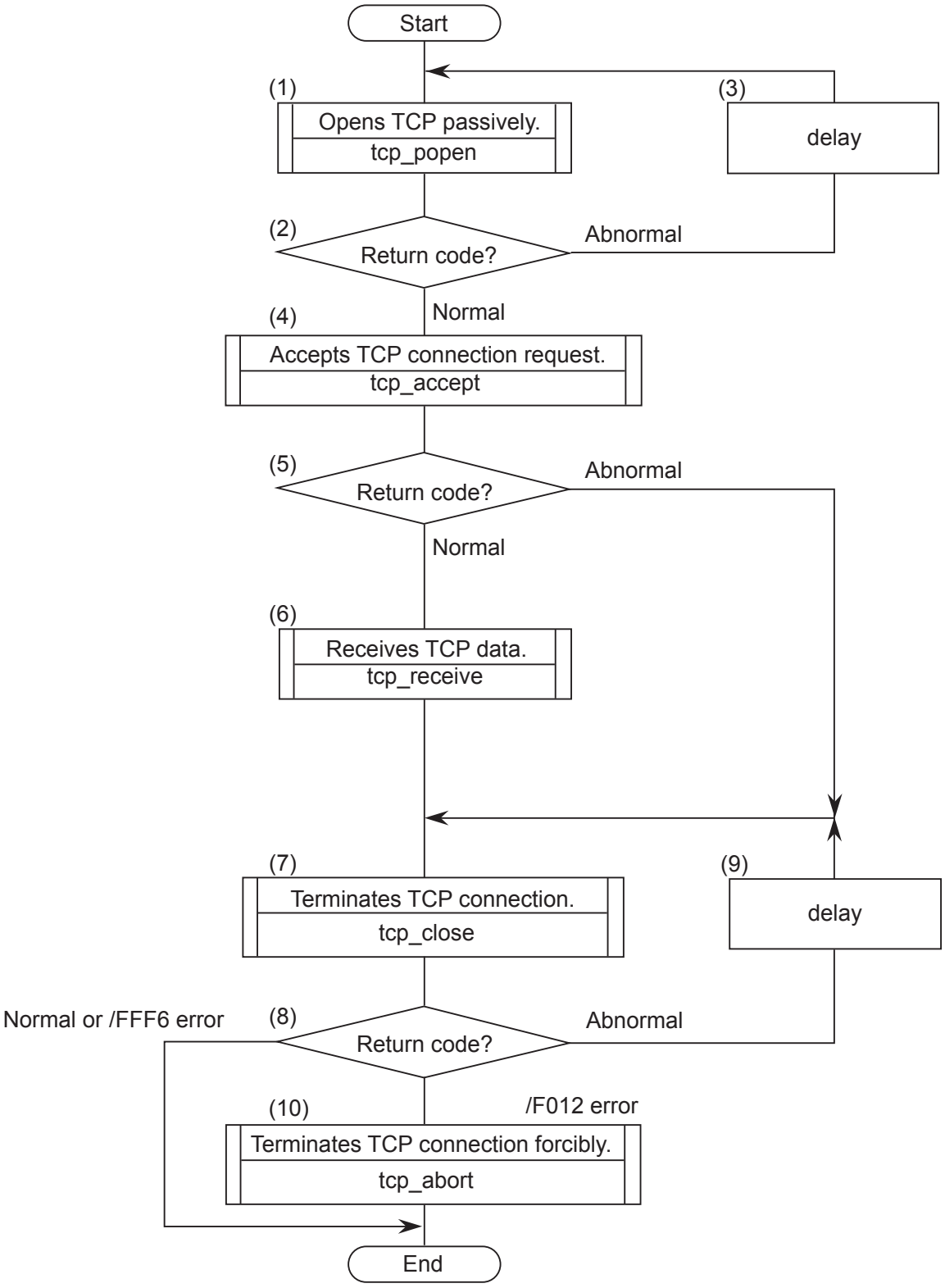


Figure 5-6 CPU01 Program Flowchart

■ Flowchart explanation

(1) Register a socket with port number 10000 and set the socket into passive state.
(2) The registered socket ID is released as a return code. If the return code is positive, it indicates that the socket has been registered successfully.
(3) If the return code points to an error in (2), issue a delay macro and repeat (1) and (2).
(4) Accept a connection request originating from CPU02.
(5) Test the return code for validity.
(6) Get the data sent from CPU02 into the receive buffer.
(7) Terminate the connection.
(8) Test the return code to see if the connection has terminated successfully or not. Error code /FFF6 indicates the connection has already closed after terminating successfully.
(9) If the return code points to an error in (8), issue a delay macro and repeat (7) and (8).
(10) If a socket driver timeout error (error code: /F012) is encountered in (8), force the connection to terminate.

■ Sample program

```
#define   TCP_POPEN       0x874106L       /* tcp_popen( )   starting address (main)*/
#define   TCP_ACCEPT      0x87410CL       /* tcp_accept( )  starting address (main)*/
#define   TCP_CLOSE       0x874112L       /* tcp_close( )   starting address (main)*/
#define   TCP_RECEIVE     0x874136L       /* tcp_receive( ) starting address (main)*/
#define   TCP_ABORT       0x87411EL       /* tcp_abort( )   starting address( main)*/
#define   IPADDR          0xC0010002L     /* IP address of remote station        */
#define   RBUFADDR        0x01006000L     /* Starting address of receive buffer */

struct   popen_p {
     long       dst_ip;                    /* IP address of remote station        */
     short      dst_port;                  /* Port number of remote station       */
     short      src_port;                  /* Port number of local station        */
     char       listennum;                 /* Fixed at 0                          */
     char       ttl;                       /* Time to live                        */
};

struct   accept_p{
  short     s_id;                          /* Socket ID                           */
};

struct   receive_p{
  short     s_id;                          /* Socket ID                           */
  short     len;                           /* Buffer length                       */
  char      *buf;                          /* Starting address of buffer          */
  long      tim;                           /* Receive wait time (ms)              */
};

struct   close_p{
  short     s_id;                          /* Socket ID                           */
};

struct   abort_p{
  short     s_id;                          /* Socket ID                           */
};
/*************************/
/*   task2: Server (CPU01)   */
/*************************/
main()
{
  register      short ( *tcp_popen ) ( );
  register      short ( *tcp_accept ) ( );
  register      short ( *tcp_receive ) ( );
  register      short ( *tcp_close ) ( );
  register      short ( *tcp_abort ) ( );
  long          time;
  short         rtn;
  struct        popen_p         popen;
  struct        accept_p        accpt;
  struct        receive_p       recv;
  struct        close_p         close;
```

```
struct      abort_p             abort;
 while( 1 ) {
        popen.dst_ip          = IPADDR;            /* IP address of remote station          */
        popen.dst_port        = 10000;             /* Port number of remote station         */
        popen.src_port        = 10000;             /* Port number of local station          */
        popen.listennum       = 0;                 /* Fixed at 0                            */

        popen.ttl   = 0;                           /* Time to live                          */
        tcp_popen= ( short (*) () ) TCP_POPEN;
        rtn         = (tcp_popen) (&popen);        /* Return code normal?                   */
        if( rtn > 0 ) {
           break;
        }
        time = 100;                                /* Issue of 100-ms Delay macro           */
        delay( &time);
 }
 accpt.s_id        = rtn;                          /* Socket ID                             */
 tcp_accept        = ( short (*) ( ) ) TCP_ACCEPT;
 rtn               =(tcp_accept) (&accpt);         /* Accepts TPC connection request.       */
 recv.s_id         =rtn;                           /* Socket ID                             */
 if( rtn > 0 ) {                                   /* Return code normal?                   */
   recv.len = 1024;                                /* Receive buffer length(bytes)          */
   recv.buf          = ( char *)RBUFADDR;          /* Starting address of receive buffer    */
   recv.tim = 60000;                               /* Receive wait time(ms)                 */
   tcp_receive = (short (*) ( ) )TCP_RECEIVE;
   rtn               = (tcp_receive) (&recv);      /* Receives TCP                          */
   close.s_id        = recv.s_id;                  /* Socket ID                             */
 }   else   {
         close.s_id = accpt.s_id;                  /* Socket ID                             */
 }
 while( 1 ) {
   tcp_close = ( short (*) ( ) ) TCP_CLOSE;
   rtn = (tcp_close) (&close);                     /* Terminates TCP connection.            */
   if( rtn == 0 || rtn == ( short )0xFFF6 ) {
      break;
   }
   else if ( rtn == ( short )0xF012 ) {
      tcp_abort = ( short (*) ( ) ) TCP_ABORT;
      rtn = (tcp_abort) (&abort);                  /* Terminates TCP connection forcibly    */
      break;
   }
   time = 100;                                     /* Issue of 100-ms Delay macro           */
   delay( &time);
 }
 return;
}
```

5.5.3 CPU02 program flowchart and sample program

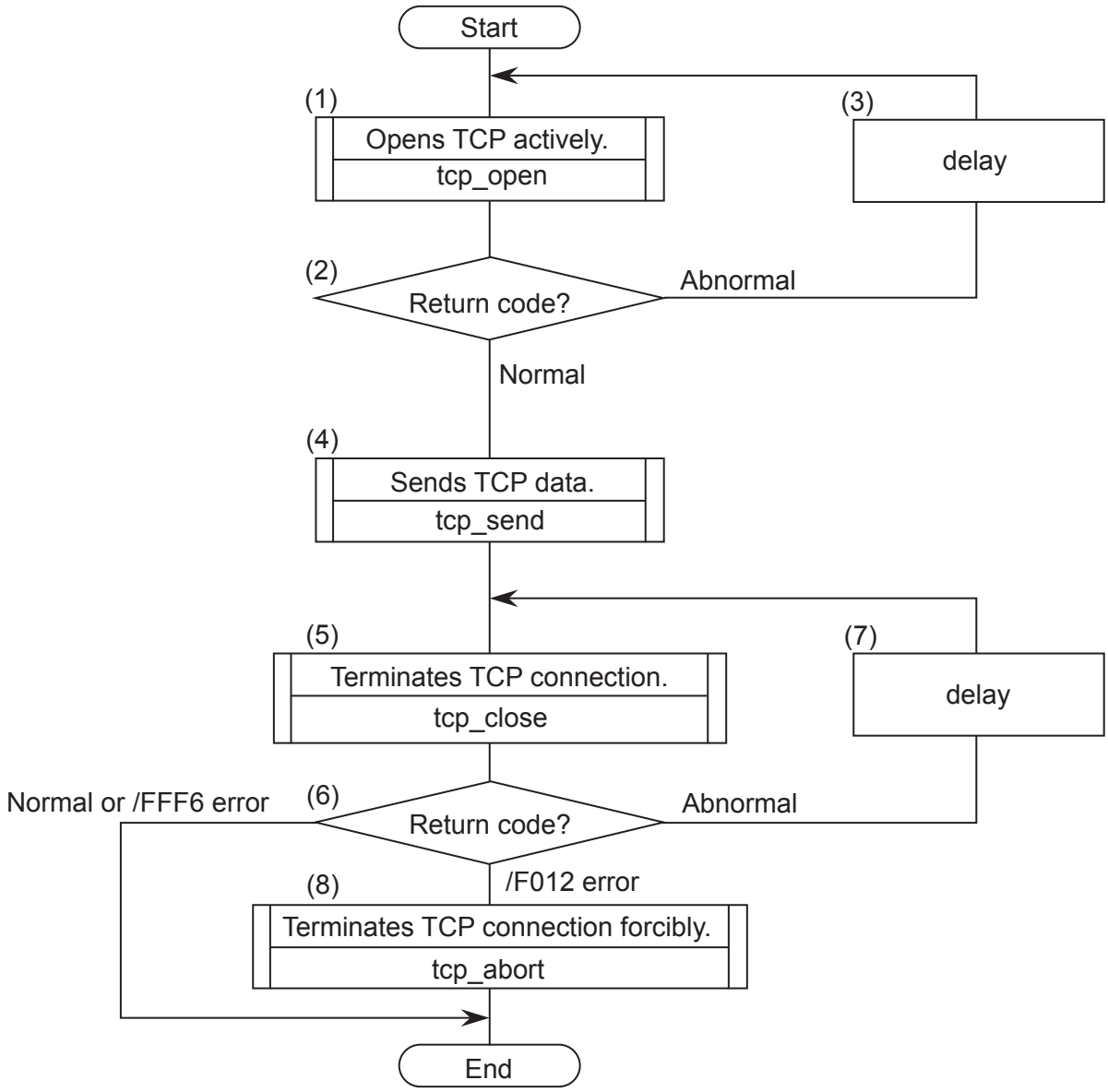


Figure 5-7 CPU02 Program Flowchart

■ Flowchart explanation

(1) Register a socket with port number 10000 and set the socket into active state.

(2) The registered socket ID is released as a return code. If the return code is positive, it indicates that the socket has been registered successfully.

(3) If the return code points to an error in (2), issue a delay macro and repeat (1) and (2).

(4) Send data from the send buffer to CPU01.

(5) Terminate the connection.

(6) Test the return code to see if the connection has terminated successfully or not. Error code /FFF6 indicates the connection has already closed after terminating successfully.

(7) If the return code points to an error in (6), issue a delay macro and repeat (5) and (6).

(8) Since no response is returned from the remote station, force the connection to terminate.

■ Sample program

```
#define   TCP_OPEN        0x874100L       /* tcp_open( )    starting address (main)*/
#define   TCP_CLOSE       0x874112L       /* tcp_close( )   starting address (main)*/
#define   TCP_SEND        0x874130L       /* tcp_send( )    starting address (main)*/
#define   TCP_ABORT       0x87411EL       /* tcp_abort( )   starting address (main)*/
#define   IPADDR          0xC0010001L     /* IP address of remote station        */
#define   SBUFADDR        0x01006000L     /* Starting address of send buffer     */

struct   open_p {
     long       dst_ip;                   /* IP address of remote station        */
     short      dst_port;                 /* Port number of remote station       */
     short      src_port;                 /* Port number of local station        */
     char       notuse;                   /* Unused(0)                           */
     char       ttl;                      /* Time to live                        */
  };

struct   send_p{
  short     s_id;                         /* Socket ID                           */
  short     len;                          /* Send data length(bytes)             */
  char      *buf;                         /* Starting address of send data       */
};

struct   close_p{
  short     s_id;                         /* Socket ID                           */
};

struct   abort_p{
  short     s_id;                         /* Socket ID                           */
};
/***************************/
/*     task3: Client (CPU02)    */
/***************************/
main()
{
  register      short ( *tcp_open ) ( );
  register      short ( *tcp_send ) ( );
  register      short ( *tcp_close ) ( );
  register      short ( *tcp_abort ) ( );
  long          time;
  short         rtn;
  struct        open_p          open;
  struct        send_p          send;
  struct        close_p         close;
  struct        abort_p         abort;

     while( 1 ) {
          open.dst_ip         = IPADDR;       /* IP address of remote station        */
          open.dst_port       = 10000;        /* Port number of remote station       */
```

```
        open.src_port        = 10000;          /* Port number of local station        */
        open.notuse          = 0;              /* Unused                              */
        open.ttl     = 0;                      /* Time to live                        */
        tcp_open     = ( short (*) () ) TCP_OPEN;
        rtn          = (tcp_open) (&open);     /* Opens TCP actively.                 */
        if( rtn > 0 ) {                        /* Return code normal?                 */
           break;
        }
        time = 100;                            /* Issue of 100-ms Delay macro         */
        delay( &time);
  }
  send.s_id          = rtn;                    /* Socket ID                           */
  send.len           =1024;                    /* Send data length(bytes)             */
  send.buf           = ( char *) SBUFADDR;     /* Starting address of send data       */
  tcp_send           = (short (*) ( ) ) TCP_SEND;
  rtn                =(tcp_send) (&send);      /* Sends TCP data.                     */
  close.s_id         = send.s_id;              /* Socket ID                           */
  while( 1 ) {
    tcp_close = ( short (*) ( ) ) TCP_CLOSE;
    rtn = (tcp_close) (&close);                /* Terminates TCP connection.          */
    if( rtn == 0 || rtn == ( short )0xFFF6 ) {
      break;
    }
    else if ( rtn == ( short )0xF012 ) {
      tcp_abort = ( short (*) ( ) ) TCP_ABORT;
      rtn = (tcp_abort) (&abort);              /* Terminates TCP connection forcibly  */
      break;
  }
  time = 100;                                  /* Issue of 100-ms Delay macro         */
  delay( &time);
 }
 return;
}
```

5.6 Inter-CPU Continuous Communication Sample Program

5.6.1 System configuration and program configuration

Figure 5-8 shows the system configuration. The program has the ET.NET module of CPU01 and that of CPU02 connected with each other by a logical line and allows 1024 bytes of data to be transmitted between CPU02 and CPU01.
To run this program, be sure to invoke a user program from CPU01.

