User's Manual

Sequence CPU – Network Functions
(for F3SP66-4S, F3SP67-6S)

YOKOGAWA ◆
Yokogawa Electric Corporation

Blank Page

Applicable Product:

■ Range-free-controller FA-M3

- Model Name: F3SP66-4S, F3SP67-6S
- Name: Sequence CPU Module (with network functions)

The document number and document model code for this manual are given below.
Refer to the document number in all communications; also refer to the document
number or the document model code when purchasing additional copies of this manual.

Document No. : IM 34M6P14-02E
Document Model Code : DOCIM

Important

■ About This Manual

- This Manual should be passed on to the end user.

- Before using the controller, read this manual thoroughly to have a clear understanding of the controller.

- This manual explains the functions of this product, but there is no guarantee that they will suit the particular purpose of the user.

- Under absolutely no circumstances may the contents of this manual be transcribed or copied, in part or in whole, without permission.

- The contents of this manual are subject to change without prior notice.

- Every effort has been made to ensure accuracy in the preparation of this manual. However, should any errors or omissions come to the attention of the user, please contact the nearest Yokogawa Electric representative or sales office.

■ Safety Precautions when Using/Maintaining the Product

- The following safety symbols are used on the product as well as in this manual.

Danger. This symbol on the product indicates that the operator must follow the instructions laid out in this user's manual to avoid the risk of personnel injuries, fatalities, or damage to the instrument. Where indicated by this symbol, the manual describes what special care the operator must exercise to prevent electrical shock or other dangers that may result in injury or the loss of life.

Protective Ground Terminal. Before using the instrument, be sure to ground this terminal.

Function Ground Terminal. Before using the instrument, be sure to ground this terminal.

Alternating current. Indicates alternating current.

Direct current. Indicates direct current.

The following symbols are used only in the user's manual.

⚠ WARNING

Indicates a "Warning".

Draws attention to information essential to prevent hardware damage, software damage or system failure.

⚠ CAUTION

Indicates a "Caution"

Draws attention to information essential to the understanding of operation and functions.

TIP

Indicates a "TIP"

Gives information that complements the present topic.

SEE ALSO

Indicates a "SEE ALSO" reference.

Identifies a source to which to refer.

- For the protection and safe use of the product and the system controlled by it, be sure to follow the instructions and precautions on safety stated in this manual whenever handling the product. Take special note that if you handle the product in a manner other than prescribed in these instructions, the protection feature of the product may be damaged or impaired. In such cases, Yokogawa cannot guarantee the quality, performance, function and safety of the product.

- When installing protection and/or safety circuits such as lightning protection devices and equipment for the product and control system as well as designing or installing separate protection and/or safety circuits for fool-proof design and fail-safe design of processes and lines using the product and the system controlled by it, the user should implement it using devices and equipment, additional to this product.

- If component parts or consumable are to be replaced, be sure to use parts specified by the company.

- This product is not designed or manufactured to be used in critical applications which directly affect or threaten human lives and safety — such as nuclear power equipment, devices using radioactivity, railway facilities, aviation equipment, air navigation facilities, aviation facilities or medical equipment. If so used, it is the user's responsibility to include in the system additional equipment and devices that ensure personnel safety.

- Do not attempt to modify the product.

■ Exemption from Responsibility

- Yokogawa Electric Corporation (hereinafter simply referred to as Yokogawa Electric) makes no warranties regarding the product except those stated in the WARRANTY that is provided separately.

- Yokogawa Electric assumes no liability to any party for any loss or damage, direct or indirect, caused by the use or any unpredictable defect of the product.

■ Software Supplied by the Company

- Yokogawa Electric makes no other warranties expressed or implied except as provided in its warranty clause for software supplied by the company.
- Use the software with one computer only. You must purchase another copy of the software for use with each additional computer.
- Copying the software for any purposes other than backup is strictly prohibited.
- Store the original media, such as floppy disks, that contain the software in a safe place.
- Reverse engineering, such as decompiling of the software, is strictly prohibited.
- No portion of the software supplied by Yokogawa Electric may be transferred, exchanged, or sublet or leased for use by any third party without prior permission by Yokogawa Electric.

■ General Requirements for Using the FA-M3 Controller

■ Avoid installing the FA-M3 controller in the following locations:

- Where the instrument will be exposed to direct sunlight, or where the operating temperature exceeds the range 0°C to 55°C (32°F to 131°F).
- Where the relative humidity is outside the range 10 to 90%, or where sudden temperature changes may occur and cause condensation.
- Where corrosive or flammable gases are present.
- Where the instrument will be exposed to direct mechanical vibration or shock.
- Where the instrument may be exposed to extreme levels of radioactivity.

■ Use the correct types of wire for external wiring:

- Use copper wire with temperature ratings greater than 75°C.

■ Securely tighten screws:

- Securely tighten module mounting screws and terminal screws to avoid problems such as faulty operation.
- Tighten terminal block screws with the correct tightening torque as given in this manual.

■ Securely lock connecting cables:

- Securely lock the connectors of cables, and check them thoroughly before turning on the power.

■ Interlock with emergency-stop circuitry using external relays:

- Equipment incorporating the FA-M3 controller must be furnished with emergency-stop circuitry that uses external relays. This circuitry should be set up to interlock correctly with controller status (stop/run).

■ Ground for low impedance:

- For safety reasons, connect the [FG] grounding terminal to a Japanese Industrial Standards (JIS) Class D Ground[*1] (Japanese Industrial Standards (JIS) Class 3 Ground). For compliance to CE Marking, use braided or other wires that can ensure low impedance even at high frequencies for grounding.

  *1 Japanese Industrial Standard (JIS) Class D Ground means grounding resistance of 100 Ω max.

■ Configure and route cables with noise control considerations:

- Perform installation and wiring that segregates system parts that may likely become noise sources and system parts that are susceptible to noise. Segregation can be achieved by measures such as segregating by distance, installing a filter or segregating the grounding system.

■ Configure for CE Marking Conformance:

- For compliance to CE Marking, perform installation and cable routing according to the description on compliance to CE Marking in the "Hardware Manual" (IM34M6C11-01E).

■ **Keep spare parts on hand:**

- We recommend that you stock up on maintenance parts including spare modules.

