HITACHI

USER'S MANUAL

OPTION

ET.NET

（LQE520）

S10mini S10V

SVE-1-103（G）

USER'S MANUAL

OPTION

ET.NET

（LQE520）

S10mini S10V

First Edition, February 2003, SVE-1-103(B) (out of print)
Second Edition, July 2003, SVE-1-103(C) (out of print)
Third Edition, June 2005, SVE-1-103(D) (out of print)
Fourth Edition, September 2008, SVE-1-103(E) (out of print)
Fifth Edition, February 2009, SVE-1-103(F) (out of print)
Sixth Edition, February 2011, SVE-1-103(G)

The contents of this publication may be revised without prior notice.

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 3-7)

⚠ DANGER

- To communicate under 10BASE5, ground the FG terminal of the ET.NET module by Class D grounding via the mount base FG terminal.
- Use ground lines whose size is 2mm^2 or more.

(Page 6-17)

⚠ DANGER

Never fail to discharge electricity after the test, otherwise you will get an electric shock.

(Page 6-22)

⚠ DANGER

Ground all stations by Class D or better grounding.
Leaving any station ungrounded might incur electrical shock hazards.

(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 3-3)

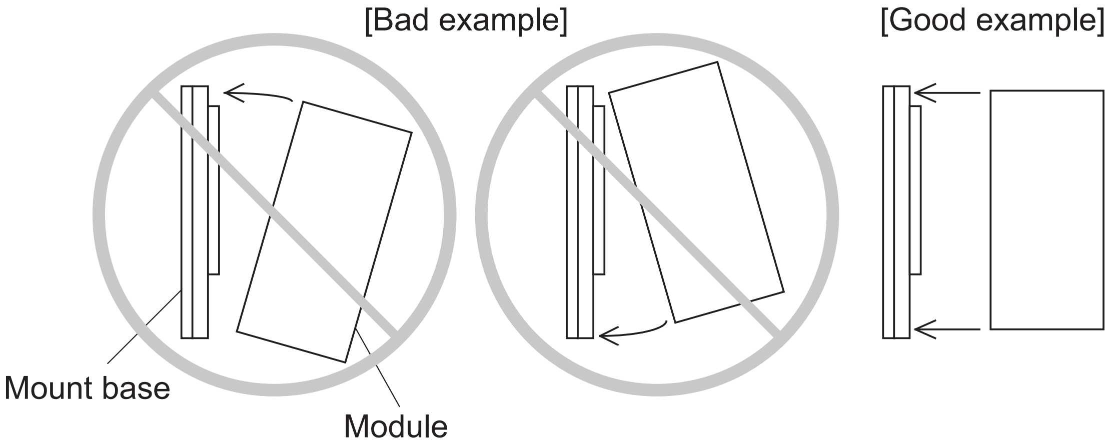
⚠ CAUTION

- 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 3-5)

⚠ 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 3-6)

⚠ 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 module number 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 10BASE5 connector, check whether the locking post is locked by the retainer.
- Do not touch the10BASE5 connector during power on. Otherwise, the system may malfunction due to static electricity, etc.

(Page 4-4)

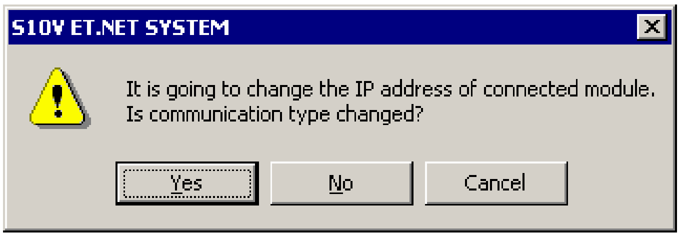
NOTICE

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 4-16)

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.


S10V ET.NET SYSTEM

It is going to change the IP address of connected module.
Is communication type changed?

If you want to reset the PCs and connect the PCs with 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 4-19)

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 Register in that window, the routing information will not be registered in the PCs or file.

(Page 5-7)

NOTICE

- When mounted on a S10V, the ET.NET modules are subject to the limitations described below.
  - ET.NET module (LQE520) before Module Rev.B (Ver-Rev: 0005-0001) doesn't provide for communication function from task using socket handler, but provides for communication with tool only. When utilizing socket handler in combination with S10V, you should use the module after Module Rev.F (Ver-Rev: 0006-0000).
  - ET.NET modules LQE520 and LQE720 cannot be used together on a single LPU unit.

  Moreover, the above-mentioned Ver-Rev is a micro program Ver-Rev of ET.NET module which is shown on "Module List" of S10V BASE SYSTEM.
- The maximum number of sockets that can be used simultaneously by one single module is 12 for TCP and 8 for UDP.
- The port numbers 0 to 9999 are reserved by the system; the user can use port numbers 10000 to 65535.
- 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 CPU or LPU. When the CPU or LPU is replaced or the operating system is reloaded, 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, take any one of the following actions for the floating socket or module:

- Execute tcp_close( ) or udp_close( ) for the floating socket by another task or built-in subroutine.
- Reset the CPU.
- Switch off the power supply, then recover it.

(Page 5-29)

NOTICE

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

(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 product and program products:

<Hardware product>
ET.NET (LQE520)

<Program products>
S-7890-29, ET.NET SYSTEM, 07-01
S-7895-29, S10V ET.NET SYSTEM, 02-03

Change Record (for SVE-1-103(F))

Description of added changes	Chapter/Section/ Subsection
The following information is newly added: setting the module no. setting switch (or, simply, MODU No. switch) in 4- or 5-position may result in a route information setting error.	2.1, 7.3.4

Change Record (for SVE-1-103(G))

Description of added changes	Chapter/Section/ Subsection
A procedure for using the ET.NET system in offline mode is added.	4.4.2
A command for loading IP address setup information in from a file is added.	4.4.5
A command for saving IP address setup information in a file is added.	4.4.6
A command for printing IP address setup information is added.	4.4.7
A command for outputting IP address setup information in CSV-format is added.	4.4.8
A command for comparing IP address setup information is added.	4.4.9

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
B	First Edition	February 2003	
E	Replacing or adding on the module is newly added.	September 2008	
F	The following information is newly added: setting the module no. setting switch (or, simply, MODU No. switch) in 4- or 5-position may result in a route information setting error.	February 2009	
G	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 S10mini and 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 S10mini and 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: LQE520
Environmentally resistant model: LQE520-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.

When mounted on a S10V, the ET.NET modules are subject to the limitations described below.

- ET.NET module (LQE520) before Module Rev.B (Ver-Rev: 0005-0001) doesn't provide for communication function from task using socket handler, but provides for communication with tool only. When utilizing socket handler in combination with S10V, you should use the module after Module Rev.F (Ver-Rev: 0006-0000).
- ET.NET modules LQE520 and LQE720 cannot be used together on a single LPU unit.

Moreover, the above-mentioned Ver-Rev is a micro program Ver-Rev of ET.NET module which is shown on "Module List" of S10V BASE SYSTEM.

<Trademarks>

- Microsoft® Windows® operating system, Microsoft® Windows® 95 operating system, Microsoft® Windows® 98 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: LQE520) is used in conjunction with an S10V LPU module or S10mini CPU module to provide communication under TCP/IP or the UDP/IP protocols by way of a local area network conforming to the IEEE802.3 specifications.

- An ET.NET module can be used together with an S10mini ET.NET module (LQE020) when used on an S10mini CPU unit (but not when used on a S10V LPU unit).
- ET.NET modules LQE520 and LQE720 cannot be used together on a single S10V LPU unit.

1.2 Specifications

1.2.1 General specifications

Item	Specifications
Model	LQE520
Maximum number of installable mount base (*)	S10mini: 2 per CPU (to be mounted, left-justified) S10V: 2 per LPU (no need to mount left-justified)
Mass	240 g

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

1.2.2 Communication specifications

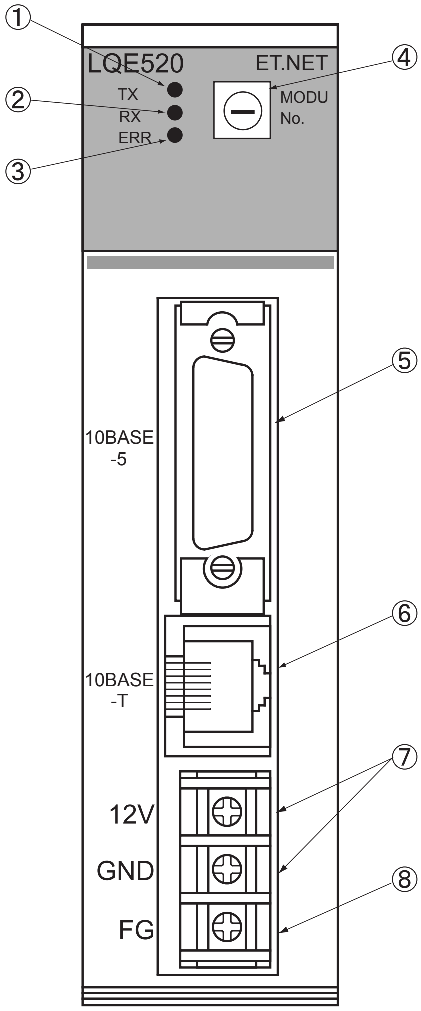
Item	Specifications
Transmission method	Serial (bit serial) transmission
Electrical interface	Conforming to IEEE802.3 (conforming to CSMA/CD standard)
Coding system	Manchester
Protocol	TCP/IP or UDP/IP
Maximum number of connectable units	10BASE5: 100 per segment 10BASE-T: n per hub (The value of n depends on the hub.)
Maximum number of stations	1024 per network
Communication cable	10BASE5 coaxial cable: Up to 500 m per segment 10BASE5 transceiver cable: Up to 50 m 10BASE-T twisted-pair cable: Up to 100 m per segment
Data transmission rate	10 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.

2 NAMES AND FUNCTIONS OF EACH PART

2.1 Names and Functions of Each Part



<table>
<tr><th>No.</th><th>Name</th><th>Function</th></tr>
<tr><td>①</td><td>TX LED</td><td>Glows when data is transmitted.</td></tr>
<tr><td>②</td><td>RX LED</td><td>Glows when data is present on the line.</td></tr>
<tr><td>③</td><td>ERR LED</td><td>Glows when a hardware fault has occurred.</td></tr>
<tr><td>④</td><td>Module number setting switch (MODU No.)</td><td>Sets the types of main module, submodule and communication port. Sets this switch with the module switched off.
<table>
<tr><th colspan="2">Module number setting</th><th rowspan="2">Explanation</th></tr>
<tr><th>Main</th><th>Sub</th></tr>
<tr><td>0</td><td>1</td><td>10BASE5 communication</td></tr>
<tr><td>2</td><td>3</td><td>10BASE-T communication</td></tr>
<tr><td>4</td><td>5</td><td>10BASE-T communication with the tool system (*1)</td></tr>
<tr><td>6</td><td>7</td><td>Error (*2)</td></tr>
<tr><td>8</td><td>9</td><td>Error (*2)</td></tr>
<tr><td>A</td><td>B</td><td>Error (*2)</td></tr>
<tr><td>C</td><td>D</td><td>Not customer-serviceable: reserved for maintenance use.</td></tr>
<tr><td>E</td><td>F</td><td>Not customer-serviceable: reserved for maintenance use.</td></tr>
</table>
The following IP address will be set when the Module number setting switch is set to 4 or 5.
IP address: 192.192.192.001</td></tr>
<tr><td>⑤</td><td>10BASE5 connector</td><td>Connector used to communicate under 10BASE5</td></tr>
<tr><td>⑥</td><td>10BASE-T connector</td><td>Connector used to communicate under 10BASE-T</td></tr>
<tr><td>⑦</td><td>Power terminal</td><td>Terminal through which power is supplied to the transceiver for communication under 10BASE5.</td></tr>
<tr><td>⑧</td><td>FG terminal</td><td>Grounding terminal. Carry out grounding as instructed in “3.4 Ground Wiring.”</td></tr>
</table>

(*1) Setting the MODU No. switch in one of these two positions may result in a route information setting error. For more information, see “7.3.4 Route information setting error table.”

(*2) For how to identify errors see “7.3 Error and Action To Be Taken.”

NOTICE

Switch off the power supply before operating the module number setting switch.
If you operate while the power supply is applied, it may result in a malfunction.

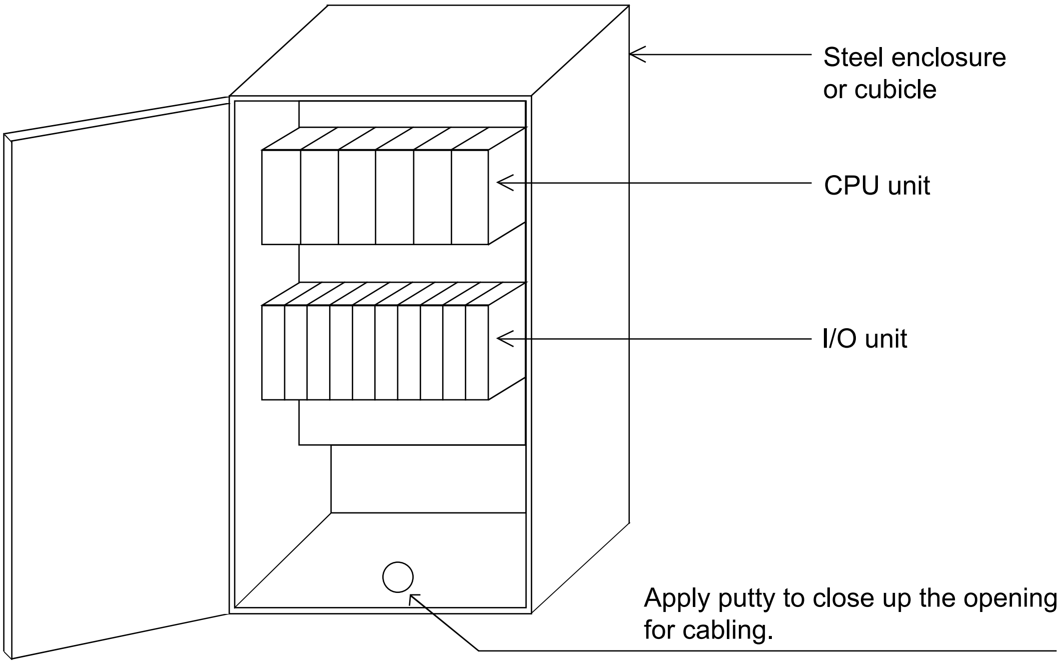
3 MOUNTING AND WIRING

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
S10mini	2-slot CPU mount base	HSC-1020
	4-slot CPU mount base	HSC-1040
	8-slot CPU mount base	HSC-1080

3.3 Mounting the Module

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

- With the S10mini Series, mount the option module at the leftmost positions without an intervening space from the CPU module. Further leave no open slots between option modules mounted.
- The S10V Series places no limitations on the mounting location and available slots.
- ET.NET modules LQE520 and LQE720 cannot be used together.
- Mount an ET.NET module in a pair with a CMU module (except when communicating with a tool system).

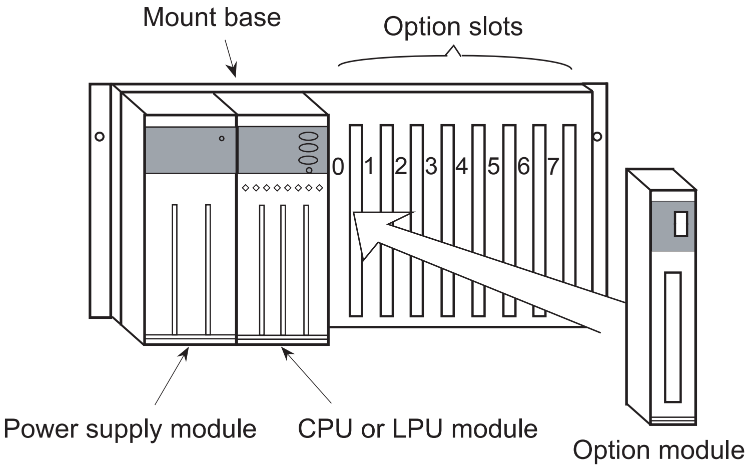


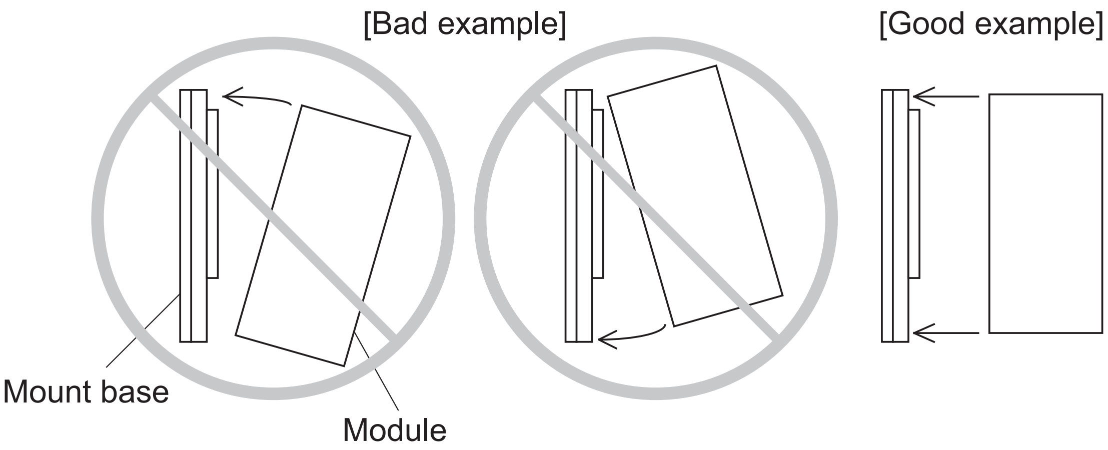
Figure 3-1 Mounting the Option Module

⚠ 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

(1) Grounding for 10BASE5

Ground the FG terminal on the front of the ET.NET module as shown below. For the wire size and length, refer to the manual pertaining to the CPU or LPU.

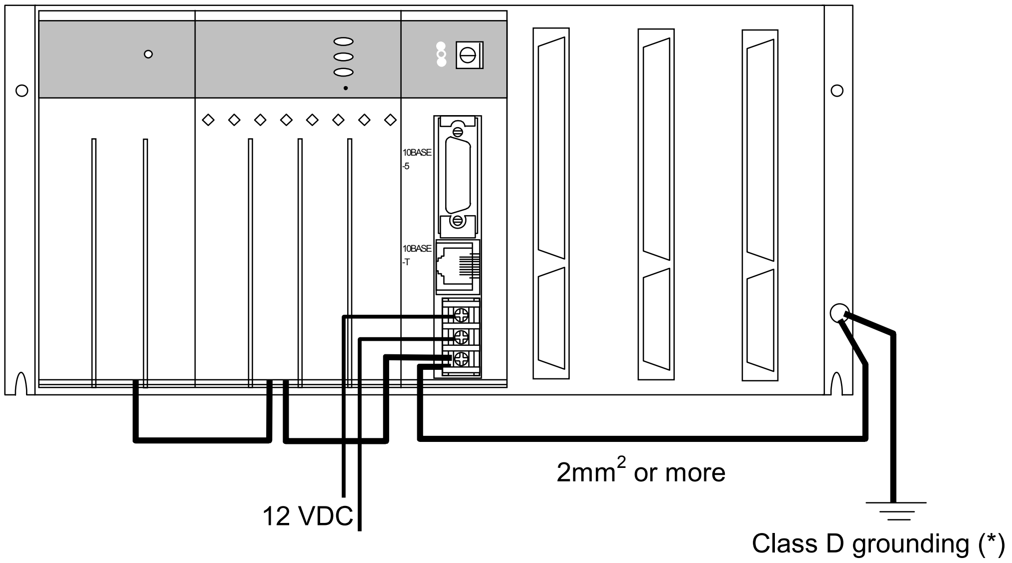


Figure 3-2 Ground Wiring

(*) 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.

(2) Grounding for 10BASE-T

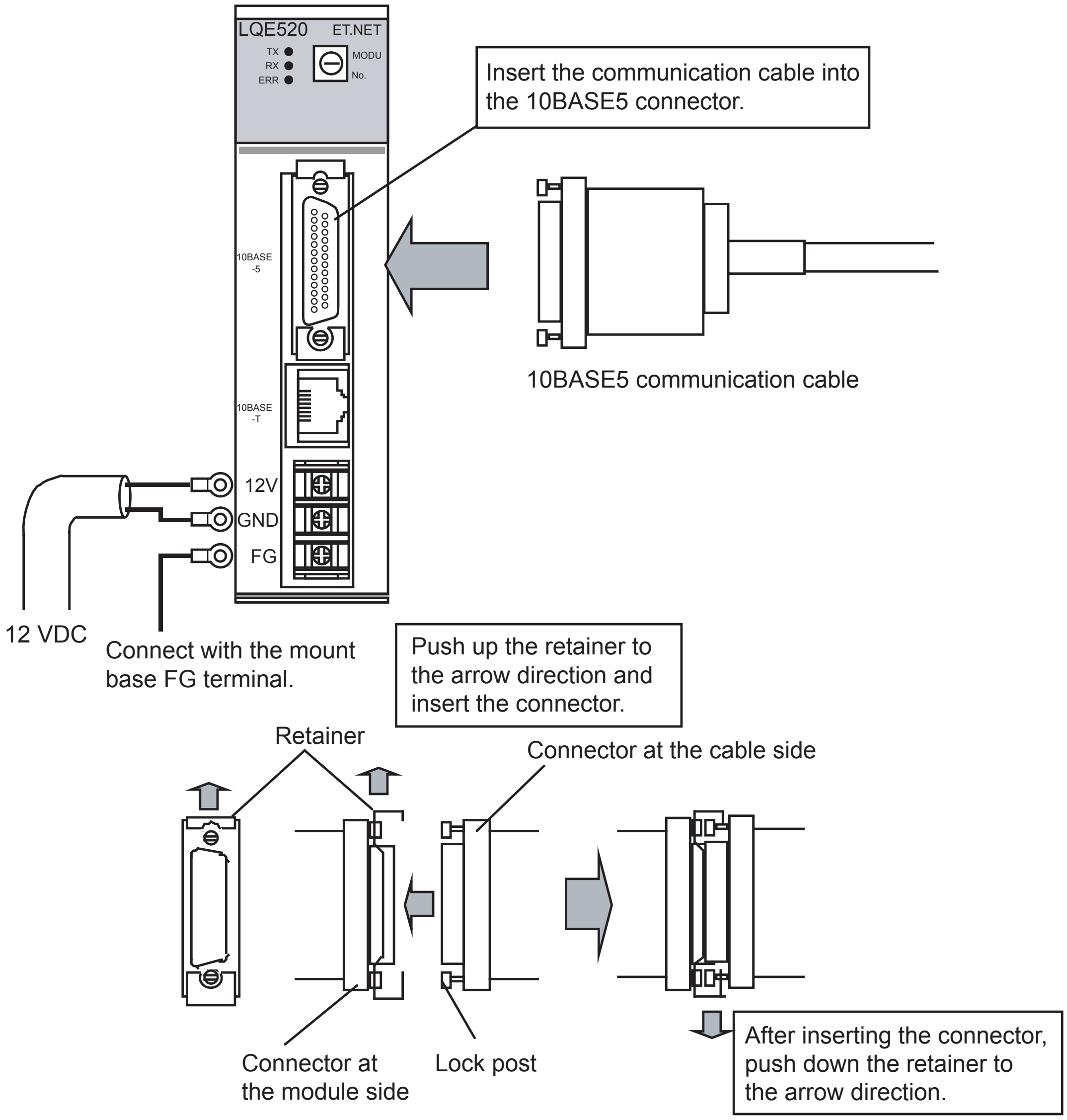
Grounding to the ET.NET module is unnecessary.

DANGER

- To communicate under 10BASE5, ground the FG terminal of the ET.NET module by Class D grounding via the mount base FG terminal.
- Use ground lines whose size is 2mm^2 or more.

3.5 Communication Cable Wiring

(1) 10BASE5 communications cable wiring



Insert the communication cable into the 10BASE5 connector.

Connect with the mount base FG terminal.

Push up the retainer to the arrow direction and insert the connector.

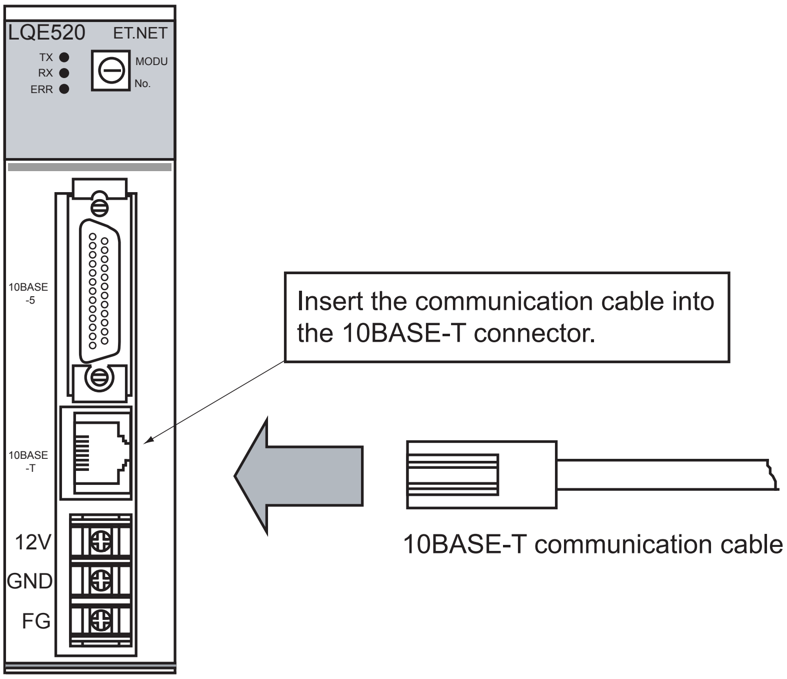
After inserting the connector, push down the retainer to the arrow direction.

Figure 3-3 10BASE5 Communication Cable Wiring

NOTICE

- This hardware unit may malfunction if it is connected poorly or has a broken line. After connecting the 10BASE5 connector, check whether the locking post is locked by the retainer.
- Do not touch the10BASE5 connector during power on. Otherwise, the system may malfunction due to static electricity, etc.

(2) 10BASE-T communication cable wiring



Insert the communication cable into the 10BASE-T connector.

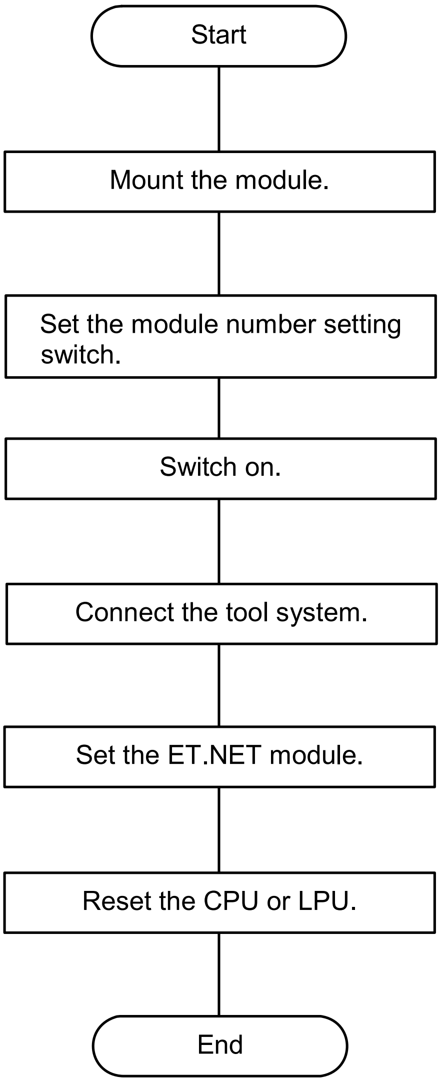
Figure 3-4 10BASE-T Communication Cable Wiring

This Page Intentionally Left Blank

4 OPERATION

4.1 Startup Procedure

To start up the ET.NET module, follow these steps:



Switch off the CPU or LPU and mount the ET.NET module.

Set the module number setting switch on the ET.NET module.

Switch on the S10mini CPU unit or S10V LPU unit.

Connect the CPU or LPU to the tool system with an RS-232C interface cable.

Set the ET.NET module.

Reset the CPU or LPU.

4.2 Installing and Starting Up the System

4.2.1 Installing

(1) Installing the S10mini ET.NET system

First check that the correct CD is on hand. The S10mini ET.NET system runs on the Microsoft® Windows® 95 operating system, Microsoft® Windows® 98 operating system, Microsoft® Windows® 2000 operating system, and Microsoft® Windows® XP operating system.

To install the S10mini ET.NET system, you must execute the setup program that is stored in the S10V ET.NET system DISK1 folder on the CD.

From the Start button, select [Settings] and then click [Control Panel]. When the Control Panel opens, double-click [Add/Remove Programs], and then click the Install button on the [Install/Uninstall] tab.

Double-click "setup.exe" that is stored in the DISK1 folder on the S10mini ET.NET system CD. Since no window opens upon completion of installation, attach a shortcut to the desktop as needed.

- Microsoft® Internet Explorer 4.01 or later is required for operating the S10mini ET.NET system. If it is not installed, the S10mini ET.NET system does not operate normally.
- Before installing the S10mini 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 S10mini ET.NET system without exiting such programs, an error may occur during installation. If such an error occurs, first uninstall the S10mini ET.NET system as directed in "4.2.2 Uninstalling," exit all the Windows®-based programs, and then install the S10mini ET.NET system again.

(2) Installing the S10V ET.NET system

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 S10V 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 Panel]. When the Control Panel opens, double-click [Add/Remove Programs], select [ET.NET SYSTEM] for S10mini or [S10V ET.NET SYSTEM] for S10V from the [Install/Uninstall] tab, and then click the Remove button. When the [Confirm File Deletion] window opens, click the Yes button.