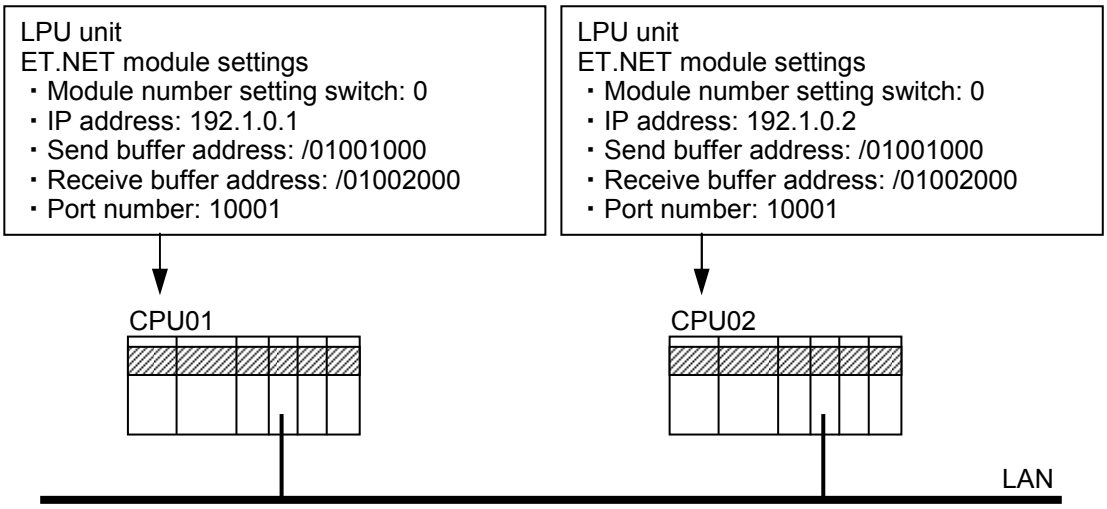


Figure 5-8 A Sample System Configuration for the Inter-CPU Continuous Communication Program

5.6.2 CPU01 program flowchart and sample program

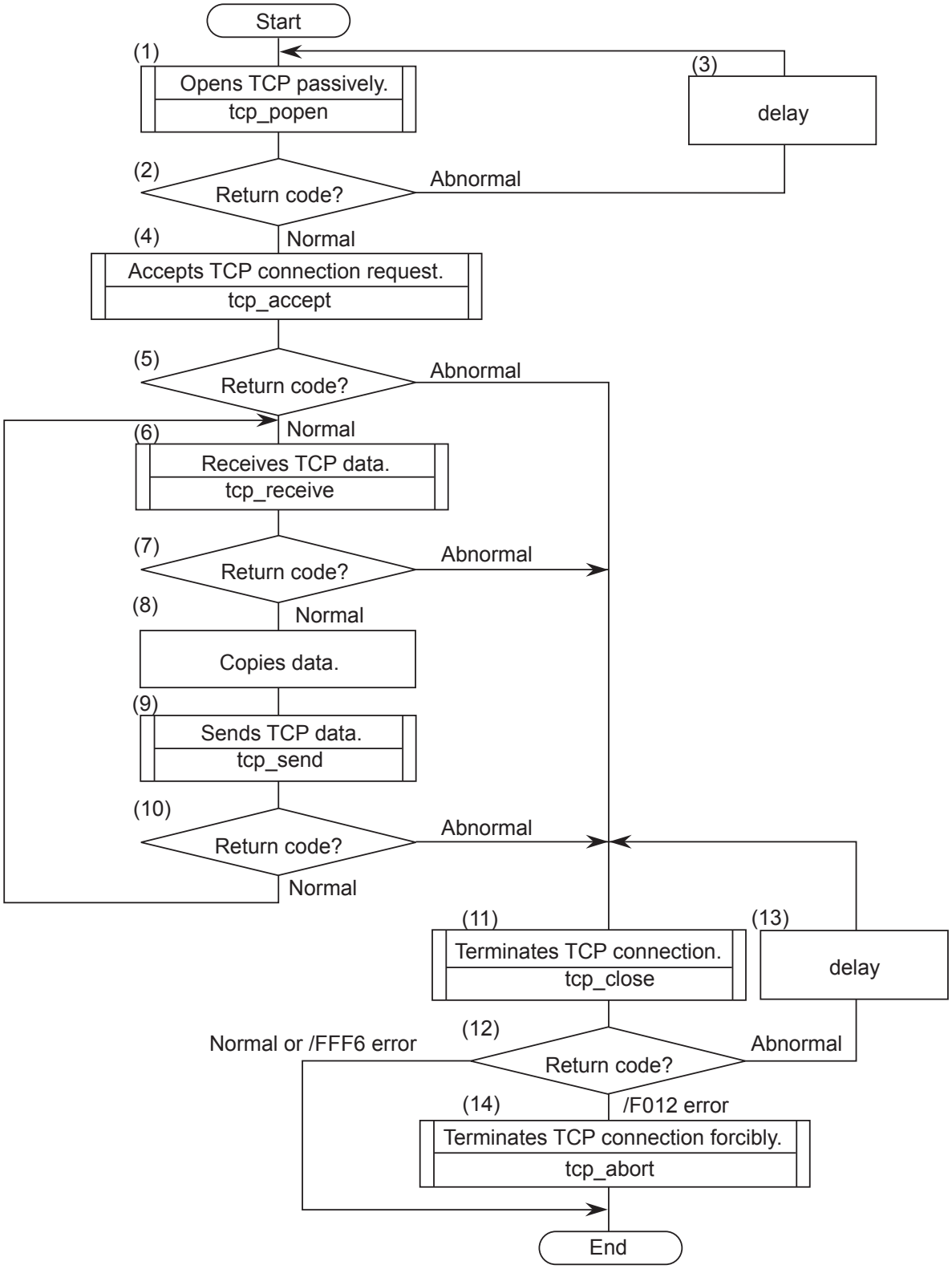


Figure 5-9 CPU01 Program Flowchart

■ Flowchart explanation

(1) Register a socket with port number 10001 and set the socket into passive state.
(2) The registered socket ID is released as a return code. If the return code is positive, it indicates that the socket has been registered successfully.
(3) When the return code points to an error in (2), issue a delay macro and repeat (1) and (2).
(4) Accept a connection request originating from CPU02.
(5) Test the return code for validity.
(6) Get the data sent from CPU02 into the receive buffer.
(7) When the return code points to an error or no received data is available, carry out (11).
(8) Copy the data in the receive buffer to the send buffer.
(9) Send the data in the send buffer to CPU02.
(10) Test the return code to see if the transmission has completed successfully or not. If it has, repeat (6) to (10).
(11) Terminate the connection.
(12) Test the return code to see if the connection has terminated successfully or not. Error code /FFF6 indicates the connection has already closed after terminating successfully.
(13) When the return code points to an error in (12), issue a delay macro and repeat (11) and (12).
(14) Since no response is returned from the remote station, force the connection to terminate.

■ Sample program

```
#define   TCP_POPEN      0x874106L      /* tcp_popen( )   starting address (main) */
#define   TCP_ACCEPT     0x87410CL      /* tcp_accept( )  starting address (main) */
#define   TCP_RECEIVE    0x874136L      /* tcp_receive( ) starting address (main) */
#define   TCP_SEND       0x874130L      /* tcp_send( )    starting address (main) */
#define   TCP_CLOSE      0x874112L      /* tcp_close( )   starting address (main) */
#define   TCP_ABORT      0x87411EL      /* tcp_abort( )   starting address (main) */
#define   IPADDR         0xC0010002L    /* IP address of remote station           */
#define   SBUFADDR       0x01001000L    /* Starting address of send buffer        */
#define   RBUFADDR       0x01002000L    /* Starting address of receive buffer     */

struct   popen_p {
    long       dst_ip;                   /* IP address of remote station           */
    short      dst_port;                 /* Port number of remote station          */
    short      src_port;                 /* Port number of local station           */
    char       listennum;                /* Fixed at 0                             */
    char       ttl;                      /* Time to live                           */
};

struct   accept_p{
 short     s_id;                         /* Socket ID                              */
};

struct   receive_p{
 short     s_id;                         /* Socket ID                              */
 short     len;                          /* Buffer length                          */
 char      *buf;                         /* Starting address of buffer             */
 long      tim;                          /* Receive wait time                      */
};

struct   send_p{
 short     s_id;                         /* Socket ID                              */
 short     len;                          /* Send data length (bytes)               */
 char      *buf;                         /* Starting address of send data          */
};

struct   close_p{
 short     s_id;                         /* Socket ID                              */
};

struct   abort_p{
 short     s_id;                         /* Socket ID                              */
};
/***************************/
/*     task2: Server (CPU01)    */
/***************************/
main()
{
 register       short ( *tcp_popen ) ( );
 register       short ( *tcp_accept ) ( );
 register       short ( *tcp_receive ) ( );
 register       short ( *tcp_send ) ( );
 register       short ( *tcp_close ) ( );
 register       short ( *tcp_abort ) ( );
```

```
  long        time;
  short       rtn, i;
  char        *sbuf, *rbuf;
  struct      popen_p      popen;
  struct      accept_p     accpt;
  struct      receive_p    recv;
  struct      send_p       send;
  struct      close_p      close;
  struct      abort_p      abort;

while( 1 ) {
        popen.dst_ip          = IPADDR;          /* IP address of remote station          */
        popen.dst_port        = 10001;           /* Port number of remote station         */
        popen.src_port        = 10001;           /* Port number of local station          */
        popen.listennum       = 0;               /* Fixed at 0                            */

        popen.ttl     = 0;                       /* Time to live                          */
        tcp_popen     = ( short (*) () ) TCP_POPEN;
        rtn           = (tcp_popen) (&popen);    /* Return code normal?                   */
        if( rtn > 0 ) {
          break;
        }
        time = 100;                              /* Issue of 100-ms Delay macro           */
        delay( &time);
}
accpt.s_id          = rtn;                       /* Socket ID                             */
tcp_accept          = ( short (*) ( ) ) TCP_ACCEPT;
rtn                 =(tcp_accept) (&accpt);      /* Accepts TCP connection request.       */
if( rtn > 0 ) {                                  /* Return code normal?                   */
recv.s_id           =rtn;                        /* Socket ID                             */
while( 1 ) {
   recv.len = 1024;                              /* Receive buffer length (bytes)         */
   recv.buf         = ( char *)RBUFADDR;         /* Starting address of receive buffer    */
   recv.tim = 60000;                             /* Receive wait time (ms)                */
   tcp_receive = (short (*) ( ) )TCP_RECEIVE;
   rtn              = (tcp_receive) (&recv);     /* Receives TCP data.                    */
   if( rtn < 0 ) {                               /* Return code abnormal?                 */
      break;
   }
   sbuf = ( char *)SBUFADDR;
   rbuf = ( char *)RBUFADDR;
   for( i = 0 ; i < 1024 ; i++ ) {
      sbuf[ i ] = rbuf[ i ];
   }
   send.s_id = recv.s_id;                        /* Socket ID                             */
   send.len         =1024;                       /* Send data length (bytes)              */
   send.buf         =( char *)SBUFADDR;          /* Starting address of send data         */
   tcp_send         = ( short (*) ( ) ) TCP_SEND;
   rtn              = (*tcp_send) (&send);       /* Sends TCP data.                       */
   if ( rtn < 0 ) {                              /* Return code abnormal?                 */
      break;
   }
}
   close.s_id       = recv.s_id;                 /* Socket ID                             */
}   else   {
```

```
        close.s_id = accpt.s_id;                    /* Socket ID                          */
    }
    while( 1 ) {
        tcp_close = ( short (*) ( ) ) TCP_CLOSE;
        rtn = (tcp_close) (&close);                 /* Terminates TCP connection.         */
        if( rtn == 0 || rtn == ( short )0xFFF6 ) {
            break;
        }
        else if ( rtn == ( short )0xF012 ) {
            tcp_abort = ( short (*) ( ) ) TCP_ABORT;
            rtn = (tcp_abort) (&abort);             /* Terminates TCP connection forcibly */
            break;
        }
        time = 100;                                 /* Issue of 100-ms Delay macro        */
        delay( &time);
    }
    return;
}
```

5.6.3 CPU02 program flowchart and sample program

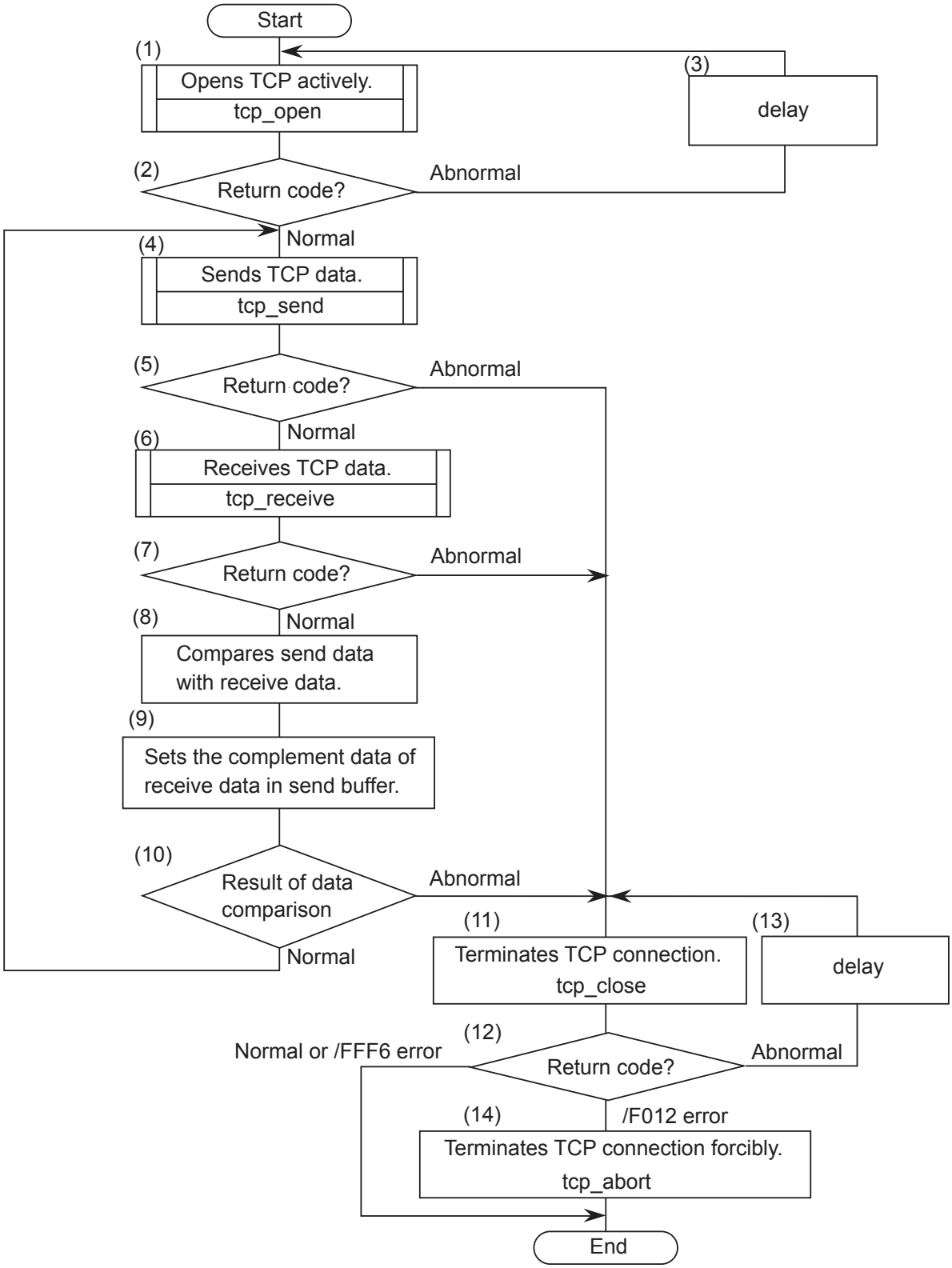


Figure 5-10 CPU02 Program Flowchart

■ Flowchart explanation

(1) Register a socket with port number 10001 and set the socket into active state.
(2) The registered socket ID is released as a return code. If the return code is positive, it indicates that the socket has been registered successfully.
(3) When the return code points to an error in (2), issue a delay macro and repeat (1) and (2).
(4) Send the data in the send buffer to CPU01.
(5) Test the return code to see if the transmission has completed successfully or not.
(6) Get the data sent from CPU01 into the receive buffer.
(7) Test the return code to see if the transmission has completed successfully or not.
(8) Compare the data in the send buffer and that in the receive buffer at the location.
(9) Copy a reversed version of the data in the receive buffer to the send buffer.
(10) Test the comparison and, if the data has been transmitted successfully, repeat (4) to (10).
(11) Terminate the connection.
(12) Test the return code to see if the connection has terminated successfully or not. Error code /FFF6 indicates the connection has already closed after terminating successfully.
(13) When the return code points to an error in (12), issue a delay macro and repeat (11) and (12).
(14) Since no response is returned from the remote station, force the connection to terminate.

■ CPU02 sample program

```
#define     TCP_OPEN        0x874100L     /* tcp_open( )     starting address (main) */
#define     TCP_CLOSE       0x874112L     /* tcp_close( )    starting address (main) */
#define     TCP_SEND        0x874130L     /* tcp_send( )     starting address (main) */
#define     TCP_RECEIVE     0x874136L     /* tcp_receive( )  starting address (main) */
#define     TCP_ABORT       0x87411EL     /* tcp_abort( )    starting address (main) */
#define     IPADDR          0xC0010001L   /* IP address of remote station          */
#define     SBUFADDR        0x01001000L   /* Starting address of send buffer       */
#define     RBUFADDR        0x01002000L   /* Starting address of receive buffer    */

struct  open_p {
    long        dst_ip;                   /* IP address of remote station          */
    short       dst_port;                 /* Port number of remote station         */
    short       src_port;                 /* Port number of local station          */
    char        notuse;                   /* Unused (0)                            */
    char        ttl;                      /* Time to live                          */
  };

struct  send_p{
    short       s_id;                     /* Socket ID                             */
    short       len;                      /* Send data length (bytes)              */
    char        *buf;                     /* Starting address of send data         */
};

struct  receive_p{
    short       s_id;                     /* Socket ID                             */
    short       len;                      /* Buffer length                         */
    char        *buf;                     /* Starting address of buffer            */
    long        tim;                      /* Receive wait time (ms)                */
};

struct  close_p{
    short       s_id;                     /* Socket ID                             */
};

struct  abort_p{
    short       s_id;                     /* Socket ID                             */
};
/***************************/
/*    task3: Client (CPU02)    */
/***************************/
main()
{
  register      short ( *tcp_open ) ( );
  register      short ( *tcp_send ) ( );
  register      short ( *tcp_receive ) ( );
  register      short ( *tcp_close ) ( );
  register      short ( *tcp_abort ) ( );
  long          time;
  short         rtn, i, cerr_flg;
  char          *sbuf, *rbuf;
  struct        open_p          open;
  struct        send_p          send;
  struct        receive_p       recv;
```

```
  struct        close_p        close;
  struct        abort_p        abort;

   sbuf = ( char *) SBUFADDR;
   for( i = 0 ; i < 1024 ; i++ ) {
        sbuf[ i ] = 0x55;
   }

while( 1 ) {
        open.dst_ip    = IPADDR;                    /* IP address of remote station        */
        open.dst_port  = 10001;                     /* Port number of remote station       */
        open.src_port  = 10001;                     /* Port number of local station        */
        open.notuse    = 0;                         /* Unused                              */
        open.ttl       = 0;                         /* Time to live                        */
        tcp_open       = ( short (*) () ) TCP_OPEN;
        rtn            = (tcp_open) (&open);        /* Opens TCP actively.                 */
        if( rtn > 0 ) {                             /* Return code normal?                 */
        break;
        }
        time = 100;                                 /* Issue of 100-ms Delay macro         */
        delay( &time);
   }
   send.s_id           = rtn;                       /* Socket ID                           */
   recv.s_id           = rtn;                       /* Socket ID                           */
   while( 1 ) {
      send . len       = 1024 ;                     /* Send data length (bytes)            */
      send . buf       = ( char * ) SBUFADDR ;      /* Starting address of send data       */
      tcp_send         = ( short (*) ( ) ) TCP_SEND;
      rtn              = (tcp_send) (&send);        /* Sends TCP data.                     */
      if ( rtn < 0 ) {                              /* Return cond abnormal?               */
         break;
      }
      recv . len       = 1024;                      /* Receive buffer length (bytes)       */
      recv . buf       = ( char *) RBUFADDR ;       /* Starting address of receive buffer  */
      recv . tim       = 60000 ;                    /* Receive wait time (ms)              */
      tcp_receive      = ( short (*) ( ) ) TCP_RECEIVE;
      rtn              = (tcp_receive) ( &recv );   /* Receive TCP data.                   */
      if( rtn < 0 ) {                               /* Return cond abnormal?               */
         break;
      }
      cerr_flg = 0;                                 /* Clears compare error flag.          */
      sbuf             = ( char *)SBUFADDR;         /* Starting address of send buffer     */
      rbuf             = ( char *)RBUFADDR;         /* Starting address of receive buffer  */
      for( i = 0 ; i < 1024 ; i++ ) {
         if( sbuf[ i ] != rbuf[ i ] ) {
            cerr_flg = 1;                           /* Sets compare error flag.            */
            break;
         }
         sbuf[ i ] = ~rbuf[ i ];                    /* Sets complement                     */
      }
      if( cerr_flg == 1 ) {                         /* Compare error?                      */
         break;
      }
   }
   close . s_id = send . s_id;                      /* Socket ID                           */
```

```
    while( 1 ) {
        tcp_close = ( short (*) ( ) ) TCP_CLOSE;
        rtn = (tcp_close) (&close);                    /*  Terminates TCP connection.          */
        if( rtn == 0 || rtn == ( short )0xFFF6 ) {
            break;
        }
        else if ( rtn == ( short )0xF012 ) {
            tcp_abort = ( short (*) ( ) ) TCP_ABORT;
            rtn = (tcp_abort) (&abort);                /*  Terminates TCP connection forcibly  */
            break;
        }
        time = 100;                                    /*  Issue of 100-ms Delay macro         */
        delay( &time);
    }
    return;
}
```

6 USER GUIDE

6.1 Recommended Network Components

The LQE720 is a standard product conformed to the global standard of IEEE802.3. It may happen, however, that the LQE720 does not function successfully when used in conjunction with certain network components conforming to the same standard. To avoid this inconvenience, use network components of the make recommended by us to connect to the LQE720.
Table 6-1 give listings of the kinds of network components recommended by us.
Ethernet® specifications are available in two versions: IEEE802.3 and original Ethernet®.
Equipment made to the original Ethernet® specifications cannot be connected to the LQE720.