- Preventive maintenance (replacement of the module or its battery) is required for using the module beyond 10 years. For enquiries on battery replacement service, contact your nearest Yokogawa Electric representative or sales office. (The module has a built-in lithium battery. Lithium batteries may exhibit decreased voltage, and in rare cases, leakage problems after ten years.)

■ **Discharge static electricity before operating the system:**

- Because static charge can accumulate in dry conditions, first touch grounded metal to discharge any static electricity before touching the system.

■ **Never use solvents such as paint thinner for cleaning:**

- Gently clean the surfaces of the FA-M3 controller with a cloth that has been soaked in water or a neutral detergent and wringed.

- Do not use volatile solvents such as benzine or paint thinner or chemicals for cleaning, as they may cause deformity, discoloration, or malfunctioning.

■ **Avoid storing the FA-M3 controller in places with high temperature or humidity:**

- CPU modules and temperature control modules (F3CT04-□N, F3CR04-□N, F3CV04-1N) have built-in batteries. Avoid storing modules with built-in batteries under high temperature or humidity conditions. Storage at room temperature is recommended.

- The service life of batteries may be shortened when installed or stored at locations of extreme low or high temperatures. Beware that service life of batteries may be drastically shortened under high-temperature conditions (storage temperature should be between −20°C and 75°C).

■ **Always turn off the power before installing or removing modules:**

- Failing to turn off the power supply when installing or removing modules, may result in damage.

■ **Do not touch components in the module:**

- In some modules you can remove the right-side cover and install ROM packs or change switch settings. While doing this, do not touch any components on the printed-circuit board, otherwise components may be damaged and modules may fail to work.

■ **Do not use unused terminals:**

- Do not connect wires to unused terminals on a terminal block or in a connector. Doing so may adversely affect the functions of the module.

■ Waste Electrical and Electronic Equipment

Waste Electrical and Electronic Equipment (WEEE), Directive 2002/96/EC

(This directive is only valid in the EU.)

This product complies with the WEEE Directive (2002/96/EC) marking requirement. The following marking indicates that you must not discard this electrical/electronic product in domestic household waste.

Product Category

With reference to the equipment types in the WEEE directive Annex 1, this product is classified as a "Monitoring and Control instrumentation" product.

Do not dispose in domestic household waste.

When disposing products in the EU, contact your local Yokogawa Europe B. V. office.

Introduction

■ Overview of the Manual

This manual describes the network functions of the F3SP66-4S and F3SP67-6S sequence CPU modules for use with the Range-free Multi-controller FA-M3. For details on sequencing, debug, storage and other non-network functions of the modules, see "Sequence CPU – Functions (for F3SP66-4S, F3SP67-6S)" (IM34M6P14-01E).

■ How to Read the Manual

Start with Chapter 1, "CPU Built-in Network Functions," which outlines the network functions of the modules.

Chapters 2 and beyond describe the main protocols and network functions in separate chapters. You may read selected chapters that relate to your application. Each chapter is further divided by sub-function with detailed descriptions on usage, setup and instructions/commands. References to other manuals for more details are included wherever appropriate.

■ Other User's Manuals

Be sure to read each of the following manuals, in addition to this manual.

■ For information on the general functions of the F3SP66-4S or F3S67-6S sequence CPU module, refer to:

- Sequence CPU - Functions (for F3SP66-4S, F3SP67-6S) (IM 34M6P14-01E)

■ For information on the instructions used with sequence CPUs, refer to:

- Sequence CPU - Instructions (IM34M6P12-03E)

■ For information on the commands and responses of personal computer link functions, refer to:

- Personal Computer Link Command (IM34M6P41-01E).

■ When creating programs using ladder language, refer to:

- FA-M3 Programming Tool WideField2 (IM34M6Q15-01E)

■ For information on the specifications*, configuration*, installation, wiring, trial operation, maintenance and inspection of the FA-M3, as well as information on the system-wide limitation of module installation, refer to:

- Hardware Manual (IM34M6C11-01E).

*: For information on the specifications of products other than the power supply module, base module, I/O module, cable and terminal block unit, refer to their respective user's manuals.

Read the following user's manuals, as required.

■ **For information on the functions of fiber-optic FA-bus modules, refer to:**

- Fiber-optic FA-bus Module and Fiber-optic FA-bus Type 2 Module (IM34M6H45-01E).

■ **For information on the functions of FA link H and fiber-optic FA link H modules, refer to:**

- FA Link H Module, Fiber-optic FA Link H Module (IM34M6H43-01E).

■ **For information on the FL-net functions, refer to:**

- FL-net (OPCN-2) Interface Module (IM 34M6H32-02E)

■ **For information on the functions of the Ethernet Interface Module, refer to:**

- Ethernet Interface Module (IM 34M6H24-01E)

■ **For information on the functions of BASIC CPU modules, refer to:**

- BASIC CPU Modules and YM-BASIC/FA Programming Language (IM34M6Q22-01E).

Copyrights and Trademarks

■ Copyrights

Copyrights of the programs and online manual included in this CD-ROM belong to Yokogawa Electric Corporation.

This online manual may be printed but PDF security settings have been made to prevent alteration of its contents.

This online manual may only be printed and used for the sole purpose of operating this product. When using a printed copy of the online manual, pay attention to possible inconsistencies with the latest version of the online manual. Ensure that the edition agrees with the latest CD-ROM version.

Copying, passing, selling or distribution (including transferring over computer networks) of the contents of the online manual, in part or in whole, to any third party, is strictly prohibited. Registering or recording onto videotapes and other media is also prohibited without expressed permission of Yokogawa Electric Corporation.

■ Trademarks

The trade and company names that are referred to in this document are either trademarks or registered trademarks of their respective companies.

FA-M3
Sequence CPU – Network Functions
(for F3SP66-4S, F3SP67-6S)

IM 34M6P14-02E   1st Edition

CONTENTS

1. CPU Built-in Network Functions

This chapter describes the built-in network functions of the CPU module.

1.1 Overview of CPU Built-in Network Functions

The CPU module comes ready with built-in network capability, which supports the functions described below.

● Socket Communications Function

The socket communications function enables communications between the CPU module and a PC or other equipment using the TCP/IP or UDP/IP protocol. It also allows broadcasting using UDP/IP.

● FTP Function

The FTP function performs file transfer between the CPU module and a PC or other equipment. Both FTP client function and FTP server function are supported.

In addition, a virtual directory function is provided as a proprietary extension of the FTP server function. The virtual directory function enables various FA-M3 operations to be added to FTP commands. These operations include manipulating device data and switching the operating mode.