NOTICE

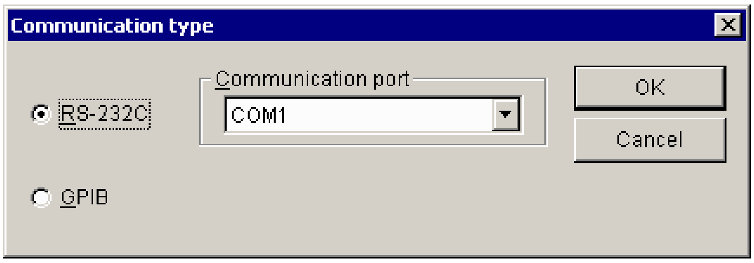
If Windows® opens a window during the uninstall process to display the question "Remove Shared File?," click the No button to retain shared files.

4.2.3 Starting up the system

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

① Select Start button – [Program] – [ET.NET SYSTEM] – [ET.NET SYSTEM] for S10mini. Select Start button – [Program] – [Hitachi S10V] – [S10V ET.NET SYSTEM] – [S10V.ET NET SYSTEM] for S10V. When a shortcut of the [ET.NET SYSTEM] or [S10V ET.NET SYSTEM] has been created on the desktop, double-click this shortcut to start the system.

② The [Communication type] window is displayed. The [Communication type] window described in Figure 4-1 will be shown when the ET.NET system is S-7890-29. The [[S10V] ET.NET] window described in Figure 4-2 will be shown when the ET.NET system is S-7895-29.



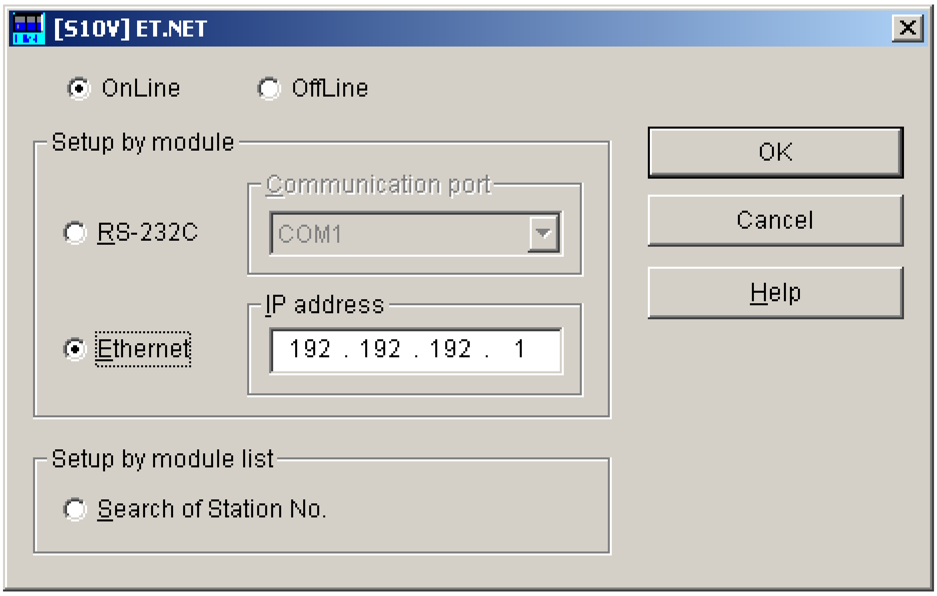
Communication type

● RS-232C

○ GPIB

Communication port: COM1

Figure 4-1 [Communication Type] Window (S10mini ET.NET System)



[S10V] ET.NET

● OnLine ○ OffLine

Setup by module

○ RS-232C — Communication port: COM1

● Ethernet — IP address: 192 . 192 . 192 . 1

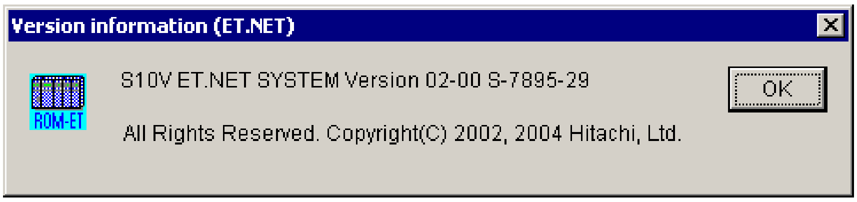
Setup by module list

○ Search of Station No.

Figure 4-2 [[S10V] ET.NET] Window

See “4.3 Commands (S10mini ET.NET System)” for further operations when the ET.NET system is S-7890-29. And see “4.4 Commands (S10V ET.NET System)” when the ET.NET system is S-7895-29.

Confirm the version of the ET.NET system in the [Version information (ET.NET)] window. The [Version information (ET.NET)] window will be displayed by selecting “Version information (ET.NET)” in the icon located in the upper left corner of the [ET.NET] window or [[S10V] ET.NET] window.



S10V ET.NET SYSTEM Version 02-00 S-7895-29

All Rights Reserved. Copyright(C) 2002, 2004 Hitachi, Ltd.

4.2.4 Shutting down the system

(1) Shutting down the S10mini ET.NET system

If you are a user of the S10mini ET.NET system, it can be shut down by clicking ☒ or the Close button in the [ET.NET SYSTEM] window (Figure 4-3).

(2) Shutting down the S10V ET.NET system

If you are a user of the S10V ET.NET system, it can be shut down by clicking ☒ or the Cancel button in the [[S10V] ET.NET] window (Figure 4-2).

4.3 Commands (S10mini ET.NET System)

4.3.1 Changing PCs connection

Function: Specify the communication type between PCs and personal computers.
Operation: The operation procedure is described below.

① The [Communication type] window will be displayed by clicking the Change connection button in the [ET.NET SYSTEM] window, or displayed automatically during the start-up process of the ET.NET system.

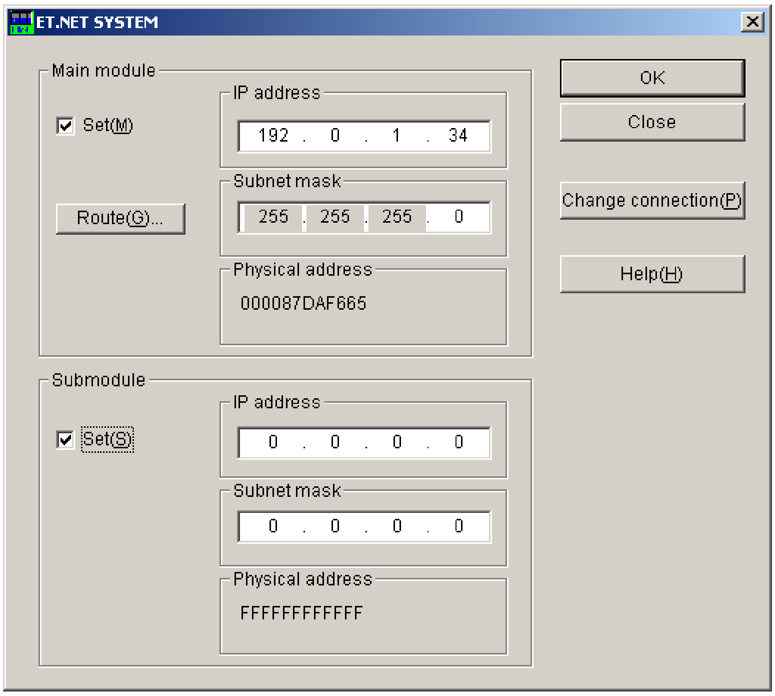
② Click the [RS-232C] radio button, and select a [Communication port] from the menu that pulls down, and click the OK button.
To leave the current setting unchanged, click the Cancel button.

> The S10mini does not support GPIB. Select “RS232C” on the [Communication type] window.

4.3.2 IP address and a subnet mask setting

Function: Set the IP addresses and subnet masks of the main module and submodule.
Operation: The operating procedure is shown below.

① Start up the [ET.NET SYSTEM] window. Set “IP address” and “Subnet mask.”



ET.NET SYSTEM

Main module
☑ Set(M)
IP address: 192 . 0 . 1 . 34
Subnet mask: 255 . 255 . 255 . 0
Physical address: 000087DAF665

Submodule
☑ Set(S)
IP address: 0 . 0 . 0 . 0
Subnet mask: 0 . 0 . 0 . 0
Physical address: FFFFFFFFFFFF

(This window is displayed when no submodule is mounted.)

Figure 4-3 [ET.NET SYSTEM] Window

② After completing the above setting, click the OK button. To cancel the setting, click the Close button.

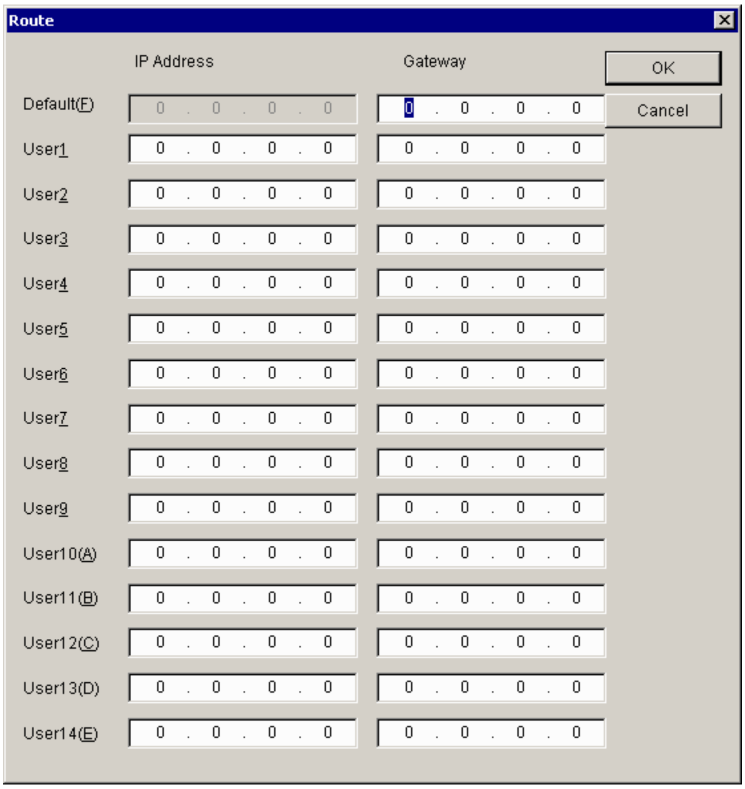
On the [ET.NET SYSTEM] window, the setting and the display method vary depending on whether the ET.NET module is installed or not as shown in the following table.

Installation status	IP address	Subnet mask	Route information	Physical address
Installed	Settable (The contents of setting are displayed.)	Settable (The contents of setting are displayed.)	Settable (Button is displayed.)	Can be referenced. (Physical address is displayed.)
Not installed	Settable (The contents of setting are displayed.)	Settable (The contents of setting are displayed.)	Not settable (Button is not displayed.)	Cannot be referenced. (FFFFFFFF is displayed.)

4.3.3 Routing information setting

Function: Set a routing table.
Operation: The procedure is shown below.

① Starting up the [ET.NET SYSTEM] window, click the Route button.
② The [Route] window is displayed. Set “IP Address” and “Gateway.”



	IP Address	Gateway
Default(F)	0 . 0 . 0 . 0	0 . 0 . 0 . 0
User1	0 . 0 . 0 . 0	0 . 0 . 0 . 0
User2	0 . 0 . 0 . 0	0 . 0 . 0 . 0
User3	0 . 0 . 0 . 0	0 . 0 . 0 . 0
User4	0 . 0 . 0 . 0	0 . 0 . 0 . 0
User5	0 . 0 . 0 . 0	0 . 0 . 0 . 0
User6	0 . 0 . 0 . 0	0 . 0 . 0 . 0
User7	0 . 0 . 0 . 0	0 . 0 . 0 . 0
User8	0 . 0 . 0 . 0	0 . 0 . 0 . 0
User9	0 . 0 . 0 . 0	0 . 0 . 0 . 0
User10(A)	0 . 0 . 0 . 0	0 . 0 . 0 . 0
User11(B)	0 . 0 . 0 . 0	0 . 0 . 0 . 0
User12(C)	0 . 0 . 0 . 0	0 . 0 . 0 . 0
User13(D)	0 . 0 . 0 . 0	0 . 0 . 0 . 0
User14(E)	0 . 0 . 0 . 0	0 . 0 . 0 . 0

Figure 4-4 [Route] Window

③ When the setup is complete, click the OK button to save the settings. Click the Cancel button to abandon the settings.

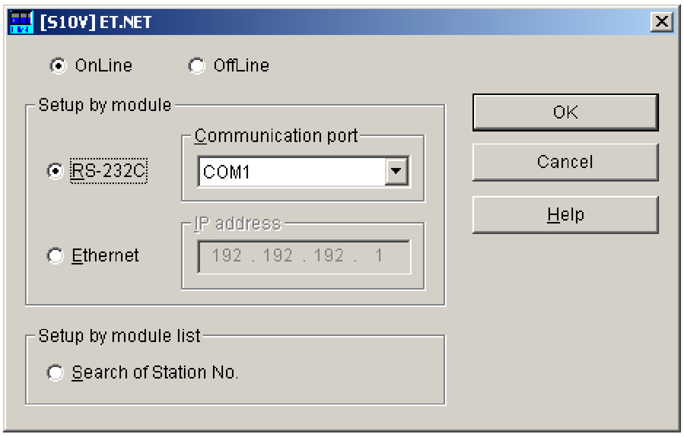
4.4 Commands (S10V ET.NET System)

4.4.1 Changing PCs Connection

Function: Specify the communication type between PCs and personal computers.
Operation: The operation procedure is described below.

① Select the type of communication between PCs and personal computer in the ET.NET system startup window.

- RS-232C connection
  Click the [RS-232C] radio button and select a [Communication port] from the menu that pulls down. The communication ports available are “COM1” through “COM4”. The default port is “COM1”.



● OnLine ○ OffLine

Setup by module

● RS-232C — Communication port: COM1

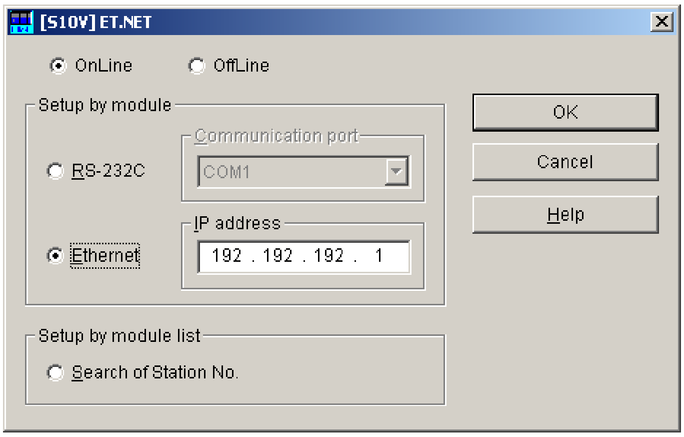
○ Ethernet — IP address: 192 . 192 . 192 . 1

Setup by module list

○ Search of Station No.

Figure 4-5 [[S10V] ET.NET] Window with [RS-232C] Radio Button Clicked

- Ethernet connection
  Click the “Ethernet” radio button and enter the IP address of the PCs that is connected.



● OnLine ○ OffLine

Setup by module

○ RS-232C — Communication port: COM1

● Ethernet — IP address: 192 . 192 . 192 . 1

Setup by module list

○ Search of Station No.

Figure 4-6 [[S10V] ET.NET] Window with [Ethernet] Radio Button Clicked

② Click the OK button when the setup has completed.　The [Setup by module] window will then appear.

The “Search of Station No.” is a function dedicated to ET.NET (LQE720).　This function is not available for ET.NET (LQE520).

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

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

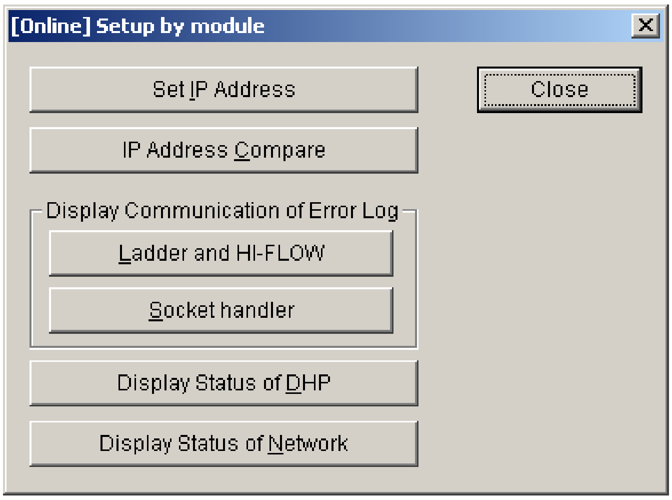


Figure 4-7 [Setup by module] Window

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 “4.4.3 IP address setting.”

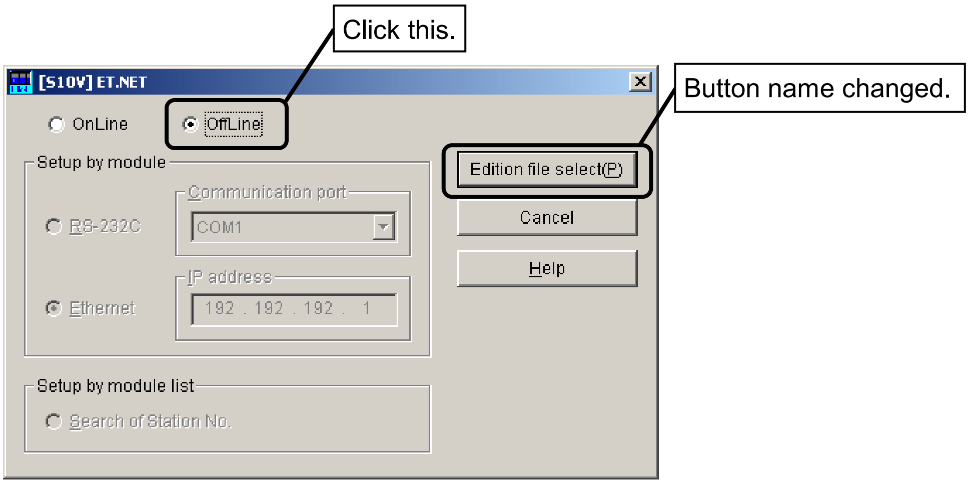
④ If you want to exit the [Setup by module] window, click the Close button. The [[S10V] ET.NET] window will then become active again.

When you use an RS-232C connection or an Ethernet connection via a hardware module other then the model-LQE720 ET.NET module, the following commands are not available:

- Ladder and HI-FLOW (Display Communication of Error Log)
- Socket handler (Display Communication of Error Log)
- Display Status of DHP
- Display Status of Network

(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.



Click this.

Button name changed.

Figure 4-8 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.

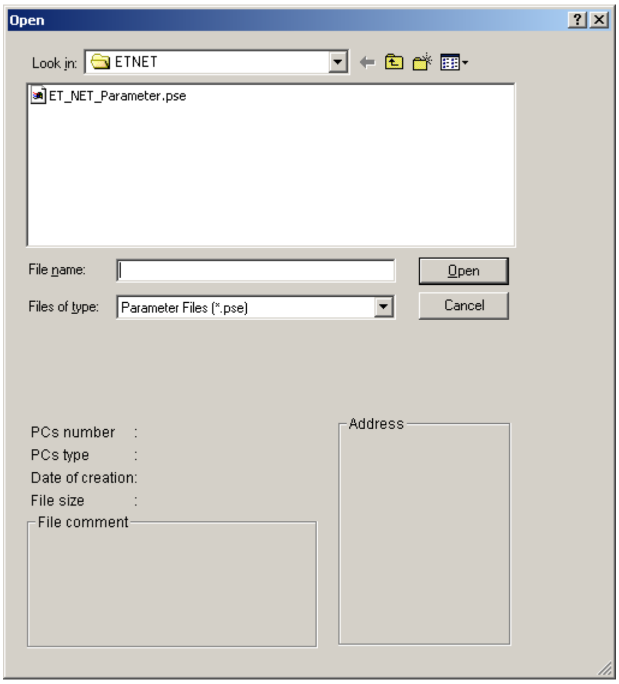
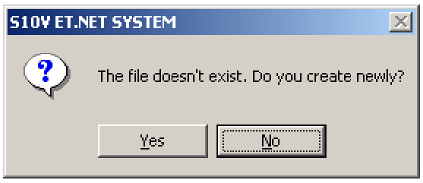


Figure 4-9 [Open] Window

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:



The file doesn't exist. Do you create newly?

Figure 4-10 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.

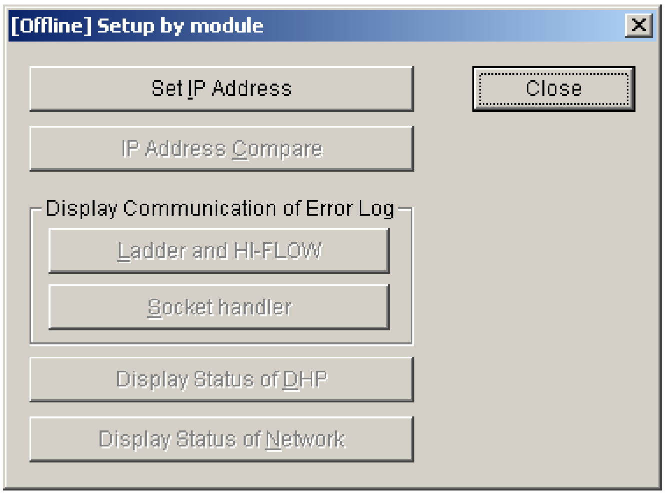


Figure 4-11 [[Offline] Setup by module] Window

③ If you want to set an IP address and other information for the ET.NET module, 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 “4.4.3 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.

4.4.3 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.

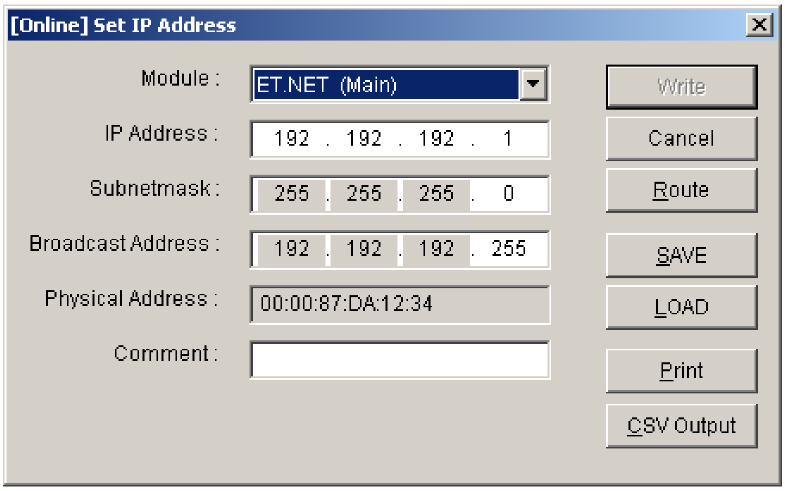
In offline mode: It is the file that is selected via the [[S10V] ET.NET] window.

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] window. The [[Online]Set IP Address] window will then appear.

② Set the desired IP address and other information.


[Online] Set IP Address

Module : ET.NET (Main)

IP Address : 192 . 192 . 192 . 1

Subnetmask : 255 . 255 . 255 . 0

Broadcast Address : 192 . 192 . 192 . 255

Physical Address : 00:00:87:DA:12:34

Comment :

Figure 4-12 [[Online] Set IP Address] Window

- Module

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

Possible choice	Remarks
ET.NET (Main)	Default
ET.NET (Sub)	

- IP Address/Subnetmask

  Set an IP address, and subnet mask for the ET.NET module. For details, see "6.5 System Definition Information."

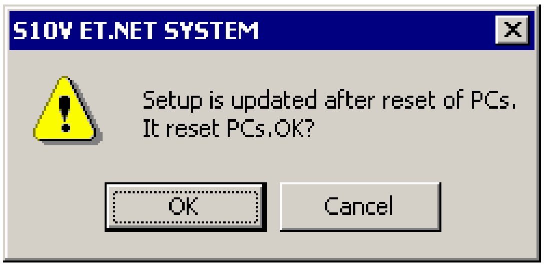
- Physical Address

  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.

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 IP addresses for all necessary routes.
For details, see “4.4.4 Routing information.”

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:



S10V ET.NET SYSTEM

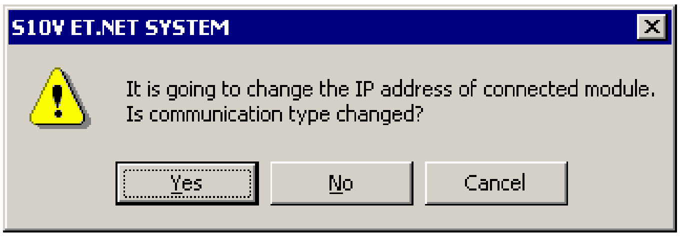
Setup is updated after reset of PCs.
It reset PCs.OK?

Figure 4-13 A “Reset” Confirmation Message

To reset the PCs, click the OK button. Then, the newly set IP address and routing 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.



S10V ET.NET SYSTEM

It is going to change the IP address of connected module.
Is communication type changed?

If you want to reset the PCs and connect the PCs with 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

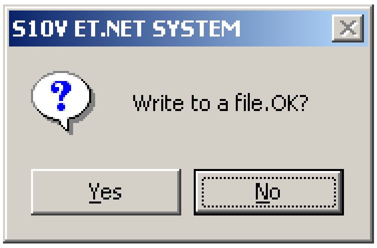
① In the [[Offline] Setup by module] window displayed, click the Set IP Address button. The [[Offline] Set IP Address] window will then appear.

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.

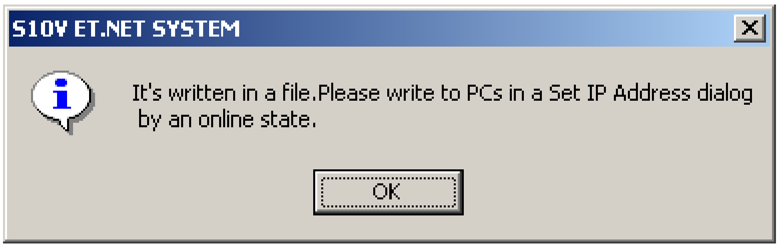


S10V ET.NET SYSTEM

Write to a file.OK?

Figure 4-14 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.



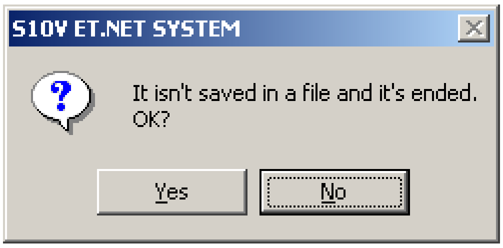
S10V ET.NET SYSTEM

It's written in a file.Please write to PCs in a Set IP Address dialog by an online state.

Figure 4-15 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.



It isn't saved in a file and it's ended.
OK?

Figure 4-16 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.

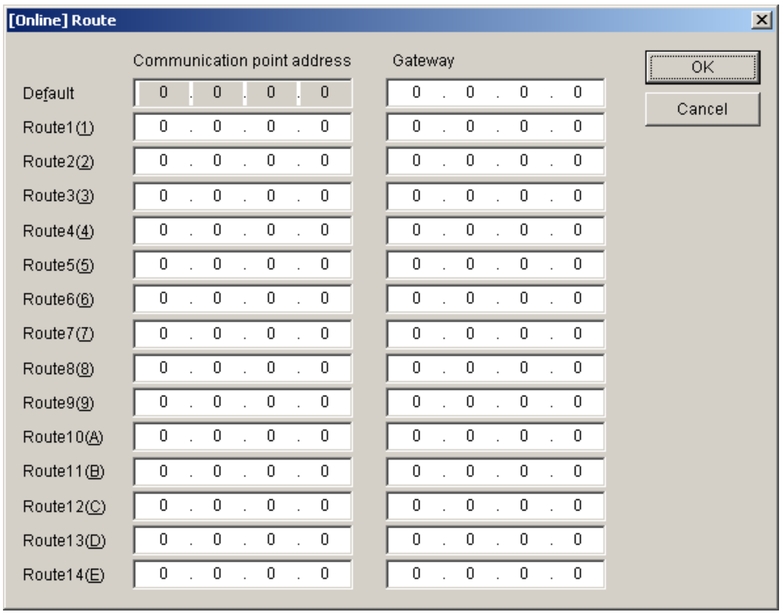
If the Set IP Address command is not in an editing state, clicking the Cancel button will not present the confirmation message shown in Figure 4-16. It will simply cause the [[Offline] Setup by module] window to become active again.

4.4.4 Routing information

Function: Set routing information for the ET.NET module.
Operation: The procedure used is shown below.

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.



	Communication point address	Gateway
Default	0 . 0 . 0 . 0	0 . 0 . 0 . 0
Route1(1)	0 . 0 . 0 . 0	0 . 0 . 0 . 0
Route2(2)	0 . 0 . 0 . 0	0 . 0 . 0 . 0
Route3(3)	0 . 0 . 0 . 0	0 . 0 . 0 . 0
Route4(4)	0 . 0 . 0 . 0	0 . 0 . 0 . 0
Route5(5)	0 . 0 . 0 . 0	0 . 0 . 0 . 0
Route6(6)	0 . 0 . 0 . 0	0 . 0 . 0 . 0
Route7(7)	0 . 0 . 0 . 0	0 . 0 . 0 . 0
Route8(8)	0 . 0 . 0 . 0	0 . 0 . 0 . 0
Route9(9)	0 . 0 . 0 . 0	0 . 0 . 0 . 0
Route10(A)	0 . 0 . 0 . 0	0 . 0 . 0 . 0
Route11(B)	0 . 0 . 0 . 0	0 . 0 . 0 . 0
Route12(C)	0 . 0 . 0 . 0	0 . 0 . 0 . 0
Route13(D)	0 . 0 . 0 . 0	0 . 0 . 0 . 0
Route14(E)	0 . 0 . 0 . 0	0 . 0 . 0 . 0

Figure 4-17 [[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 Register in that window, the routing information will not be registered in the PCs or file.