Table 6-1 Network Component List

No.	Product name	Manufacturer	Model	Remarks
①	ET.NET	Hitachi, Ltd.	LQE720	
②	Hub	Hitachi, Ltd.	H-7612-90	Switching hub
③	Twisted pair cable	Hitachi Cable, Ltd.	HUTP-CAT5E-4P***	*** is the cable length.

6.2 System Configuration

Data communication between Ethernet network devices becomes possible if you connect such devices to a hub, as shown in Figure 6-1. To connect Ethernet devices to the hub, use twisted-pair cables.

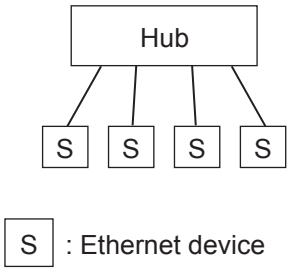


Figure 6-1 Hub and Connected Devices

■ Configuration using 10-Mbps dedicated hubs

- Where 10-Mbps dedicated hubs are used to connect Ethernet devices together in a multi-stage form, the following rule must be observed: no more than four hubs, and no more than five link segments, may be involved in the communication path between any two Ethernet devices in the multi-stage connection.

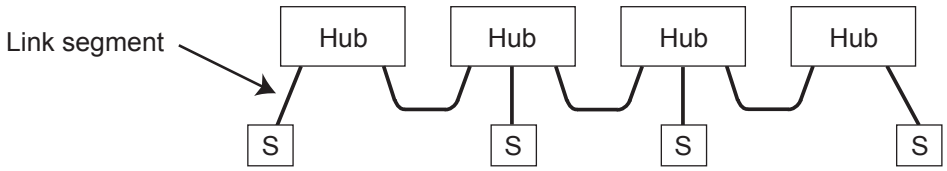


Figure 6-2 A Sample Configuration Using 10-Mbps Dedicated Hubs Only

■ Configurations using 100-Mbps hubs

- If a 100-Mbps hub of class 1 is used to connect Ethernet devices, it may not be connected in a multi-stage form. In this case, connect the hub and Ethernet devices in the following way:

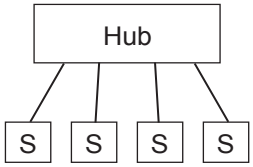


Figure 6-3 A Sample Configuration Using a 100-Mbps Hub of Class 1

- If a 100-Mbps hub of class 2 is used to connect Ethernet devices, it may be connected in a multi-stage form, although the maximum number of 100-Mbps hubs of the same class that can be connected together is 2. In this case, connect the hubs and Ethernet devices in the following way (the maximum allowable length of cable connecting the two hubs together is 5 m):

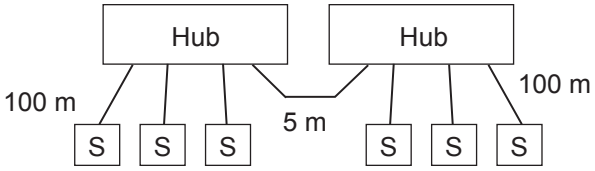


Figure 6-4 A Sample Configuration Using 100-Mbps Hubs of Class 2

- If 100-Mbps switching hubs are used to connect Ethernet devices together in a multi-stage cascade form, there is basically no limitation on the number of stages used in the multi-stage connection. In this case, consult the switching hub manual and connect them according to the instructions given in the manual.

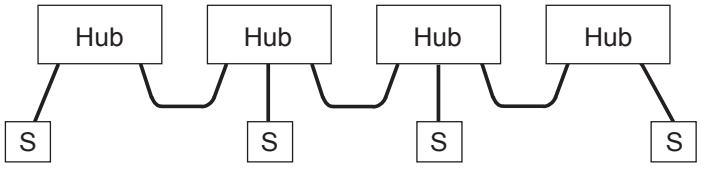
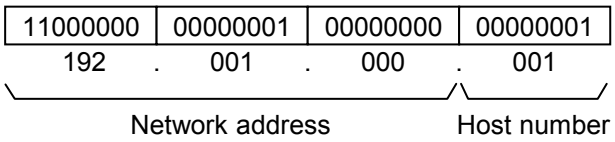


Figure 6-5 A Sample Configuration Using 100-Mbps Switching Hubs

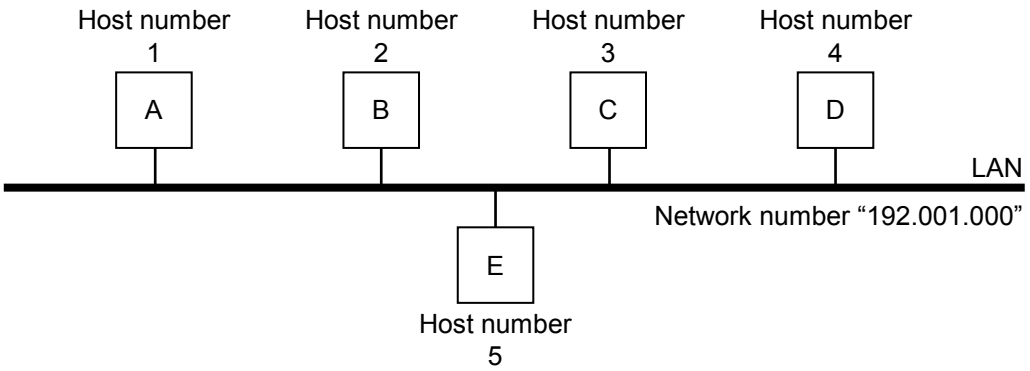
NOTICE
The only port setting that is supported by the ET.NET module (model LQE720) is auto-negotiation. Do not use 100-Mbps full-duplex setting for the port of the switching hub. Disregarding this rule may result in a failure of data communication when the load on the communication line builds up.

The IP address is divided into four 8-bit fields, separated by dots from each other and represented in decimal. For example, in Class C, the address is represented as follows:



Networks are identified by their network number and each host in the network is identified by its uniquely defined host number. Class C is selected when there are 200 or more hosts in a given network. For example, if the network number used is “192.001.000” and there are five hosts connected to the network, then the IP address of each station is set as shown below.

Station A: 192.001.000.001
Station B: 192.001.000.002
Station C: 192.001.000.003
Station D: 192.001.000.004
Station E: 192.001.000.005



There are two special IP addresses used: one is an IP address whose host number bits are all set to 0 and, as a whole, represents the entire network; and the other is an IP address whose host number bits are all set to 1 and, as a whole, is used as a broadcast address. The broadcast address is used when data needs to be transmitted to all the stations in the network at once -- this can be accomplished by UDP/IP communication.

6.3 System Definition Information

Set the following ② and ③ information for ET.NET (LQE520). To connect a station to another network through a router, define item ④, too. Do not set ② in duplicate with another station. Item ③ needs to have a consistent value throughout one single subnetwork.

① Physical address: An original number is set for each ET.NET.
② IP address: Define these items for each ET.NET by using the ET.NET system tool
③ Subnetwork mask: Define these items for each ET.NET by using the ET.NET system tool
④ Route information: Define this item when connecting a station to another network through a router. The item can be set by the ET.NET system tool or by a user program.

6.3.1 Physical address

A 48-bit physical address is assigned to each ET.NET.
This is a unique address; the user cannot change it.

6.3.2 IP address

The IP address used for TCP/IP and UDP/IP is a 32-bit logical address. An IP address consists of a network number and a host number. There are three types of address assignment depending on the number of hosts.

(i) Class A (The high-order one bit of the network number is set to “0”.)

Network number (8 bit)	Host number (24 bits)

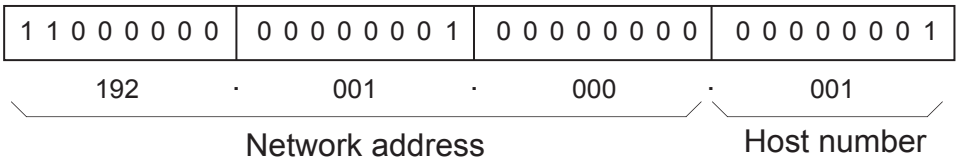
(ii) Class B (The high-order two bits of the network number are set to “10” in binary.)

Network number (16 bit)	Host number (16 bits)

(iii) Class C (The high-order three bits of the network number are set to “110” in binary.)

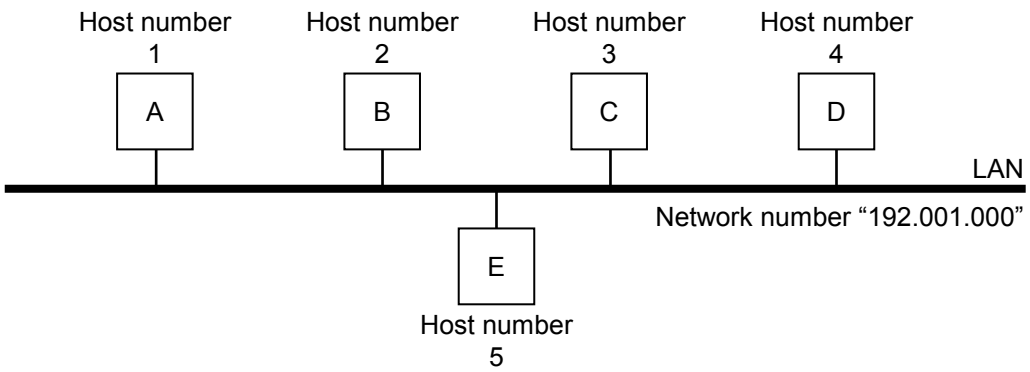
Network number (24 bit)	Host number (8 bits)

An IP address is represented in decimal; the eight-bit values are delimited from each other by a period (“.”). For example, an IP address of class C is represented as shown below.



A network is determined by a network number. Define a unique host number for each host in the network. If the number of hosts in a network is 200 or less, select class C. For example, if “192.001.000” is set as a network number and five hosts are connected to the network, set the IP address of each station as follows:

Station A: 192.001.000.001
Station B: 192.001.000.002
Station C: 192.001.000.003
Station D: 192.001.000.004
Station E: 192.001.000.005



There are two special IP addresses: one indicates the entire network by setting all bits of host number to 0, and the other is the broadcast address in which all bits of host number are set to 1. The broadcast address is used when data is sent to all stations belonging to the network. (In this case, send data by UDP/IP communication.)

6.3.3 Subnetwork mask

When splitting an IP address into subnetworks, define the boundary between subnetwork number and local host number by a subnetwork mask. If a subnetwork mask is used with other than the default value, the address is a the broadcast address as shown in the example below.

Example: For class B:

IP address	Subnetwork mask	Broadcast address
128.123.000.001	255.255.000.000	128.123.255.255
128.123.001.001	255.255.255.000	128.123.001.255

6.3.4 Route information

Route information must be defined if you want to connect a station to another network through a router. As the route information, the IP addresses of both the communication destination and router are registered in a pair.

(1) IP address of communication destination
For each communication destination, an IP address is registered. When multiple communication destinations exist in the same network, a network address may be set as a generic address. The host number of the IP address that has been set to “0” is used as the network address.

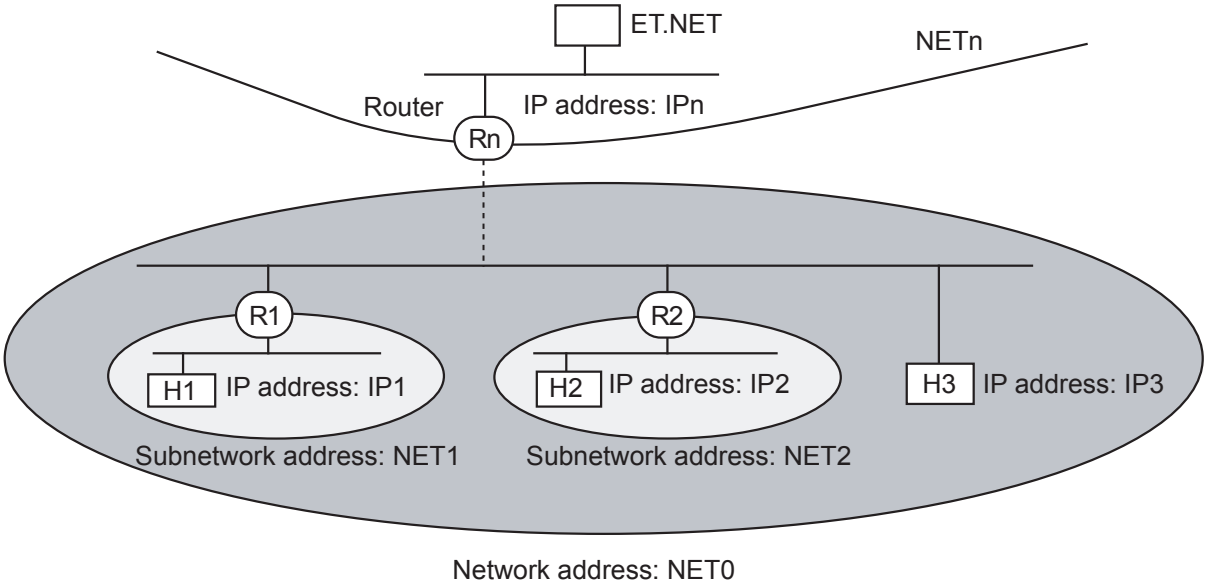
(2) IP address of router
The IP address of the router in the same network as the ET.NET module is registered. When multiple routers is involved in the communication route to the destination, register only the router in the same network as the ET.NET module.

The following two methods are available for setting route information.

- Setting in the socket handler route_add( ) in a C program
  - See “5.3.1 Socket handler function list”
- Setting by using the ET.NET or S10V ET.NET system.
  - See “4.3.4 Routing information”

A sample entry of routing information is described below.

- Route information registered for communication with host H1
  - IP address of router Rn: IPn
  - IP address of host H1: IP1
- Route information registered for communication with host H3
  - IP address of router Rn: IPn
  - IP address of host H3: IP3 or network address NET0



Restriction and limitation

- Up to 15 items of route information including both route_add( ) and tool settings can be registered.
- If the same setting is made by route_add( ) and the tool, the setting made by the latter has priority and that made by route_add( ) is invalidated. In this case, an error return code will be given back.
- The addresses that can be registered are IP and network addresses. No subnetwork address can be registered.
  This is because the ET.NET module recognizes route information as an IP address or network address but not as a subnetwork address. Even if a subnetwork address is registered, it is not recognized as an IP address, so no communication can be performed.

This Page Intentionally Left Blank

7 MAINTENANCE

7.1 Maintenance and Inspection

To keep the module running in optimal condition, it requires checks. Make checks daily or periodically (twice a year or more often).

Table 7-1 Maintenance and Inspection Items

Item	Point to check
Module appearance	Check the module case for cracks, flaws and other defects. Such defects can be a sign of breakage in the internal circuitry, causing the system to malfunction.
LED	Check to see if the module ERR LED has not glowed.
Loose mounting screws	Check the module and communications cable mounting screws for tightness. Give additional tightening to screws found loose. Loose screws could cause the system to malfunction and lead to burnouts after heating.
Cable covering status	Check cable coverings for defects. A cable covering out of position could cause the system to malfunction, incur electrical shock hazards, or develop short circuits, resulting in burnouts.
Dust adhesion	Check to see if the module has not caught dust. If dust is noticed, remove it with a vacuum cleaner or other apparatus. Dust could cause short circuits in its internal circuitry, resulting in burnouts.
Module replacement	Replacing the module without switching it off could cause damage to its hardware and software. Before replacing the module, switch it off first.
Connector status	Connectors might have their characteristics degraded to cause failures if their contacts catch dust or foreign matter. Cover connectors out of use with the protective cap supplied.

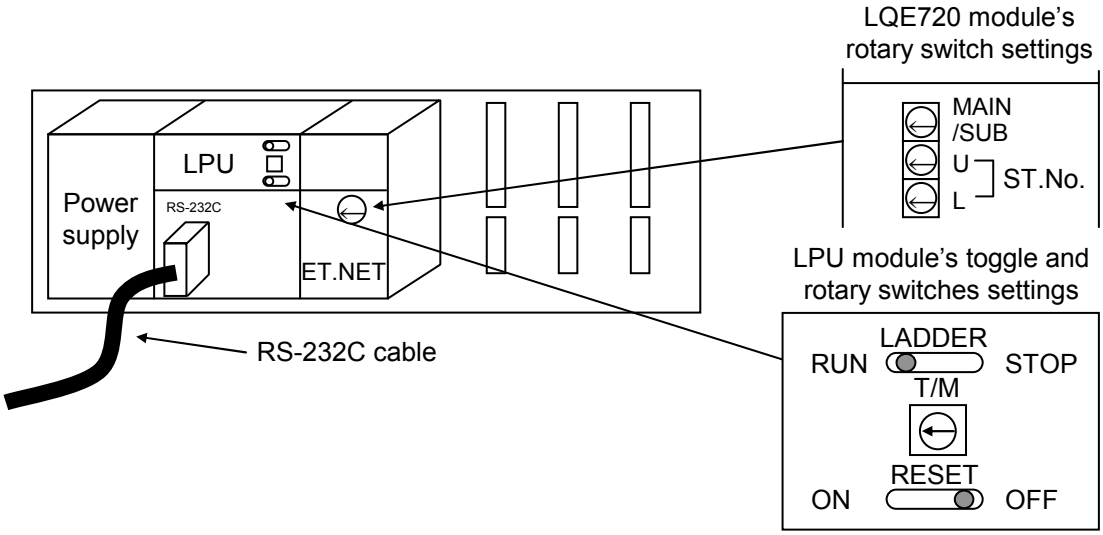
NOTICE

- Static electricity could cause damage to the module. Before handling the module allow static charges on the human body to discharge.
- Before replacing the module, switch it off to avoid electrical shock hazards and also to prevent it from being damaged or malfunctioning.

7.1.1 Replacing or adding on the module

- What you should get in preparation
  ① Personal computer (with Hitachi's S10V ET.NET System installed in it)
  ② RS-232C cable
  ③ New or add-on ET.NET module (LQE720)
  ④ Copies of the parameter values for the module to be replaced. (These copies are prepared for use in cases where the parameters are not accessible for some reason.)