● Higher-level Link Service (PC Link Function)

The Higher-level Link Service (PC Link function) is a FA-M3 proprietary protocol, which enables various FA-M3 operations such as reading and writing device data or switching operating mode to be performed from a higher-level computer (e.g. PC).

● Remote Programming Service

The remote programming service enables connection between the Programming Tool WideField2 software and the CPU module.

● DNS Function

The DNS function allows DNS clients to access the network using hostnames in place of IP addresses in FTP client instructions and socket instructions. To use DNS, a DNS server must be present on the network.

1.2 10BASE-T/100BASE-TX Connector Specifications

This section describes the 10BASE-T/100BASE-TX connector, which is located on the front panel of the module.

1.2.1 CPU Built-in 10BASE-T/100BASE-TX Connector Specifications

■ General Specifications

Table 1.2.1   10BASE-T/100BASE-TX Connector Specifications

Item		Specifications	
		Ethernet	
Interface		10BASE-T	100BASE-TX
Transmission Specifications	Access control	CSMA/CD	
	Transmission rate	10 Mbps	100 Mbps
	Transmission method	Baseband	
	Maximum distance between nodes [*1]	100 m	
Protocols [*2]		TCP, UDP, IP, ICMP, ARP, FTP	
Transmission rate setting		Auto-detect (10 Mbps or100Mbps)	
Transmission mode		Full duplex or half duplex	

*1: Distance between a hub and the module.
*2: Some parts of the software of the Regents of University of California have been incorporated.

TIP

The CPU module is not counted as a module in the maximum limit of six installed Ethernet Interface Modules defined in the "Hardware Manual" (IM34M6C11-01E).

■ Special Relays and Special Registers

The tables below list the special relays and special registers related to the 10BASE-T/100BASE-TX connector.

SEE ALSO

- For details on special relays and special registers related to the Socket Communications function, see Subsection 2.2.3, "Special Relays and Special Registers."

- For details on special relays and special registers related to the FTP Client function, see "■ Resource Relays" of Subsection 3.4.1, "Using FTP Client Instructions."

● Special Relays

Table 1.2.2   Special Relays Related to 10BASE-T/100BASE-TX Connector

Category	10BASE-T/100BASE-TX Connector Special Relays		
No.	Name	Function	Description
M0241	Link Status	ON: Link is up OFF: Link is down	Indicates the status of the link. Its status is interlocked with the ON/OFF status of the LNK LED located on the front panel of the module.

● Special Registers

Table 1.2.3   Special Registers Related to 10BASE-T/100BASE-TX Connector

Category	10BASE-T/100BASE-TX Connector Special Registers		
No.	Name	Function	Description
Z0114	MAC Address	MAC address low word	Low-order 16 bits [$xxxx]
Z0115		MAC address mid word	Mid-order 16 bits [$64xx]
Z0116		MAC address high word	High-order 16 bits [$0000]

1.2.2 Cable Connection

This subsection describes how to connect a cable to the 10BASE-T/100BASE-TX connector.

■ Connecting using 10BASE-T

10BASE-T is an Ethernet connection method using twisted-pair cables (STP/UTP cables).

- The transmission rate is 10 Mbps.
- Uses category 3, 4 or 5 cables.
- In a 10BASE-T network, equipment such as PCs are connected to a hub using a star topology.
- Hubs and twisted-pair cables used must conform to the Ethernet (10BASE-T) specifications.
- Up to 4 segments are allowed for cascade connections to the hub.
- The maximum length allowed for the twisted-pair cables is 100 m.

■ Connecting using 100BASE-TX

100BASE-TX is an Ethernet connection method using twisted-pair cables (STP/UTP cables).

- The transmission rate is 100 Mbps.
- Uses category 5 cables.
- In a 100BASE-TX network, equipment such as PCs are connected to a hub using a star topology.
- Hubs and twisted-pair cables used must conform to the Ethernet (100BASE-TX) specifications.
- Up to 2 segments are allowed for cascade connections to the hub.
- The maximum length allowed for the twisted-pair cables is 100 m.

■ How to Connect the Cable

Push the modular plug of the cable into the modular jack of the10BASE-T/100BASE-TX connector until it clicks into place.

· Insert the cable with the latch facing left (towards the power supply module) until it locks into place with a clicking sound.
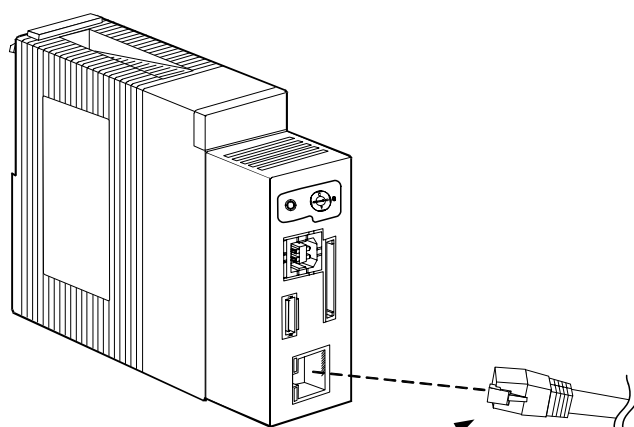
F0101.VSD

Figure 1.2.1   Orientation for Cable Insertion

1.2.3　Network Setup before Operation

This subsection describes the basic network setup (IP address, etc.) before operation. The setup involves changing CPU property values using any of the following means:

- Setup using WideField2 (via USB)
- Setup using WideField2 (via Ethernet)
- Setup using rotary switch function
- Setup using card batch file function

SEE ALSO

For details of CPU properties, see Chapter A9, "Setup Description" of "Sequence CPU – Functions (for F3SP66-4S, F3SP67-6S)" (IM34M6P14-01E)

● **Setup using WideField2 (via USB)**

Connect the PC and the CPU module using USB, and change the required CPU property values using the Programming Tool WideField2 software (hereinafter abbreviated as "WideField2").

SEE ALSO

For details on how to change CPU property values using WideField2, see "FA-M3 Programming Tool WideField2" (IM34M6Q15-01E).

● **Setup using WideField2 (via Ethernet)**

Connect the PC and the module using a crossed cable, and change the required CPU property values using WideField2. Beware that the network configuration on the PC must match the factory network settings of the module, a subset of which is shown in the table below.