4.4.5 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 operation procedure used is described below.

① Establish a connection with the programmable controller in online mode. (For details, see "4.4.1 Changing PCs connection.")

② In the [[Online] Setup by module] window displayed, click the Set IP Address button.

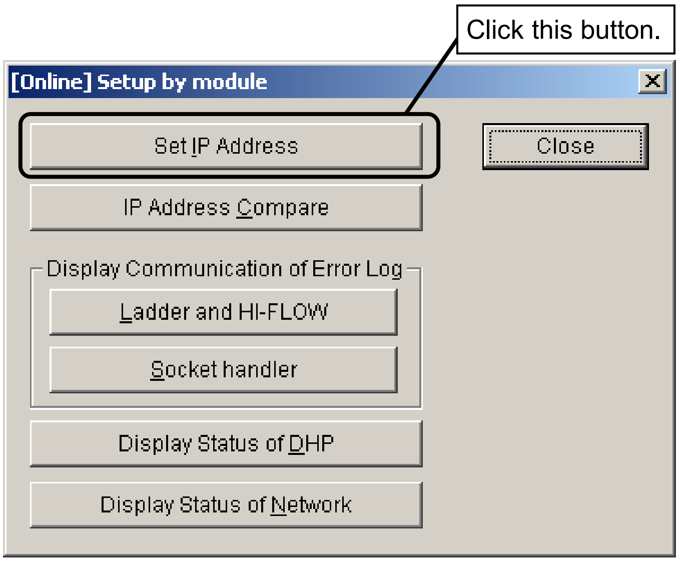
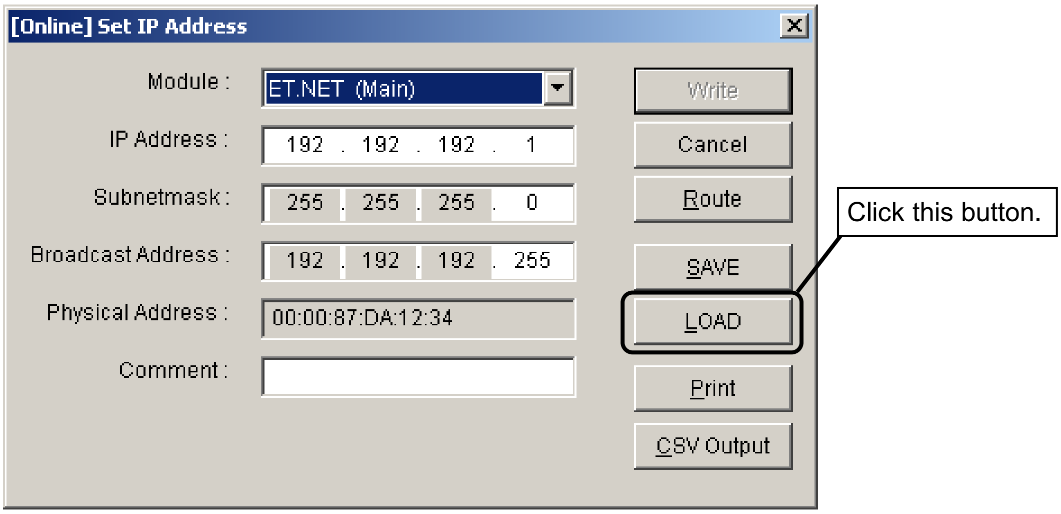


Figure 4-18 [[Online] Setup by module] Window and Clicking the Set IP Address Button

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


[Online] Set IP Address

Module : ET.NET (Main)

IP Address : 192 . 192 . 192 . 1

Subnetmask : 255 . 255 . 255 . 0

Broadcast Address : 192 . 192 . 192 . 255

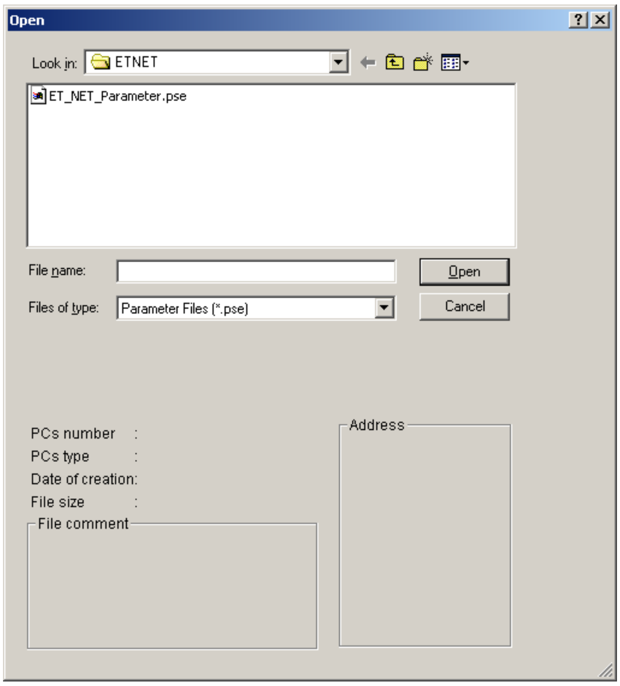
Physical Address : 00:00:87:DA:12:34

Comment :

Click this button.

Figure 4-19 [[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.


Open

Look in: ETNET

ET_NET_Parameter.pse

File name:

Files of type: Parameter Files (*.pse)

PCs number :

PCs type :

Date of creation:

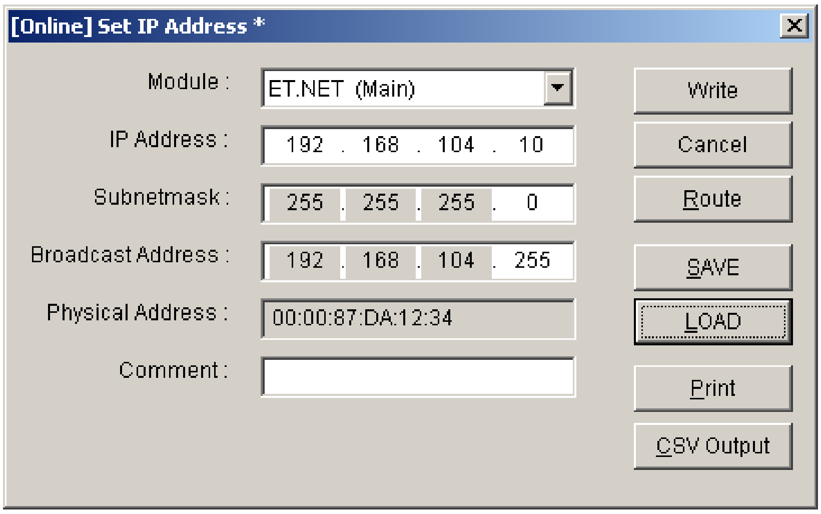
File size :

File comment

Address

Figure 4-20 [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.



[Online] Set IP Address *

Module : ET.NET (Main)

IP Address : 192 . 168 . 104 . 10

Subnetmask : 255 . 255 . 255 . 0

Broadcast Address : 192 . 168 . 104 . 255

Physical Address : 00:00:87:DA:12:34

Comment :

Figure 4-21 [[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.

4.4.6 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 operation procedure used is described below.

① Establish a connection with the programmable controller in online mode. (For details, see "4.4.1 Changing PCs connection.")

② In the [[Online] Setup by module] window displayed, click the Set IP Address button.

Click this button.

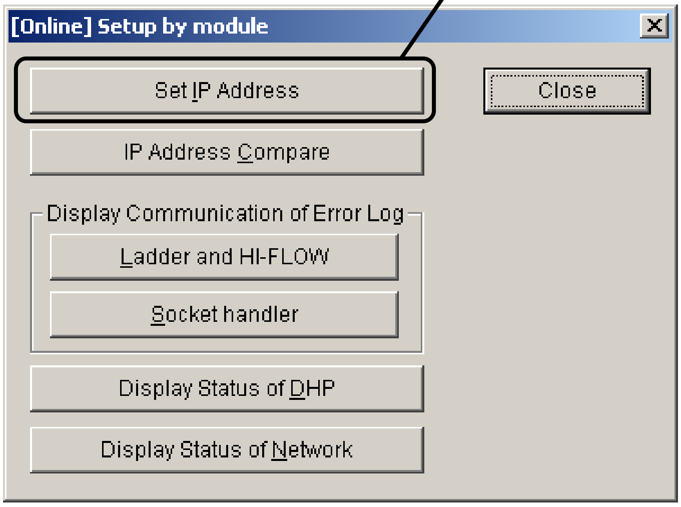
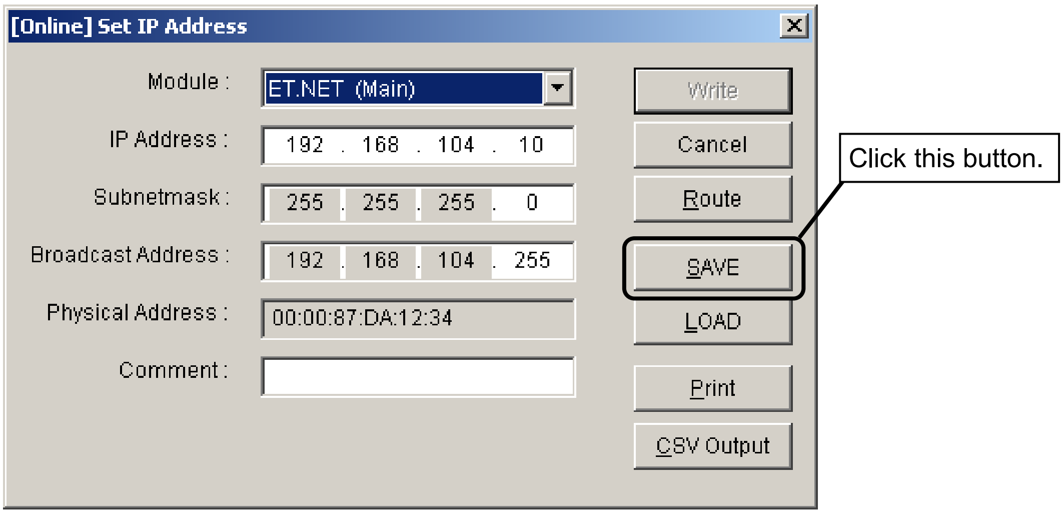


Figure 4-22 [[Online] Setup by module] Window and Clicking the Set IP Address Button

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


[Online] Set IP Address

Module : ET.NET (Main)

IP Address : 192 . 168 . 104 . 10

Subnetmask : 255 . 255 . 255 . 0

Broadcast Address : 192 . 168 . 104 . 255

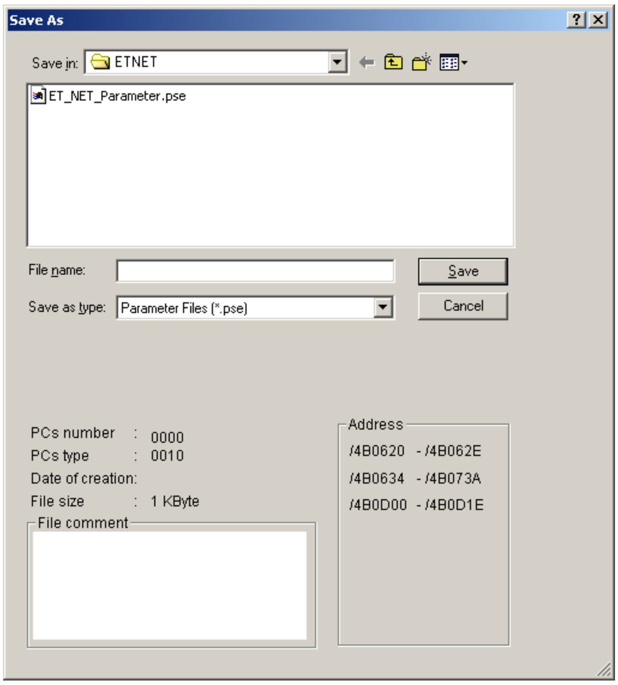
Physical Address : 00:00:87:DA:12:34

Comment :

Click this button.

Figure 4-23 [[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.


Save As

Save in: ETNET

ET_NET_Parameter.pse

File name:

Save as type: Parameter Files (*.pse)

PCs number : 0000
PCs type : 0010
Date of creation:
File size : 1 KByte

File comment

Address
/4B0620 - /4B062E
/4B0634 - /4B073A
/4B0D00 - /4B0D1E

Figure 4-24 [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.

4.4.7 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 operation procedure used is described below.

① Establish a connection with the programmable controller in online mode. (For details, see "4.4.1 Changing PCs connection.")

② In the [[Online] Setup by module] window displayed, click the Set IP Address button.

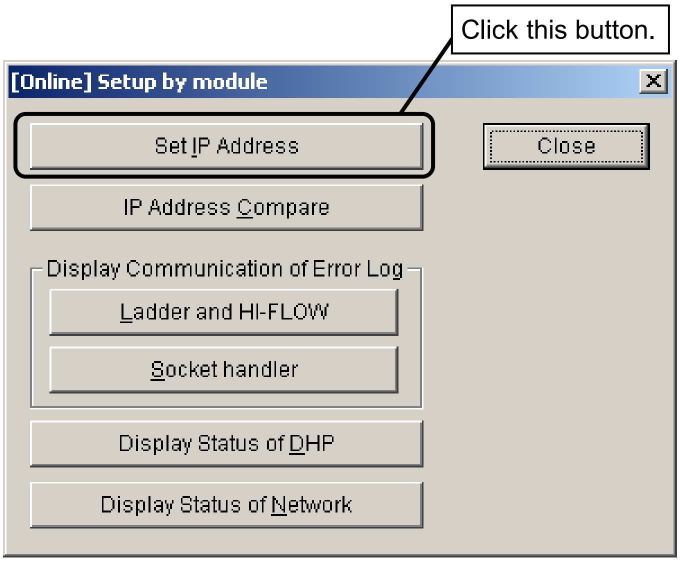
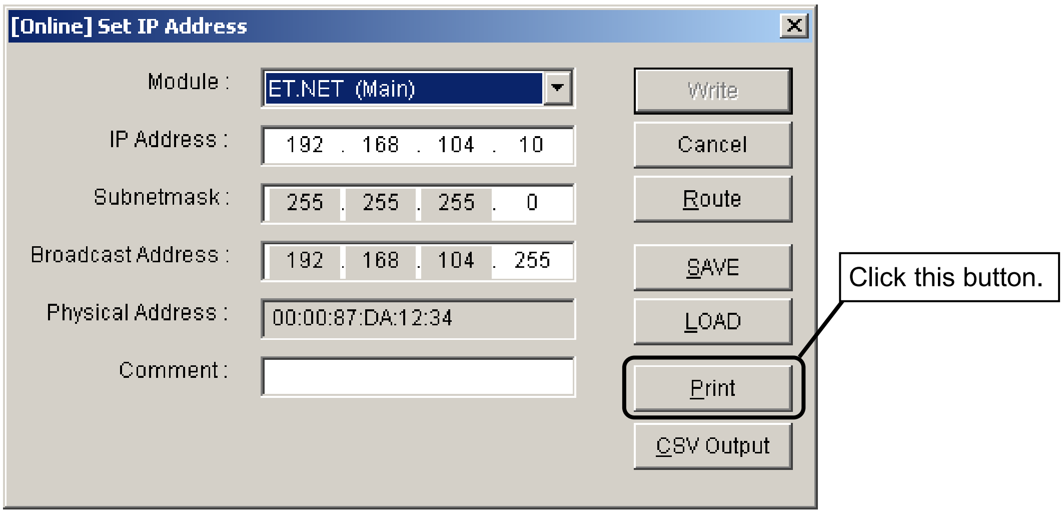


Figure 4-25 [[Online] Setup by module] Window and Clicking the Set IP Address Button

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



[Online] Set IP Address

Module : ET.NET (Main)

IP Address : 192 . 168 . 104 . 10

Subnetmask : 255 . 255 . 255 . 0

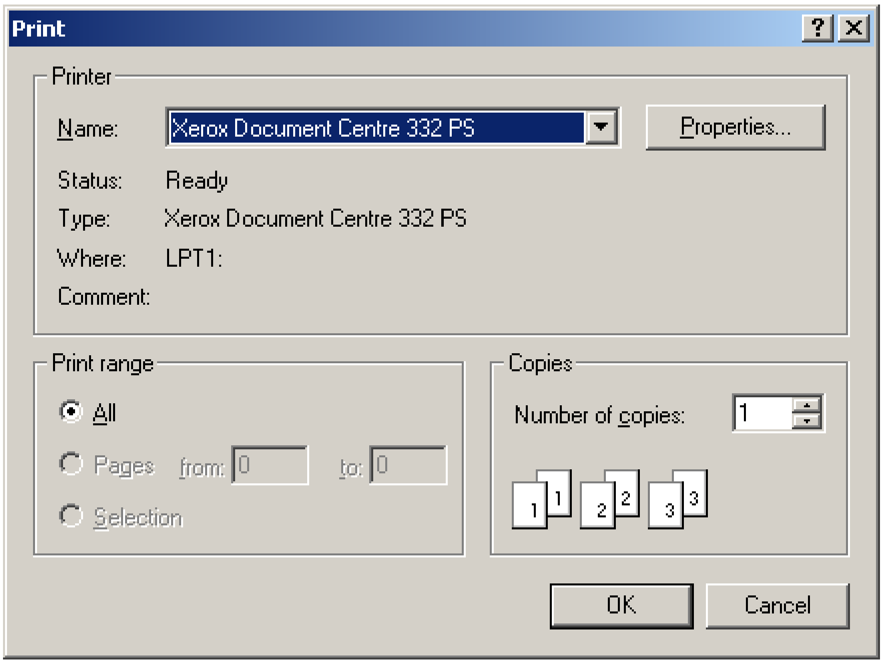
Broadcast Address : 192 . 168 . 104 . 255

Physical Address : 00:00:87:DA:12:34

Comment :

Figure 4-26 [[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.



Print

Printer

Name: Xerox Document Centre 332 PS

Status: Ready

Type: Xerox Document Centre 332 PS

Where: LPT1:

Comment:

Print range

● All

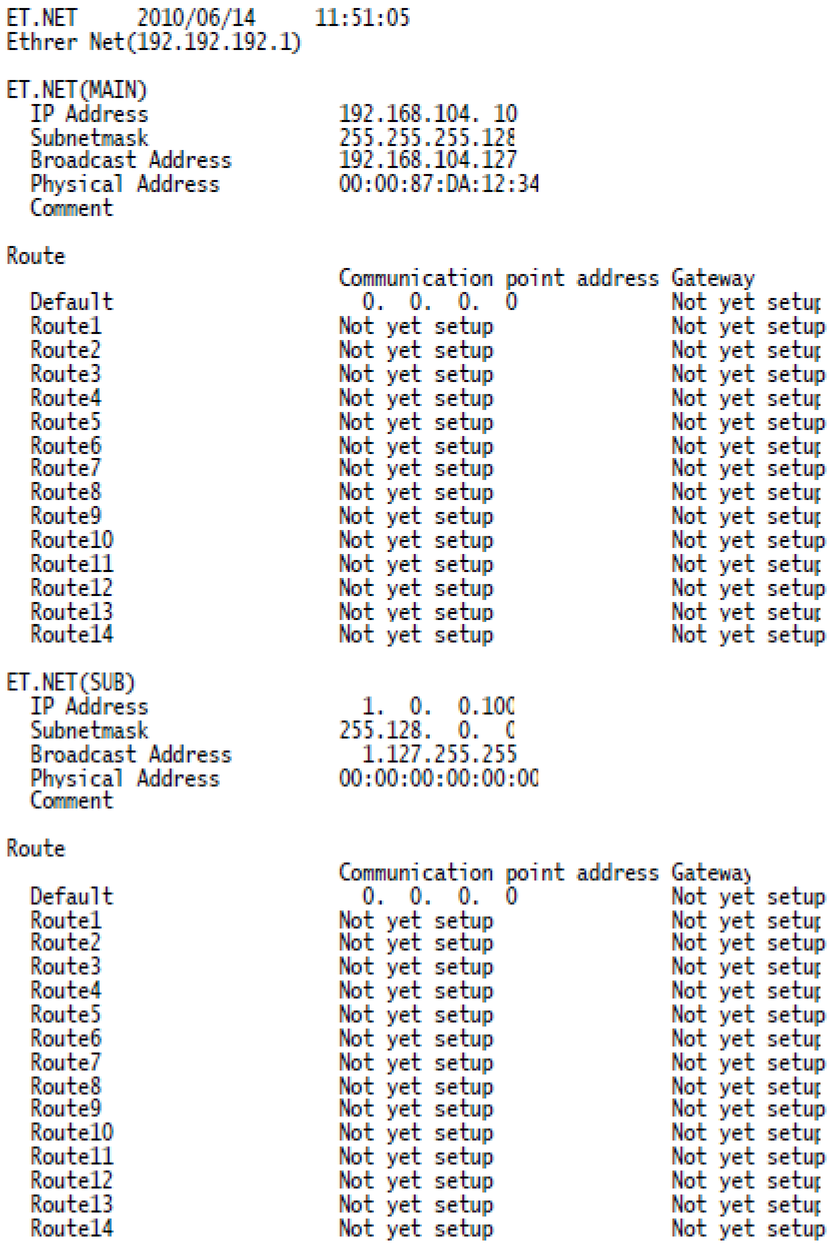
○ Pages from: 0 to: 0

○ Selection

Copies

Number of copies: 1

Figure 4-27 [Print] Dialog Box



```
ET.NET     2010/06/14     11:51:05
Ethrer Net(192.192.192.1)

ET.NET(MAIN)
  IP Address                192.168.104. 10
  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 4-28 A Printout of the Displayed IP Address Setup Information

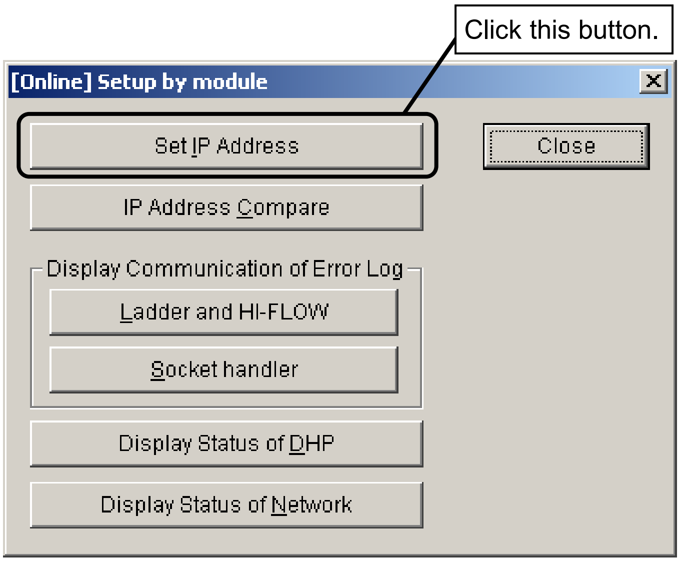
4.4.8 Output IP address setup information in CSV format

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 operation procedure used is described below.

① Establish a connection with the programmable controller in online mode. (For details, see "4.4.1 Changing PCs connection.")

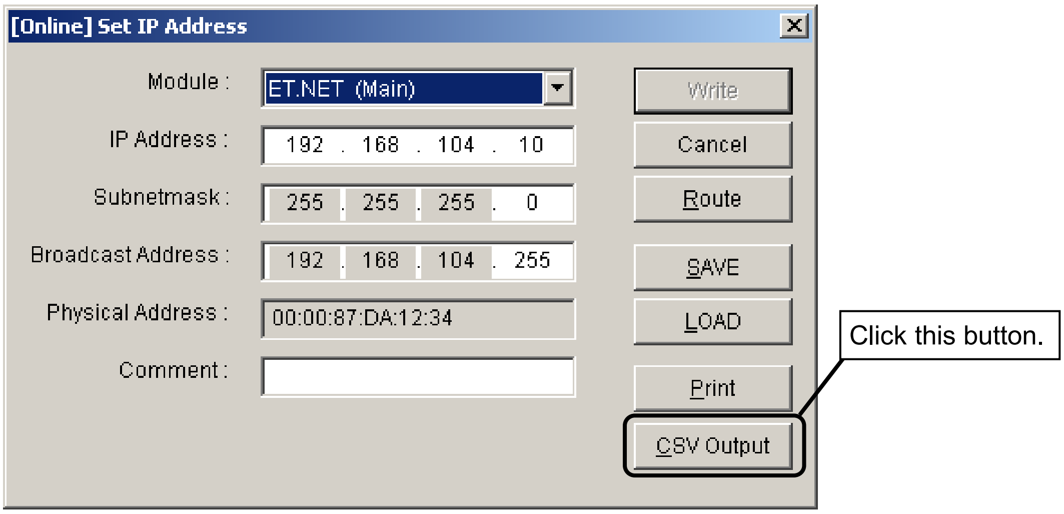
② In the [[Online] Setup by module] window displayed, click the Set IP Address button.



Click this button.

Figure 4-29 [[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.



[Online] Set IP Address

Module : ET.NET (Main)

IP Address : 192 . 168 . 104 . 10

Subnetmask : 255 . 255 . 255 . 0

Broadcast Address : 192 . 168 . 104 . 255

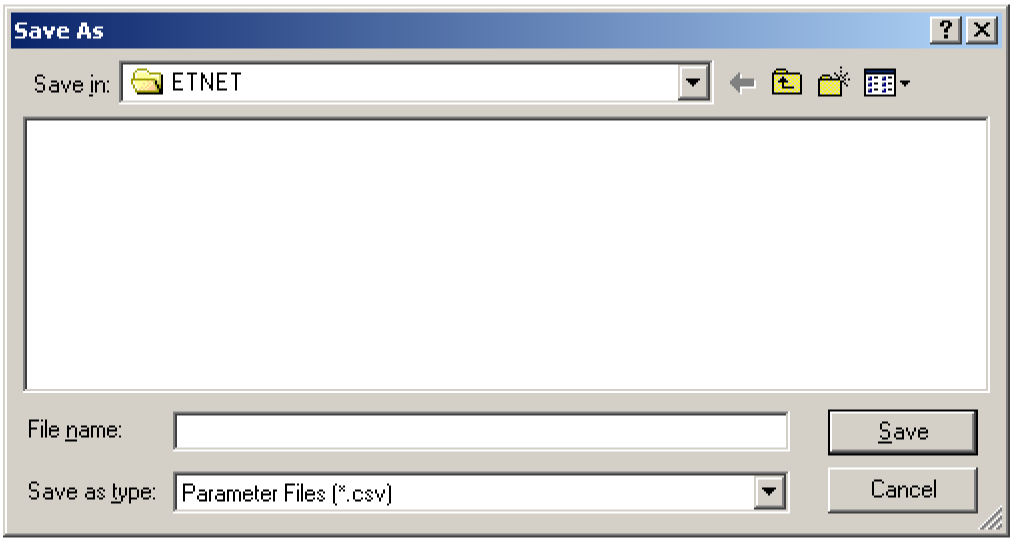
Physical Address : 00:00:87:DA:12:34

Comment :

Click this button.

Figure 4-30 [[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.



Save As

Save in: ETNET

File name:

Save as type: Parameter Files (*.csv)

Figure 4-31 [Save As] Window for Output in CSV Format

4.4.9 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 operation procedure used is described below.

① Establish a connection with the programmable controller in online mode. (For details, see "4.4.1 Changing PCs connection.")

② In the [[Online] Setup by module] window displayed, click the IP Address Compare button.

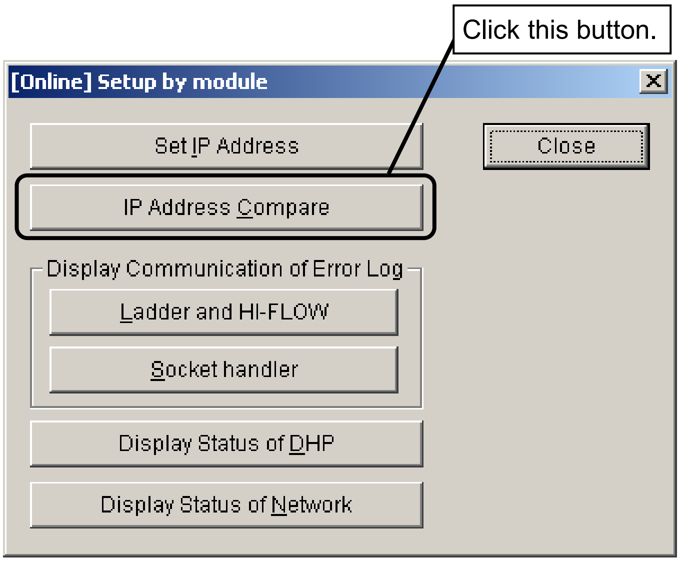


Figure 4-32 [[Online] Setup by module] Window and Clicking the IP Address Compare Button

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

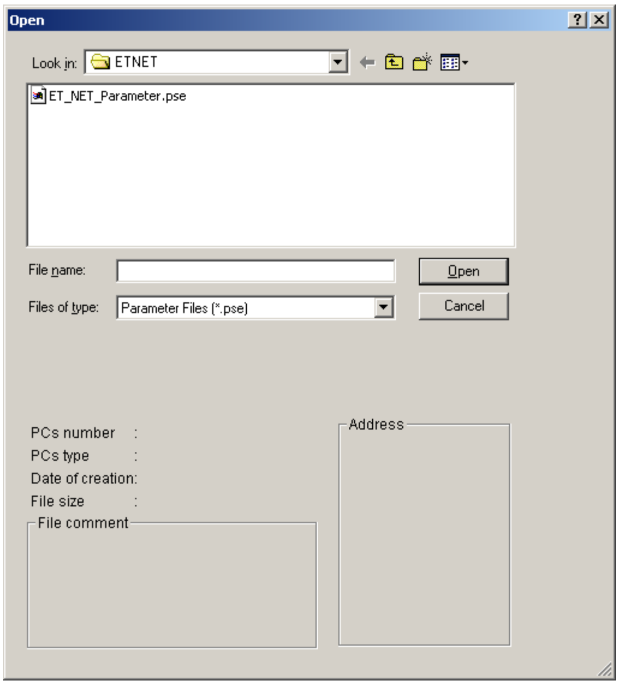
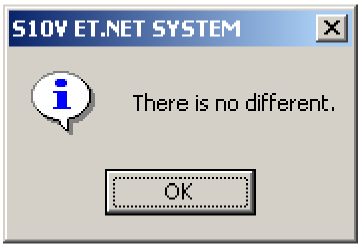


Figure 4-33 [Open] Window

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.



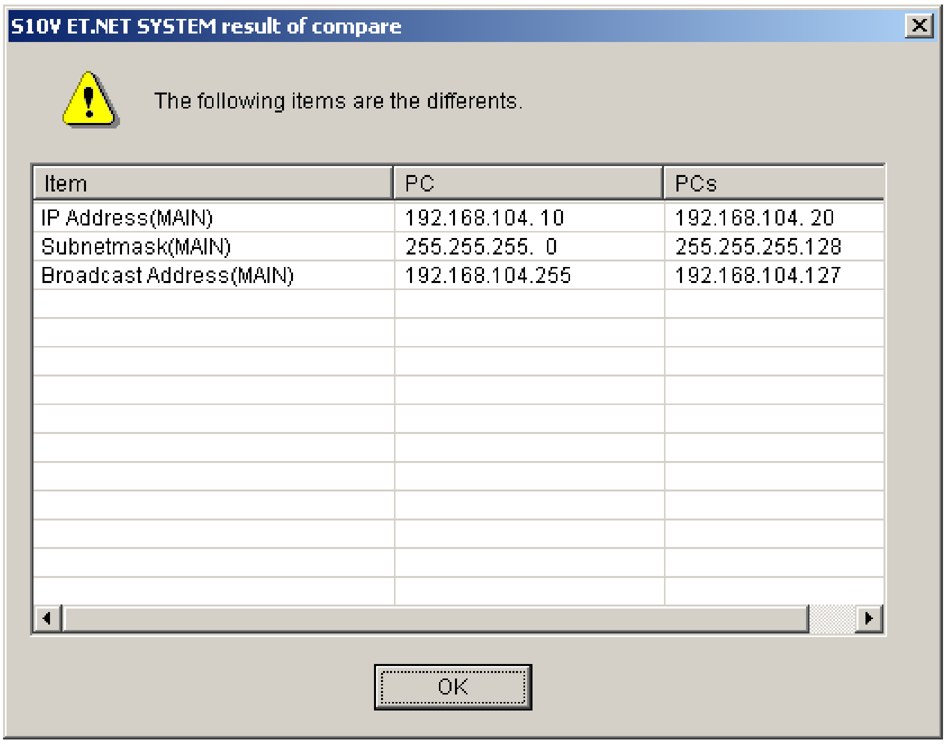
There is no different.

Figure 4-34 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.



S10V ET.NET SYSTEM result of compare

The following items are the differents.

Item	PC	PCs
IP Address(MAIN)	192.168.104. 10	192.168.104. 20
Subnetmask(MAIN)	255.255.255. 0	255.255.255.128
Broadcast Address(MAIN)	192.168.104.255	192.168.104.127

Figure 4-35 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.

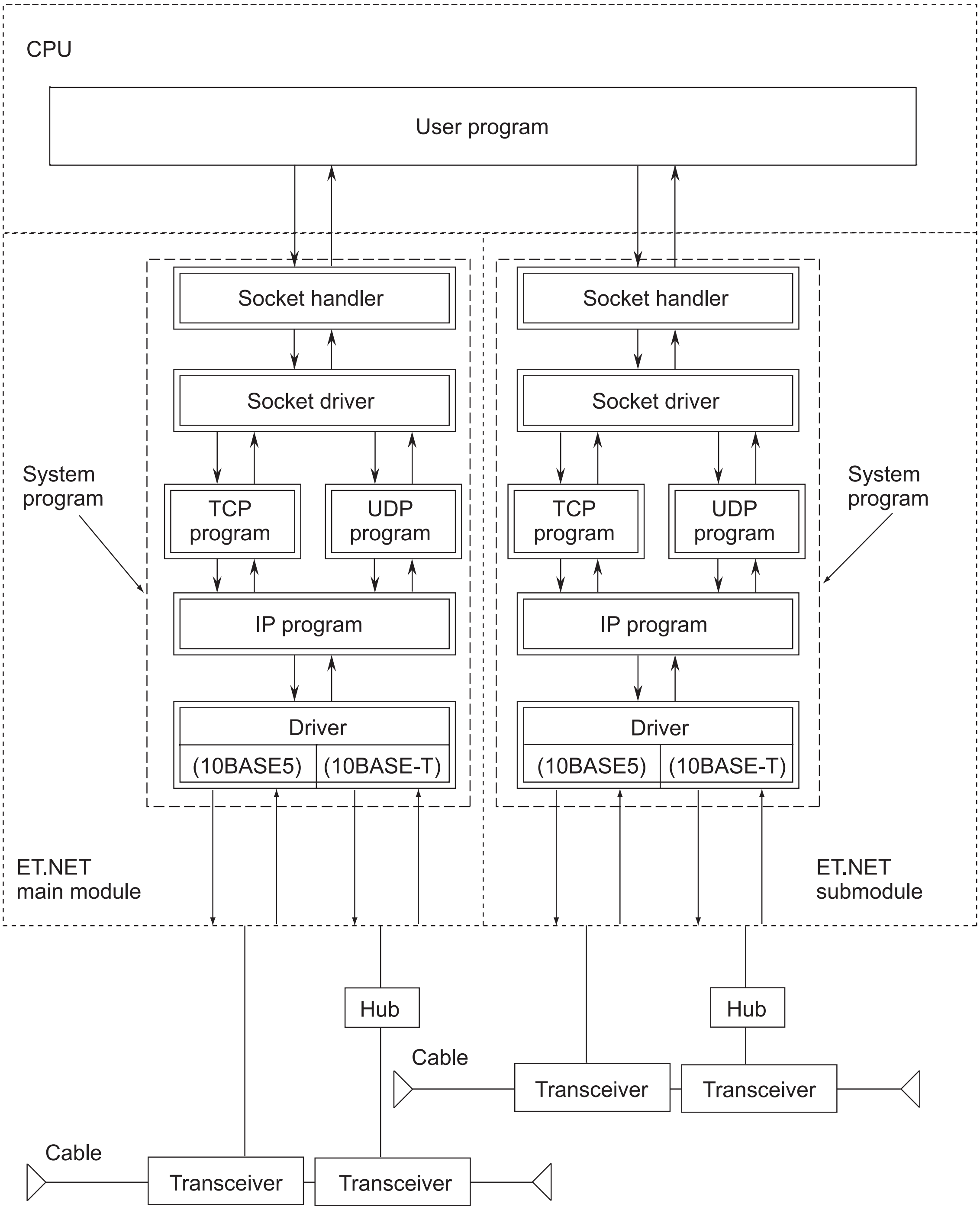


Figure 5-1 ET.NET Software Map

5.2 System Programs

The system programs are classified into the six types.

- Socket handler
- Socket driver
- TCP program
- UDP program
- IP program
- Driver

5.2.1 Socket handler

The socket handler, invoked as a function in C, controls the ET.NET module for user program. By using the socket handler, the user can create programs without considering the hardware specifications and communication protocol.

5.2.2 Socket driver

The socket driver passes commands from the socket handler to the TCP or UDP program via the memory interface for subsequent processing.

5.2.3 TCP program

The TCP program as a higher-level protocol conducts high-reliability data transmission/reception management.

The functions of the TCP program are listed below.

- Reliability check
  - Confirmation of reception response signal (ACK)
  - Sequence check by sequence numbers
  - Data checksum check
- Data retransmission (when an error is detected by reliability check)
- Flow control for receivable data amount
- Simultaneous communication with multiple processes (multiplexing)
- Logical connection by connection establishment
- Data security and priority management

5.2.4 UDP program

The UDP program as a higher-level protocol manages high-speed transmission/reception of a large amount of data.

The UDP program has the following functions:

- Connectionless communication
- Simultaneous communication
- Packet-based data transmission

5.2.5 IP program

The IP program as a low-level protocol conducts logical connection of communication paths.

The IP program has the following functions:

- Disassembling or reassembling data according to the maximum packet length
- Exchanging IP address and physical address

5.2.6 Driver

The driver controls the communication circuit, and sends data to and receives data from lines (transceivers).

The driver has the following functions:

- CRC (Cyclic Redundancy Check) for transmission/reception of data
- Data collision detection during transmission/reception and retransmission

5.3 User Program

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

5.4 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.

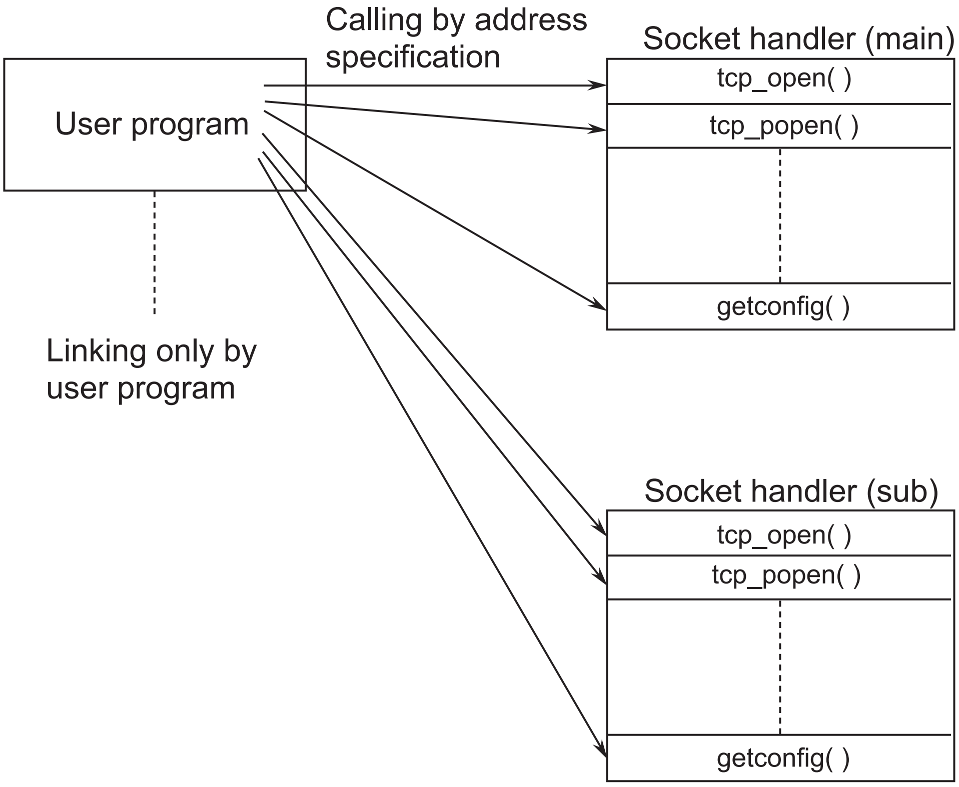


Figure 5-2 Calling a Socket Handler from a User Program

5.4.1 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 S10mimi and S10V.

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

- When mounted on a S10V, the ET.NET modules are subject to the limitations described below.
  - ET.NET module (LQE520) before Module Rev.B (Ver-Rev: 0005-0001) doesn't provide for communication function from task using socket handler, but provides for communication with tool only. When utilizing socket handler in combination with S10V, you should use the module after Module Rev.F (Ver-Rev: 0006-0000).
  - ET.NET modules LQE520 and LQE720 cannot be used together on a single LPU unit.

  Moreover, the above-mentioned Ver-Rev is a micro program Ver-Rev of ET.NET module which is shown on “Module List” of S10V BASE SYSTEM.
- The maximum number of sockets that can be used simultaneously by one single module is 12 for TCP and 8 for UDP.
- The port numbers 0 to 9999 are reserved by the system; the user can use port numbers 10000 to 65535.
- 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 CPU or LPU. When the CPU or LPU is replaced or the operating system is reloaded, 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, take any one of the following actions for the floating socket or module:

- Execute tcp_close( ) or udp_close( ) for the floating socket by another task or built-in subroutine.
- Reset the CPU.
- Switch off the power supply, then recover it.

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 75 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 for S10V.)

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 ->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.

(0 to /000F): Registered socket ID

(/F000 to /FFFF): Error occurred.

For a definition of the error code, see "7.3.3 Error codes of errors detected by socket handlers."

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

Parameters

Input parameters:

padr: Starting address of input parameters (Specify an even address for S10V.)

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.

(0 to /000F): Registered socket ID

(/F000 to /FFFF): Error occurred.

For a definition of the error code, see “7.3.3 Error codes of errors detected by socket handlers.”

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

Parameters

Input parameters:

padr: Starting address of input parameters (Specify an even address for S10V.)

padr -> s_id: Socket ID

Output parameters:

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

(0 to /000F): Registered socket ID

(/F000 to /FFFF): Error occurred.

For a definition of the error code, see “7.3.3 Error codes of errors detected by socket handlers.”

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

Parameters

Input parameters:

padr: Starting address of input parameters (Specify an even address for S10V.)

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 “7.3.3 Error codes of errors detected by socket handlers.”

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

Parameters

Input parameters:

padr: Starting address of input parameters (Specify an even address for S10V.)

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 “7.3.3 Error codes of errors detected by socket handlers.”

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

Parameters

Input parameters:

padr: Starting address of input parameters (Specify an even address for S10V.)

padr -> s_id: Socket ID

Output parameters:

outinf: Starting address of output parameters (Specify an even address for S10V.)

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 “7.3.3 Error codes of errors detected by socket handlers.”

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 for S10V.)

padr -> s_id: Socket ID

Output parameters:

outinf: Starting address of output parameters (Specify an even address for S10V.)

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 "7.3.3 Error codes of errors detected by socket handlers."

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

Parameters

Input parameters:

padr: Starting address of input parameters (Specify an even address for S10V.)

padr -> s_id: Socket ID

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

padr -> buf: Starting address of sent data

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 “7.3.3 Error codes of errors detected by socket handlers.”

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

Parameters

Input parameters:

padr: Starting address of input parameters (Specify an even address for S10V.)

padr -> s_id: Socket ID

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

padr -> buf: Starting address of receive buffer

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 "7.3.3 Error codes of errors detected by socket handlers."

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 2048 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 for S10V.)

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 “7.3.3 Error codes of errors detected by socket handlers.”

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 for S10V.)

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 “7.3.3 Error codes of errors detected by socket handlers.”

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

Parameters

Input parameters:

padr: Starting address of input parameters (Specify an even address for S10V.)

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 for S10V.)

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 “7.3.3 Error codes of errors detected by socket handlers.”

■ 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 for S10V.)

padr -> s_id: Socket ID

padr -> notuse: Fixed at 0 (unused)

padr -> buf: Starting address of receive buffer
(Specify an even address for S10V.)

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 “7.3.3 Error codes of errors detected by socket handlers.”

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 16 [routes].) The number of obtained entries is returned as the return value. If a 0 is specified for parameter len, only the number of obtained entries is returned. For len, specify a multiple of 16 (bytes).

Linking procedure

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

Parameters

Input parameters:

padr: Starting address of input parameters (Specify an even address for S10V.)

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 for S10V.)

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

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 for S10V.)

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 “7.3.3 Error codes of errors detected by socket handlers.”

route_add( )

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

Linking procedure

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

Parameters

Input parameters:

padr: Starting address of input parameters (Specify an even address for S10V.)

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 “7.3.3 Error codes of errors detected by socket handlers.”

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, only the number of obtained entries is returned. For len, specify a multiple of 12 (bytes).

Linking procedure

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

Parameters

Input parameters:

padr: Starting address of input parameters (Specify an even address for S10V.)

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 for S10V.)

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

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 for S10V.)

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 “7.3.3 Error codes of errors detected by socket handlers.”

arp_add( )

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

Linking procedure

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

Parameters

Input parameters:

padr: Starting address of input parameters (Specify an even address for S10V.)

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.

(3): Normal termination

(/F000 to /FFFF): Error occurred.

For a definition of the error code, see “7.3.3 Error codes of errors detected by socket handlers.”

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 for S10V.)

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 (8)
        char udp_num: Maximum number of UDP sockets (8)
        char rt_num: Size of routing information table (16)
        char arp_num: Size of ARP information table (32)
        short tcp_win: Size of TCP send/receive window (1024)
};
```

5.5 Examples of Socket Handler Issuance Procedure

5.5.1 Example of using TCP/IP program

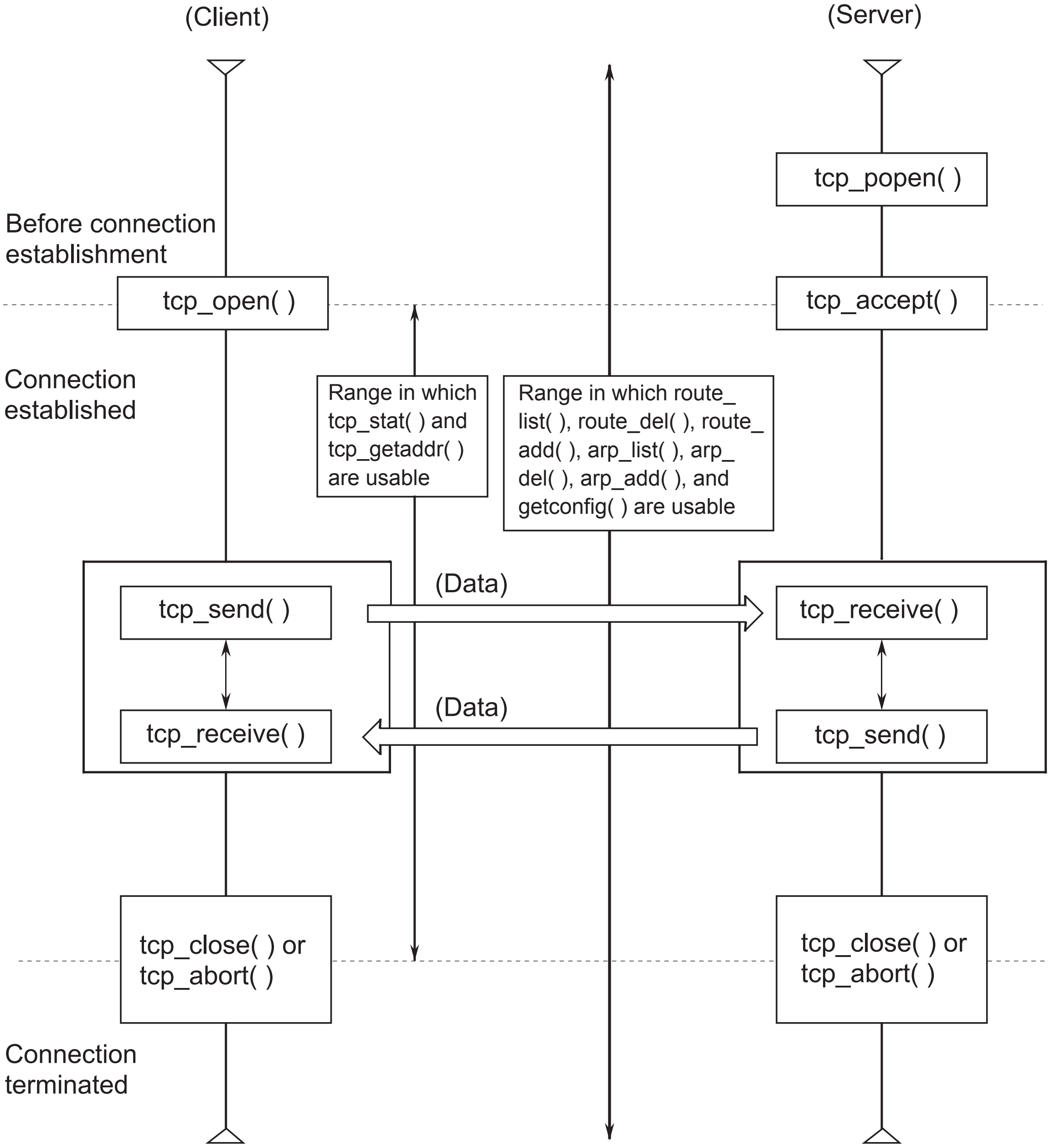

(Client)
(Server)
tcp_popen( )
Before connection establishment
tcp_open( )
tcp_accept( )
Connection established
Range in which tcp_stat( ) and tcp_getaddr( ) are usable
Range in which route_list( ), route_del( ), route_add( ), arp_list( ), arp_del( ), arp_add( ), and getconfig( ) are usable
tcp_send( )
(Data)
tcp_receive( )
tcp_receive( )
(Data)
tcp_send( )
tcp_close( ) or tcp_abort( )
tcp_close( ) or tcp_abort( )
Connection terminated

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

5.5.2 Example of using UDP/IP program

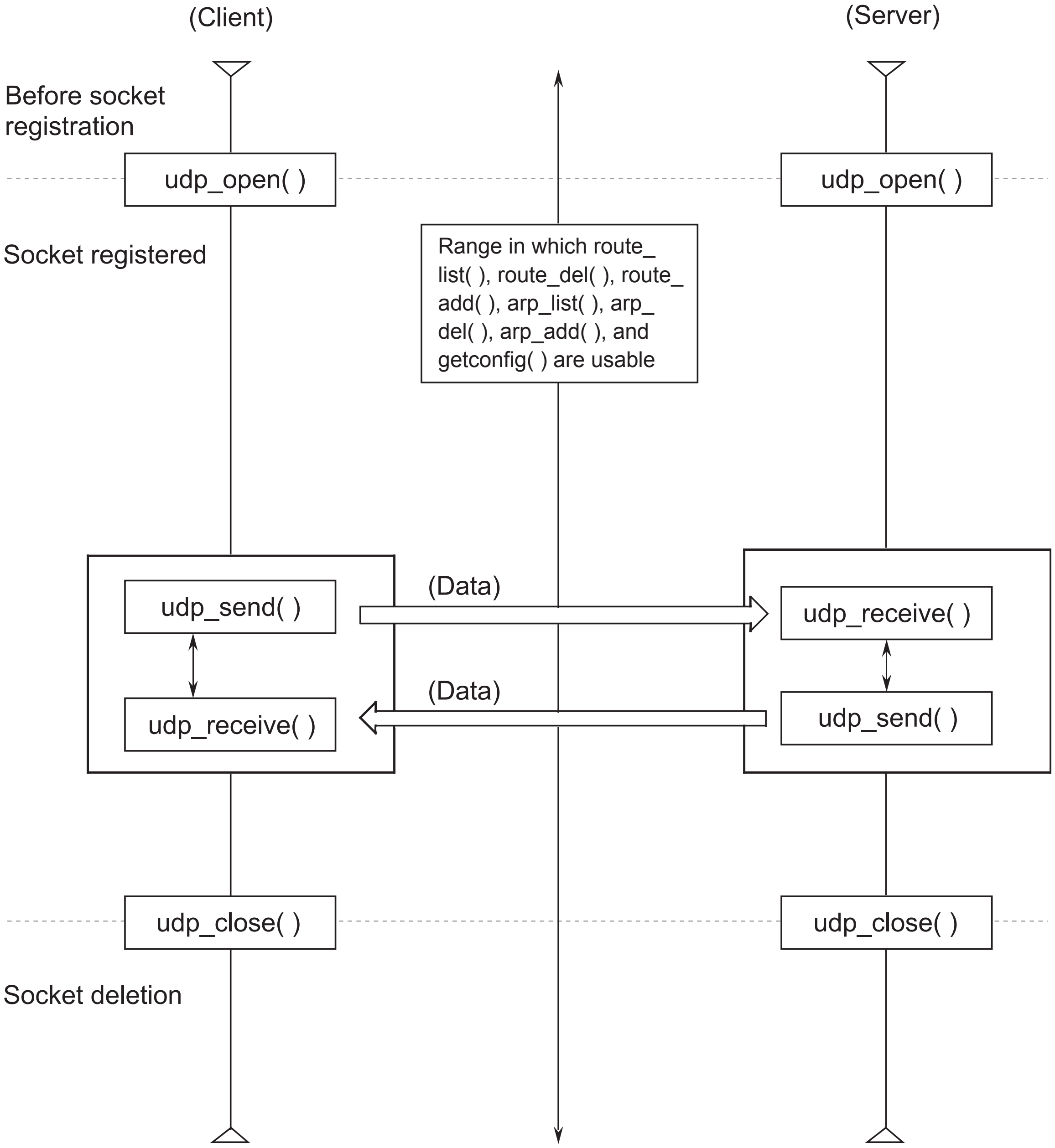
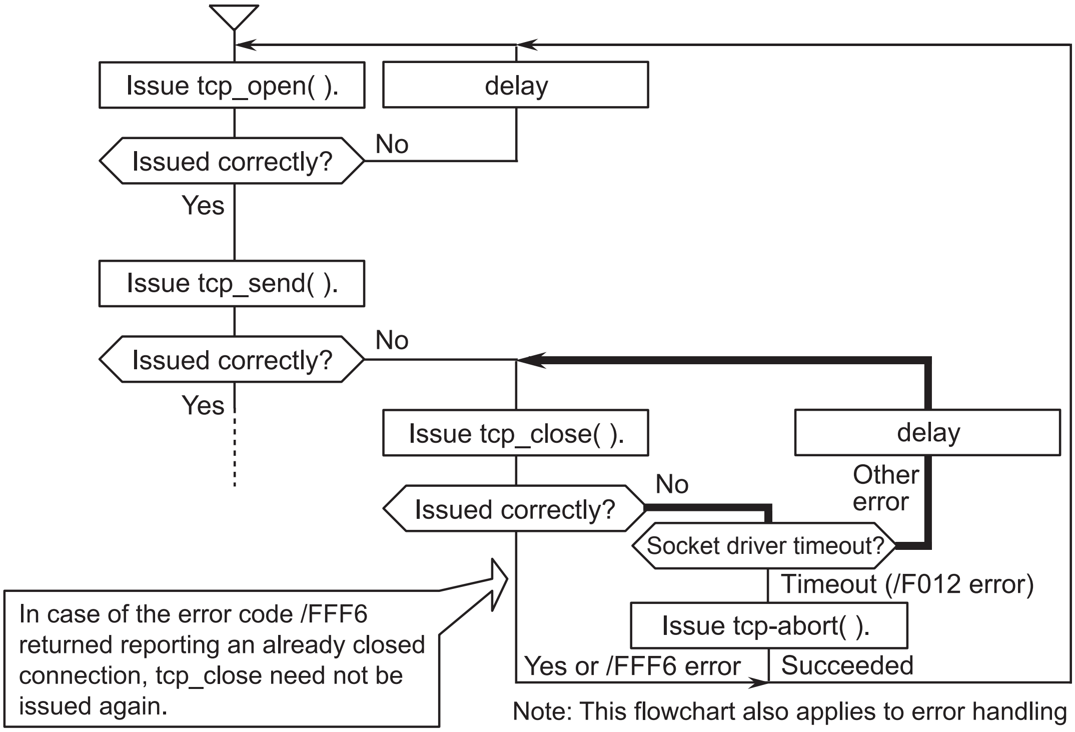


Figure 5-4 Example of Releasing a Socket Handler When Running a UDP/IP Program

- 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.



Note: This flowchart also applies to error handling for udp_close.

- 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

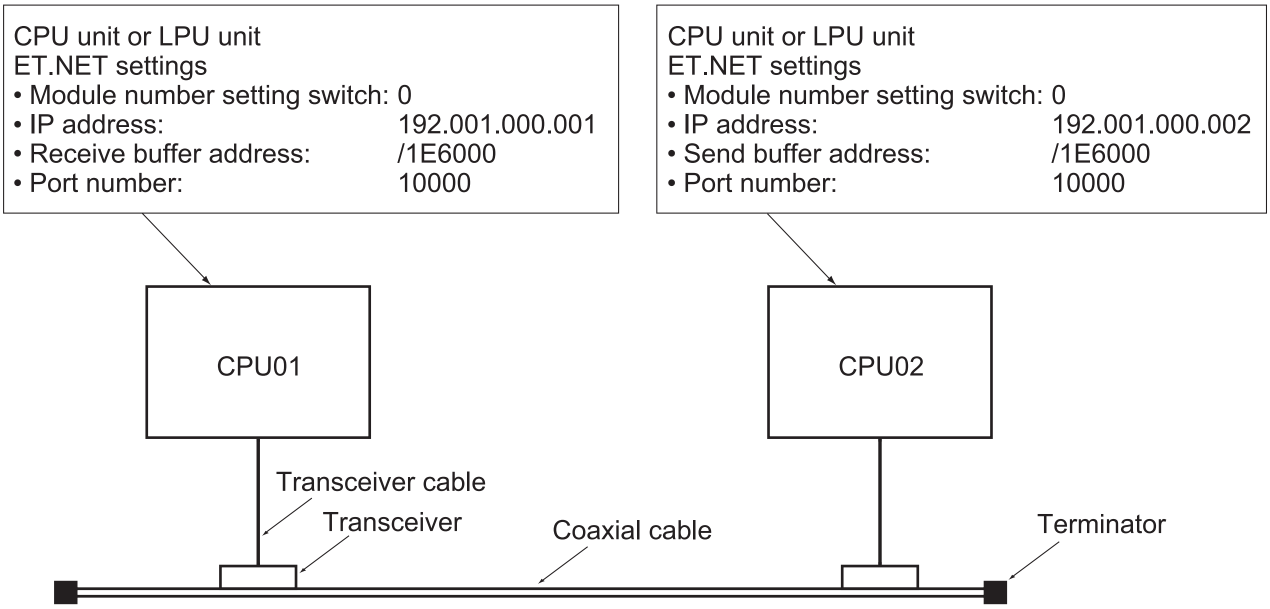
  When the LQE520 issues a socket library function, an ACK packet may cause a timeout due to a communication error or a failure in the remote device. It takes time to detect a timeout as indicated in the table below. Therefore, at least the time in the table is required after a timeout of the socket handler 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 (LQE520) 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	

5.6 Inter-CPU Communication Sample Program

5.6.1 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.