- Replacement procedure
  ① Write down, on a piece of paper, the current settings of the rotary switches that are, as shown below, accessible at the front side of the ET.NET module to be replaced.
  ② Write down also the current settings of two switches, labeled LADDER (toggle switch) and T/M (rotary switch), respectively, that are, as shown below, accessible at the front side of the LPU module.



  ③ Connect the personal computer and the LPU module together with the RS-232C cable.
  ④ Start Hitachi's S10V ET.NET System and make a hand-written record of the currently used IP address. (If the existing parameters are not accessible for some reason, use the copies of their set values [item ④] that were obtained in preparation.)
  ⑤ Set the LPU module's LADDER switch in STOP position and turn off the power supply of the controller unit.
  ⑥ Remove the connecting cables from the ET.NET module to be replaced.
  ⑦ Replace the existing ET.NET module with the new one and set the new ET.NET module's rotary switches in the same way as you wrote down in Step ①.

⑧ Turn on the power supply of the controller unit. Then, set the same IP address as you recorded in Step ④, by using the S10V ET.NET System.
⑨ Check that the set IP address is identical to the one that was recorded in Step ④.
⑩ Turn off the power supply of the controller unit.
⑪ Remove the RS-232C cable from both the personal computer and LPU module, which were connected together in Step ③.
⑫ Connect to the new ET.NET module the connecting cables that you removed in Step ⑥.
⑬ Set the LPU module's LADDER and T/M switches in the same way as you wrote down in Step ②.
⑭ Turn on the power supply of the controller unit and check that the new ET.NET module is running normally.

- Add-on procedure

① Write down, on a piece of paper, the current settings of two switches, labeled LADDER (toggle switch) and T/M (rotary switch), respectively, that are accessible at the front side of the LPU module, the one that is installed in the controller unit in which you are adding on a ET.NET module.
② Ensure that your application system has been shut down. Then, set the LPU module's LADDER switch in STOP position and turn off the power supply of the controller unit.
③ Mount the add-on ET.NET module in place according to the instructions given under "3.3 Mounting the Module."
④ Set the add-on ET.NET module's rotary switches in such a way that a new module No. setting, which must be a sub-module No. setting, will not duplicate with the current rotary switch settings of the existing main ET.NET module.
⑤ Connect the personal computer and the LPU module together with the RS-232C cable. Then, turn on the power supply of the controller unit and set IP address for the add-on ET.NET module by using the S10V ET.NET System.
⑥ Turn off the power supply of the controller unit and connect the connecting cables to the add-on ET.NET module.
⑦ Set the LPU module's LADDER and T/M switches in the same way as you wrote down in Step ①.
⑧ Remove the RS-232C cable from both the personal computer and LPU module, which were connected together in Step ⑤.
⑨ Turn on the power supply of the controller unit and check that the add-on ET.NET module is running normally.

7.2 Troubleshooting

7.2.1 Procedure

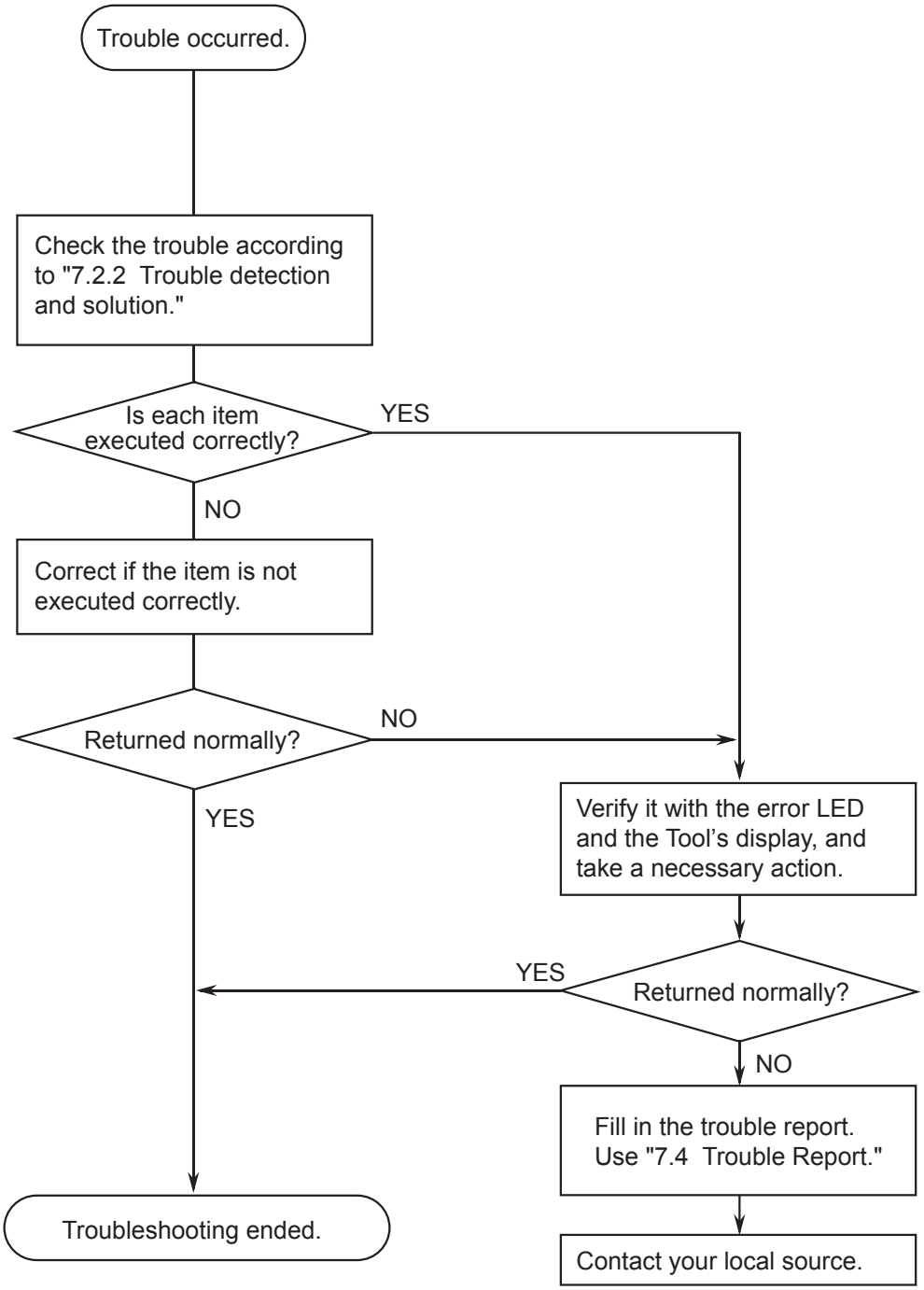
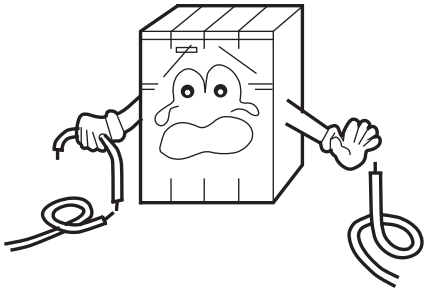


Figure 7-1 Troubleshooting Flow

7.2.2 Trouble detection and solution

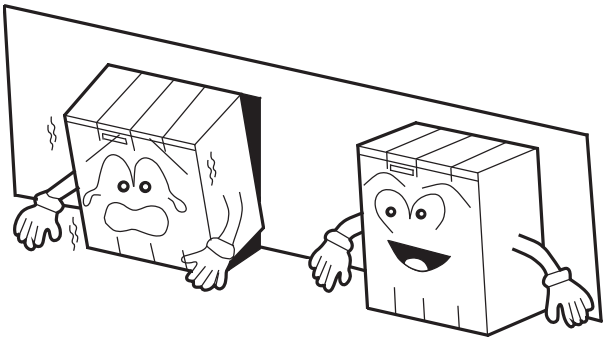
(1) Is the cabling correct?

- Check cables for disconnection or incorrect connection.
- Check that a cable with shielded ground wire is used as the transceiver cable.



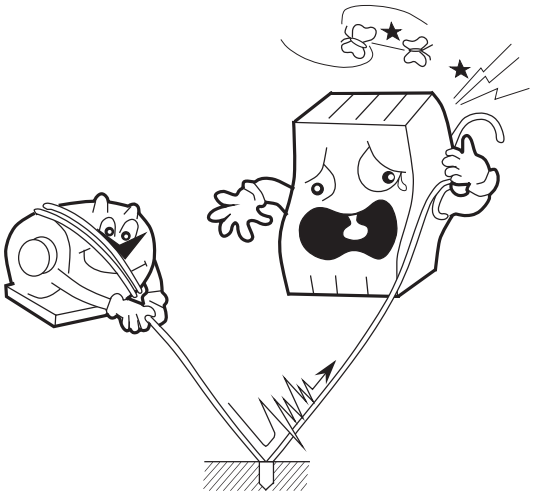
(2) Are the modules mounted correctly?

- Check that no set screws loosen.
- Check if a model LQE720 module is intermixed with a model LQE520 module among the same series of similar modules installed. The two different models may not be intermixed on the same mount base.



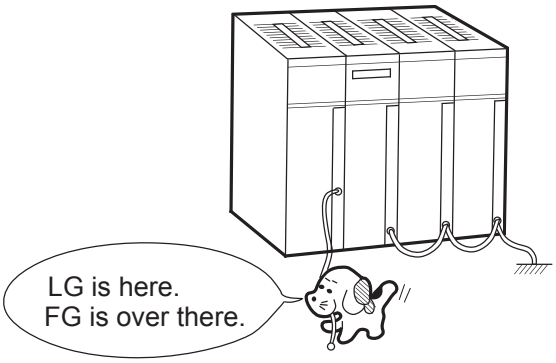
(3) Is grounding correct?

- Do not ground the ET.NET module in the same place where high-voltage equipment is grounded. They must be grounded in separate place.
- Perform grounding work conforming to Class D grounding or higher.



(4) Are LG and FG separated?

- Be sure to separate the LG from the FG or vice versa because power noise enters the FG via the LG. Failure to observe this rule may result in an equipment malfunction.
- Ground the LG at the power supply side.



7.3 Errors and Actions To Be Taken

7.3.1 Meanings of error log information items

Error information on the errors detected by ET.NET can be viewed in the [Error Log Information] window displayed by the S10V base system. This error log information is presented in the format shown below.
For information on how to start the S10V base system and how to display the error log information on screen, refer to the S10V USER'S MANUAL, BASIC MODULE (manual number SVE-1-100).

<Panic log error message>

```
[*] ******* (PC=0x********,FADR=0x********)
①    ②            ③                  ④
```

Table 7-2 Panic Log Error Message Format (Components)

Format type	Error message format
System down (system error)	①+②+③+④

where:

① Error severity indication
[F]: Fatal error

② Error message
For information on the displayed error message, see Table 7-6. Any error code not listed in the error messages list is reported by displaying the following default error message:

Table 7-3 Panic Log Default Error Messages

Format type	Error message
System down (system error)	System down

③ Program counter

④ Fault address

<Error message other than the panic log>

(Pattern 1)

```
[*] ******* (UNO=**,DEV=0x********) (TN=***) (SLOT=**)
①    ②                ③                ④        ⑤
```

Table 7-4 Non-Panic Log Error Message Format (Components)

Format type	Error message format
Program error	①+②+④
Macro parameter check error	①+②+④
I/O error	①+②+③
Watchdog timer timeout error	①+②
Module error	①+②+⑤
Kernel warning	①+②+④
Kernel information	①+②+④
System down (kernel trap)	①+②
Memory error	①+②+④
System bus error	①+②+⑤
Message frame error	①+②
Buffer status report	①+②
Socket error	①+②

where:

① Error severity indication

[F]: Fatal error; [W]: Warning;

[E]: Error; [I]: Information

② Error message

For information on the displayed error message, see Table 7-6. Any error code not listed in the error messages list is reported by displaying the default error message shown in Table 7-5.

Table 7-5 Non-Panic Log Default Error Messages

Format type	Error message
Program error	Program error
Macro parameter check error	Macro parameter error
I/O error	I/O error
Watchdog timer timeout error	WDT timeout error
Module error	Module Error
Kerning warning	Kernel Warning
Kernel information	Kernel Information
System down (kernel trap)	System down
Memory error	Memory error
System bus error	System Bus Error
Message frame error	Message frame error
Buffer status report	Buffer status
Socket error	Socket error

③ Unit number, device number

Unit number range: 1 to 24

Device number range: 0x00000000 to 0xFFFFFFFF

④ Task number

Range: 1 to 255

⑤ Slot number

Range: 0 to 7

(Pattern 2)

Error messages other than the panic log and the non-panic log pattern-1 error messages are displayed in the following format:

```
%****-*-****-****
   ①  ②  ③    ④
```

where:

① Identification of the system that detected the error
CPMS: CPMS (basic OS)
LNET: Network driver

② Error severity indication
F: Fatal error; E: Error;
W: Warning; I: Information;
?: Other type of error

③ Fault classification
HARD: Hardware
CPMS: CPMS
SOFT: Software item other than CPMS
????: Other item

④ Code
A code that denotes a format type. Displayed as a 4-digit hexadecimal number.

Table 7-6 Error Messages (1/4)

No.	Error log title	Error code	Error message	Meaning	ALARM LED	ERR LED	Required recovery action
1	%CPMS-E-SOFT-0001	EC=03620000	Program error (Invalid Data Access)	Data access error	– (Turned off)	– (Turned off)	Hardware item replacement
2	%CPMS-E-SOFT-0001	EC=03660000	Program error (Data Access Protection)	Data access protection error	–	–	Hardware item replacement
3	%CPMS-E-SOFT-0001	EC=03600000	Program error (Data Page Fault)	Data access page fault	–	–	Hardware item replacement
4	%CPMS-E-SOFT-0001	EC=03420000	Program error (Invalid Inst. Access)	Instruction access error	–	–	Hardware item replacement
5	%CPMS-E-SOFT-0001	EC=03460000	Program error (Inst. Access Protection)	Instruction access protection error	–	–	Hardware item replacement
6	%CPMS-E-SOFT-0001	EC=03400000	Program error (Instruction Page Fault)	Instruction access page fault	–	–	Hardware item replacement
7	%CPMS-E-SOFT-0001	EC=03030000	Program error (Inst. Alignment Error)	Instruction alignment error	–	–	Hardware item replacement
8	%CPMS-E-SOFT-0001	EC=03080000	Program error (Privileged Instruction)	Privileged-instruction error	–	–	Hardware item replacement
9	%CPMS-E-SOFT-0001	EC=03040000	Program error (Illegal Instruction)	Illegal-instruction error	–	–	Hardware item replacement
10	%CPMS-E-SOFT-0001	EC=03390000	Program error (FP Program Error)	Floating-point arithmetic error	–	–	Hardware item replacement
11	%CPMS-E-SOFT-0001	EC=03470000	Program error (Data Alignment Error)	Data alignment error	–	–	Hardware item replacement
12	%CPMS-E-SOFT-0002	EC=05130000	Macro parameter error	Undefined macro issued	–	–	Hardware item replacement
13	%CPMS-E-SOFT-0002	EC=05110000	Macro parameter error	Incorrect macro parameter used	–	–	Hardware item replacement
14	%CPMS-E-SOFT-0005	EC=05C70000	WDT timeout error	Watchdog timer timeout	–	Turned on	Hardware item replacement
15	%CPMS-E-HARD-0006	EC=03B70000	Module error (Bus Target Abort)	Bus target aborted	–	–	Hardware item replacement
16	%CPMS-E-HARD-0006	EC=05000000	Module error (Invalid Interrupt)	Invalid interrupt detected	–	–	Hardware item replacement
17	%CPMS-E-HARD-0006	EC=05000001	Module error (Undefined Invalid Interrupt)	Undefined invalid interrupt detected	–	–	Hardware item replacement
18	%CPMS-E-HARD-0006	EC=05000002	Module error (INTEVT Invalid Interrupt)	INTEVT invalid interrupt detected	–	–	Hardware item replacement
19	%CPMS-E-HARD-0006	EC=0500F001	Module error (HERST Invalid Interrupt)	Serious-error invalid interrupt detected	–	–	Hardware item replacement
20	%CPMS-E-HARD-0006	EC=0500F002	Module error (HERST2 Invalid Interrupt)	Serious-error invalid interrupt of type 2 detected	–	–	Hardware item replacement
21	%CPMS-E-HARD-0006	EC=0500F003	Module error (BUERRSTAT Invalid Interrupt)	Bus serious-error interrupt status invalid	–	–	Hardware item replacement
22	%CPMS-E-HARD-0006	EC=0500F006	Module error (NHPMCLG Invalid Interrupt)	Memory serious-error interrupt status invalid	–	–	Hardware item replacement
23	%CPMS-E-HARD-0006	EC=0500F007	Module error (ECC 2bit Master Invalid Interrupt)	Memory ECC 2-bit serious-error status invalid	–	–	Hardware item replacement
24	%CPMS-E-HARD-0006	EC=0500F008	Module error (RERRMST Invalid Interrupt)	RERR interrupt status invalid	–	–	Hardware item replacement
25	%CPMS-E-HARD-0006	EC=0500C001	Module error (NINTR Invalid Interrupt)	NINT status invalid	–	–	Hardware item replacement
26	%CPMS-E-HARD-0006	EC=0500B001	Module error (PUINTR Invalid Interrupt)	PUINT status invalid	–	–	Hardware item replacement
27	%CPMS-E-HARD-0006	EC=05005001	Module error (RINTR Invalid Interrupt)	RINT status invalid	–	–	Hardware item replacement
28	%CPMS-E-HARD-0006	EC=05003001	Module error (LV3 INTST Invalid Interrupt)	Level-3 interrupt status invalid	–	–	Hardware item replacement
29	%CPMS-E-HARD-0006	EC=05003002	Module error (RQI6 INF Invalid Interrupt)	RQI6 status invalid	–	–	Hardware item replacement
30	%CPMS-E-HARD-0006	EC=05001001	Module error (RQI3 INT Invalid Interrupt)	RQI3 status invalid	–	–	Hardware item replacement

Table 7-6 Error Messages (2/4)