Table 1.2.4　　Factory Network Settings (subset)

Setup Item	Initial Value
IP address	192.168.0.2
Subnet mask	255.255.255.0

TIP

If the module is shipped with no configuration, the IP address and other settings may be different from its normal factory settings. In such situations, you can perform network setup using other means described in this section or restore the module to its factory settings using the rotary switch function.

SEE ALSO

- For details on how to change CPU properties using WideField2, see "FA-M3 Programming Tool WideField2" (IM34M6Q15-01E).
- For details on network configuration of the PC, read the documentation of the PC and the operating system.
- For details on how to restore the module to factory settings using the rotary switch function, see Subsection B1.5.4, "Restore Factory Settings" of "Sequence CPU – Functions (for F3SP66-4S, F3SP67-6S)" (IM34M6P14-01E).

● Setup using rotary switch function

You can load a CPU property file stored on a SD memory card into the module using the rotary switch. To find out the current module settings, you can also save CPU property data from the module to a CPU property file on the SD memory card. The CPU property file can be edited using any generic text editor.

SEE ALSO

- For details on how to load data using the rotary switch function, see Subsection B1.4.6, "Load Project from CARD1" of "Sequence CPU – Functions (for F3SP66-4S, F3SP67-6S)" (IM34M6P14-01E).

- For details on how to save data using the rotary switch function, see Subsection B1.4.4, "Save Project to CARD1" of "Sequence CPU – Functions (for F3SP66-4S, F3SP67-6S)" (IM34M6P14-01E).

● Setup using card batch file function

You can load a CPU property file stored on the SD memory card to the module by executing a card batch file stored on an SD memory card. To check current module settings, you can save CPU property data from the module to a CPU property file on the SD memory card. The CPU property file can be edited using any generic text editor.

SEE ALSO

- For details on how to load data using the card batch file function, see Subsection B2.8.2.1, "Load Project (LOAD)" of "Sequence CPU – Functions (for F3SP66-4S, F3SP67-6S)" (IM34M6P14-01E).

- For details on how to save data using the card batch file function, see Subsection B2.8.2.2, "Save Project (SAVE)" of "Sequence CPU – Functions (for F3SP66-4S, F3SP67-6S)" (IM34M6P14-01E).

1.3 CPU Built-in Network Diagnosis Function

This section describes the built-in network diagnosis function of the CPU module.

1.3.1 Self Diagnosis

The module has no self diagnosis function related to its network function.

1.3.2 ping

"ping" is a standard command used for verifying connections between machines connected using Ethernet. The module is capable of replying to ping commands. By executing a "ping" command from the command prompt of a PC, specifying the IP address or hostname of the module, you can determine whether the connection between the module and the PC is normal. If connection is normal, the module returns a reply within the timeout interval.

Example: Executing a "ping" command to IP address 192.168.0.6 from the command prompt of a PC and receiving a reply:

```
C:\>ping 192.168.0.6

Pinging 192.168.0.6 with 32 bytes of data:

Reply from 192.168.0.6: bytes=32 time<10ms TTL=128
Reply from 192.168.0.6: bytes=32 time<10ms TTL=128
Reply from 192.168.0.6: bytes=32 time<10ms TTL=128
Reply from 192.168.0.6: bytes=32 time<10ms TTL=128

Ping statistics for 192.168.0.6:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum =  0ms, Average =  0ms
```

Example: Executing a "ping" command to IP address 10.0.142.79 from the command prompt of a PC and receiving no reply:

```
C:\>ping 10.0.142.79

Pinging 10.0.142.79 with 32 bytes of data:

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

Ping statistics for 10.0.142.79:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum =  0ms, Average =  0ms
```

Example: Executing a "ping" command to hostname "abc001" from the command prompt of a PC but the hostname or DNS setup is invalid:

```
C:\>ping abc001
Unknown host abc001.
```

1.3.3   Troubleshooting Communications Problems

The subsection describes how to troubleshoot communications problems using the CPU built-in network functions.

■ If "ping" Fails:

If no reply is received for a "ping" command executed as described in Subsection 1.3.2, "ping," perform the following checks:

● **Are cables attached securely?**

- Ensure that all cables to the module and network equipment (hub, routers, etc.) are securely attached.
- If the remote equipment is directly connected to the module, ensure that a crossed cable is used.

● **Is power supplied to network equipment?**

- Ensure that power is supplied to the module and network equipment (hub, router, etc.).

● **Further checks**

Perform the checks described in "If "Ping" is Successful:" below

■ If "ping" is Successful:

If a reply is returned for a "ping" command executed as described in Subsection 1.3.2, "ping," perform the following checks:

● **Are CPU property values valid?**

- Ensure that the IP address and subnet mask values are valid.
- If hostnames are used for communications, check DNS related setup. Check with the network administrator on the required DNS settings. To narrow down possibilities, we recommend that you perform a test using the IP address in place of the hostname.
- If you are using FTP client or socket communications functions of the module, check the destination IP address and port number setup.

● **Are network equipment configured properly?**

- Check the setup of all network equipment by reading the related documentation.
- If the PC is running as a client through higher-level link service or tool service, ensure that the IP address or port number is correctly specified for the module.
- If a router is used, check whether its specific port number is blocked. Similarly, if a PC is used, check whether its specific port is blocked by a firewall.

● **Is network configuration valid?**

- Check network configuration details with the network administrator.
- If hostnames are used for communications, check the use of the DNS server.

2. Socket Communications Function

This chapter describes the socket communications function.

2.1 Overview of Socket Communications Function

This section gives a general overview of socket communications as a standard communications protocol, followed by description of the socket communications functions supported by the module.

2.1.1 Overview of Socket Communications

Socket communications provides an interface for easy communications using the TCP/IP and UDP/IP protocols. It is known as "Winsock" in Microsoft Windows terminology. With socket communications, PCs installed with UNIX or Windows can communicate with each other using the TCP/IP or UDP/IP protocols.