CPU unit or LPU unit
ET.NET settings
- Module number setting switch: 0
- IP address: 192.001.000.001
- Receive buffer address: /1E6000
- Port number: 10000

CPU unit or LPU unit
ET.NET settings
- Module number setting switch: 0
- IP address: 192.001.000.002
- Send buffer address: /1E6000
- Port number: 10000

Figure 5-5 System Architecture Example

5.6.2 CPU01 program flowchart and sample program

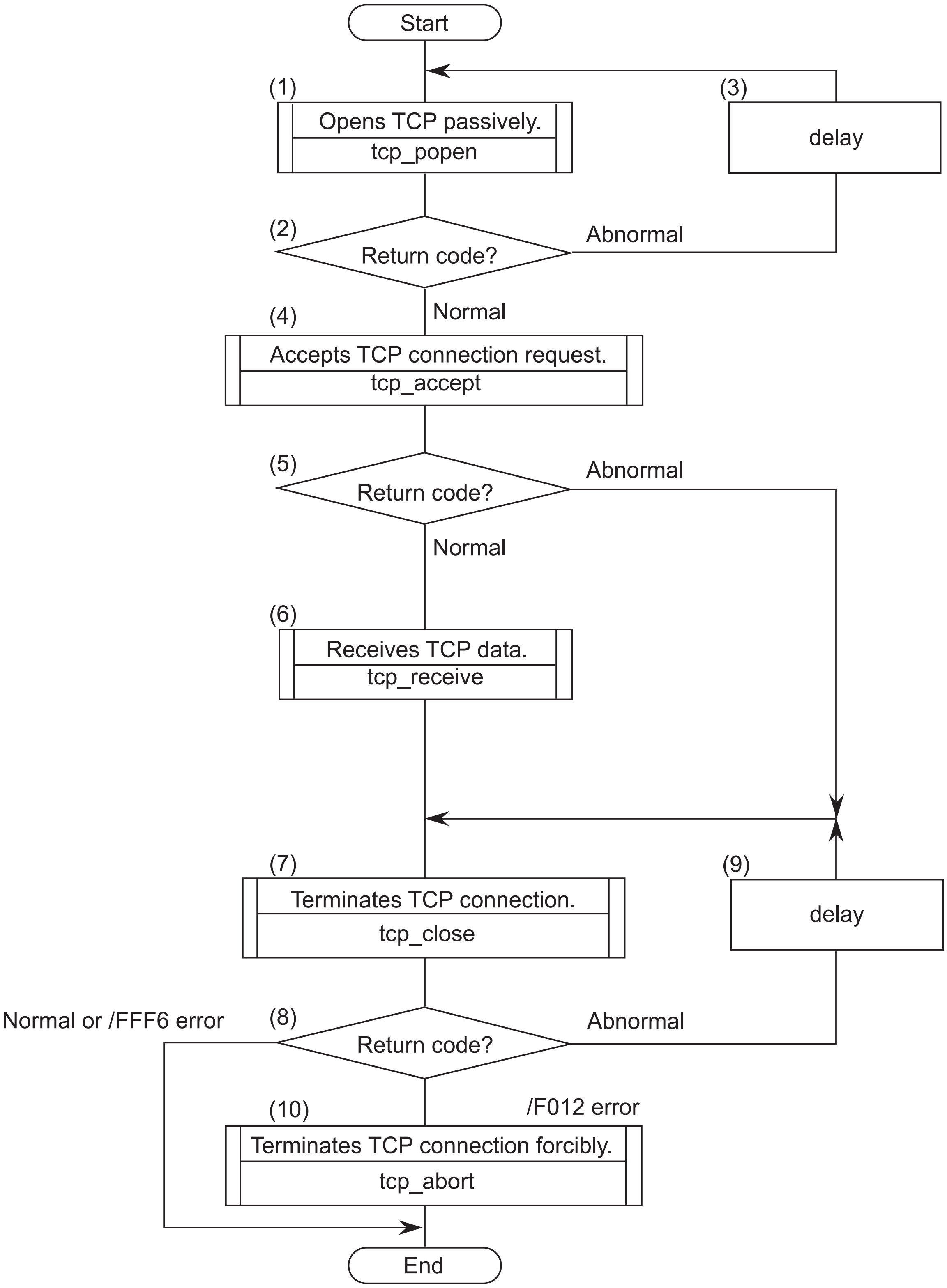


Figure 5-6 CPU01 Program Flowchart

■ 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           */
#define IPADDR      0xC0010002L /* IP address of remote station              */
#define RBUFADDR    0x1E6000L   /* Starting address of receive buffer        */
#define PARADDR     0x1E5000L   /* Starting address of parameter strage area */

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;
    char    *rbuf;
    struct  popen_p     *popen;
    struct  accept_p    *accpt;
    struct  receive_p   *recv;
    struct  close_p     *close;
    struct  abort_p     *abort;

    popen = (struct popen_p   *)PARADDR;       /* Starting address of input parameter storage area */
    accpt = (struct accept_p  *)(popen + 1);
    recv  = (struct receive_p *)(accpt + 1);
    close = (struct close_p   *)(recv  + 1);
    abort = (struct abort_p   *)(close + 1);
    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);       /* Opens TCP passively                */
        if( rtn > 0 ){                         /* Return code normal?                */
            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;
}
```

5.6.3 CPU02 program flowchart and sample program

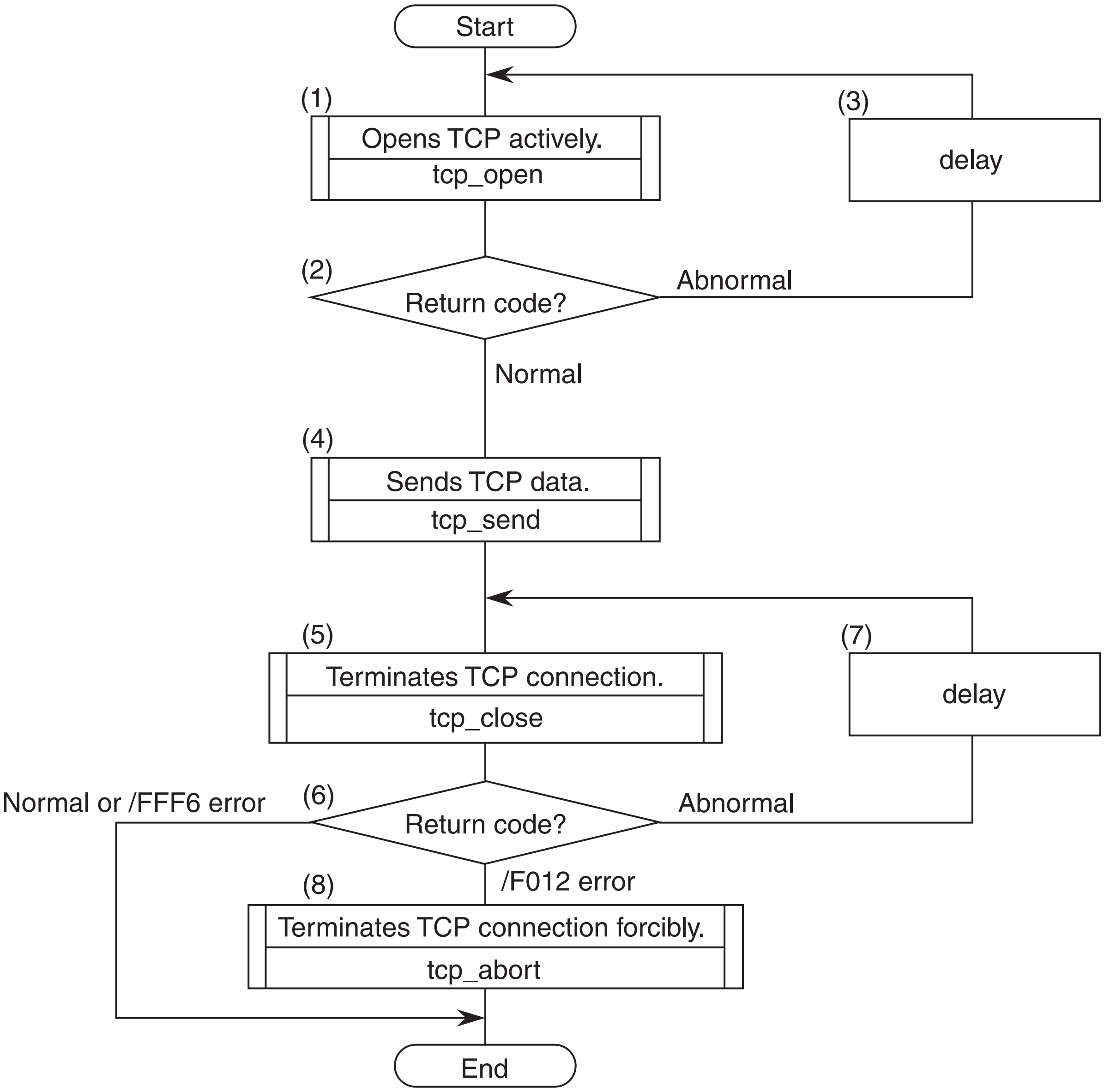


Figure 5-7 CPU02 Program Flowchart

■ 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          */
#define IPADDR      0xC0010001L /* IP address of remote station             */
#define SBUFADDR    0x1E6000L   /* Starting address of send buffer          */
#define PARADDR     0x1E5000L   /* Starting address of parameter strage area */

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;
     open  = (struct open_p    *)PARADDR;    /* Starting address of input parameter storage area */
     send  = (struct send_p    *)(open  + 1);
     close = (struct close_p   *)(send  + 1);
     abort = (struct abort_p   *)(close + 1);

     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;
```

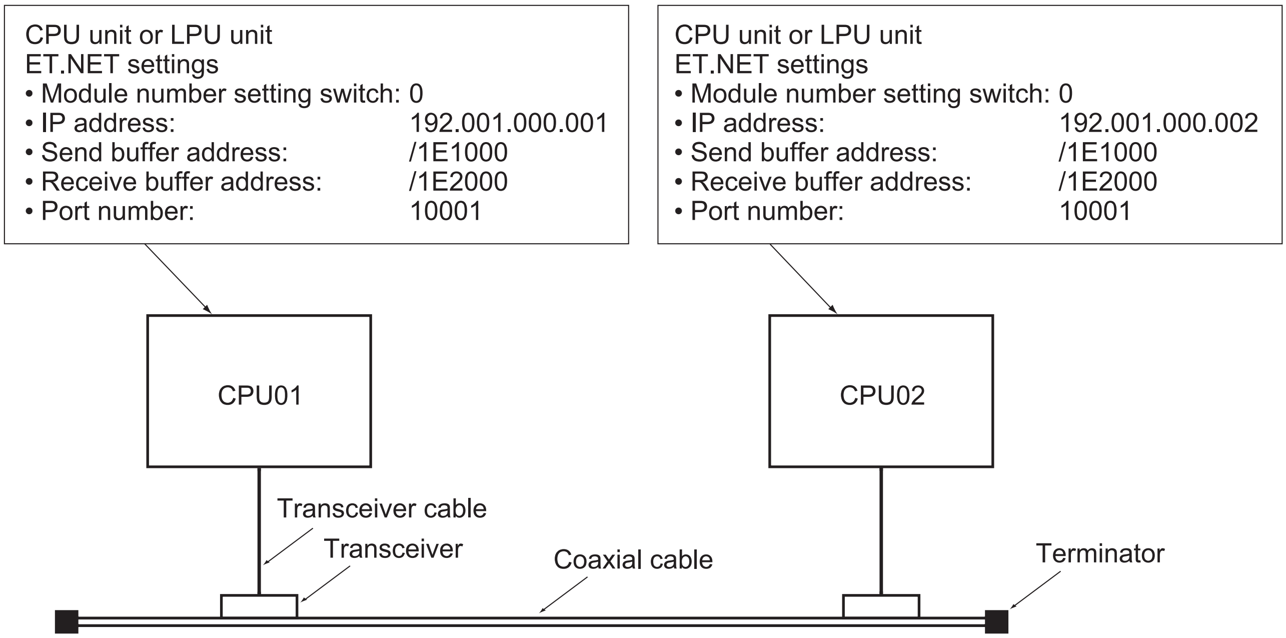
```
        }
        time = 100;                         /* Issue of 100-ms Delay macro       */
        delay( &time);

    }
    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;
}
```

5.7 Inter-CPU Continuous Communication Sample Program

5.7.1 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.



CPU unit or LPU unit
ET.NET settings
- Module number setting switch: 0
- IP address: 192.001.000.001
- Send buffer address: /1E1000
- Receive buffer address: /1E2000
- Port number: 10001

CPU unit or LPU unit
ET.NET settings
- Module number setting switch: 0
- IP address: 192.001.000.002
- Send buffer address: /1E1000
- Receive buffer address: /1E2000
- Port number: 10001

Figure 5-8 System Architecture Example

5.7.2 CPU01 program flowchart and sample program

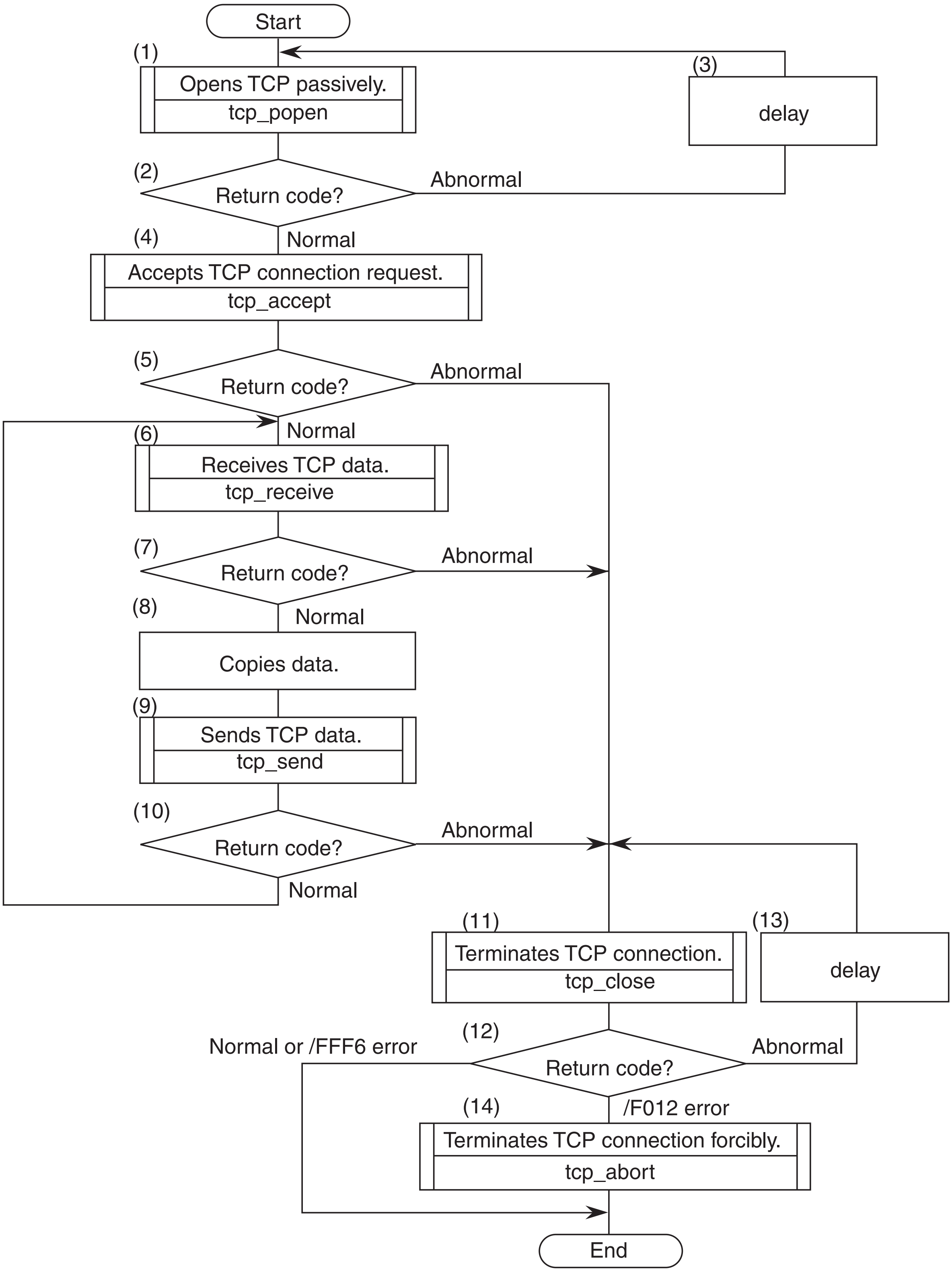


Figure 5-9 CPU01 Program Flowchart

■ 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            */
#define IPADDR      0xC0010002L /* IP address of remote station               */
#define SBUFADDR    0x1E1000L   /* Starting address of send buffer            */
#define RBUFADDR    0x1E2000L   /* Starting address of receive buffer         */
#define PARADDR     0x1E5000L   /* Starting address of parameter storage area */

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;
     popen = (struct popen_p   *)PARADDR;         /* Starting address of input parameter storage area */
     accpt = (struct accept_p  *)(popen + 1);
     recv  = (struct receive_p *)(accpt + 1);
     send  = (struct send_p    *)(recv  + 1);
     close = (struct close_p   *)(send  + 1);
     abort = (struct abort_p   *)(close + 1);