No.	Error log title	Error code	Error message	Meaning	ALARM LED	ERR LED	Required recovery action
31	%CPMS-E-HARD-0006	EC=05001002	Module error (RQI3 Link Invalid Interrupt)	RQI3 link status invalid	–	–	Hardware item replacement
32	%CPMS-E-HARD-0006	EC=05001003	Module error (RQI3 Module Invalid Interrupt)	RQI3 module status invalid	–	–	Hardware item replacement
33	%CPMS-E-HARD-0006	EC=07D00001	Module error (INVALID EXCEPTION)	Invalid exception detected	–	Turned on	Hardware item replacement
34	%CPMS-E-HARD-0006	EC=07D00010	Module error (INVALID MAIN/SUB SWITCH SETTING)	Main-/submodule setting switch operation error	–	Turned on	Set the switch correctly.
35	%CPMS-E-HARD-0006	EC=07D00011	Module error (INVALID MAC ADDRESS)	MAC address error	–	Turned on	Hardware item replacement
36	%CPMS-E-HARD-0006	EC=07D00012	Module error (MAIN/SUB SWITCH SETTING DUPLICATION)	Duplicate setting of main-/ submodule setting switch	Blinking	Turned on	Set the switch correctly.
37	%CPMS-E-HARD-0006	EC=07D00013	Module error (ETHERNET LSI CHECK ERROR)	LANCE diagnosis error	–	Turned on	Hardware item replacement
38	%CPMS-E-HARD-0006	EC=07D00014	Module error (SDRAM CHECK ERROR)	SDRAM initialization error	–	Turned on	Hardware item replacement
39	%CPMS-E-HARD-0006	EC=07D00015	Module error (OS-ROM CHECKSUM ERROR)	ROM checksum error (in CPMS)	–	Turned on	Hardware item replacement
40	%CPMS-E-HARD-0006	EC=07D00016	Module error (CAN NOT MOUNTING WITH LQE520 MODULE)	Intermixed with a model LQE520 module among the similar installed modules	–	Turned on	The module cannot be mounted on the same mount base on which a model LQE520 module is mounted. Remove the model LQE520 module from the mount base.
41	%CPMS-E-HARD-0006	EC=07D00018	Module error (TASK-ROM CHECKSUM ERROR)	ROM checksum error (communication task)	–	Turned on	Hardware item replacement
42	%CPMS-W-HARD-0006	EC=07D01003	Module error (THE VERSION OF CMU MODULE IS OLD)	Installed along with a CMU module not supporting the model LQE720	Turned on	–	Replacement of the CMU module with a supporting one
43	%CPMS-W-HARD-0006	EC=0D010000	Module error (Memory Alarm)	Memory single-bit error (solid)	–	–	Hardware item replacement
44	%CPMS-E-HARD-0006	EC=0D330000	Module error (Hardware WDT timeout)	Hardware watchdog timer timeout	–	–	Hardware item replacement
45	%CPMS-E-HARD-0006	EC=0D340000	Module error (Software WDT Timeout)	Software watchdog timer timeout	–	–	Hardware item replacement
46	%CPMS-F-HARD-0009	EC=0D810000	System down (BPU Error)	BPU error	–	Turned on	Hardware item replacement
47	%CPMS-F-HARD-0009	EC=03820000	System down (Memory Error)	Memory error	–	Turned on	Hardware item replacement
48	%CPMS-F-HARD-0009	EC=038A0000	System down (Memory Access Error)	Memory access error	–	Turned on	Hardware item replacement
49	%CPMS-F-HARD-0009	EC=038B0000	System down (Internal Bus Parity)	Internal bus parity error	–	Turned on	Hardware item replacement
50	%CPMS-F-HARD-0009	EC=038C0000	System down (System Bus Parity)	System bus parity error	–	Turned on	Hardware item replacement
51	%CPMS-F-HARD-0009	EC=038F0000	System down (Undefined Machine Check)	Undefined machine check error	–	Turned on	Hardware item replacement
52	%CPMS-F-CPMS-0009	EC=03620000	System down (Invalid Data Access)	Data access error	–	Turned on	Hardware item replacement
53	%CPMS-F-CPMS-0009	EC=03660000	System down (Data Access Protection)	Data access protection error	–	Turned on	Hardware item replacement
54	%CPMS-F-CPMS-0009	EC=03600000	System down (Data Page Fault)	Data access page fault	–	Turned on	Hardware item replacement
55	%CPMS-F-CPMS-0009	EC=03420000	System down (Invalid Inst. Access)	Instruction access error	–	Turned on	Hardware item replacement
56	%CPMS-F-CPMS-0009	EC=03460000	System down (Inst. Access Protection)	Instruction access protection error	–	Turned on	Hardware item replacement

Table 7-6 Error Messages (3/4)

No.	Error log title	Error code	Error message	Meaning	ALARM LED	ERR LED	Required recovery action
57	%CPMS-F-CPMS-0009	EC=03400000	System down (Instruction Page Fault)	Instruction access page fault	–	Turned on	Hardware item replacement
58	%CPMS-F-CPMS-0009	EC=03030000	System down (Inst. Alignment Error)	Instruction alignment error	–	Turned on	Hardware item replacement
59	%CPMS-F-CPMS-0009	EC=03040000	System down (Illegal Instruction)	Illegal-instruction error	–	Turned on	Hardware item replacement
60	%CPMS-F-CPMS-0009	EC=03380000	System down (FP Unavailable)	Floating-point unavailability exception	–	Turned on	Hardware item replacement
61	%CPMS-F-CPMS-0009	EC=03390000	System down (FP System down)	Floating-point arithmetic error	–	Turned on	Hardware item replacement
62	%CPMS-F-CPMS-0009	EC=03470000	System down (Data Alignment Error)	Data alignment error	–	Turned on	Hardware item replacement
63	%CPMS-F-CPMS-0009	EC=030F0000	System down (Illegal Exception)	Illegal-exception error	–	Turned on	Hardware item replacement
64	%CPMS-F-CPMS-0009	EC=05700000	System down (System Error)	System down (system error)	–	Turned on	Hardware item replacement
65	%CPMS-F-CPMS-000A	EC=05800000	System down (Kernel Trap)	System down (kernel trap)	–	Turned on	Hardware item replacement
66	%LNET-W-HARD-004	EC=07801308	I/O error (SEND_TIMEOUT)	Transmission timeout error	–	–	Automatically recovered
67	%LNET-E-HARD-004	EC=07801308	I/O error (SEND_TIMEOUT)	Transmission timeout error	–	Turned on	Hardware item replacement in case it is not recovered along with power recovery.
68	%LNET-W-HARD-004	EC=0780130A	I/O error (RESET_ERROR)	Hardware resetting error	–	–	Automatically recovered
69	%LNET-E-HARD-004	EC=0780130A	I/O error (RESET_ERROR)	Hardware resetting error	–	Turned on	Hardware item replacement in case it is not recovered along with power recovery.
70	%LNET-W-HARD-004	EC=0780130E	I/O error (MEMORY)	Memory error	–	–	Automatically recovered
71	%LNET-E-HARD-004	EC=0780130E	I/O error (MEMORY)	Memory error	–	Turned on	Hardware item replacement in case it is not recovered along with power recovery.
72	%LNET-W-HARD-004	EC=07801310	I/O error (LOSS)	Carrier loss error	–	–	Transmission path checkup (*1)
73	%LNET-W-HARD-004	EC=07801311	I/O error (RETRY)	Retry error	–	–	Transmission path checkup (*3)
74	%LNET-W-HARD-004	EC=07801312	I/O error (LATE)	Late-collision error	–	–	Transmission path checkup (*5)
75	%LNET-W-HARD-004	EC=07801351	I/O error (TX_ABORT)	Abnormal end of transmission	–	–	Transmission path checkup
76	%LNET-W-HARD-004	EC=07801353	I/O error (TX_DEFER)	Transmission error due to transmission delay	–	–	Transmission path checkup
77	%LNET-W-HARD-004	EC=07801370	I/O error (EC_PCI_ERROR)	PCI error detected in communication LSI	–	–	Hardware item replacement (*6)
78	%LNET-W-HARD-004	EC=07801376	I/O error (TX_DATA_UNDER)	Communication data FIFO underrun	–	–	Transmission path checkup (*6)
79	%LNET-W-HARD-004	EC=07801375	I/O error (RX_STAT_OVER)	Reception-status FIFO overrun	–	–	Check the load on the communication line. (*6)
80	%LNET-W-HARD-004	EC=07801377	I/O error (RX_DATA_OVER)	Received-data FIFO overrun	–	–	Check the load on the communication line. (*7)
81	%LNET-E-HARD-004	EC=07D01001	I/O error (IP_ADDERSS_NOT_REGISTERED)	IP address not set yet	Turned on	–	Register the IP address.

Table 7-6 Error Messages (4/4)

No.	Error log title	Error code	Error message	Meaning	ALARM LED	ERR LED	Required recovery action
82	%LNET-E-HARD-004	EC=07801400	I/O error (PCI_BUS_ERR)	PCI bus error	–	–	Hardware item replacement
83	%LNET-E-HARD-004	EC=07801505	I/O error (INV_INTR)	Invalid interrupt generated from the communication line	–	–	Hardware item replacement
84	%LNET-E-SOFT-004	EC=07801508	I/O error (BUF_OVF)	OS-managed send/receive buffer overflowed.	–	–	Check the load on the communication line. (*2)
85	%LNET-W-SOFT-004	EC=0780150F	I/O error (SOCKET_OVF)	Socket table is full.	–	–	Review the user program.
86	%LNET-W-SOFT-004	EC=07801510	I/O error (IFCONFIG_UP)	Network interface initialization error	–	–	Review the settings made.
87	%LNET-W-SOFT-004	EC=07801512	I/O error (IPADDR_DUPL)	Duplicated-IP address error	Blinking (*4)	–	Review the settings made. (*4)

(*1) This message is displayed once when a total of 32 LSI carrier losses are detected. The LSI carrier loss occurs when an attempt is made to transmit data in the OFF state of the LINK LED indicator, which indicates that a “link not established yet” condition exists. The LSI carrier loss also occurs when, after the startup of the CPU, a total of 32 send requests are issued by an application program before the LINK LED indicator comes on. In this case, the application program needs to be corrected so that it will not transmit data until the LINK LED indicator comes on.

(*2) This type of error occurs due to an insufficient buffer space in a "high communication load" condition.

(*3) This message is displayed once when a total of 32 retries are made successively.

(*4) The IP address is duplicated with some other computer’s. If the local computer is connected to the network after the other computer, the ALARM LED indicator will be lit.

(*5) This message is displayed once when a total of eight such collisions are detected successively. If a total of 16 or more such collisions are detected successively, the message will be displayed once when a total of 256 such collisions are detected.

(*6) This message is displayed once when a total of five such errors or underruns/overruns are detected successively.

(*7) This message is displayed once when a total of 10 such overruns are detected successively.

7.3.2 Meanings of DHP trace information items

DHP trace information is displayed in the following way:

- It is displayed in a backward chronological order.
- It is classified into three groups of task, idle, and OS, where each group of information begins with the event DISPATCH_E.
- The DISPATCH_E line presents numbers in the range 0x00000001 to 0x0000012C in the DATA1 field. These numbers are the task numbers of tasks.
- The time is expressed in seconds and microseconds as a real value with six digits after the decimal point.
- The DHP events and data displayed have the relationships shown in Table 7-7.

<DHP information display example>

The following is an example of a DHP trace information display, which is shown along with a brief explanation of which task was executed and of the operation of the operating system (OS) which took place at the time of task switching.

									Explanation
New	165	40.901912	TASK_PRI	112	10	00000071	00000032		Task 112 was executed.
	166	40.901901	RLEAS	112	10	00000071			
	167	40.901883	DISPATCH_E	112	10	00000070	00000032	84DB2000 00000002	Aborting task 111, the OS switched to task 112.
	168	40.901868	DISPATCH	111	10	0000006F	00000032	84DAF000	
	169	40.901832	DISPATCH_E	111	10	0000006F	00000032	84DAF000 00000002	
	170	40.901815	RUNQ	112	10	00000070			
	171	40.901810	DISPATCH	112	10	00000070	00000032	84DB2000	
	172	40.901796	RUNQ	112	10	0000006F			
	173	40.901785	WAKEUP	112	10	F0000000			
	174	40.901771	ABORT	112	10	0000006F			Task 112 was executed.
	175	40.901748	GFACT	112	10	00000003			
	176	40.901727	DISPATCH_E	112	10	00000070	00000032	84DB2000 00000002	Delaying task 111, the OS switched to task 112.
	177	40.901703	DISPATCH	111	10	0000006F	0000001C	84DAF000	
	178	40.901691	TASK_PRI	111	10	0000006F	0000001C	00000000	
	179	40.901611	DELAY	111	10	00000BB8			Task 111 was executed.
	180	40.901600	RUNQ	111	10	00000070			
	181	40.901590	QUEUE	111	10	00000070	00000003		
	182	40.901579	TASK_PRI	111	10	00000070	00000032		
	183	40.901568	RLEAS	111	10	00000070			
	184	40.901546	GFACT	111	10	00000002			
	185	40.901525	DISPATCH_E	111	10	0000006F	00000032	84DAF000 00000002	Placing task 110 into wait state, the OS switched to task 111.
	186	40.901507	DISPATCH	110	10	0000006E	00000032	84DAC000	
	187	40.901493	SLEEP	110	10	841C982C	00000032		
	188	40.901483	WAIT	110	10	5004502C			Task 110 was executed.
	189	40.901471	RUNQ	110	10	0000006F			
	190	40.901459	QUEUE	110	10	0000006F	00000002		
	191	40.901446	TASK_PRI	110	10	0000006F	00000032		
	192	40.901434	RLEAS	110	10	0000006F			
	193	40.901408	DISPATCH_E	110	10	0000006E	00000032	84DAC000 00000001	Terminating task 119, the OS switched to task 110.
	194	40.901399	RUNQ	110	10	0000006E			
	195	40.901393	DISPATCH	110	10	0000006E	00000032	84DAC000	
	196	40.901373	DISPATCH_E	110	10	0000006E	00000032	84DAC000 00000002	
	197	40.901348	DISPATCH	119	10	00000077	00000032	84DC7000	
	198	40.901323	EXIT	119	10				Task 119 was executed.
	199	40.901311	RUNQ	119	10	0000006E			
	200	40.901300	WAKEUP	119	10	841C982C			
Old	201	40.901288	POST	119	10	5004502C	00001234		

Table 7-7 DHP Codes (1/3)

Processing by CPMS (tracing)

Code value	DHP-displayed name	Trace point	DATA1	DATA2	DATA3	DATA4	DATA5
0x010001	TRACE_ON	Start of tracing					
0x010002	TRACE_OFF	End of tracing					
0x010003	TRACE_TBU	Time recording	old tbu（Time Base Upper）	new tbu（Time Base Upper）			

Processing by CPMS (scheduling)

Code value	DHP-displayed name	Trace point	DATA1	DATA2	DATA3	DATA4	DATA5
0x020001	WAKEUP	WAKEUP process	wchan				
0x020002	SLEEP	SLEEP event	wchan	pri (priority level)			
0x020003	DISPATCH	Before thread_invoke process	tn (task number)	pri (priority level)	cont (CPMS stack information)		
0x020083	DISPATCH_E	After thread_invoke process	tn (task number)	pri (priority level)	cont (CPMS stack information)		
0x020004	RUNQ	RUNQ connection	tn (task number)				
0x020005	IDLE	IDLE process					
0x020006	TASK_PRI	Priority level control	tn (task number)	pri (priority level)			

Processing by CPMS (error logging, built-in subroutine processing)

Code value	DHP-displayed name	Trace point	DATA1	DATA2	DATA3	DATA4	DATA5
0x030002	ELSETK	elset process	Error type	Error class	Error format	Error code	
0x030003	IOERR	I/O error handling	uno	Device number	Device address	Detailed error code	
0x030004	PRGERR	Program error handling	tn (task number)	Fault address	PC	expevt register	
0x030005	WDTERR	WDT error handling	time				
0x030006	PIOERR	PI/O error handling	slot				
0x030007	MODERR	Module error handling	Error code	slot	HERST register	INTST register	
0x030008	KERN_PANIC	Panic handling	tn (task number)	Fault address	PC	Extension error code	
0x03000a	ASSERT	Assertion panic handling	Place where the error occurred	line	Test conditions		
0x03000b	CPUSTOP	CPU termination process	Built-in subroutine nest count	Built-in subroutine point number	Built-in subroutine return value		

Processing by CPMS (startup/termination)

Code value	DHP-displayed name	Trace point	DATA1	DATA2	DATA3	DATA4	DATA5
0x040001	SETUP_MAIN	Startup process	1 (fixed)				
0x040002	HDUTL_STOP	Termination process					
0x040003	HDUTL_RSUM	Restart process					
0x040004	HDUTL_ERR	ERROR handling					

Processing by CPMS (exception handling)

Code value	DHP-displayed name	Trace point	DATA1	DATA2	DATA3	DATA4	DATA5
0x050001	EXCEPTION	Exception handling	Type of exception				
0x050002	SLIH_SRES	System reset exception	NMI factor register	PC			
0x050005	SLIH_SM	System management interrupt exception	MSW register				
0x050007	SLIH_HERR	Serious-error interrupt handling	Serious-error cause register				

Table 7-7 DHP Codes (2/3)

Processing by CPMS (macro processing)

Code value	DHP-displayed name	Trace point	DATA1	DATA2	DATA3	DATA4	DATA5
0x100000	NOSYS	Issuing of undefined macro					
0x100001	QUEUE	Issuing of queue	tn (task number)	fact (initiation factor)			
0x100002	RLEAS	Issuing of rleas	tn (task number)				
0x100003	SFACT	Issuing of sfact	tn (task number)	fact (initiation factor)			
0x100004	ABORT	Issuing of abort	tn (task number)				
0x100005	SUSP	Issuing of susp	tn (task number)				
0x100006	RSUM	Issuing of rsum	tn (task number)				
0x100007	CTIME	Issuing of ctime	tn (task number)	fact (initiation factor)			
0x100008	WAIT	Issuing of wait	ecb (ECB address)				
0x100009	POST	Issuing of post	ecb (ECB address)	pcode (post code)			
0x10000a	TIMER	Issuing of timer	id (event type)	tn (task number)	fact (initiation factor)	t (time period/point in time)	cyt (cycle time)
0x10000b	DELAY	Issuing of delay	t (milliseconds)				
0x10000c	STIME	Issuing of stime	Year	Month	Day	Milliseconds	
0x10000d	CHAP	Issuing of chap	tn (task number)	chgp (priority level)			
0x10000e	RSERV	Issuing of rserv	n (number of shared resources)	para1	para2	para3	para4
0x10000f	FREE	Issuing of free	n (number of shared resources)	para1	para2	para3	para4
0x100010	PRSERV	Issuing of prserv	n (number of shared resources)	para1	para2	para3	para4
0x100011	PFREE	Issuing of pfree	n (number of shared resources)	para1	para2	para3	para4
0x100012	GFACT	Before/After issuing of gfact	fact (initiation factor)				
0x100013	GTIME	Issuing of gtime	time (time_t address)				
0x100014	EXIT	Issuing of exit					
0x100015	ASUSP	Issuing of asusp					
0x100016	ARSUM	Issuing of arsum					
0x100017	OPEN	Issuing of open	uno	flag			
0x100018	CLOSE	Issuing of close	uno				
0x100019	READ	Issuing of read	uno	Logical address	cnt		
0x10001a	WRITE	Issuing of write	uno	Logical address	cnt		
0x10001e	DHPCTL	Issuing of dhpctl	cmd (command)	id (major ID)	info		
0x10001f	DHPREAD	Issuing of dhpread	Logical address	size			
0x100023	CHML	Issuing of chml	Logical address	para1	para2	para3	para4

Processing by CPMS

Code value	DHP-displayed name	Trace point	DATA1	DATA2	DATA3	DATA4	DATA5
0x200004	SETTCB	Issuing of settcb	toptn	cnt	tcbaddr		
0x20000c	REGSET	Task register setting	reg	Data address			

Table 7-7 DHP Codes (3/3)

Processing by RCTLNET (network driver)