In socket communications, destinations are viewed as sockets (or plugs), each represented by a combination of an IP address and a port number. By issuing send requests and receive requests to a socket, user programs can accomplish TCP/IP communications and UDP/IP communications without complex programming.
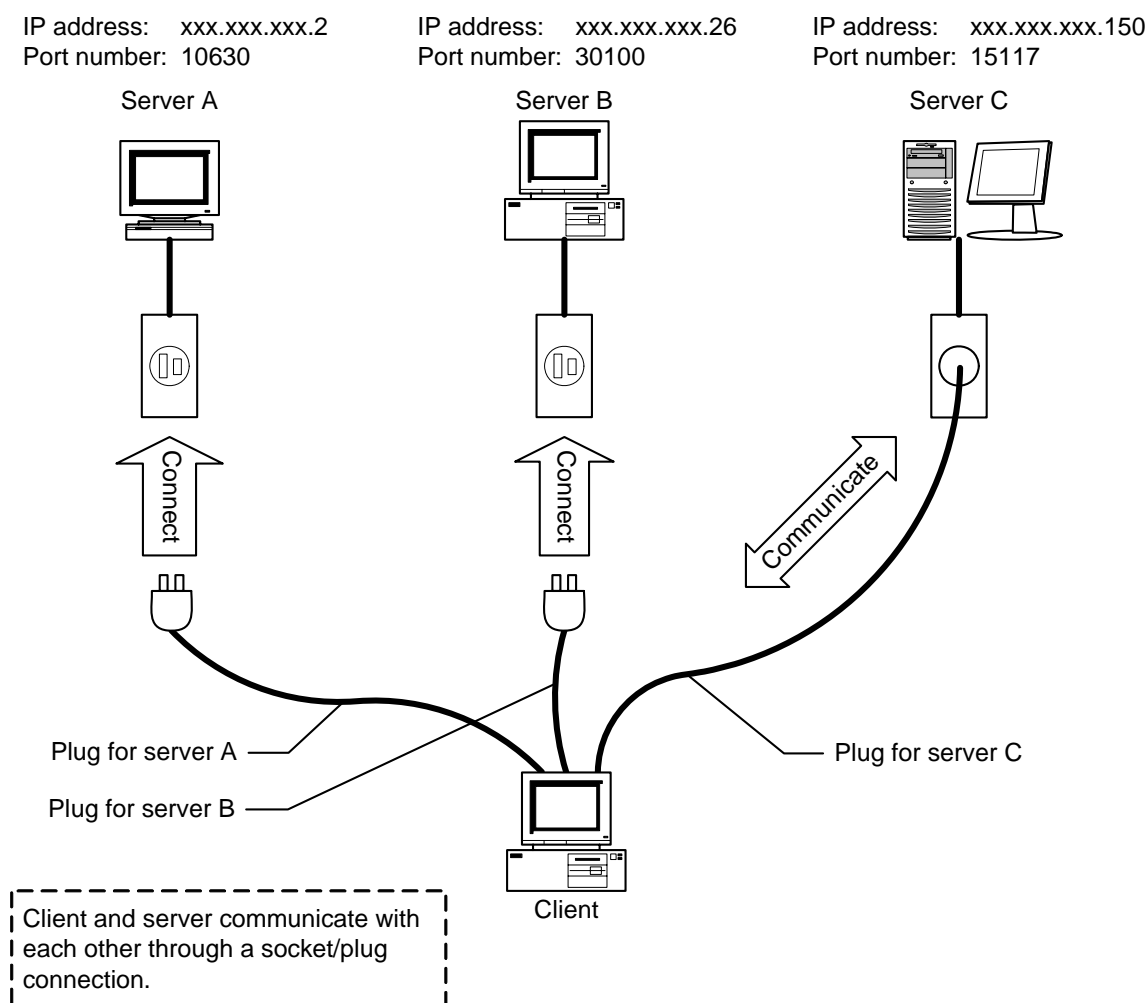

Figure 2.1.1   Roles of Sockets (Illustrative Diagram)

2.1.2 Socket Communications Functions Supported by the Module

Socket communications functions supported by the module are described below.

● **TCP/IP socket communications**

TCP/IP socket communications performs TCP/IP communications by establishing a virtual telephone-line-like one-to-one transmission channel between two nodes. Data retransmission is carried out within the protocol, which is transparent to the application. The application can safely assume a reliable transmission channel. Both client communications and server communications can be carried out.

The price for reliable data transmission is lower transmission speed compared to UDP due to heavier protocol processing.

● **UDP/IP socket communications**

UDP/IP socket communications differ from TCP/IP socket communications in the following ways:

- Reliable data transmission is not guaranteed by the protocol (throw-and-forget)
- Multiple nodes can communicate using the same socket without establishing a one-to-one transmission channel.
- Supports broadcast feature.

Reliable data transmission is the responsibility of the application. This allows a simpler protocol and thus a higher transmission speed than TCP.

● **UDP/IP broadcast**

UDP/IP broadcast allows the same packet to be sent to a large number of unspecified network nodes using a special IP address representing the entire network.

The multicast function is not supported.

● **Features**

- Socket communications instructions

    A wide range of socket communications instructions are provided. This simplifies programming and improves program readability when compared to the traditional relay interface.

- Network filter function

    Enables creation of secure applications by restricting the IP addresses that are allowed to establish connections. A mask can also be used to grant permissions to a class of network addresses.

- Automatic allocation of socket ID

    Socket IDs are automatically allocated when sockets are opened, so that management of socket ID numbers is transparent to the programmer when creating or reusing communications programs.

- Error handling by status notification

    At the end of instruction execution, any error status is reported to a user for further handling. Programming for reconnecting after disconnection and retransmission is relatively easy.

2.2 Socket Communications Function Specifications

This section describes the specifications of the socket communications function.

2.2.1 Socket Communications Function Specifications

Table 2.2.1　　Socket Communications Function Specifications

	TCP/IP Socket Service	UDP/IP Socket Service
Number of sockets	8	8
Send buffer size[4]	4K bytes per socket	4K bytes per socket
Receive buffer size[4]	4K bytes per socket	4K bytes per socket
Maximum size for sending[1]	2K bytes per socket	2K bytes per socket
Maximum size for receiving[1]	2K bytes per socket	4K bytes per socket
Number of registered IP addresses[2]	16 (Socket address settings 1-16 of CPU properties)	
Number of TCP/IP server connections[3]	7	–

*1: This is the maximum data size that can be handled per send or receive operation for TCP/IP sockets or the maximum packet size that can be handled for UDP/IP sockets.
*2: This limit applies only if socket address settings of CPU properties are used in socket instructions. There is no limit if IP addresses and port numbers are directly specified in socket instructions.
*3: This limit applies to the case where one TCP/IP socket is used for listening and the rest are used for client connections.
*4: In the event of multiple send and receive events within a short duration, the buffer size available for send/receive data is somewhat reduced due to consumption by protocol headers.