    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);  /* Opens TCP passively.               */
        if( rtn > 0 ){                         /* Return code normal？               */
            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;                   /* SocketID                           */
        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;          /* Starting address of send buffer    */
            rbuf = ( char *)RBUFADDR;          /* Starting address of receive buffer */
            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;
}
```

5.7.3 CPU02 program flowchart and sample program

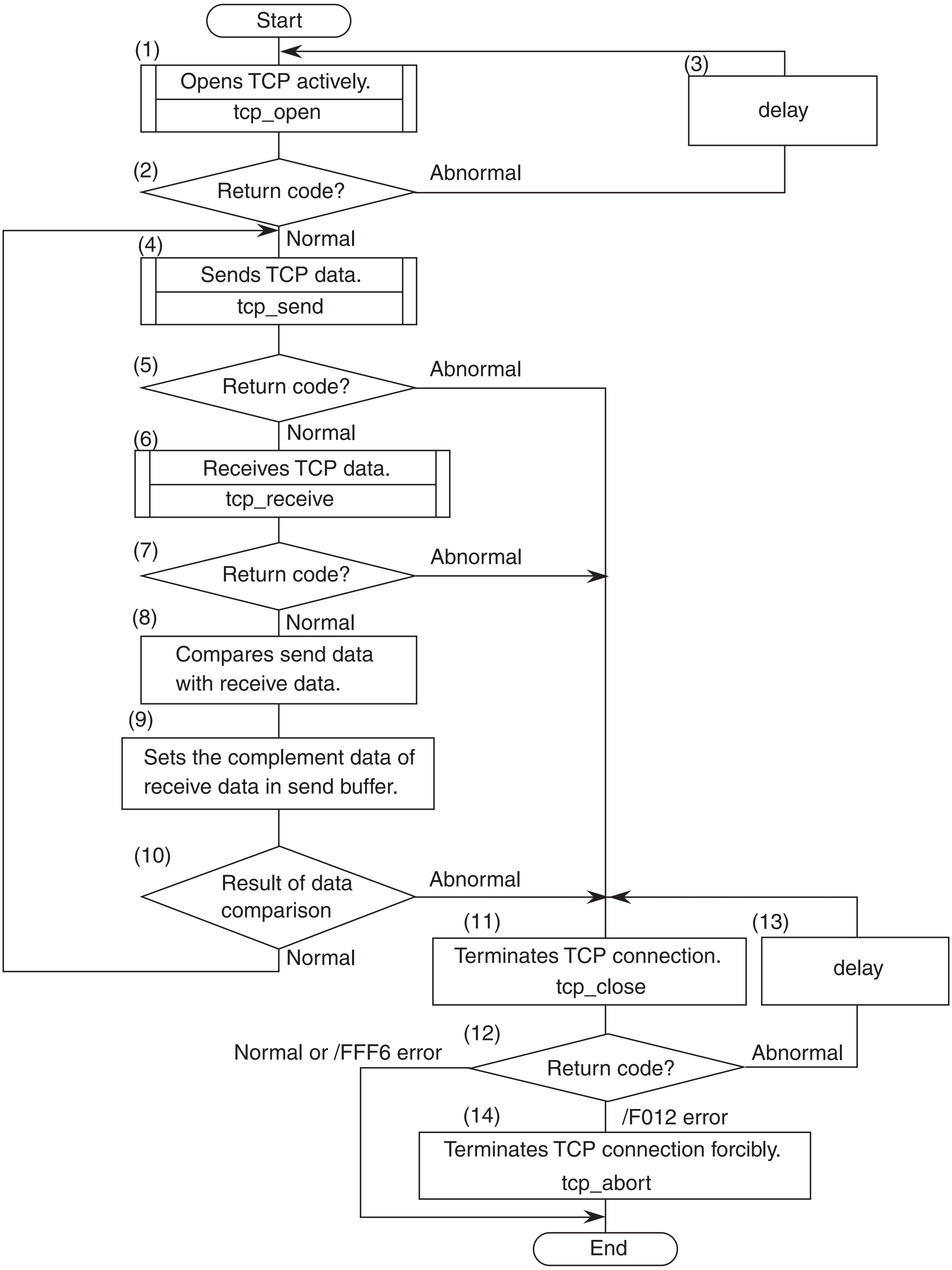


Figure 5-10 CPU02 Program Flowchart

■ 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            */
#define IPADDR      0xC0010001L /* IP address of remote station               */
#define SBUFADDR    0x1E1000L   /* Starting address of send buffer            */
#define RBUFADDR    0x1E2000L   /* Starting address of receive buffer         */
#define PARADDR     0x1E5000L   /* Starting address of parameter storage area */

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;
    struct  open_p      *open;
    struct  send_p      *send;
    struct  receive_p   *recv;
    struct  close_p     *close;
    struct  abort_p     *abort;

    open  = (struct open_p    *)PARADDR;    /* Starting address of input parameter storage area */
    send  = (struct send_p    *)(open  + 1);
    recv  = (struct receive_p *)(send  + 1);
    close = (struct close_p   *)(recv  + 1);
    abort = (struct abort_p   *)(close + 1);
    sbuf = ( char *)SBUFADDR;              /* Starting address of send buffer    */
    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 LQE520 is a standard product conformed to the global standard of IEEE802.3. It may happen, however, that the LQE520 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 LQE520.
Table 6-1 and Figure 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 LQE520.

Table 6-1 Network Component List

No.	Product name	Manufacturer	Model	Remarks
①	ET.NET	Hitachi, Ltd.	LQE520	
②	Transceiver	Hitachi Cable, Ltd.	HLT-200TB HBN200TZ HBN200TD	Tap type
③	Transceiver	Hitachi Cable, Ltd.	HLT-200	Connector type
④	Repeater	Hitachi Cable, Ltd.	HLR-200H	Repeater for extending transmission distance of coaxial cable
⑤	Multi-port transceiver	Hitachi, Ltd.	H-7612-64 H-7612-68	4 port/8 port AC power supply built in
⑥	Coaxial cable	Hitachi Cable, Ltd.	HBN-CX-100	For indoor Up to 500 m
⑦	Coaxial connector	Hitachi Cable, Ltd.	HBN-N-PC	For coaxial cable
⑧	Relay connector	Hitachi Cable, Ltd.	HBN-N-AJJ	For coaxial cable
⑨	Terminator	Hitachi Cable, Ltd.	HBN-T-NJ	J type
⑩	Terminator	Hitachi Cable, Ltd.	HBN-T-NP	P type
⑪	Ground terminal	Hitachi Cable, Ltd.	HBN-G-TM	For coaxial cable
⑫	Transceiver cable	Hitachi Cable, Ltd.	HBN-TC-100	With male and female D-sub 15-pin connectors Up to 50 m
⑬	Twisted pair cable	Hitachi Cable, Ltd.	HUTP-CAT5-4P	
⑭	Multi-port transceiver	Hitachi Cable, Ltd.	HBM-400TZ	4 port

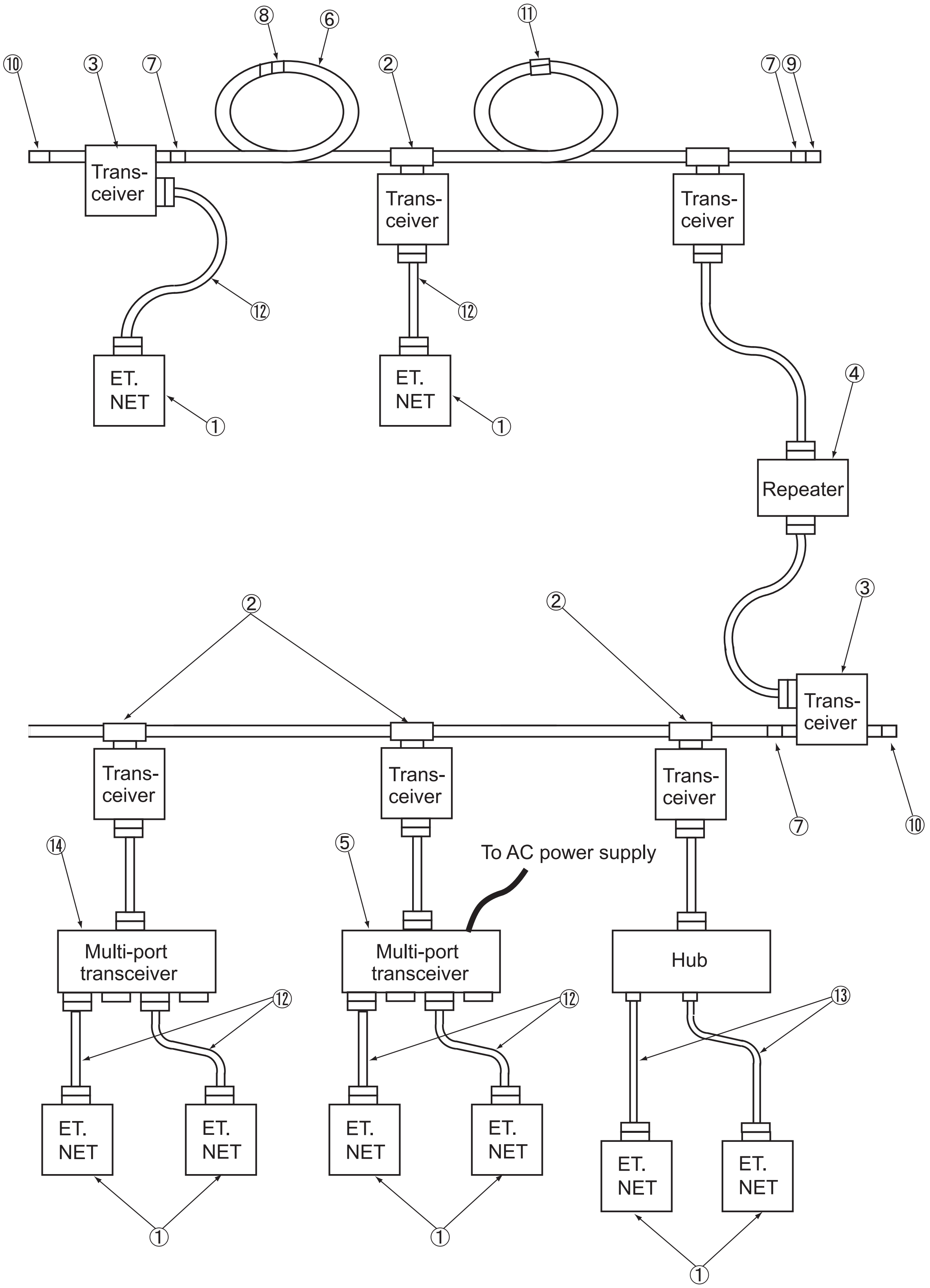


Figure 6-1 Network Components

6.2 System Configuration of 10BASE5

6.2.1 10BASE5 system configuration overview

As shown in Figure 6-2, a basic configuration consists of a single coaxial cable of up to 500 m and stations connected to this cable. Each station is connected to the coaxial cable via a transceiver cable and a transceiver. (The station means Ethernet equipment including LQE520.) This basic configuration is also called a segment; up to 100 stations can be connected in one segment.

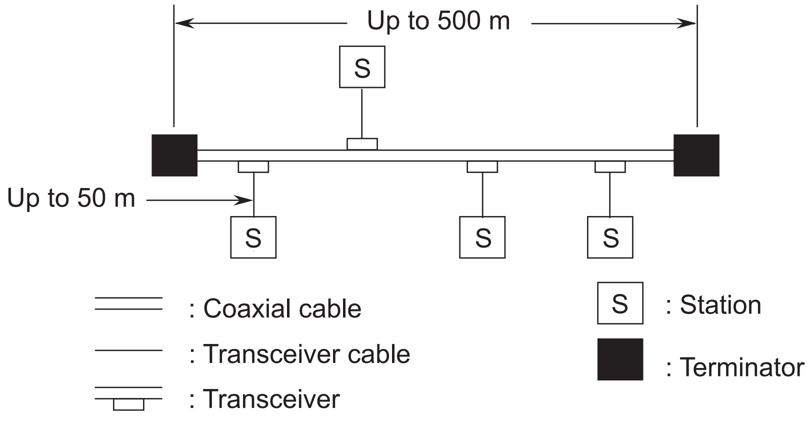


Figure 6-2 Minimum Configuration of 10BASE5

When the distance between stations exceeds 500 m, the number of segments can be increased by branching by using repeaters (see Figure 6-3.) This figure shows an example of a system in which the maximum distance between stations does not exceed 1,500 m. Construct the system so that the number of repeaters between any two stations is two or less.

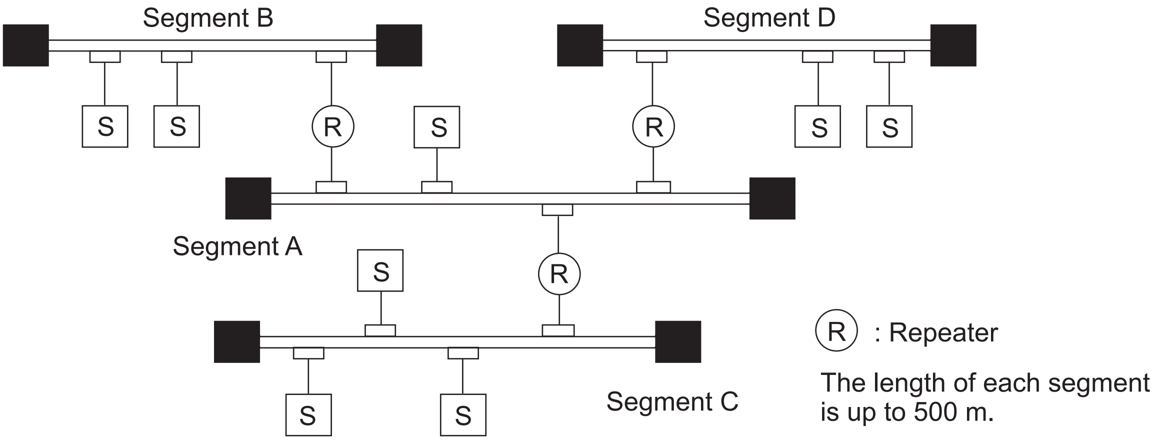


Figure 6-3 Medium-scale Configuration of 10BASE5

Figure 6-4 assumes a maximum inter-station transmission distance of 2500 m. Link cables each terminated at both ends by a repeater (up to 500 m for a coaxial cable), called "link segments," are used to extend the transmission distance. Link segments are not attached to a station. Instead, each link segment with its repeaters at both ends inclusive, as enclosed with a dotted line below, is viewed as one repeater to save the total number of repeaters installed between two stations.

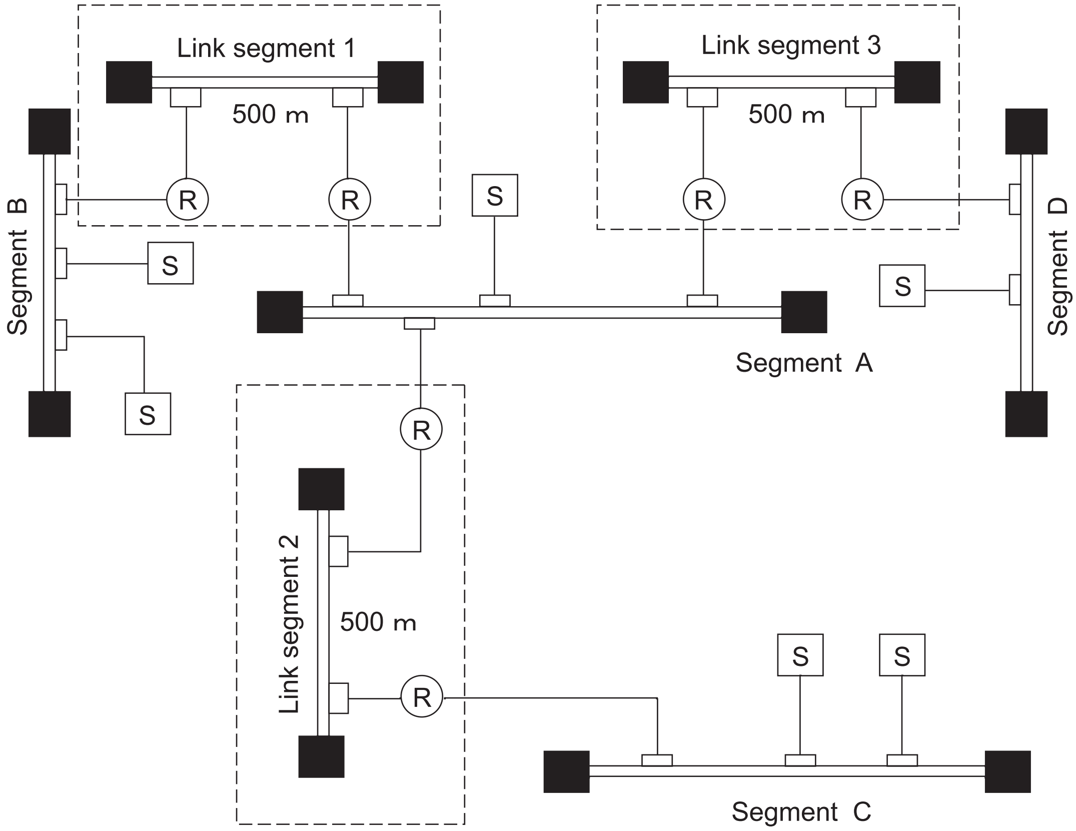


Figure 6-4 Large-scale Configuration of 10BASE5

System configuration parameters are listed below.

Table 6-2 10BASE5 System Configuration Parameters

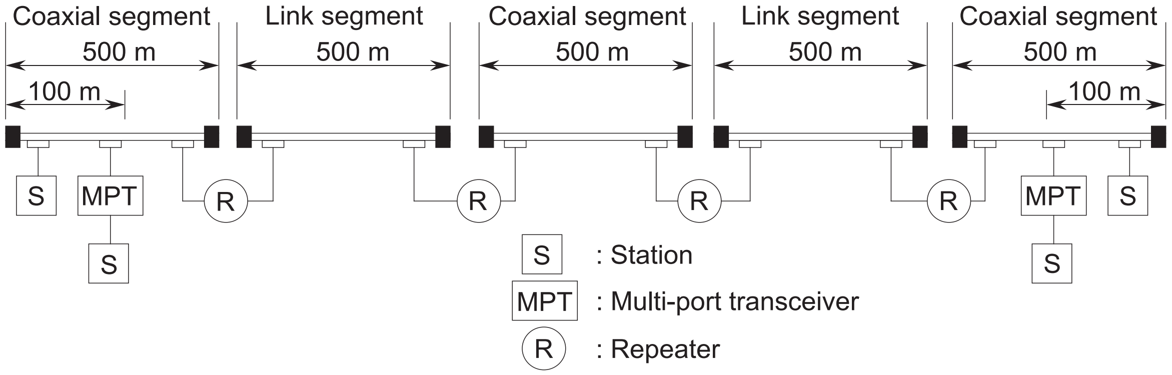
Item	Specifications
Maximum segment length	500 m
Maximum number of transceivers in segment	100 m
Maximum distance between stations	2,500 m or less (excluding transceiver cable)
Maximum number of stations in system	1,024
Maximum length of transceiver cable	50 m
Maximum number of repeaters in route between stations	2 (Each link segment with its repeaters at both ends inclusive is viewed as one repeater.)

6.2.2 Tips on configuring a 10BASE5 system

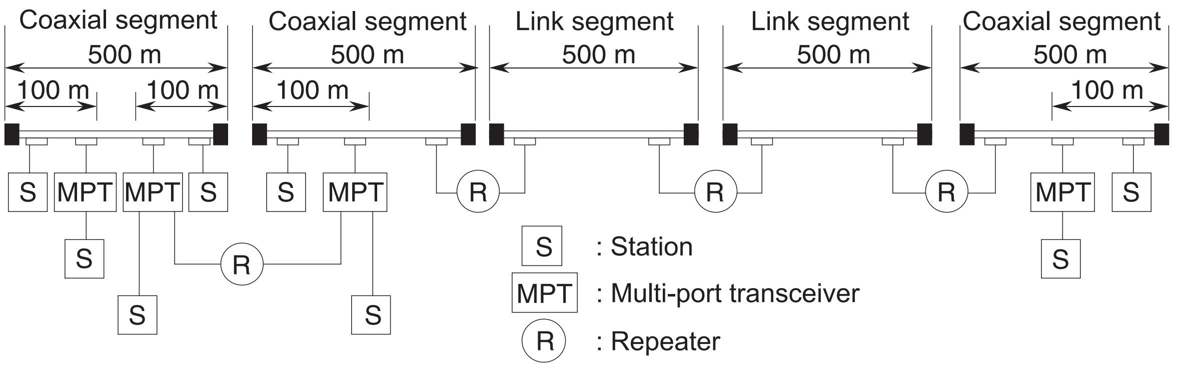
- Connect each repeater to a coaxial cable by way of a transceiver cable and a transceiver.
- Keep transceivers an integer multiple of 2.5 m apart from each other.
- Do not attach a station to a link cable.
- A repeater can be attached to the transceiver at any position.
- Do not allow more than two repeaters between two stations.
- Allow only one segment to have more than two segments connected to it.
- Up to four MCS and other screens can be opened where the ET.NET module is used in connection with a tool system.
- The maximum distance between the two stations that are connected by way of multi-port transceivers is reduced by a coaxial length equivalent of 100 m for each multi-port transceiver intervening between them. If the length of the coaxial cable laid between the two stations is L and the total number of multi-port transceivers intervening between them is N, then the following relation holds between L and N:

$L[m] \leq 2{,}500[m] - 100 \times N[m]$

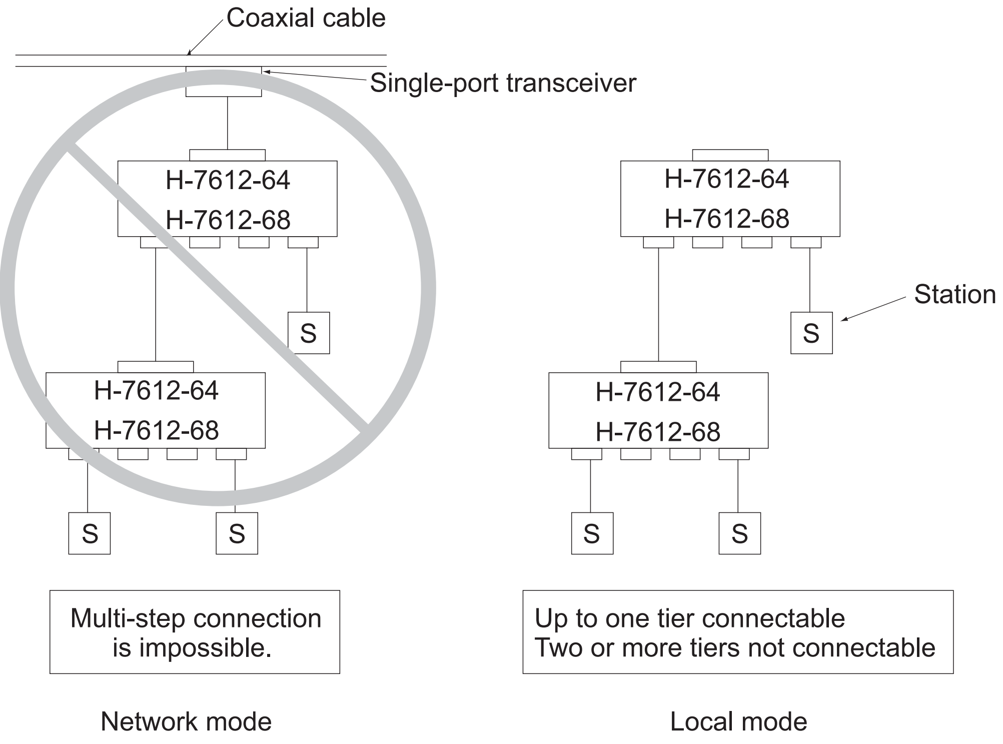
(Example 1) On a system that is built of a 2500-meter-long coaxial cable, install multi-port transceivers at least 100 inside the remotest terminators (where the station-to-station distance is reduced).



(Example 2) If repeaters are connected by way of multi-port transceivers, locate the transceivers to allow the remotest station-to-station distance to be reduced by 100 m for each transceiver intervening.



- When multi-port transceivers (H-7612-64/68) are used in network mode, multi-step connection is impossible due to the restrictions on transmission characteristics.



- Choose network components, such as coaxial cables, transceiver cables, and transceivers, from the recommendations given in Table 6-1 and Figure 6-1.

6.3 10BASE-T System Configuration

Connecting the hub (multi-port repeater) to a transceiver through a transceiver cable (AUI cable) enables connecting multiple stations to the hub, as shown in Figure 6-5. For connecting stations to the hub, use twisted-pair cables.

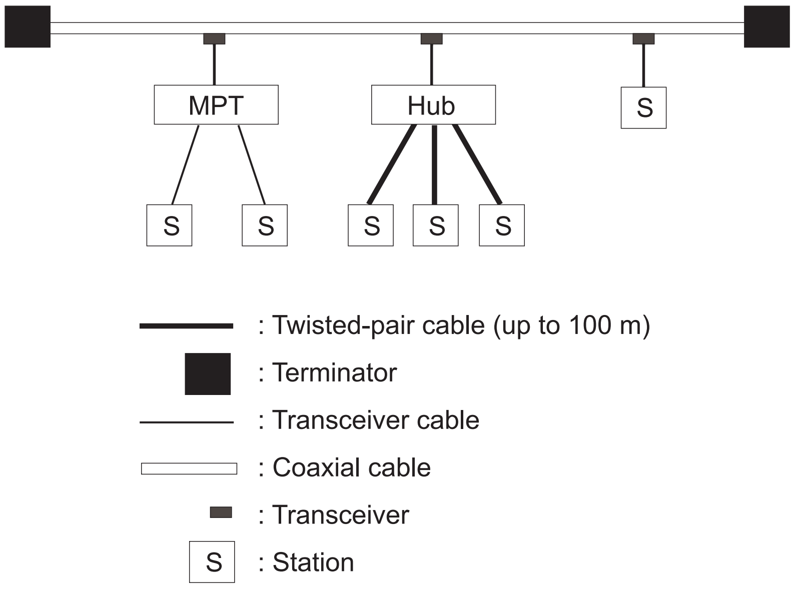


Figure 6-5 10BASE-T System Configuration

When the distance between stations is relatively short, each station can be connected directly to the hub through twisted-pair cables without using any coaxial cable or transceiver, as shown in Figure 6-6.

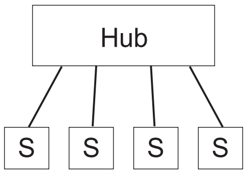
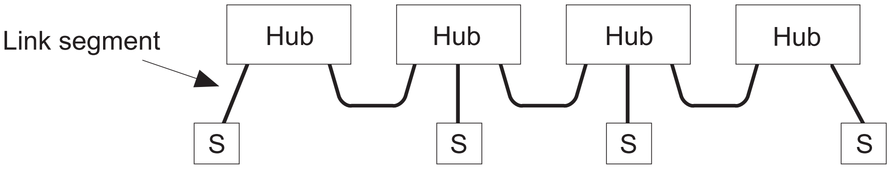


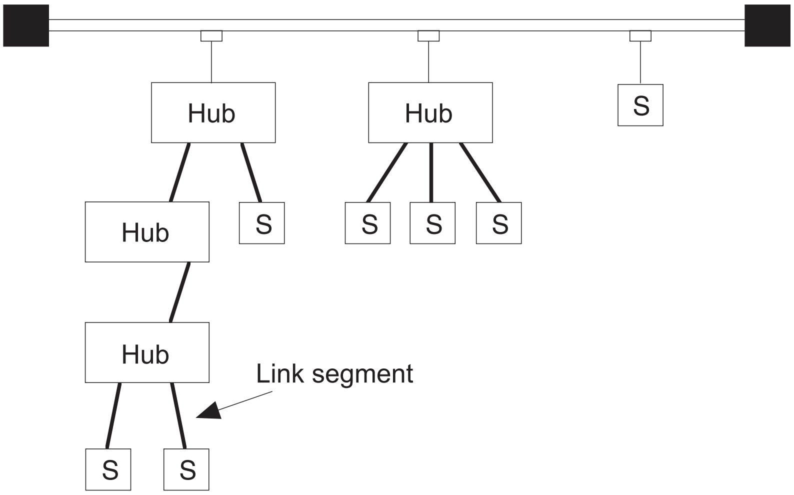
Figure 6-6 Typical Multi-Hub 10BASE-T System Configuration

■ Constraints on multi-hub connection

- When using multiple hubs, configure the system so that the number of hubs are up to four and the number of link segments up to five for any routing between stations.



- When connecting hubs with a coaxial cable, also configure the system so that the number of hubs are up to four and the number of link segments up to five (three for coaxial segments) for any routing between stations.



6.4 Installing, Wiring, and Setting Network Components

6.4.1 Wiring of coaxial cable

(1) The coaxial cable shall be laid in an indoor wiring duct and must be separated from 100 V or higher wiring. Before laying the cable, never fail to check that there is no short circuit nor break.

(2) The methods of laying the cable depending on the cabling location. The major methods are listed below.

- Rolling wiring in ceiling
- Wiring in cable rack
- Open wiring on wall surface
- Free-access wiring in floor pit
- Wiring in conduit

(3) The notes on wiring work are described below.

- In principle, lay this cable indoors.
- The mass of the cable is about 1.9 kg per 10 m.
- Do not add the tension of 245N or more to the cable body during cable laying.
- The bend radius of the cable should be 250 mm (150 mm when unavoidable) or more both when the cable is being laid and when it is finally fixed.
- Use a saddle when fixing the cable to a wall surface or ceiling. Except for special cases, the standard fixing interval is 1 m. When fixing the cable, take care not to deform the cable by tightening the saddle.
- When fixing the cable to a cable rack, the standard fixing interval is 2 m.
- For wiring in conduit, use a conduit whose inside diameter is 22 mm or more except for special cases (e.g., when it is used in the penetrated part of a fire wall).
- The bend radius of the conduit used shall be 300 mm or more.
- When the cable is laid on a floor or floor edge, it is apt to be deformed or damaged by walking or heavy objects. Protect the cable by tying or the like.
- For safety, ground the external conductor of the cable. Ground it at one point on a segment. Class D grounding or higher shall be applied. Insulate the connectors and terminators by covering them with the attached boots or by winding insulating tapes onto them, so that the exposed metallic parts of the cable except those at the grounding point do not touch the earth or other metallic parts.

6.4.2 Installation and wiring of transceiver

(1) For the transceiver, the installation location and wiring method differ depending on the conditions of the site. The major installation locations may be as follows:

- On a wall
- Beside a station

(2) Notes on transceiver installation are given below.

- Fix the transceiver by wood screws or the like via metal fittings.
- The installation interval of the transceiver shall be 2.5 mm or more.

(3) Fix the transceiver at the four tapped holes so that excessive force is not added to the cable.

(4) Use a coaxial cable connector to attach a coaxial cable connector to a transceiver. Because the external conductor of the coaxial cable connector is elevated from the ground potential, insulate the connector from other metals as, by fitting it with a rubber boot or covering it with PVC tape. The casing of the transceiver itself is equalized to the ground potential when a transceiver cable is connected to it. The transceiver casing should, therefore, require insulation from other metals as well.

(5) When selecting installation location, strictly observe the following rules:

- The looseness of the connectors and terminators can be checked.
- The looseness of the transceiver cable connectors can be checked.
- The attached LED can be checked.

Installation examples of transceivers and transceiver cables

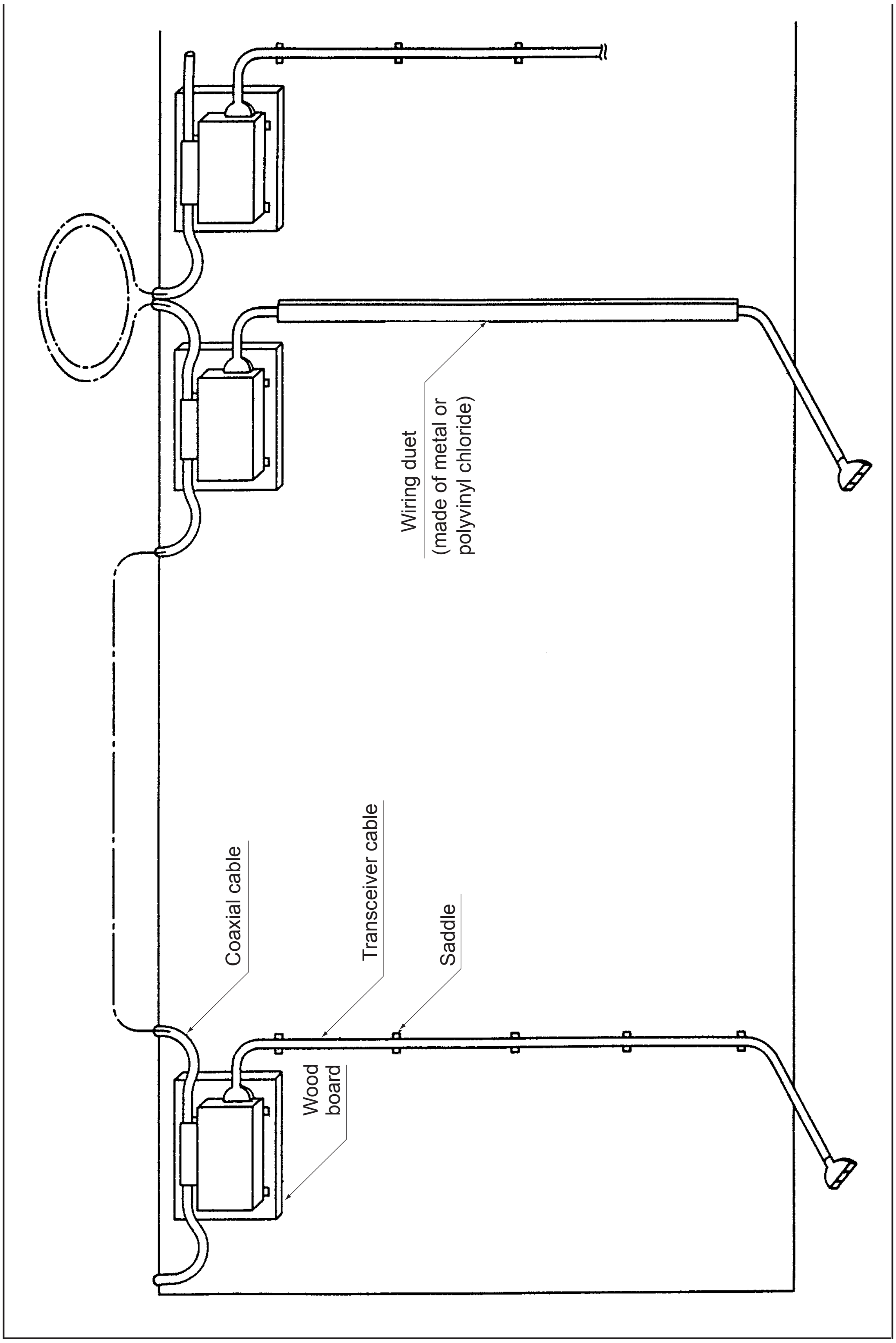


Figure 6-7 Wall-mounting Example (1)

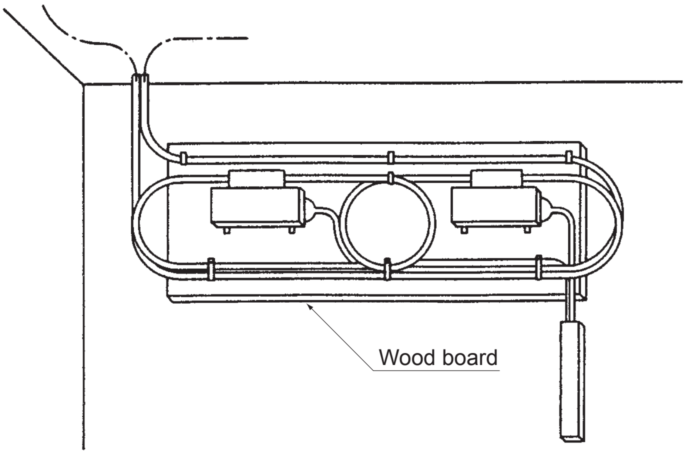


Figure 6-8 Wall-mounting Example (2)

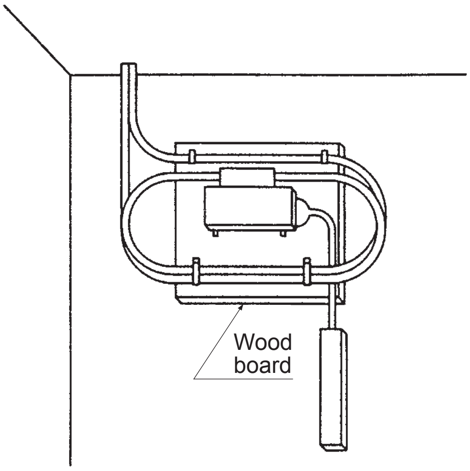


Figure 6-9 Wall-mounting Example (3)

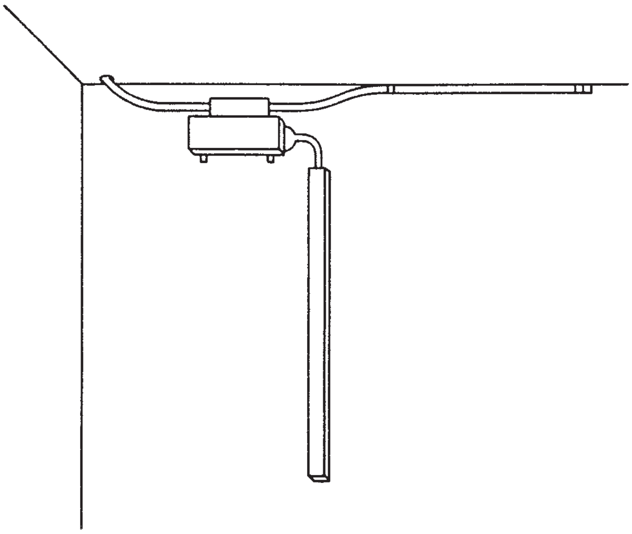


Figure 6-10 Wall-mounting Example (4)

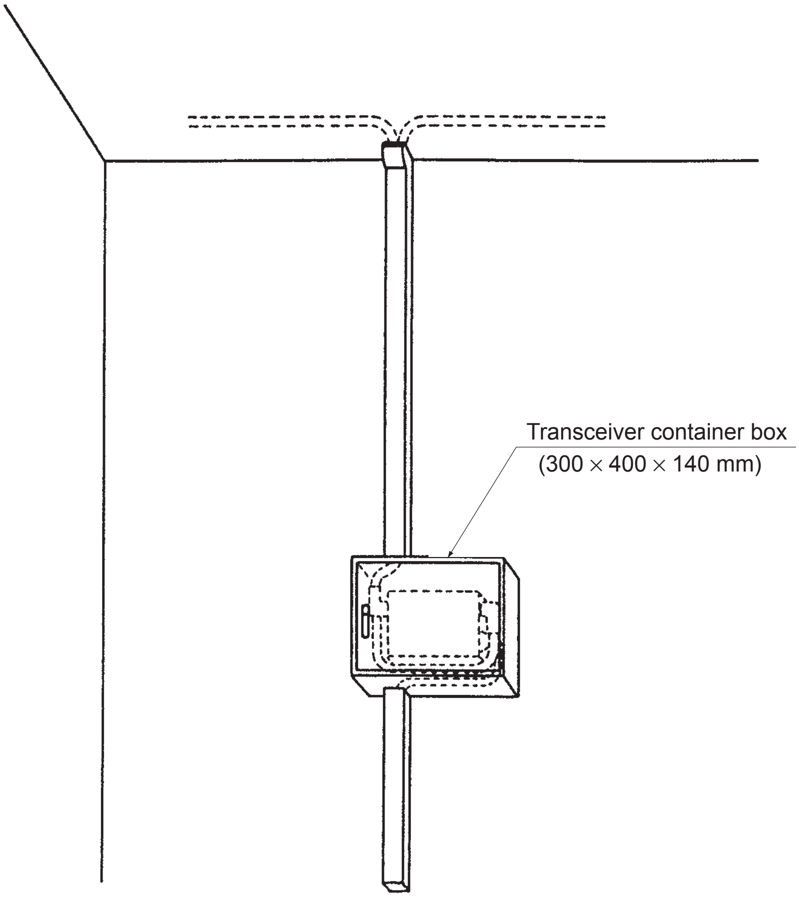


Figure 6-11 Mounting in Box (1)

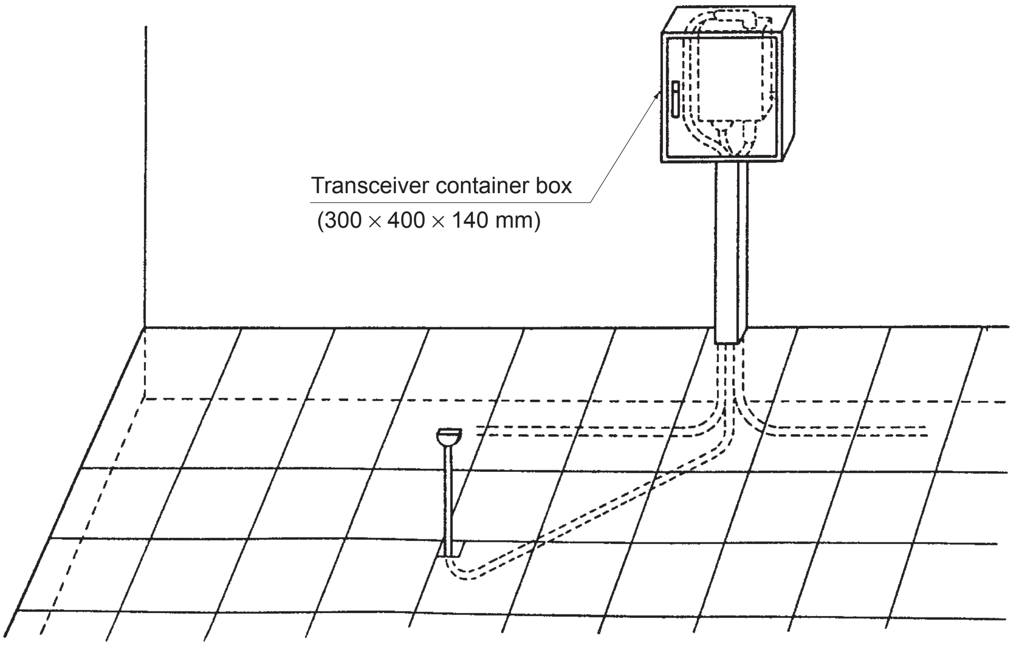


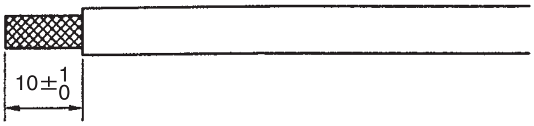
Figure 6-12 Mounting in Box (2)

6.4.3 Attaching coaxial connector

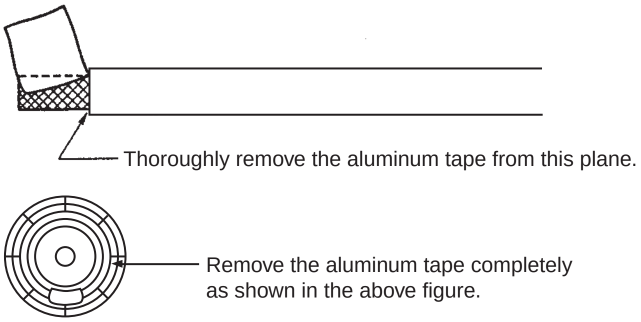
(1) Coaxial connector attachment procedure

The procedure for attaching the coaxial connector is shown below.