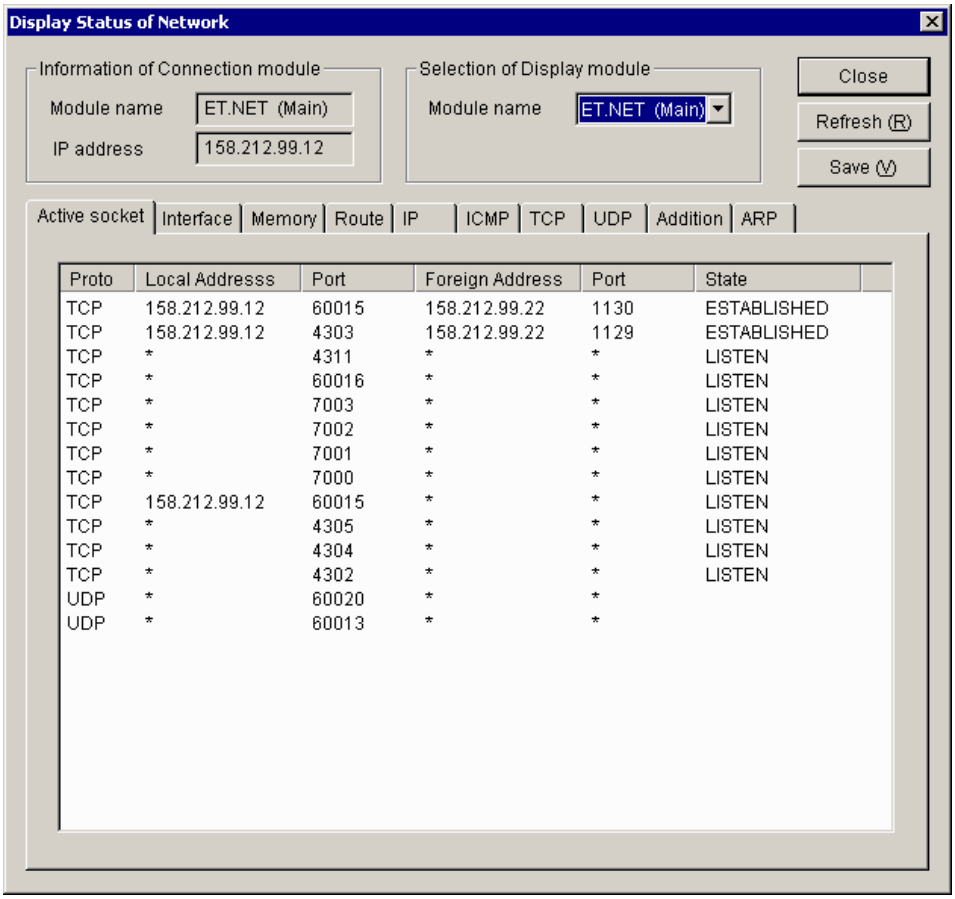
Code value	DHP-displayed name	Trace point	DATA1	DATA2	DATA3	DATA4	DATA5
0x300001	SOCKET	Issuing of SOCKET	uno (unit number)	Type	Protocol	Work data	Work data
0x300002	BIND	Issuing of BIND	Socket ID	Port number	IP address	Work data	Work data
0x300003	LISTEN	Issuing of LISTEN	Socket ID	Maximum number of connection requests than can wait for a connection to be established	Work data	Work data	Work data
0x300004	ACCEPT	Issuing of ACCEPT	Socket ID	Address information pointer	Address information length	Work data	Work data
0x300005	CONNECT	Issuing of CONNECT	Socket ID	Port number	IP address	Work data	Work data
0x300006	SEND	Issuing of SEND	Socket ID	Buffer address	High-order word: Data length Low-order word: Transmission flag	Work data	Work data
0x300007	SENDTO	Issuing of SENDTO	Socket ID	High-order word: Data length Low-order word: Transmission flag	Port number	IP address	Internal task information
0x300008	RECV	Issuing of RECV	Socket ID	Buffer address	High-order word: Data length Low-order word: Reception flag	Work data	Work data
0x300009	RECVFROM	Issuing of RECVFROM	Socket ID	Buffer address	High-order word: Data length Low-order word: Reception flag	Address information pointer	Address information length
0x30000a	SETSOCKOPT	Issuing of SETSOCKOPT	Socket ID	Level	Option	Option information address	Option information length address
0x30000b	GETSOCKOPT	Issuing of GETSOCKOPT	Socket ID	Level	Option	Option information address	Option information length address
0x30000c	SHUTDOWN	Issuing of SHUTDOWN	Socket ID	Socket shutdown method	Work data	Work data	Work data
0x30000d	NET_END	Abnormal end of macro	Socket ID	Error number	Work data	Work data	Work data
0x300010	NET_CTLR	Issuing of IOCTL	Unit number plus slot number	Control information	Control information	Control information	Control information
0x300010	NET_CTLR	Acceptance of remote CPU control request	Station number plus command	Frame length plus transmission number	Target type plus data length	Data address	Work data
0x300011	NET_START	Start of NCP-F I/O	Socket ID	Task information	Command code plus socket status	Initiation information 1	Initiation information 2
0x300011	NET_START	Transmission by built-in LANCE	Socket ID plus ETHER_TYPE	Packet header information			
0x300012	NET_TERM	NCP-F termination interrupt	Socket ID	Task information	Response information	Status code	Interrupt information
0x300012	NET_TERM	Built-in LANCE termination interrupt	Socket ID plus FFFF	LANCE descriptor information (TMD0, TMD1, TMD2, TMD3)			
0x300013	NET_ATEN	NCP-F attention interrupt	Socket ID	Task information	Response information	Status code	Interrupt information
0x300013	NET_ATEN	Reception by built-in LANCE	Socket ID plus ETHER_TYPE	Packet header information			
0x300014	NET_STO	Software timeout	Socket ID	Task information	Initiation information	Initiation information	Initiation information
0x300015	NET_SUB	Error detection	Error type	Error information	Error information	Error information	Error information

7.3.3 Meanings of network status information items

Network status information for the ET.NET module is displayed as shown below.

(1) Socket information

The socket information displayed as shown below is a list of the currently existing network connections.



Proto	Local Addresss	Port	Foreign Address	Port	State
TCP	158.212.99.12	60015	158.212.99.22	1130	ESTABLISHED
TCP	158.212.99.12	4303	158.212.99.22	1129	ESTABLISHED
TCP	*	4311	*	*	LISTEN
TCP	*	60016	*	*	LISTEN
TCP	*	7003	*	*	LISTEN
TCP	*	7002	*	*	LISTEN
TCP	*	7001	*	*	LISTEN
TCP	*	7000	*	*	LISTEN
TCP	158.212.99.12	60015	*	*	LISTEN
TCP	*	4305	*	*	LISTEN
TCP	*	4304	*	*	LISTEN
TCP	*	4302	*	*	LISTEN
UDP	*	60020	*	*	
UDP	*	60013	*	*	

where:

- Protocol

  The name of the protocol used over the connection.
- Local Address

  The IP address of the local host (source of connection). If the IP address is not bound with a socket, an asterisk (“*”) is displayed instead.
- Port

  The port number of the local host (source of connection).

- Foreign Address
  The IP address of the remote host (destination of connection). If the IP address is not bound with a socket, an asterisk ("*") is displayed instead.
- Port
  The port number of the remote host (destination of connection). If the IP address is not bound with a socket, an asterisk ("*") is displayed instead.
- State
  The connection status of the TCP protocol. The connection state is one of the following 11 states:

Displayed symbol	Meaning
CLOSED	Currently not in use.
LISTEN	Waiting for a port to become available.
SYN_SENT	Although it issued a connect (SYN) request to the server, has not received a response (ACK) from it.
SYN_RECEIVED	Has just received a connect (SYN) request from a client.
ESTABLISHED	Currently performing data communication using an established TCP connection.
FINWAIT1	Server has sent out a FIN.
FINWAIT2	Has received an ACK.
CLOSEWAIT	Has received a FIN from the server.
LASTACK	Waiting for an ACK response to be sent out to the FIN.
CLOSING	Has received a FIN and is closing the connection.
TIMEWAIT	Waiting for the connection to be terminated.

All possible state transitions between the connection states are as follows:

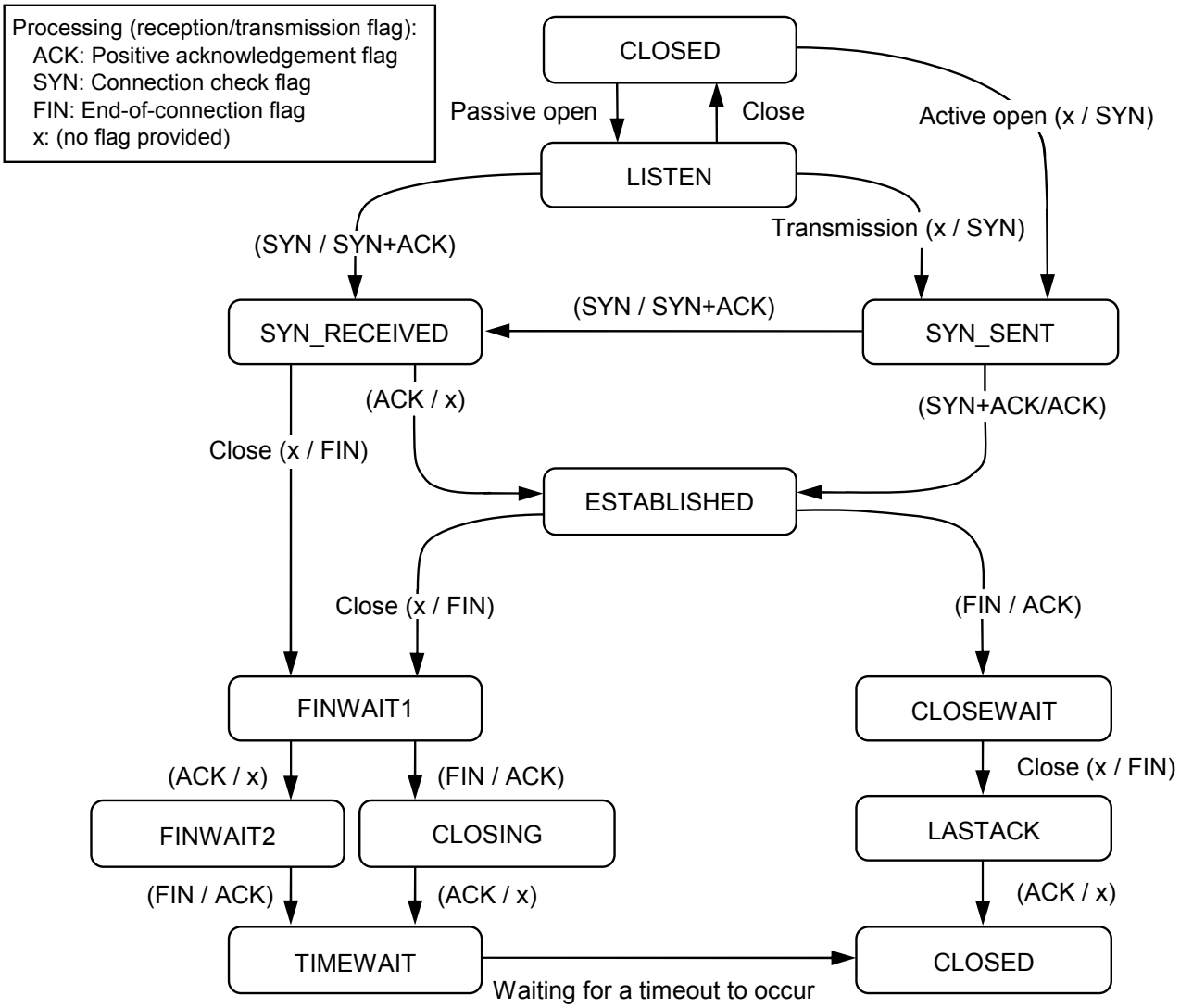
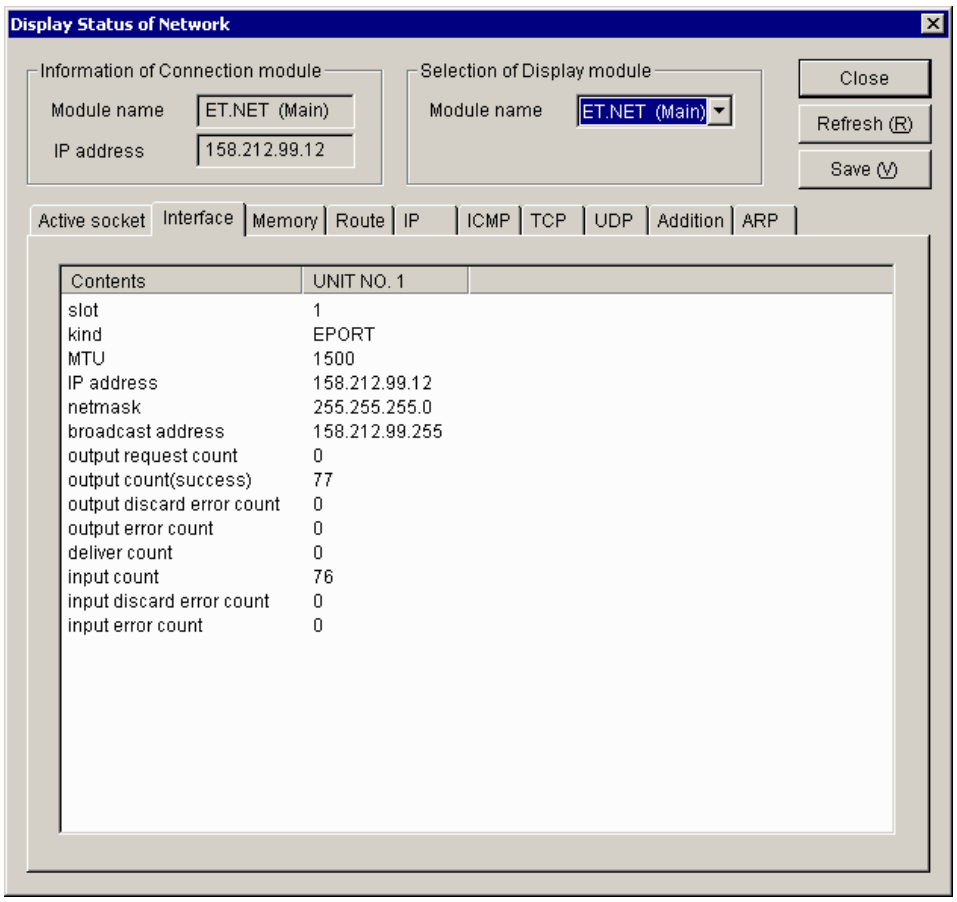


Figure 7-2 State Transitions Between Connection States

- If the TCP protocol is used over more than 150 ports, no socket information is displayed for the excess ports and the UDP protocol.
- If the TCP protocol is used over more than 80 ports, some of the socket information for the UDP protocol may not be displayed.

(2) Interface information

The interface information displayed as shown below is concerning the network interfaces currently in operation and includes input/output packet cumulative information.



where:

① slot

The slot number of the slot in which a module subjected to this display process is mounted.

② kind

Always the string “EPORT” is displayed as this item.

③ MTU

The maximum transmission unit (MTU) that refers to the maximum size of data blocks into which a set of data is divided and that is transmittable by a single transfer.

④ IP address

IP address used.

⑤ netmask

Subnet mask used.

⑥ broadcast address
Broadcast address used.

⑦ output request count
The number of send requests that were accepted for message transmission.

⑧ output count (success)
The number of message transmissions that were done successfully.

⑨ output discard error count
The number of message transmissions that failed due to memory shortage.

⑩ output error count
The number of message transmission failure reports that were made by hardware following a send request issued by the driver to the hardware.

⑪ deliver count
The number of received-message deliveries that were made to users.

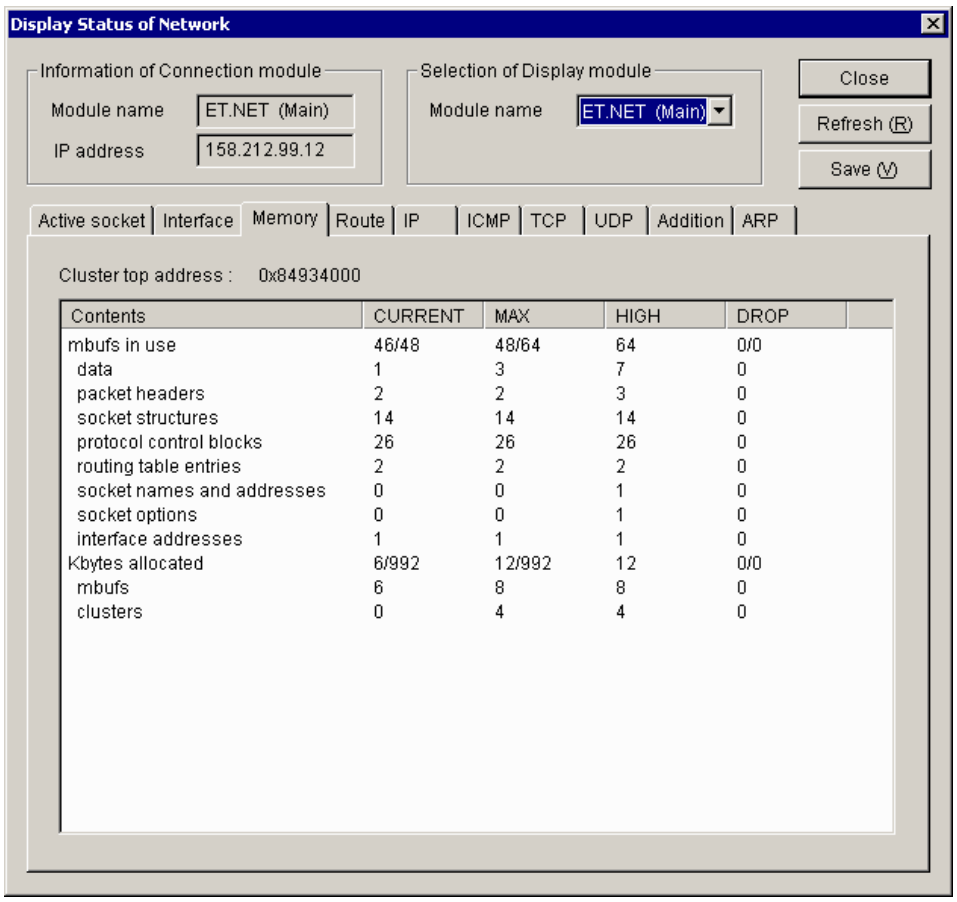
⑫ input count
The number of message reception reports that were made by hardware.

⑬ input discard error count
The number of message receptions that failed due to memory shortage.

⑭ input error count
The number of message reception failure reports that were made by hardware following a "get message" request issued by the driver to the hardware.

(3) Memory statistics

The memory statistics displayed as shown below are those which are recorded by the memory management routine.