2.2.2 List of Socket Communications Instructions

Table 2.2.2　　List of Socket Communications Instructions

Service Name	Ladder Instruction	Description
UDP/IP Open	UDPOPEN	Opens a UDP/IP socket to enable communications.
UDP/IP Close	UDPCLOSE	Closes a UDP/IP socket.
UDP/IP Send Request	UDPSND	Sends data from a specified UDP/IP socket.
UDP/IP Receive Request	UDPRCV	Receives data from a specified UDP/IP socket.
TCP/IP Open	TCPOPEN	Opens a TCP/IP socket.
TCP/IP Close	TCPCLOSE	Closes a TCP/IP socket.
TCP/IP Connect Request	TCPCNCT	Issues a connection request to a server as a client, and connects to the server if permitted to do so.
TCP/IP Listen Request	TCPLISN	Waits for a connection request from any client as a server, and establishes connection if a request is received.
TCP/IP Send Request	TCPSND	Sends data from a specified TCP/IP socket.
TCP/IP Receive Request	TCPRCV	Receives data from a specified TCP/IP socket.

2.2.3 Special Relays and Special Registers

■ Special Relays

Table 2.2.3   Special Relays Related to Socket Communications

Category	Continuous Type Application Instruction Resource Relays		
No.	Name	Function	Description
M1028	No Unused UDP Socket	No unused UDP socket is available.	Turns on when all UDP/IP sockets are in use.
M1029	No Unused TCP Socket	No unused TCP socket is available.	Turns on when all TCP/IP sockets are in use.
M1105 to M1120	Socket Open	Socket is open.	Each socket ID is associated with one special relay. The relay for a socket ID turns on while the socket ID is open. When the relay for a socket ID is OFF, the socket ID cannot be used. This is a read-only relay. Do not write to it.
M1121 to M1136	Socket Busy	Socket is busy.	Each socket ID is associated with one special relay. The relay for a socket ID turns on during execution of any socket instruction using the socket ID. When the relay for a socket ID is ON, no other socket communication instruction using the same socket ID can be executed except for concurrent execution of sending and receiving. This is a read-only relay. Do not write to it.
M1073 to M1088	Socket Sending	Socket is performing send processing.	Each socket ID is associated with one special relay. The relay for a socket ID turns on during send processing of the socket. When the relay for a socket ID is ON, no send request is allowed for the same socket ID. This is a read-only relay. Do not write to it.
M1089 to M1104	Socket Receiving	Socket is performing receive processing.	Each socket ID is associated with one special relay. The relay for a socket ID turns on during receive processing of the socket. When the relay for a socket ID is ON, no receive request is allowed for the same socket ID. This is a read-only relay. Do not write to it.

■ Special Registers

There are no special registers related the socket communications function.

2.3   Socket Communications Network Configurations

This section describes possible network configurations for using socket communications functions.

■ Network Configuration

Socket communication functions can be used in an Ethernet network environment.

■ IP Routing

IP routing using default gateway and subnet mask is supported. Both can be specified using Ethernet setup of CPU properties.

2.4 Socket Communications Setup

This section describes how to configure socket communications before use.

2.4.1 Basic Setup

The table below shows required setup for socket communications before use.

Table 2.4.1   Basic Setup for Socket Communications

Name of Setup	Type of Setup	SEE ALSO[*1]
Ethernet setup	CPU properties	A9.5.2, 'Ethernet Setup"
Socket address setup	CPU properties	A9.5.3, "Socket Setup"

*1: For details on individual setup items, see "Sequence CPU – Functions (for F3SP66-4S, F3SP67-6S)" (IM34M6P14-01E).

● **Ethernet setup**

The Ethernet setup configures the CPU module for joining an Ethernet network.

-   Minimally, you must specify the IP address and subnet mask. If you set the subnet mask to "0.0.0.0", the default mask for the class of the IP address is used.
-   If you need to access another network via a gateway, you must define the default gateway.
-   To access other network nodes by hostname, you must define the DNS related settings (DNS server, my hostname, domain name, domain suffixes).

● **Socket address setup**

Socket address setup defines the port number and IP address (or hostname) of one or more socket communications destinations. Once defined, you can specify a destination using its socket address setting number in socket instructions.

2.4.2 Optional Setup

The socket communications function may be configured as required before use.

Table 2.4.2   Optional Setup for Socket Communications

Name of Setup	Type of Setup	SEE ALSO[*1]
Socket setup	CPU properties	A9.5.3, "Socket setup"
Network filter setup	CPU properties	A9.5.8, "Network filter setup"

*1: For details on individual setup items, see "Sequence CPU – Functions (for F3SP66-4S, F3SP67-6S)" (IM34M6P14-01E).

● **Socket setup**

Perform socket setup in the following situations:

-   If using UDP/IP broadcast function
-   To modify the TCP/IP keep-alive time

● **Network filter setup**

You may perform network filter setup to restrict the IP addresses for connection to the module. By default, connections from all IP addresses are allowed. This setup affects all functions (e.g. remote programming service, FTP server, etc.) running on TCP/IP or UDP/IP.

2.5   Using Socket Communications

This section describes the procedure and precautions for using the socket communications function.

2.5.1   UDP/IP Socket Communications Procedure

This subsection describes the procedure for UDP/IP socket communications.