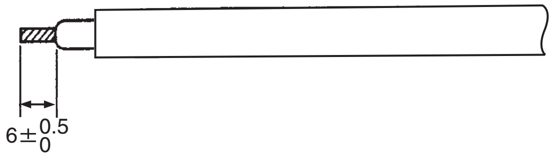
① Peeling off the PVC sheath



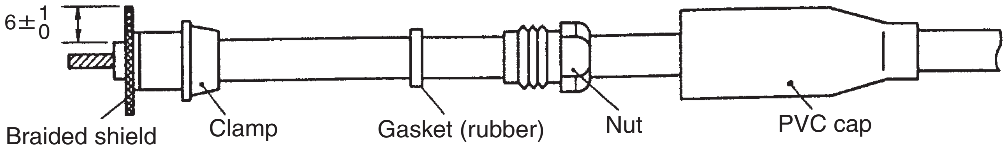
② Removing the aluminum tape


Thoroughly remove the aluminum tape from this plane.
Remove the aluminum tape completely
as shown in the above figure.

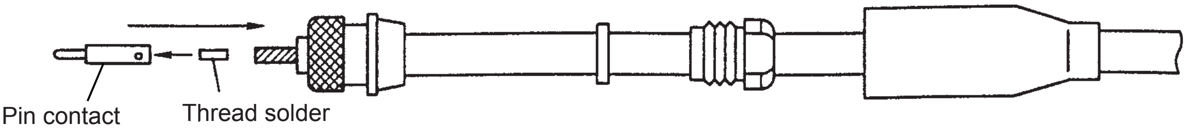
③ Peeling off the insulator



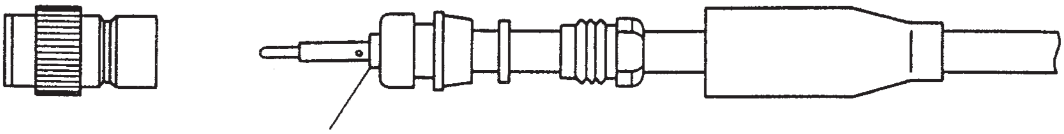
④ Parts setting and shield treatment



⑤ Soldering of pin contact



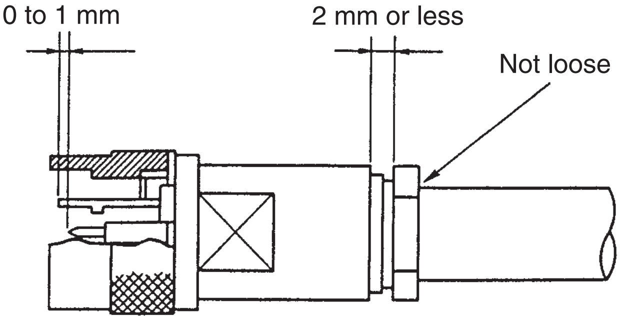
⑥ Connector attachment



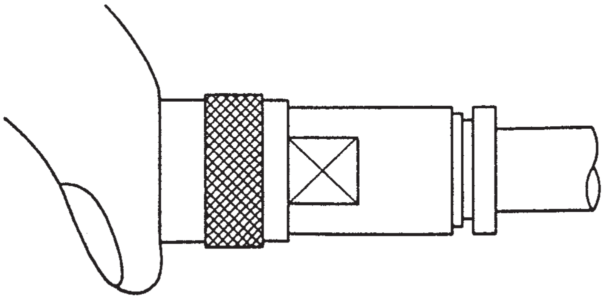
Allow a clearance of 1 mm or more between the pin contact and the insulating material. Take care not to cause the pin contact to bite into the insulating material.

(2) Check after attaching the connector

- Checking the coaxial connector opening
  - The difference between the external conductor at the top end of the coaxial connector and the inside contact shall be 0 to 1 mm.



  - When putting a thumb to the coaxial connector opening, the top end of the inside contact slightly touches the surface of the digital pulp.



  - There shall be no abnormal eccentricity of the central conductor found by visual inspection.

- Checking looseness
  - After attaching the coaxial connector, grasp and twist the coaxial connector body and coaxial cable to confirm that there is no looseness.
  - After tightening, the gap between the tightening nut and the body shall be about 2 mm or less.
- Insulation resistance measurement

  Measure the insulation resistance with the terminator out of position.
  - If a transceiver is not attached

    At least 1000 MΩ/km (500 VDC) must be present between the pin contact and the external conductor.
  - If a transceiver is attached

    Using a multimeter, measure the resistance with its + probe and – probe attached to the external conductor and the pin contact, respectively. Check for an infinite measurement reading.

⚠ DANGER

Never fail to discharge electricity after the test, otherwise you will get an electric shock.

6.4.4 Attaching tap connector

To attach a tap transceiver tap connector to a coaxial cable, follow these steps:

(1) Insert coaxial cable ① into the slot in tap connector ③ and place tap cover ② on top of it to secure coaxial cable ①.

(2) Using a box screwdriver, tighten M6 bolt ⑥ to a specified torque of 3 to 4 [N·m] to connect it to the external conductor of coaxial cable ①.

(3) Using a box screwdriver, tighten backup probe ⑤ and signal probe ④ in this order to a specified torque of 2 to 3 [N·m] to connect them to the central conductor of coaxial cable ①. Handle signal probe ④ and backup probe ⑤ with maximum cared, because their tips and threads could be easily damaged. After installing signal probe ④ and backup probe ⑤, do not retighten M6 bolt ⑥ to prevent damaging the probes under external pressure.

(4) Place included cap ⑦ on top of backup probe ⑤.

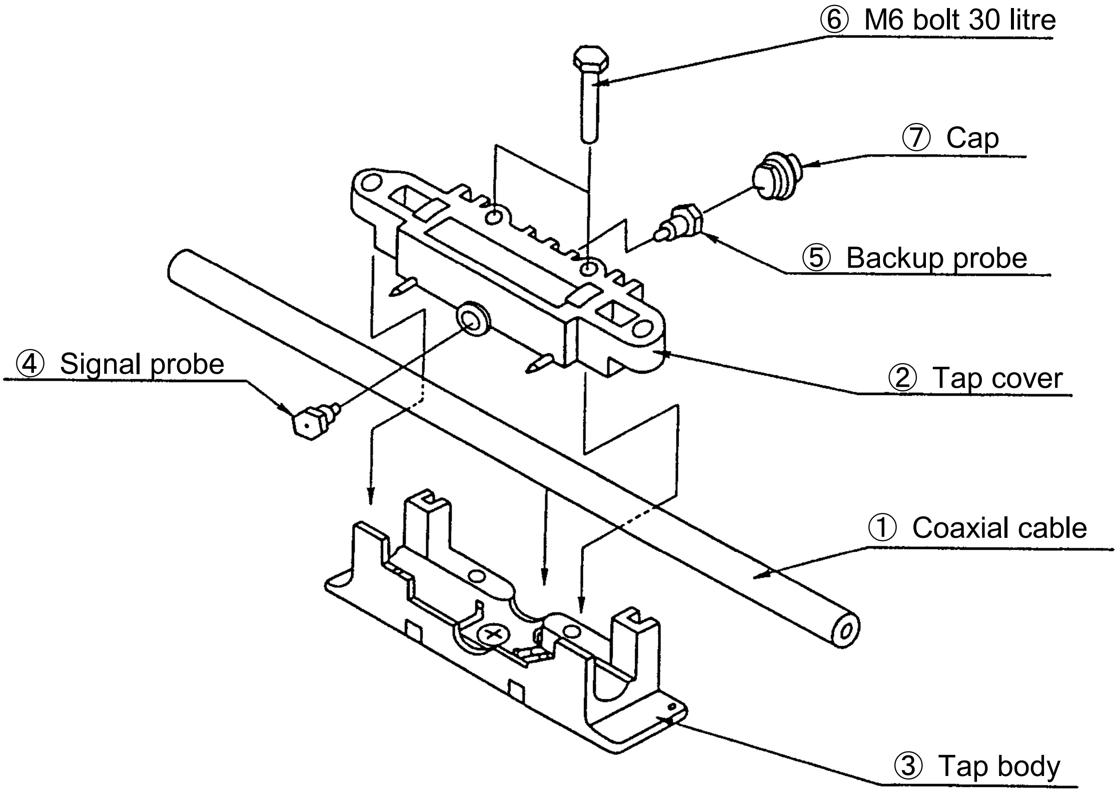


Figure 6-13 Tap Connector Assembly Drawing

To connect the tap connector and transceiver, follow these steps:

(1) Attach tap connector ① to the side of transceiver ② to let its probe and grounding screw into mounting holes in transceiver ②.

(2) Using a box screwdriver, tighten M6 bolt ③ to a specified torque of 3 to 4 [N·m] to secure transceiver ② and tap connector ③.

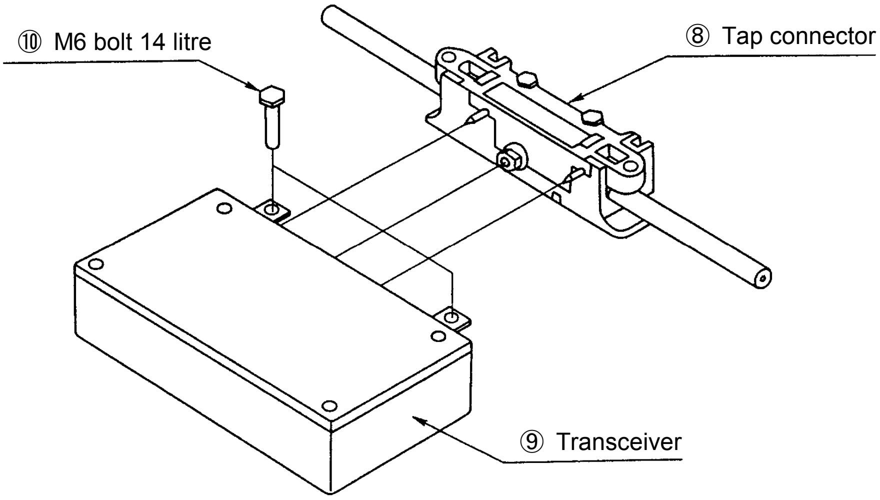


Figure 6-14 Connection of Connector and Transceiver

6.4.5 Attaching transceiver cable

The transceiver cable has a connector to attach it to a transceiver. The connector has a lock retainer. Attach the transceiver cable to allow the retainer to be fully locked in the lock post in the transceiver.

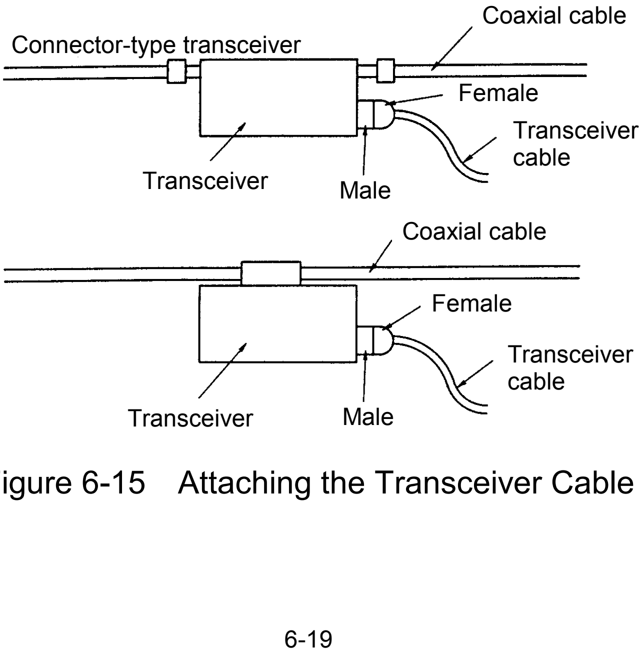


Figure 6-15 Attaching the Transceiver Cable

6.4.6 Attaching terminators

Connect the terminators to both ends of the coaxial segment without fail.

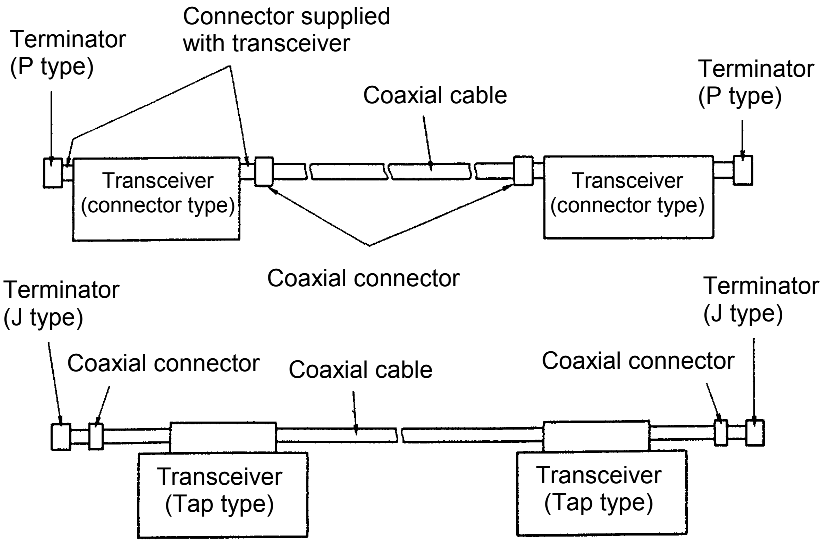
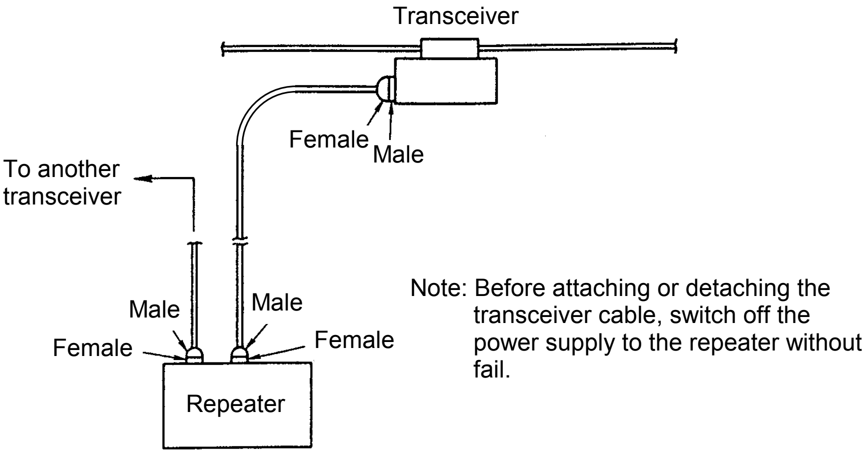


Figure 6-16 Attaching Terminator

6.4.7 Installation and attaching repeater

(1) Connection method

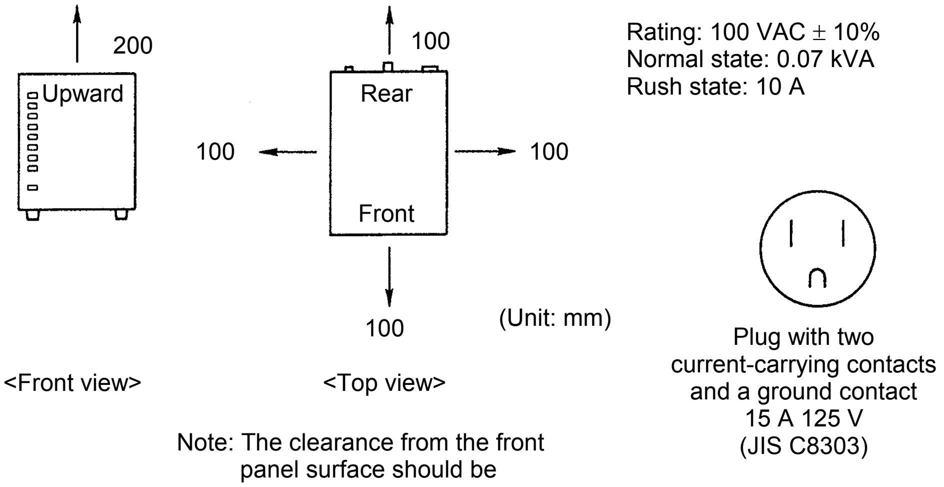


Note: Before attaching or detaching the transceiver cable, switch off the power supply to the repeater without fail.

Figure 6-17 Attaching Repeater

(2) Reserving installation location and space

- Locate the space to assure ready access for maintenance, as in a general office room. Allow clearances around the repeater as shown in Figure 6-18.
- Connect the repeater power cable to a grounded outlet.
- Do not use the repeater in a dusty environment.
- The repeater has an air inlet on the bottom and an air outlet on the top. Do not block them.
- Installation of a telephone near the repeater is recommended to facilitate its servicing.
- Ensure that the repeater is not inadvertently switched off. The transmission facility will shut down when the repeater is switched off.



Note: The clearance from the front panel surface should be great as possible.

Figure 6-18 Repeater Installation Space Requirements

6.4.8 Grounding the system

(1) Grounding the repeater

Ground the repeater by either using a grounded plug three-wire power cord or grounding terminal.

(2) Grounding stations

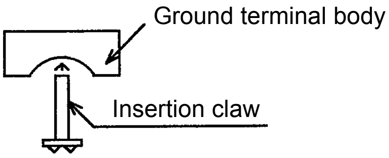
Ground all stations by Class D or better grounding.

Leaving any station ungrounded might incur electrical shock hazards and also result in data errors (CRC errors).

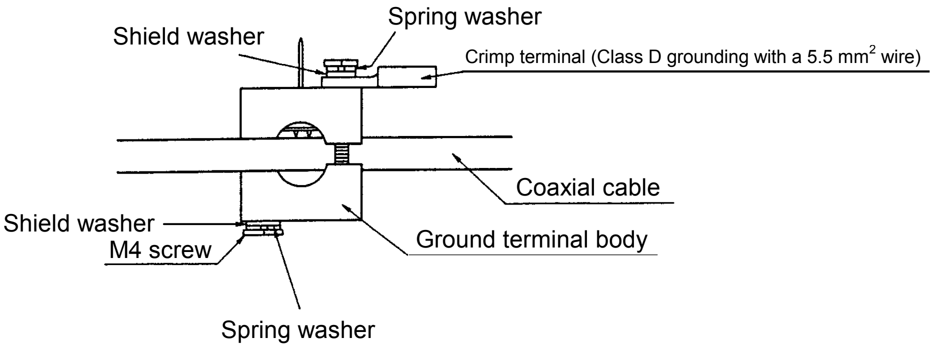
(3) Grounding the coaxial cable

Ground coaxial cables at one point per segment. Apply Class D or better grounding. Use a grounding screw for grounding. To install the grounding screw, follow these steps:

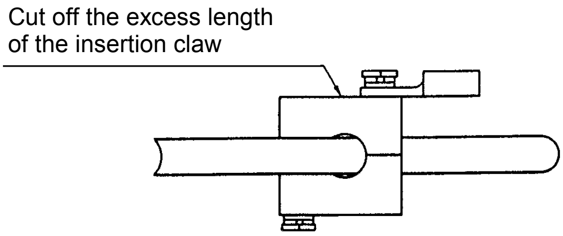
① Insert the insertion claw into the ground terminal unit.



② Attach ① to the coaxial cable, and tighten it and the M4 screw alternately. Attach a crimp terminal to either screw.



③ After tightening the M4 screw, cut off the excess length of the insertion claw.



⚠ DANGER

Ground all stations by Class D or better grounding.
Leaving any station ungrounded might incur electrical shock hazards.

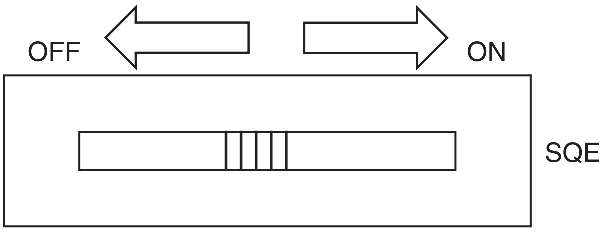
6.4.9 Setting signal-port transceiver

The single-port transceiver has an SQE switch. Set the SQE switch to suit the destination of the transceiver.

Table 6-3 SQE Switch Setting

Connected device	ET.NET controller	Multi-port transceiver	Repeater
SQE switch setting	ON	OFF	OFF

For the single transceivers HLT-200 and HLT-200TB, the SQE switch is contained in the case. When changing the setting, open the case to do the work. The switch is set to ON by turning it to the “SQE” side of silk printing on the board.



6.4.10 Setting and display of multi-port transceiver

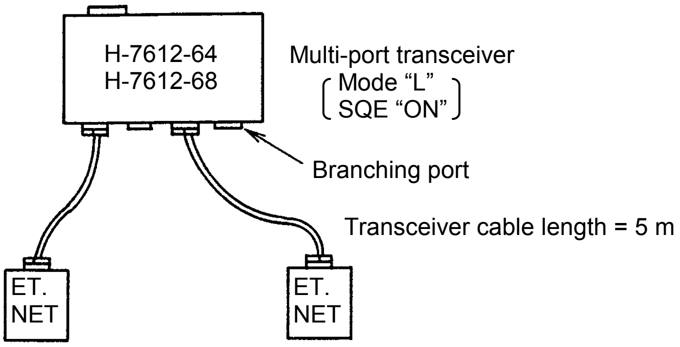
(1) Setting operation mode

The multi-port transceiver works in two modes of operation: local and network. Use the mode selector switch on the rear panel to choose between these two modes.

- Local mode

  The mode of operation in which the transceiver is used independently without using the coaxial cable. Do not attach the transceiver cable to the relay port.

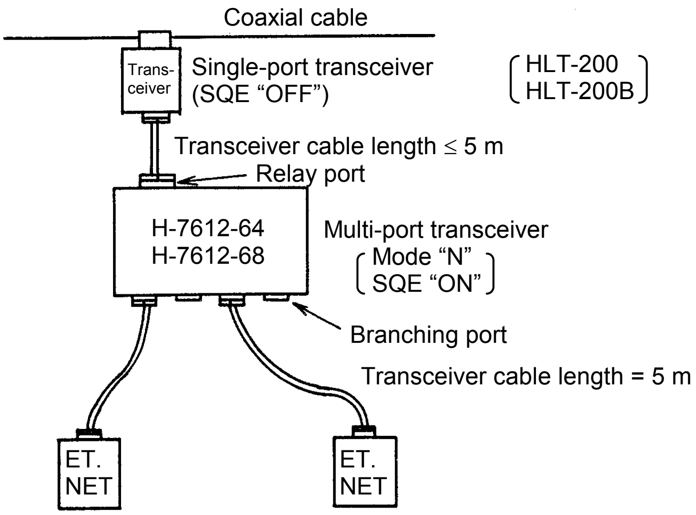
  In this mode of operation, set the mode selector switch to L (local mode) and the SQE switch to ON.



- Network mode

  The mode of operation in which the transceiver is used in connection with the coaxial cable.

  In this mode of operation, set the mode selector switch to N (network) and the SQE switch to OFF.



(2) Switch setting

The multi-port transceiver is equipped with two switches; the functions of each switch are mentioned in Table 8-1.

Table 6-4 Switch Setting

Switch type	Switch position	Function	Setting at shipment time
SQE switch	Rear panel	Setting SQE function to ON/OFF	“ON”
Operation mode switch	Rear panel	Switching operation mode	“N (network mode) ”

(3) Setting SQE switch on repeater connection

When connecting a repeater to a multi-port transceiver, set the SQE switch of the corresponding branching port of the multi-port transceiver to “OFF”.

(4) Power switch

Set the switch on the rear panel to “I”, and the power of the multi-port transceiver is turned “ON”.

(5) LED display

The “POWER” LED and the “LINK” LEDs for each branching port are placed on the front panel of the cabinet.

“POWER” LED: Lights when the power switch is “ON”.

“LINK” LED: Lights when the station is connected to the branching port of the multi-port transceiver.

6.5 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.

6.5.1 Physical address

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

6.5.2 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 bits)	Host number (24 bits)

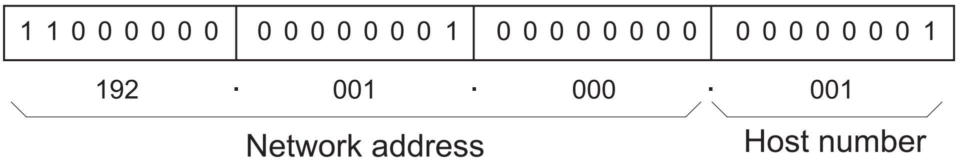
(ii) Class B (The high-order two bits of the network number are set to “10” in binary.)

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

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

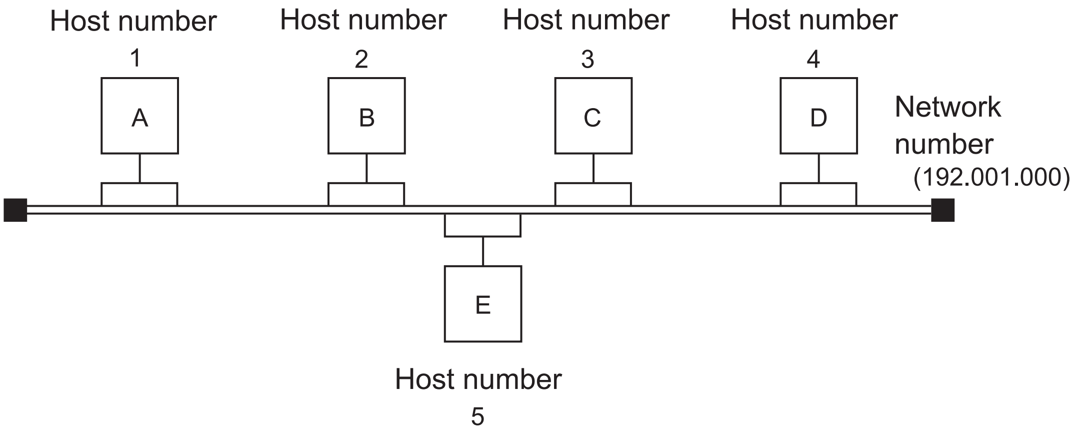
Network number (24 bits)	Host number (8 bits)

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.)

6.5.3 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

6.5.4 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.

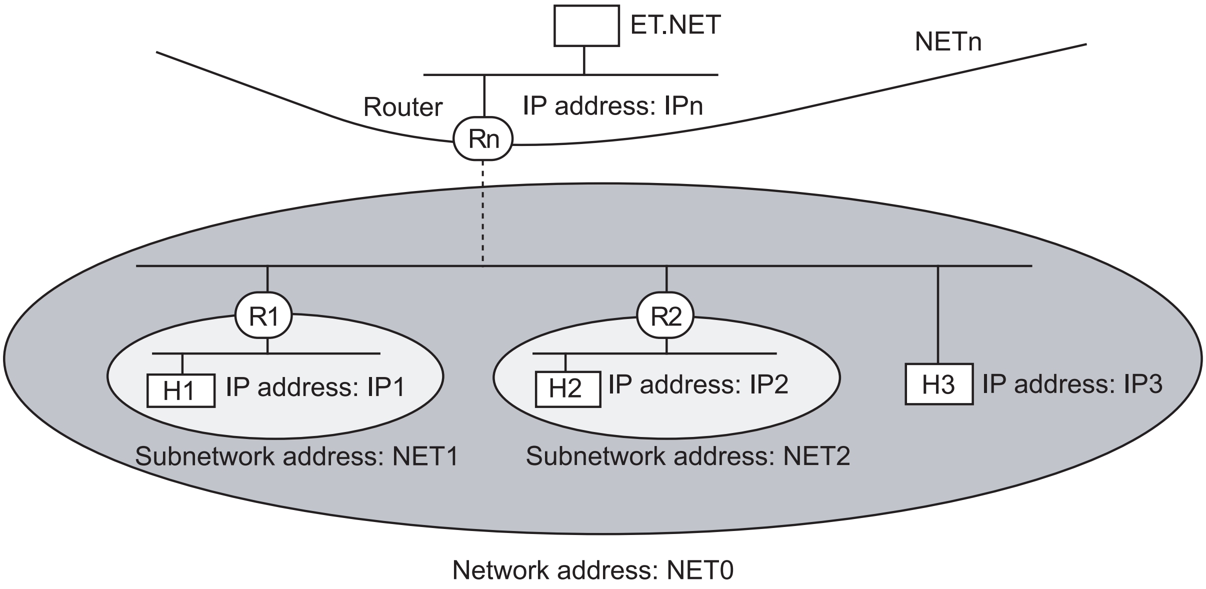
(2) IP address of router
The IP address of the router in the same network as the ET.NET module is registered. When multiple routers is involved in the communication route to the destination, register only the router in the same network as the ET.NET module.

The following two methods are available for setting route information.

- Setting in the socket handler route_add( ) in a C program
  - See “5.4.1 Socket handler function list.”
- Setting by using the ET.NET or S10V ET.NET system.
  - See “4.3.3 Routing information setting.”

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

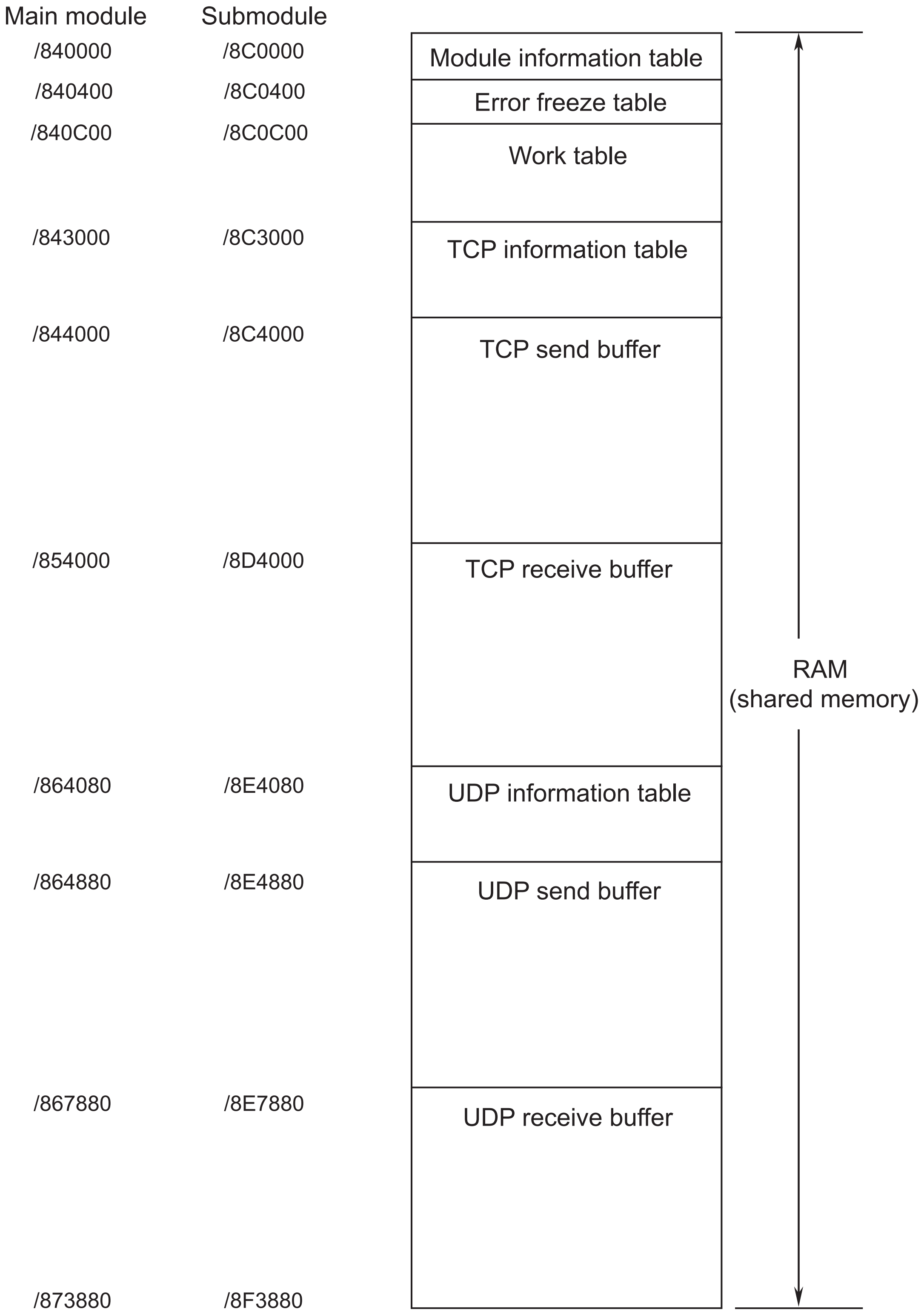


■ Restriction and limitation:

- 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.

6.6 Memory Map of ET.NET Module



Main module	Submodule	
/840000	/8C0000	Module information table
/840400	/8C0400	Error freeze table
/840C00	/8C0C00	Work table
/843000	/8C3000	TCP information table
/844000	/8C4000	TCP send buffer
/854000	/8D4000	TCP receive buffer
/864080	/8E4080	UDP information table
/864880	/8E4880	UDP send buffer
/867880	/8E7880	UDP receive buffer
/873880	/8F3880	

Figure 6-19 Memory Map

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

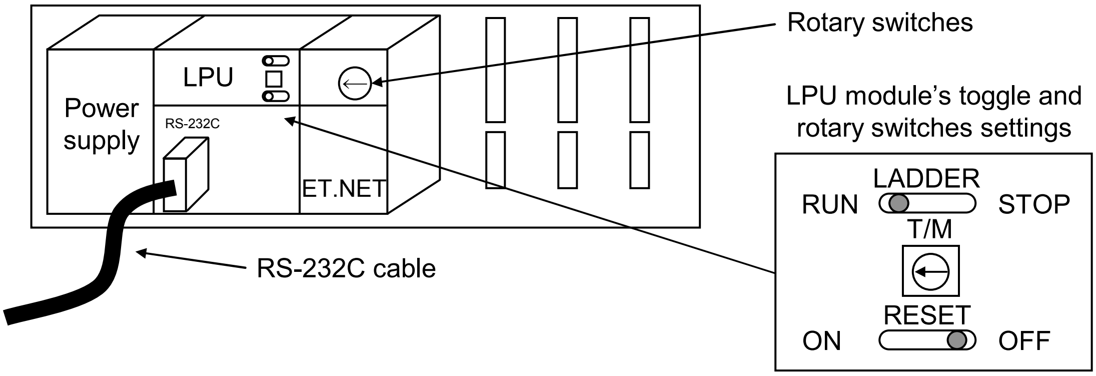
③ New or add-on ET.NET module (LQE520)

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 Reset the LPU module by setting the RESET switch in ON position and then in OFF position at its front.

⑪ Turn off the power supply of the controller unit.

⑫ Remove the RS-232C cable from both the personal computer and LPU module, which were connected together in Step ③.

⑬ Connect to the new ET.NET module the connecting cables that you removed in Step ⑥.

⑭ Set the LPU module's LADDER and T/M switches in the same way as you wrote down in Step ②.

⑮ 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 submodule 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

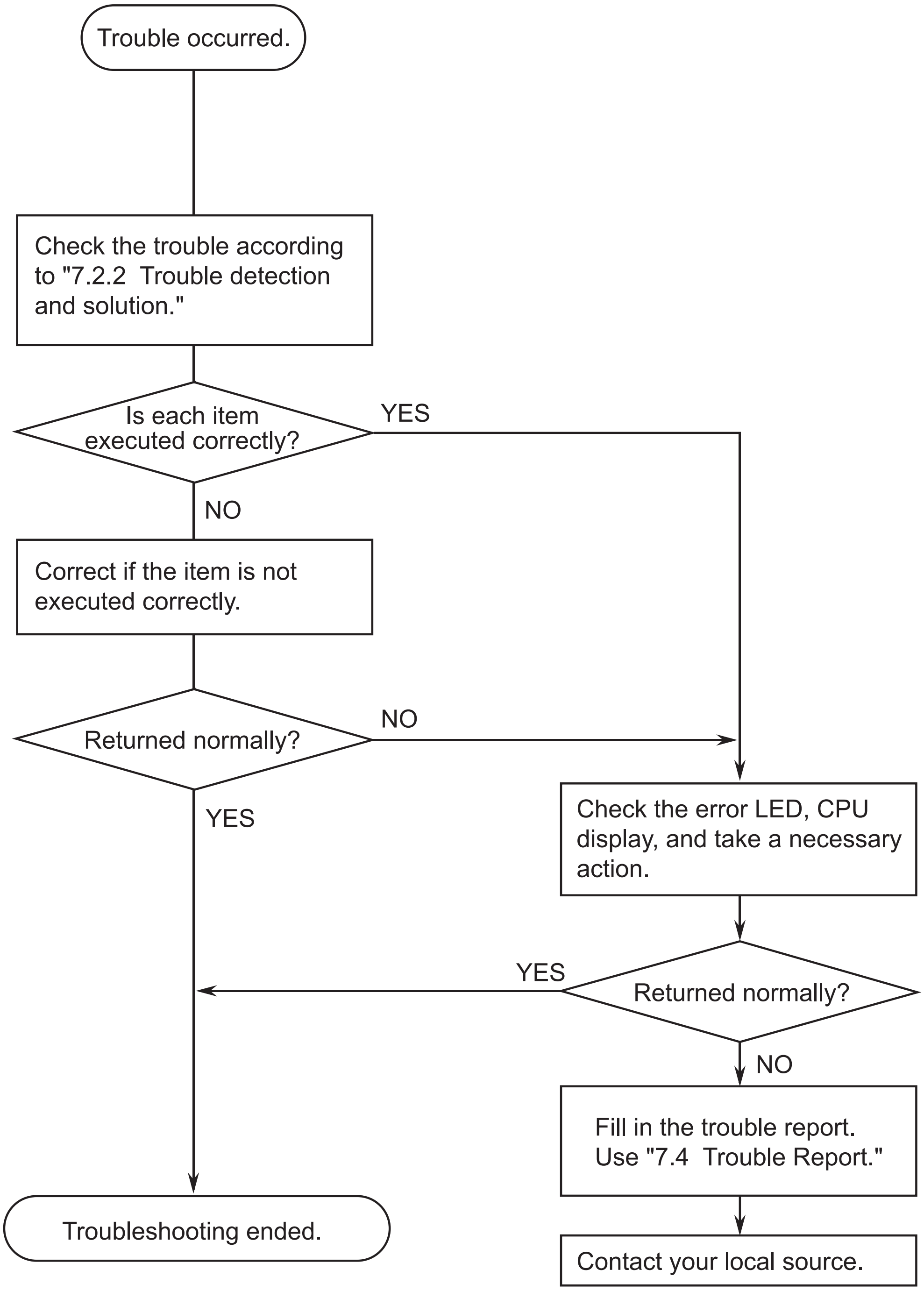
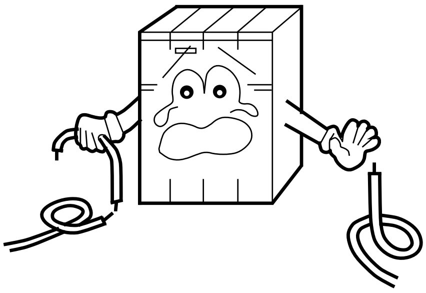

Trouble occurred.
Check the trouble according to "7.2.2 Trouble detection and solution."
Is each item executed correctly?
YES
NO
Correct if the item is not executed correctly.
Returned normally?
NO
YES
Check the error LED, CPU display, and take a necessary action.
YES
Returned normally?
NO
Fill in the trouble report. Use "7.4 Trouble Report."
Troubleshooting ended.
Contact your local source.

Figure 7-1 Troubleshooting Flow

7.2.2 Trouble detection and solution

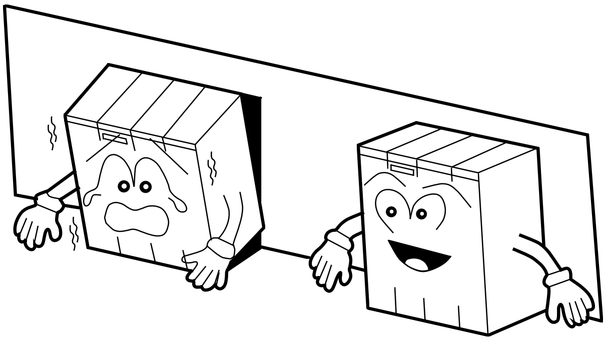
(1) Is the cabling correct?

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



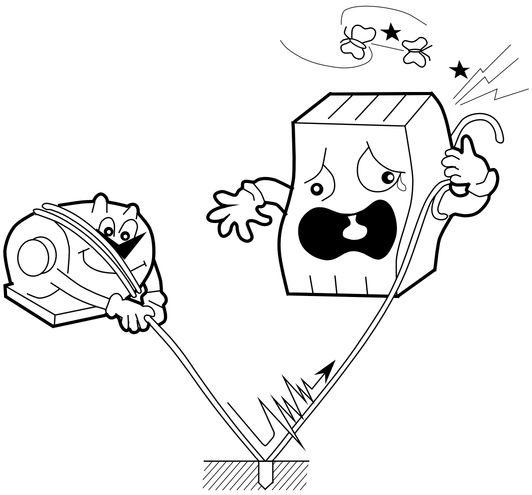
(2) Are the modules mounted correctly?

- Check if the ET.NET modules are inserted from the left so as not to be on empty space between the inserted modules and the ET.NET modules when the modules are inserted on a S10mini CPU mount base.
- Check that no set screws loosen.



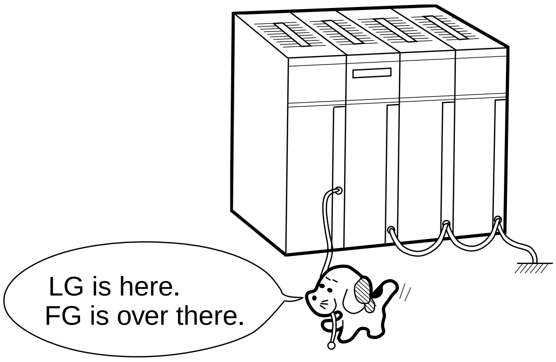
(3) Is grounding correct?

- 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 Error and Action To Be Taken

7.3.1 Indicator display messages