Contents	CURRENT	MAX	HIGH	DROP
mbufs in use	46/48	48/64	64	0/0
data	1	3	7	0
packet headers	2	2	3	0
socket structures	14	14	14	0
protocol control blocks	26	26	26	0
routing table entries	2	2	2	0
socket names and addresses	0	0	1	0
socket options	0	0	1	0
interface addresses	1	1	1	0
Kbytes allocated	6/992	12/992	12	0/0
mbufs	6	8	8	0
clusters	0	4	4	0

where:

- CURRENT: The current state of mbuf.
- MAX: The status of mbuf at its maximum utilization.
- HIGH: Peak value for each item.
- DROP: The status of mbuf in the event of an overflow.

① Cluster top address

The starting address of the cluster memory.

② mbufs in use

The number of mbufs currently in use, and the total number of allocated mbufs. The following table shows details of the mbufs currently in use.

Item	Description
data	The number of mbufs in which communication data is stored.
packet headers	The number of mbufs in which a packet header is stored.
socket structures	The number of mbufs in which a socket structure is stored.
protocol control blocks	The number of mbufs in which a protocol control block is stored.
routing table entries	The number of mbufs in which routing table entries are stored.
IP reassembly-awaiting data	The number of mbufs in which IP reassembly-awaiting data is stored.
socket addresses	The number of mbufs in which a socket address is stored.
socket options	The number of mbufs in which a socket option is stored.
interface addresses	The number of mbufs in which the address of a network interface is stored.

③ Kbytes allocated

The size of the cluster memory or mbufs currently in use, and the total size of the memory allocated to clusters. For details on the size of the cluster memory currently in use, see the following items:

Item	Description
mbufs	The size of the memory used as mbufs.
clusters	The size of the memory used as clusters.

④ mbuf/cluster allocation failures count

The number of mbuf/cluster allocation failures due to a “cluster full” condition.

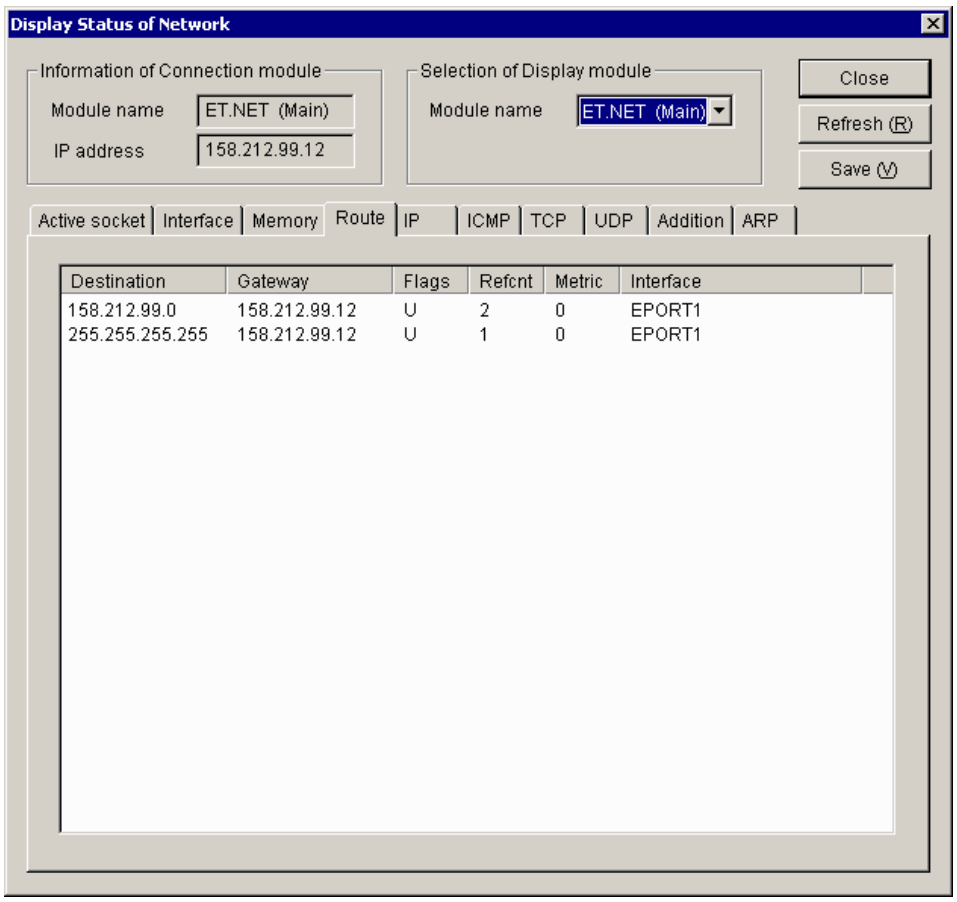
⑤ cluster request count

The number of cluster requests issued after the number of clusters used reached the upper limit.

Any item with CURRENT, MAX, HIGH, and DROP each set equal to 0 is excluded from the displayed list.

(4) Routing information

The routing information displayed as shown below is concerning the routes registered in the CMU and ET.NET modules.



① Destination

The network address of the destination. In the case of virtual network addresses, an asterisk ("*") is appended to the end of the address value.

② Gateway

The IP address of the gateway associated with the destination.

③ Flags

Information indicating the status of the route. These flags are classified into the following three types:

Flag symbol	Description
U	Indicates that the route is currently in operation.
G	Indicates that the routing is to a gateway.
H	Indicates that the routing is to a host.

④ Refcnt

The number of users who are using the route.

⑤ Metric

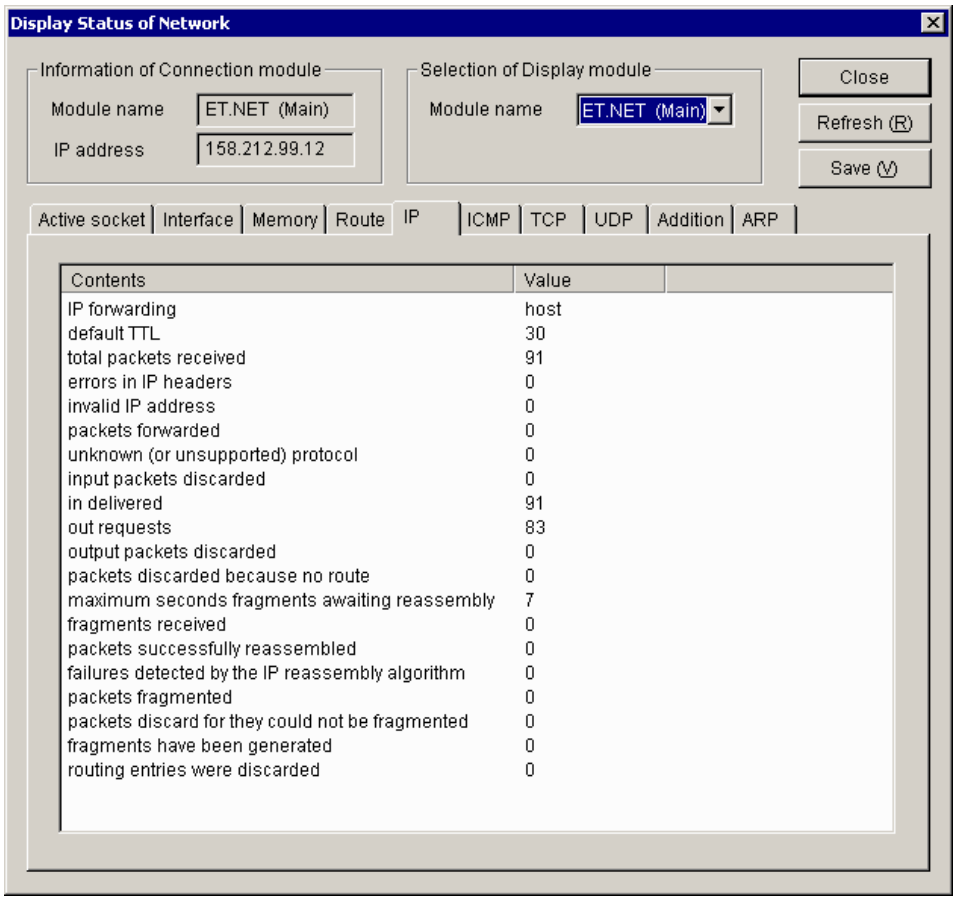
The number of gateways that are present in the route to the destination.

⑥ Interface

Always the string “EPORT” is displayed as this item.

(5) IP protocol statistics

The statistics displayed as shown below is statistical information concerning the IP protocol.



① IP forwarding

Since forwarding is not supported, the string “host” is displayed as this item. If it was supported, an indication would be displayed which indicates whether it is operating as a forwarding gateway.

② default TTL

The default value of TTL (Time To Live) that determines the maximum number of hops.

③ total packets received

The total number of IP packets that were received from all existing network interfaces.

④ errors in IP headers

The total number of IP packets that were discarded because of an error, such as a checksum or version error in the IP header.

⑤ invalid IP address

The total number of IP packets that were discarded because the destination IP address was incorrect.

⑥ packets forwarded
The total number of IP packets that were forwarded (or routed to another interface).

⑦ unknown (or unsupported) protocol
The total number of IP packets whose IP header contained a specification of an undefined higher-level protocol.

⑧ input packets discarded
The total number of IP packets that were received but discarded without being delivered to a higher-level protocol because of a buffer area shortage, or that the higher-level protocol refused to receive.

⑨ in delivered
The total number of IP packets that were delivered to a higher-level protocol, such as TCP or UDP.

⑩ out request
The total number of IP packets for which a send request was issued by a higher-level protocol.

⑪ output packets discarded
The total number of IP packets that were discarded because of a buffer shortage or some other cause.

⑫ packets discarded because no route
The total number of IP packets that were discarded because they could not be routed due to a routing information setting error or some other cause.

⑬ maximum seconds fragments awaiting reassembly
The maximum number of seconds during which a fragment awaiting reassembly may be placed in hold state.

⑭ fragment received
The total number of fragment packets that were received.

⑮ packets successfully reassembled
The number of fragments that were reassembled successfully.

⑯ failures detected by the IP reassembly algorithm
The number of failures in fragment reassembly that were caused by a timeout, resource shortage, or some other cause.

⑰ packets fragmented
The total number of transmission IP packets that were fragmented at transmission time because they exceeded the MTU size.

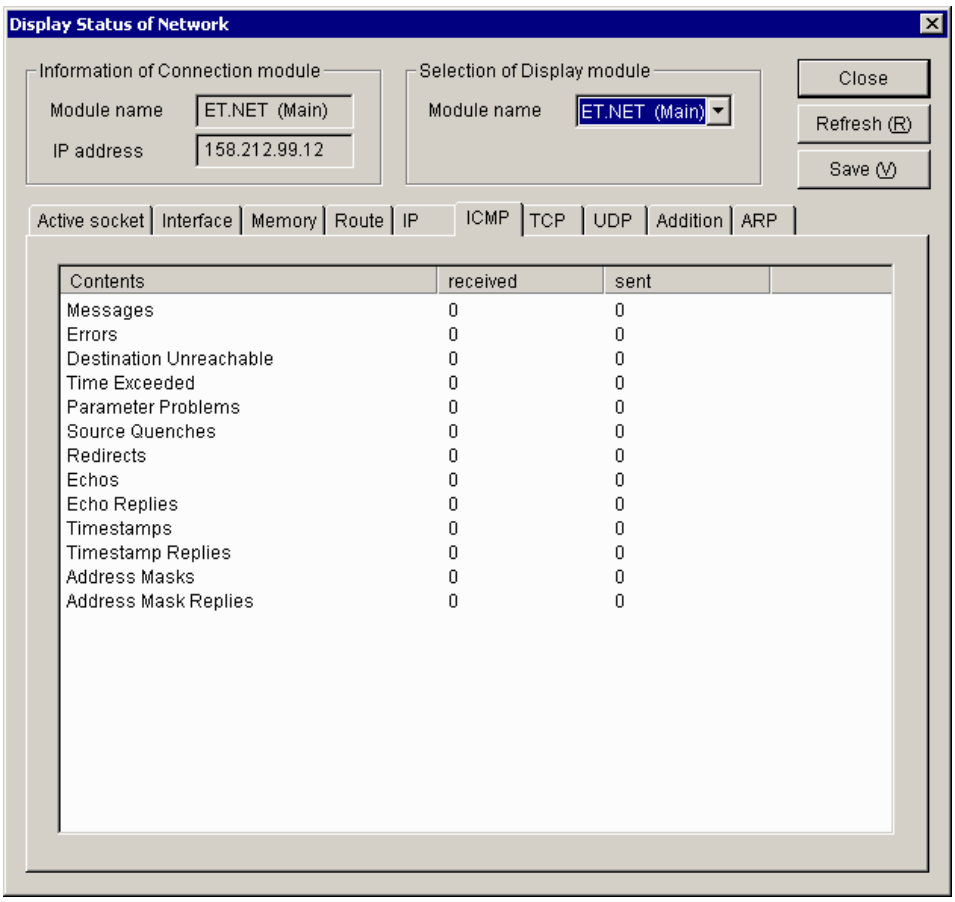
⑱ packets discard for they could not be fragmented
The total number of transmission IP packets that could not be fragmented because of a resource shortage or some other cause.

⑲ fragments have been generated
The total number of fragment packets that were created by the fragmentation of transmission IP packets.

⑳ routing entries were discarded
The number of routing entries that were discarded.

(6) ICMP protocol statistics

The statistics displayed as shown below is statistical information concerning the ICMP protocol.



① Messages

The total number of ICMP messages that were processed.

② Errors

The total number of ICMP error messages that were processed.

③ Destination Unreachable

The total number of ICMP messages that could not be transmitted to the destination.

④ Time Exceeded

The total number of ICMP messages that were discarded during routing because of a TTL (Time To Live) shortage.

⑤ Parameter Problems

The total number of ICMP messages that reported on a parameter error.

⑥ Source Quenches
The total number of ICMP messages that requested the control of transmissions because of a resource shortage on the receiving side.

⑦ Redirect
The total number of ICMP messages that reported on the existence of a more suitable route to the destination.

⑧ Echos
The total number of ICMP messages that were transmitted from the sending side of ping.

⑨ Echo Replies
The total number of ICMP messages that were returned from the receiving side of ping.

⑩ Timestamps
The total number of ICMP messages that were used as Timestamp requests.

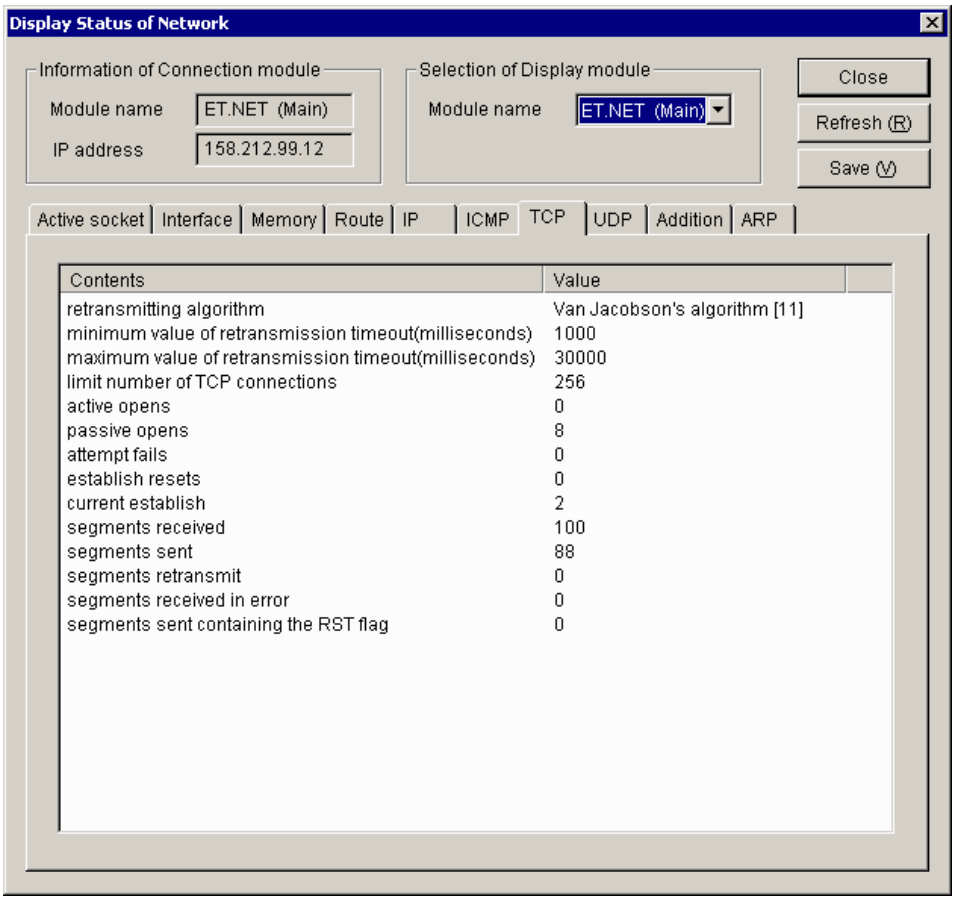
⑪ Timestamp Replies
The total number of ICMP messages that were used as responses to Timestamp requests.

⑫ Address Masks
The total number of ICMP messages that were used as Address Mask Requests.

⑬ Address Mask Replies
The total number of ICMP messages that were used as responses to Address Mask Requests.

(7) TCP protocol statistics

The statistics displayed as shown below is statistical information concerning the TCP protocol.



① retransmitting algorithm

Name of the retransmission timeout (RTO) algorithm used.

② minimum value of retransmission timeout (milliseconds)

The minimum value of retransmission timeout period expressed in milliseconds.

③ maximum value of retransmission timeout (milliseconds)

The maximum value of retransmission timeout period expressed in milliseconds.

④ limit number of TCP connections

The maximum number of connections that can be established at a time.

⑤ active opens

The number of connections that were established to satisfy the connect requests issued to the outside.

⑥ passive opens

The number of connect requests that were received from the outside.

⑦ attempt fails
The number of connect requests whose attempt to connect failed.

⑧ establish resets
The number of connect requests that were rejected during their processing.

⑨ current establish
The total number of TCP connections currently active.

⑩ segments received
The total number of segments (units of data each transmitted by TCP at a time) that were received.

⑪ segments sent
The total number of segments that were transmitted.