■ Flowchart for UDP/IP Socket Communications

The flowchart for UDP/IP socket communications is shown below.
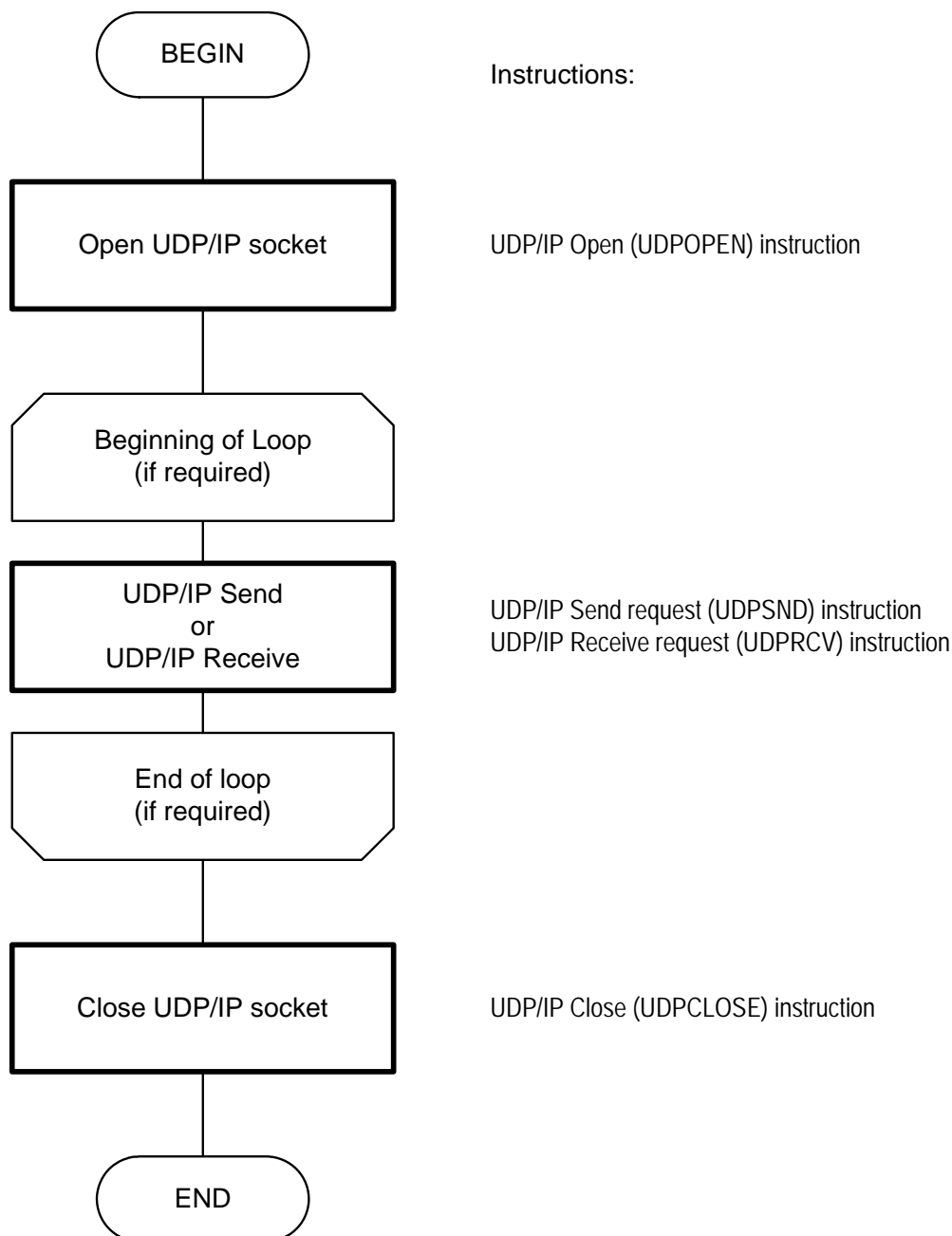
F0204.VSD

Figure 2.5.1   Flowchart for UDP/IP Socket Communications

■ Preparing for UDP/IP Socket Communications

● Instructions Used

Table 2.5.1   Instructions Used for Preparation of UDP/IP Socket Communications

Instruction Name	Instruction Mnemonic	Description
UDP/IP Open	UDPOPEN	Opens a UDP/IP socket.

● Procedure

1. Execute a UDP/IP Open (UDPOPEN) instruction to open a UDP/IP socket.

Opening a socket secures system resources required for communications.

If a socket is successfully opened, the UDP/IP Open (UDPOPEN) instruction returns a socket ID as the Status. This socket ID must be specified in subsequent send or receive instructions to identify the socket.

Up to 8 UDP/IP sockets can be open concurrently at any one time.

■ Send Procedure for UDP/IP Socket

This procedure sends data stored in devices (e.g. file registers (B)) via an open socket by executing a UDP/IP Send Request (UDPSND) instruction.
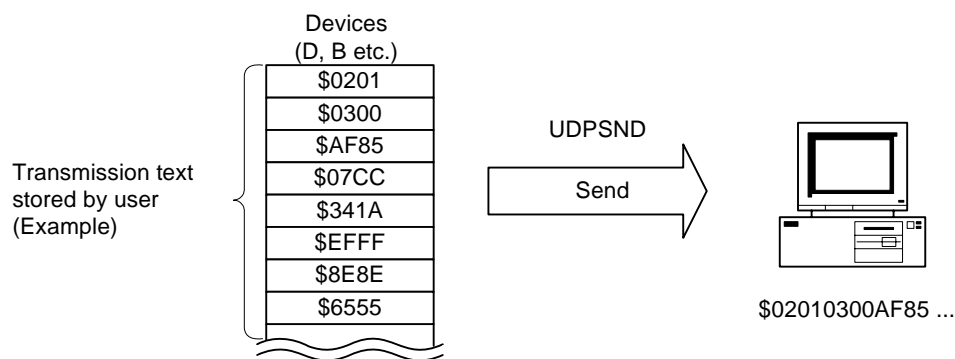
Figure 2.5.2   Send Procedure for UDP/IP Socket

● Instructions Used

Table 2.5.2   Instructions Used in Send procedure for UDP/IP Socket

Instruction Name	Instruction Mnemonic	Description
UDP/IP Send Request	UDPSND	Sends data from a specified UDP/IP socket.

● Procedure

1. Execute a UDP/IP Send Request (UDPSND) instruction, specifying the socket ID, destination, first device for send data, etc.
2. After transmission completes, the result signal is held to ON for one scan period, and the execution status is returned in the specified device.

In step 1 above, specify the socket ID returned in status by the UDP/IP Open (UDPOPEN) instruction.

If an error status is returned, perform retry processing as required.