The S10mini displays the messages listed in Table 7-2 in the CPU module indicator when certain events or errors occur in the ET.NET module. These displays are identified by the distinction between the main module and submodule settings of the ET.NET module
The S10V collects error information, but does not display errors on the LPU module. Error information collected can be referenced from the S10V Base System. For more information, see "7.3.2 User action."

Table 7-2 S10mini CPU Module Display Messages

Module	Display message	Explanation	User action
Main module	ETM @, @	The ET.NET module is functioning successfully.	This is not an error and no user action is necessary.
	ETM □□□□	An error has occurred in the ET.NET module.	Take action as instructed in "7.3.2 User action."
	EXD2 PTY	A parity error occurred when the CPU read from the ET.NET module's internal memory.	Reset the CPU using the reset switch. If the display persists, replace the ET.NET module.
Submodule	ETM @, @	The ET.NET module is functioning successfully.	This is not an error and no user action is necessary.
	ETM □□□□	An error has occurred in the ET.NET module.	Take action as instructed in "7.3.2 User action."
	EXD2 PTY	A parity error occurred when the CPU read from the ET.NET module's internal memory.	Reset the CPU using the reset switch. If the display persists, replace the ET.NET module.

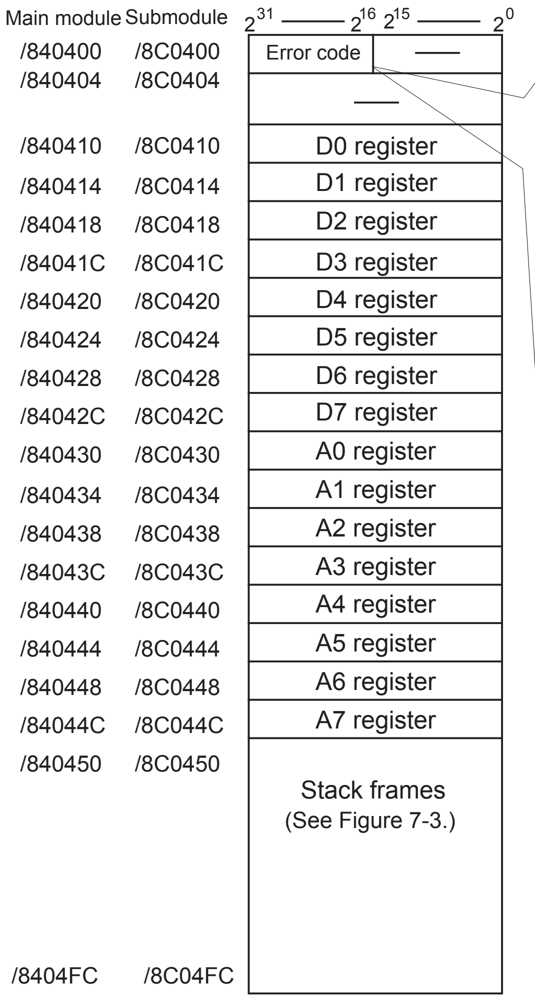
- The "@.@" above indicates the version and revision of the ET.NET module.
- The "□□□□" indicates the error display data in "7.3.2 User action."

7.3.2 User action

When the ET.NET module detects an error, the S10mini displays one of the CPU displays listed in Table 7-3 in the CPU module indicator, whereas the S10V displays one of the error codes in Table 7-3 by selecting an error log from the S10V Basic System. The ERR LED on the ET.NET module glows and error freeze information is collected at the same time. The details on error freeze information can be found in Figure 7-2. The ET.NET module shuts down its operation.
For information on how to start the tool system on the S10V and display error log information, refer to "USER'S MANUAL BASIC MODULES (manual number SVE-1-100)."

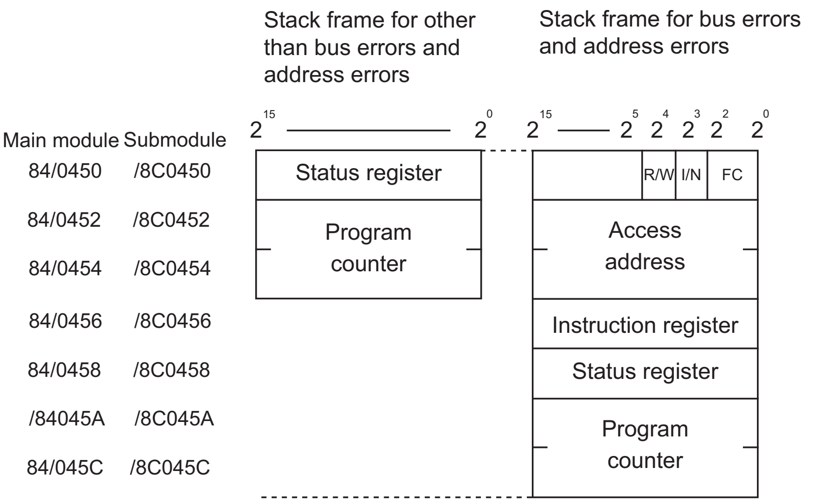
Table 7-3 Error Messages

CPU display (S10mini)	Error code (S10V)	Explanation	User action
BUS	/0010	Bus error	The ET.NET module has failed. Replace it.
ADDR	/0011	Address error	
ILLG	/0012	Invalid instruction	
ZERO	/0013	Division by zero	
PRIV	/0014	Privilege violation	
FMAT	/0016	Format error	
SINT	/0017	Spurious interrupt	
EXCP	/0018	Unsupported exception	
PTY	/0019	Party error	
ROM1	/0102	ROM1 checksum error	
RAM1	/0103	RAM1 compare error	
RAM2	/0105	RAM2 compare error	
ROM3	/010B	ROM3 checksum error	
MAC	/0114	Physical address not registered	
PRG	/0112	Micro-program error	
MDSW	/0100	Invalid Module number switch setting	The setting of the module number setting switch is invalid. See "2.1 Names and Function of Each Part," correct the switch setting.
IPNG	/0113	IP address not registered	Registered IP address
R_NG	/0200	Route information setting error	The setting of the routing information is invalid. See "7.3.4 Route information setting error table," correct the setting.


Main module
Submodule
2^{31} — 2^{16} 2^{15} — 2^{0}
/840400 /8C0400
Error code
—
/840404 /8C0404
—
/840410 /8C0410 D0 register
/840414 /8C0414 D1 register
/840418 /8C0418 D2 register
/84041C /8C041C D3 register
/840420 /8C0420 D4 register
/840424 /8C0424 D5 register
/840428 /8C0428 D6 register
/84042C /8C042C D7 register
/840430 /8C0430 A0 register
/840434 /8C0434 A1 register
/840438 /8C0438 A2 register
/84043C /8C043C A3 register
/840440 /8C0440 A4 register
/840444 /8C0444 A5 register
/840448 /8C0448 A6 register
/84044C /8C044C A7 register
/840450 /8C0450
Stack frames
(See Figure 7-3.)
/8404FC /8C04FC

No.	Error code	Error
1	/0010	Bus error
2	/0011	Address error
3	/0012	Invalid instruction
4	/0013	Division by zero
5	/0014	Privilege violation
6	/0016	Format error
7	/0017	Spurious interrupt
8	/0018	Unsupported exception (CHK, TRAPV, L1010, etc.)
9	/0019	Parity error
10	/001A	Power failure notice
11	/0100	Module switch setting error
12	/0102	ROM1 checksum error
13	/0103	RAM1 compare error
14	/0105	RAM2 compare error
15	/010B	ROM3 checksum error
16	/0112	Microprogram error
17	/0113	IP address not registered
18	/0114	MAC address error
19	/0200	Route information setting error

Figure 7-2 Error Freeze Information



R/W: (Read/Write): Write = 0 Read = 1
I/N: (Instruction/Non-instruction): Instruction = 0 Non-instruction = 1
FC: Function code

Figure 7-3 Stack Frame

7.3.3 Error codes of errors detected by socket handlers

The table below lists the error codes that may be generated by socket handlers, along with corrective action that must be taken by the user.

Table 7-4 Error Codes of Errors Detected by Socket Handlers

(1/3)

Error code	Error	Cause	User action
F000	Connection not established	When the handler was started, a connection was not yet established or a port was already opened.	Issue tcp_open( ) or tcp_popen( ) to establish a connection. Then re-call the handler function.
F002	FIN received	An FIN was received when the handler was started.	Issue tcp_close( ) to terminate the connection. Then, issue tcp_open( ) or tcp_popen( ) to re-establish a connection.
F010	Invalid socket ID	• The socket ID was out of range. (For TCP, 1 ≤ ID ≤ 15; for UDP, 0 × 20 ≤ ID ≤ 0 × 27) • The ID of an unused socket or already-opened socket was specified. • A connection was not yet established. (Applicable to tcp_accept( ) only)	Check the user program, for example, to see whether a value returned by tcp_open( ) or tcp_popen( ) is specified as the socket ID.
F011	Too many sockets	An attempt was made to register more sockets than the limit. (For TCP, 12; for UDP, 8)	Close unused sockets using tcp_close( ) or udp_close( ). Then, issue tcp_open( ) or tcp_popen( ) to re-establish a connection.
F012	Socket driver time-out	• The socket driver did not respond within the specified time. • A send operation timed out for such conditions as send window full (tcp_send() only).	Issue tcp_close( ) to terminate the connection. Then, issue tcp_open( ) or tcp_popen( ) to re-establish a connection. If communication still is not resumed, check the connectors, cables, and the remote station for any abnormality. When this error occurs due to tcp_close( ), issue tcp_abort( ), disconnect the line, and issue tcp_open( ) or tcp_popen( ) to re-establish a connection.
F013	Module stopped	When the handler was started, initialization of the socket driver was not terminated within 100 seconds.	Issue tcp_close( ) within the range allowed for the application. Then, issue tcp_open( ) or tcp_popen( ) to re-establish a connection.

(2/3)

Error code	Error	Cause	User action
F020	Invalid send data length	The send data length was out of range. (For TCP, 1 ≤ data length ≤ 4096; for UDP, 1 ≤ data length ≤ 1472)	Check the user program.
F021	Invalid receive data length	The receive data length was out of range. (1 ≤ data length ≤ 4096)	Check the user program.
F0FF	Port opened	• After the handler was started, a port was opened (RST was received). (tcp_open( )) • When an attempt was made to start the handler, the port was already opened. (tcp_send( ) or tcp_receive( ))	• Issue tcp_open( ) or tcp_popen( ) to re-establish a connection. • Issue tcp_close( ) to terminate the connection. Then, issue tcp_open( ) or tcp_popen( ) to re-establish a connection.
FFF0	Invalid address	• Both udp_open( ) and udp_send( ) set the IP address and port number of the remote station to 0. • udp_send( ) caused an Ethernet-level error such as a collision.	• Check the user program. • Retry udp_send( ) when the current amount of traffic is reduced.
FFF3	Invalid argument	An invalid argument was specified.	Check the user program.
FFF5	Connection time-out	The remote station did not respond.	Issue tcp_close( ) to terminate the connection. Then, issue tcp_open( ) or tcp_popen( ) to re-establish a connection. If communication still is not resumed, check the connectors, cables, and the remote station for any abnormality.
FFF6	Already closed	A command was issued for a socket ID for which a connection had been terminated (closed or aborted).	Issue tcp_open( ) or tcp_popen( ) to re-establish a connection.
FFF8	FIN received	An FIN was received from the remote station.	Issue tcp_close( ) to close the socket.
FFFA	Forcibly terminated connection	A connection was forcibly terminated by the remote station (RST was received). (tcp_receive( ) was issued after RST was received.)	Issue tcp_close( ) to terminate the connection. Then, issue tcp_open( ) or tcp_popen( ) to re-establish a connection.
FFFC	Invalid network handle	An attempt was made to perform transmission or reception using the number of a handle not yet opened by TCP or UDP. This error is likely to occur when an RST is received. (An RST was received during waiting for reception by tcp_receive( ).)	Issue tcp_close( ) to close the socket. Then, issue tcp_open( ) or tcp_popen( ) to re-establish a connection.

(3/3)

Error code	Error	Cause	User action
FFFD	Duplicate socket number	The same socket already existed. (The IP address of the remote station, the port number of the remote station, and the port number of the local station were duplicated.)	Check the user program.
FFFE	Invalid control block	An attempt was made to use more sockets than the limit.	Close unused sockets using tcp_close( ) or udp_close( ). Then, issue tcp_open( ) or tcp_popen( ) to re-establish a connection.

7.3.4 Route information setting error table

When a route information setting error is detected, its error code is set in the following table:

Main module	Submodule	2^{31} ——— 2^{0}
/873880	/8F3880	Default
/873884	/8F3884	User (1)
/873888	/8F3888	User (2)
/87388C	/8F388C	User (3)
/873890	/8F3890	User (4)
/873894	/8F3894	User (5)
/873898	/8F3898	User (6)
/87389C	/8F389C	User (7)
/8738A0	/8F38A0	User (8)
/8738A4	/8F38A4	User (9)
/8738A8	/8F38A8	User (10)
/8738AC	/8F38AC	User (11)
/8738B0	/8F38B0	User (12)
/8738B4	/8F38B4	User (13)
/8738B8	/8F38B8	User (14)

+0	Error code
+2	Duplicate user No.

Error code: See the table below.

Duplicate user No.: A stored user number is duplicated.
(Default = 0,
Other users = 1 to 14)

No.	Code	Contents	Duplicate user No. stored or not
1	/0010	The remote station IP address is duplicated with the local station IP address.	Not stored
2	/0011	The remote station IP address is duplicated with another gateway IP address.	Stored
3	/0012	The remote station IP address is duplicated with another remote station IP address.	Stored
4	/0013	The same network address as the local station's is set as the network address of a remote station IP address.	Not stored
5	/0014	The network address of a remote IP address is duplicated with the network address of a remote station IP address.	Stored
6	/0016	The remote station IP address is 255.255.255.255.	Not stored
7	/0020	The gateway IP address is duplicated with the local station IP address.	Not stored
8	/0022	The gateway IP address is duplicated with another local station IP address.	Stored
9	/0023	The same network address as the local station's is set as the network address of a gateway IP address.	Not stored
10	/0024	The network address of a gateway IP address is duplicated with the network address of another local station IP address.	Stored
11	/0026	The gateway IP address is 255.255.255.255.	Not stored
12	/0030	The subnetwork identified by a gateway IP address does not match the subnetwork of the local station. (*)	Not stored

(*) The user should be careful when, after setting necessary route information, he/she is connecting the ET.NET module with the tool by cable. Setting the ET.NET module's MODU No. switch in 4- or 5-position at that time may result in a route information setting error (error code /0030), except when the local station's IP address uses the network address "192.192.192.0". Recovery from this type of error can be made by simply setting the MODU No. switch back in its previous position. Even if this type of error is reported by error message, normal data communication is possible with the tool (personal computer) as long as the ET.NET module is connected directly with that tool by cable.

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			