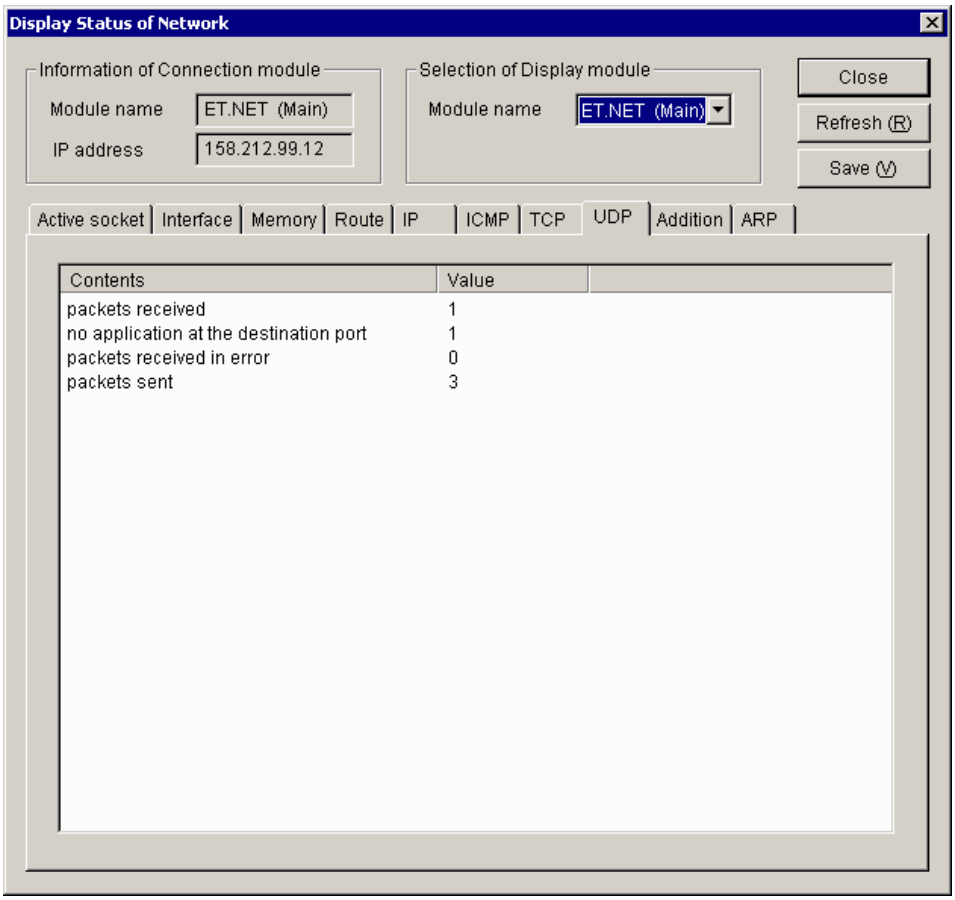
⑫ segments retransmit
The total number of segments that were retransmitted because a reception acknowledgement was received from the destination.

⑬ segments received in error
The number of received segments that contained an error.

⑭ segments send containing the RST flag
The number of received segments that contained a reset flag.

(8) UDP protocol statistics

The statistics displayed as shown below is statistical information concerning the UDP protocol.



① packets received

The total number of UDP packets that were received.

② no application at the destination port

The number of UDP packets for which no higher-level application (port number) could be found at the destination.

③ packets received in error

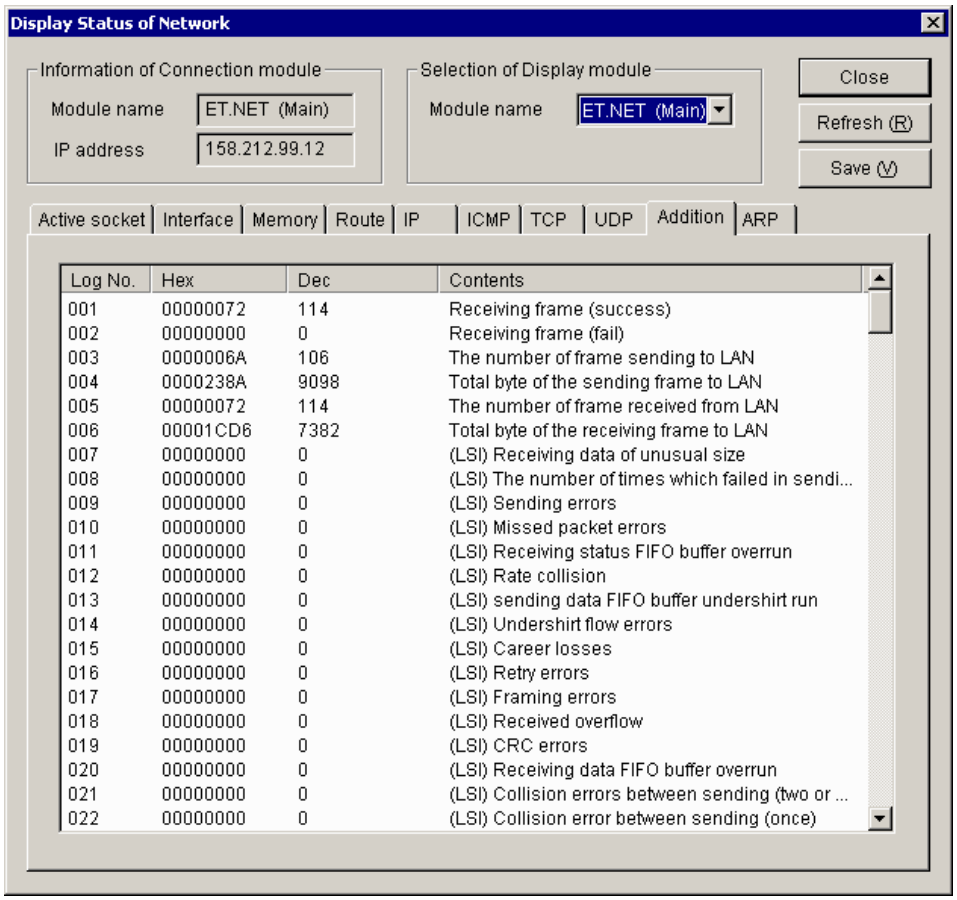
The total number of UDP packets that could not be delivered to higher-level services because of an error or some other cause.

④ packets sent

The total number of UDP packets that were transmitted.

(9) Cumulative information

The information displayed as shown below is cumulative information on the existing interfaces.



Log No.	Hex	Dec	Contents
001	00000072	114	Receiving frame (success)
002	00000000	0	Receiving frame (fail)
003	0000006A	106	The number of frame sending to LAN
004	0000238A	9098	Total byte of the sending frame to LAN
005	00000072	114	The number of frame received from LAN
006	00001CD6	7382	Total byte of the receiving frame to LAN
007	00000000	0	(LSI) Receiving data of unusual size
008	00000000	0	(LSI) The number of times which failed in sendi...
009	00000000	0	(LSI) Sending errors
010	00000000	0	(LSI) Missed packet errors
011	00000000	0	(LSI) Receiving status FIFO buffer overrun
012	00000000	0	(LSI) Rate collision
013	00000000	0	(LSI) sending data FIFO buffer undershirt run
014	00000000	0	(LSI) Undershirt flow errors
015	00000000	0	(LSI) Career losses
016	00000000	0	(LSI) Retry errors
017	00000000	0	(LSI) Framing errors
018	00000000	0	(LSI) Received overflow
019	00000000	0	(LSI) CRC errors
020	00000000	0	(LSI) Receiving data FIFO buffer overrun
021	00000000	0	(LSI) Collision errors between sending (two or ...
022	00000000	0	(LSI) Collision error between sending (once)

<Details of major cumulative info>

The following is a description of the log numbers 001 through 129 displayed as cumulative information. All log numbers other than listed are used as internal information for maintenance purposes.

- Log number 001: Receiving frame (success)
  The number of frames that were received normally.
- Log number 002: Receiving frame (fail)
  The number of frames that caused an error during reception.
- Log number 003: The number of frame sending to LAN
  The number of frames that were sent out to the communication line.
- Log number 004: Total byte of the sending frame to LAN
  The total number of bytes of the frames that were sent out to the communication line.

- Log number 005: The number of frame received from LAN
  The number of frames that were received from the communication line. This number includes the frames that were received normally or abnormally.
- Log number 006: Total byte of the receiving frame to LAN
  The total number of bytes of the frames that were received from the communication line.
- Log number 007: (LSI) Receiving data of unusual size
  The number of frames whose frame length was abnormal.
- Log number 008: (LSI) The number of times which failed in sending since 3.2 msec was exceeded
  The number of transmissions that failed due to a transmission delay (i.e., they were unable to start within 3.2 milliseconds after their initiation).
- Log number 009: (LSI) Sending errors
  The number of transmissions that were aborted.
- Log number 010: (LSI) Missed packet errors
  The number of packets that were lost during operation because the communication LSI's internal buffer was full.
- Log number 011: (LSI) Receiving status FIFO buffer overrun
  The number of reception-status FIFO overruns that occurred in the communication LSI during reception.
- Log number 012: (LSI) Late collision
  The number of late collisions (i.e., collisions detected during the transmission of the 64th or subsequent byte of data after the preamble) that occurred during transmission.
- Log number 013: (LSI) sending data FIFO buffer undershirt run
  The number of send-data FIFO underruns that occurred in the communication LSI during transmission.
- Log number 014: (LSI) Undershirt flow error
  The number of send-buffer underflow errors that occurred during transmission.
- Log number 015: (LSI) Career Sense lost
  The number of carrier losses that occurred due to a disconnected cable, a power-off condition of the hub, or some other cause during transmission.
- Log number 016: (LSI) Retry errors
  The number of retry errors (i.e., attempts to do more retries than permitted) that occurred during transmission.
- Log number 017: (LSI) Framing errors
  The number of framing errors that occurred during reception.

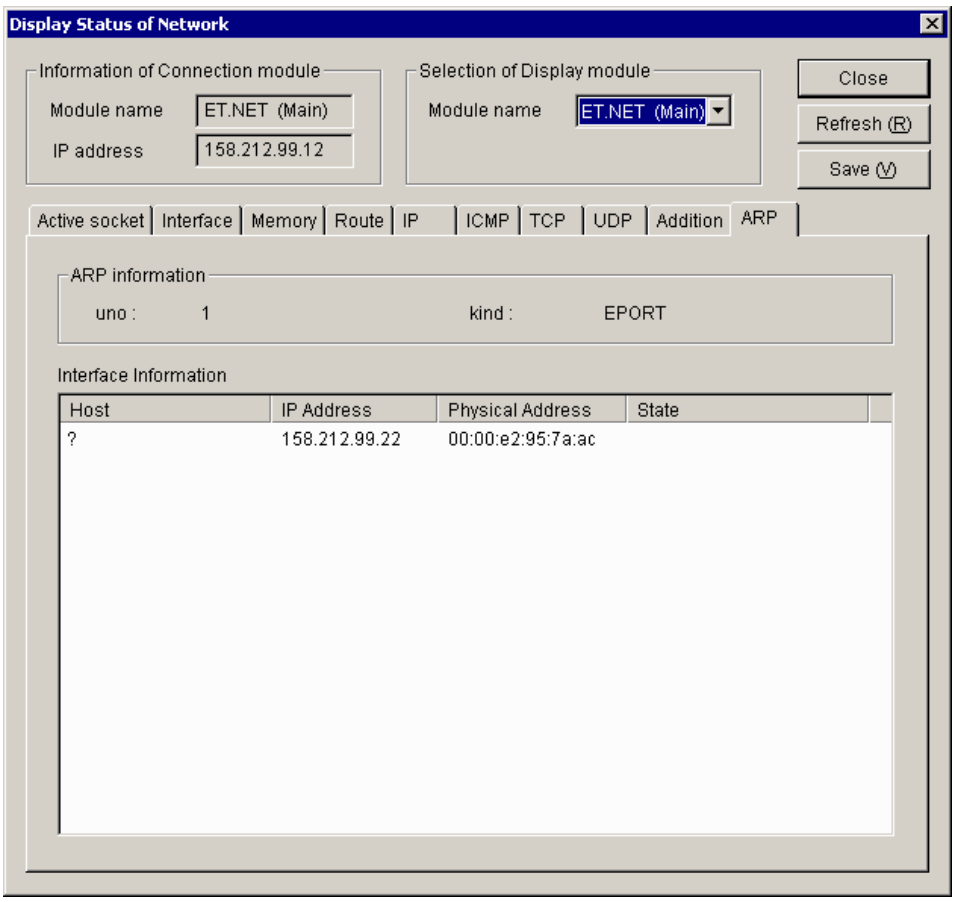
- Log number 018: (LSI) Received overflow
  The number of receive-buffer overflows that occurred during reception.
- Log number 019: (LSI) CRC errors
  The number of frame CRC errors that occurred during reception.
- Log number 020: (LSI) Receiving data FIFO buffer overrun
  The number of receive-data FIFO overruns that occurred in the communication LSI during reception.
- Log number 021: (LSI) Collision errors between sending (two or more)
  The number of times more than one collision was detected during transmission.
- Log number 022: (LSI) Collision error between sending (once)
  The number of times a single collision was detected during transmission.
- Log number 023: (LSI) Delay between sending
  The number of delays that occurred during transmission, where each transmission was terminated normally.
- Log number 024: Frame-send-timeout
  The number of frame-send-timeouts that occurred.
- Log number 129: Adapter state (top 2 byte), LINK, 10M/100Mbps, Full duplex / half-double state (bottom 2 byte)
  Data communication speed and full-duplex/half-duplex state of the ET.NET module used. Interpret this information according to the following table:

Connection type		Displayed value (*)	
		Hexadecimal	Decimal
10 Mbps	Half-duplex	00000001	1
	Full-duplex	00000005	5
100 Mbps	Half-duplex	00000003	3
	Full-duplex	00000007	7

(*) If a connection is not established over the communication line, the displayed value will be 0 (fixed).

(10) ARP table information

The information displayed as shown below is the contents of the translation table that is used by the ARP (Address Resolution Protocol) for translation of IP addresses to physical addresses.



- ARP information
  - uno

    A value of 1 is always displayed as this item.
  - kind

    The string “EPORT” is always displayed as this item.
- Interface Information

  ARP entries that are registered.
  - Host

    The host name associated with the IP address displayed.

    The host names displayed under this heading are those which are listed in the “hosts” file in the Tool currently in operation. If no host names are registered in that file, a question mark (“?”) is displayed instead.

- IP Address
  The IP address of the destination registered in the ARP table.
- Physical Address
  The physical address of the destination registered in the ARP table. If the ARP entry is invalid, the string “(incomplete)” is displayed instead.
- State
  The current state of the ARP entry. The possible states are as follows:

State name displayed	Meaning
permanent	Fixed entry
published	Proxy ARP entry

7.3.4 Error codes for socket handler-reported errors

This section provides a list of the error codes for possible errors that can be reported by socket handlers, and describes the remedial actions that the user can take to solve the error conditions.

Table 7-8 Error Codes for Socket Handler-Reported Errors

(1/3)

Error code	Meaning	Possible cause	Remedial action
F000	Connection not established yet	At the start of the handler, it is found that a connection is not established yet or the port is already released.	Establish a connection by issuing tcp_open() or tcp_popen(), and then call the handler again.
F010	Invalid socket ID	• A specified socket ID is out of the permitted range. (TCP: /01 ≤ ID ≤ 18, UDP: /20 ≤ ID ≤ /37) • A socket ID either unused or already freed is specified. • A connection is not established yet. Alternatively, it is already established (tcp_accept() only).	Review the user program. (Check that the return code returned by tcp_open() or tcp_popen() is used as the socket ID.)
F011	Too many sockets	An attempt is made to register more sockets than permitted (a total of 24 sockets may be registered for both TCP and UDP).	Close any unused sockets (by issuing tcp_close() or udp_close()), and then establish a connection again by issuing tcp_open() or tcp_popen().
F012	Socket driver timed out	• The socket driver has not responded within the fixed time period. • A requested transmission process is timed out due to a “send window full” condition or some other cause (tcp_send() only).	Terminate the connection by issuing tcp_close(), and then establish a connection again by issuing tcp_open() or tcp_popen(). If this does not solve the problem, check the cables wiring, connectors, or destination station to see if they are normal. If this error occurred in tcp_close(), terminate the connection by issuing tcp_abort(), and then establish a connection again by issuing tcp_open() or tcp_popen().
F013	Module stopped	At the start of the handler, the initialization of the socket driver is not completed within 100 seconds.	The module may have been damaged. Replace it.

(2/3)

Error code	Meaning	Possible cause	Remedial action
F020	Invalid send data length	The length of the send data is out of the permitted range. (TCP: 1 ≤ data length ≤ 4096, UDP: 1 ≤ data length ≤ 1472)	Review the user program.
F021	Invalid receive data length	The length of the receive data is out of the permitted range. (1 ≤ data length ≤ 4096)	Review the user program.
F0FF	Erroneous port release	• Port is released (by RST reception) after the start of the handler (tcp_open()). • Port is already released at the start of the handler (tcp_send() or tcp_receive()).	• Establish a connection again by issuing tcp_open() or tcp_popen(). • Issue tcp_close() and then establish a connection again by issuing tcp_open() or tcp_popen().
FFF0	Invalid address	• Both the IP address and port number of the destination station specified in udp_open() and udp_send() are zero (0). • An attempt is made to transmit to a destination station for which routing information is not set yet (udp_send() only).	• Review the user program. • Set routine information for it.
FFF3	Erroneous argument	An erroneous parameter is specified.	Review the user program, especially, the set values of the padr, buf, outinf, and tim arguments of the socket handler.
FFF5	Connection timed out	No response is received from the destination station within fixed time period.	Terminate the connection by issuing tcp_close(), and then establish a connection again by issuing tcp_open() or tcp_popen(). If this does not solve the problem, check the cable wiring, connectors, or destination station to see if they are normal.
FFF8	FIN received	FIN was received from the destination station.	Close the socket by issuing tcp_close().
FFFA	Connection forcibly terminated	The connection was forcibly terminated by the destination station (RST reception). (tcp_receive() was issued after the RST reception.)	Terminate the connection by issuing tcp_close(), and then establish a connection again by issuing tcp_open() or tcp_popen().

(3/3)

Error code	Meaning	Possible cause	Remedial action
FFFD	Socket duplicated	The same socket (port number of the destination station, or port number of the local station) is already existent.	Review the user program. This type of error can occur when a socket is terminated by issuing tcp-close() at the local station. (*)
FFFE	Illegal control block	An attempt is made to use more sockets than permitted.	Close any idle socket by issuing tcp_close() or udp_close(), and then establish a connection again by issuing tcp_open() or tcp_popen().
FFFF	Internal buffer space insufficient	• Send buffer has become full (udp_send()). • Internal registration area has become full (route_add() or arp_list()).	• Wait for a while and then issue upd_send() again. • Delete any unnecessary registration information and then issue it again.

(*) If a socket is opened by issuing tcp_open() or tcp_popen() with a specification of the local station's port number and then an attempt is made to close it by issuing tcp_close() in the same local station, the socket will enter the TIMEWAIT state (see Figure 7-2, "State Transitions Between Connection States," of "7.3.3 Meanings of network status information items"). In this situation, the socket will not be closed for approximately 20 seconds after the issuance of that tcp_close(). In the socket's TIMEWAIT state, if tcp_open() or tcp_popen() is issued with a specification of the same port number again, a "socket duplicated" error condition (error code = FFFD) will occur. In this case, take one of the following remedial actions:

- After the issuance of tcp_close(), wait for more than 20 seconds, and then issue tcp_open() or tcp_popen().
- Close the socket by issuing tcp_abort().
- Open the socket by issuing tcp_open() or tcp_popen() with no specification of the local station's port number. (In this case, a free port number will be automatically selected from among the port numbers 1024 through 2047.)

7.4 Trouble Report

Fill out this form and submit it to local source.

Your company name		Person in charge	
Data and time of occurrence	(year / month / day / hour / minute)		
Where to make contact	Address		
	Telephone		
	FAX		
	E-mail		
Model of defective module		CPU/LPU model	
OS Ver. Rev.	Program name: Ver. Rev.		
Support program	Program name: Ver. Rev.		
Symptom of defect			
Connection load	Type		
	Model		
	Wiring state		
System configuration and switch setting			
Space for correspondence			