■ Receive Procedure for UDP/IP Socket

This procedure receives data to a specified device via an open socket by executing a UDP/IP Receive Request (UDPRCV) instruction. The processing differs depending on whether receive data is present in the system control buffer when the request is issued (that is, when the instruction is executed).
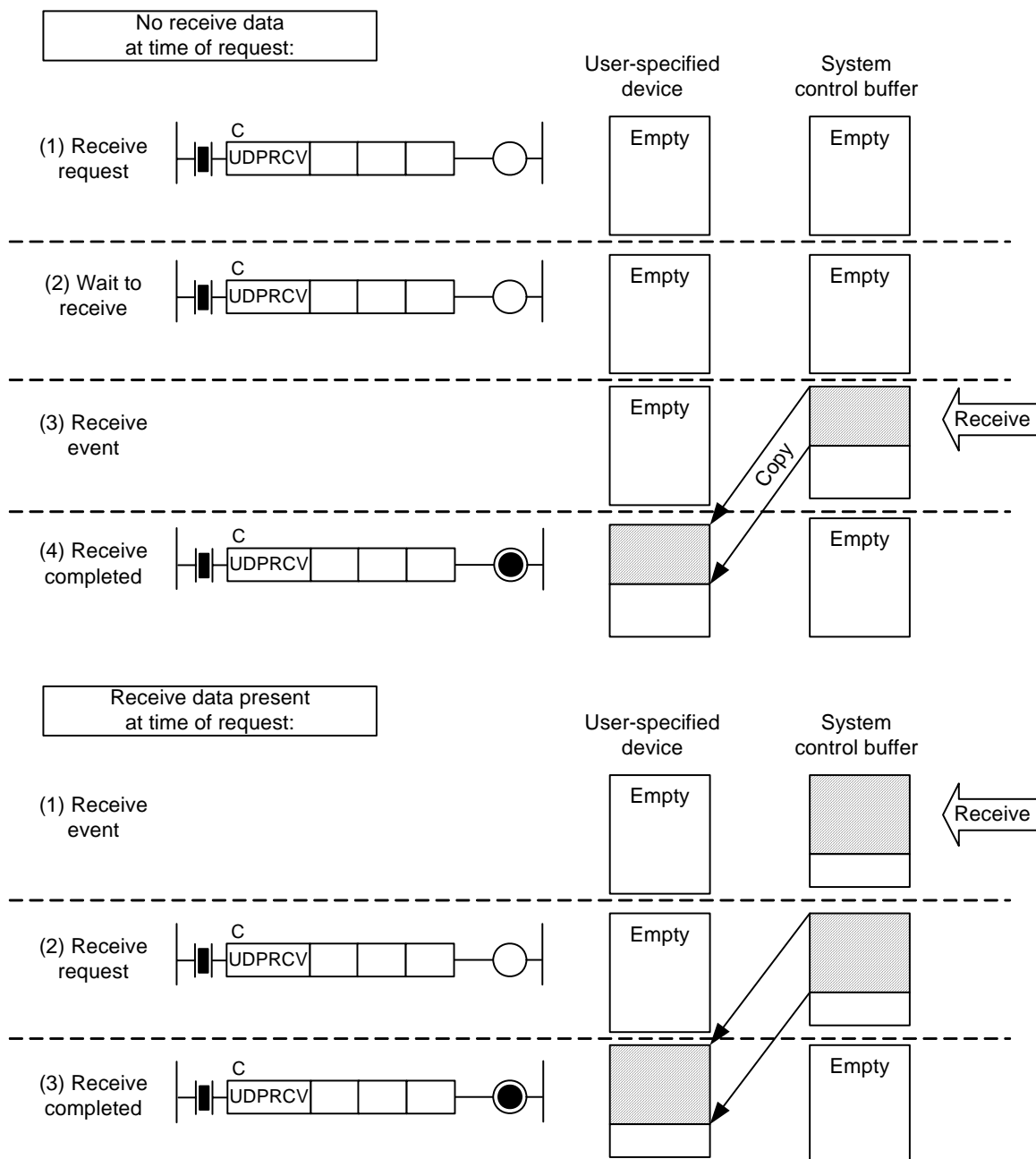
Figure 2.5.3   Receive Procedure for UDP/IP Socket

● Instructions Used

Table 2.5.3   Instructions Used in Receive Procedure for UDP/IP Socket

Instruction Name	Instruction Mnemonic	Description
UDP/IP Receive Request	UDPRCV	Receives data from a specified UDP/IP socket.

● **Procedure**

1. Execute a UDP/IP Receive Request (UDPRCV) instruction, specifying the socket ID, sender, first device for storing received data, size of receive area, etc.
2. After receiving is completed, the UDP/IP Receive Request (UDPRCV) instruction outputs ON to the connection line, and stores the execution status to the specified register. The received data is stored to device according to the specified first device and size.

In step 1 above, specify the socket ID returned as status by the UDP/IP Open (UDPOPEN) instruction.

At the end of receiving, the system control buffer is emptied by discarding all data to prepare for receiving a new packet. You can, however, override this behavior by specifying not to delete the packet using the buffer option in the instruction.

If the size of receive area specified in step 1 is smaller than the packet size, only data of the specified size is stored to device, and the packet in the system buffer (including the part of the packet that is not stored) is then discarded. You should specify the size of receive area such that it is larger than the largest expected data size to be received. You can specify not to remove the packet from the system control buffer using the buffer option in the instruction.

■ Ending UDP/IP Socket Communications

● Instructions Used

Table 2.5.4   Instruction Used for Ending UDP/IP Socket Communications

Instruction Name	Instruction Mnemonic	Description
UDP/IP Close	UDPCLOSE	Closes a specified UDP/IP socket.

● Procedure

1. Execute a UDP/IP Close (UDPCLOSE) instruction to close a UDP/IP socket after use.

When a socket is closed, system resource allocated for the socket ID is released for use by new UDP/IP open instructions.

■ Canceling UDP/IP Socket Communications

● Instructions Used

None

● Procedure

1. Set the input condition of the executing UDP/IP socket instruction to OFF.

You can cancel the execution of a continuous type application instruction including a UDP/IP socket instruction by turning off its input condition. However, when instruction processing actually terminates is indefinite and send or receive processing may actually be completed. Depending on the state of the destination, cancellation may sometimes take a long time to complete.

SEE ALSO

For details on cancellation, see "● Canceling Execution of Continuous Type Application Instructions" in Subsection 2.6.1, "Using Socket Instructions".

2.5.2 TCP/IP Socket Communications Procedure

This subsection describes the procedure for TCP/IP socket communications.

■ Flowchart for TCP/IP Socket Communications

The procedure for TCP/IP socket communications differs for a server and a client.

A server runs in a passive mode, listening for connection requests from clients and establishes a connection when it receives a request.

A client runs in an active mode, actively issuing a connection request, and establishing connection with a ready destination.

The respective flowcharts for the communications procedures under these two scenarios are shown on the following pages.
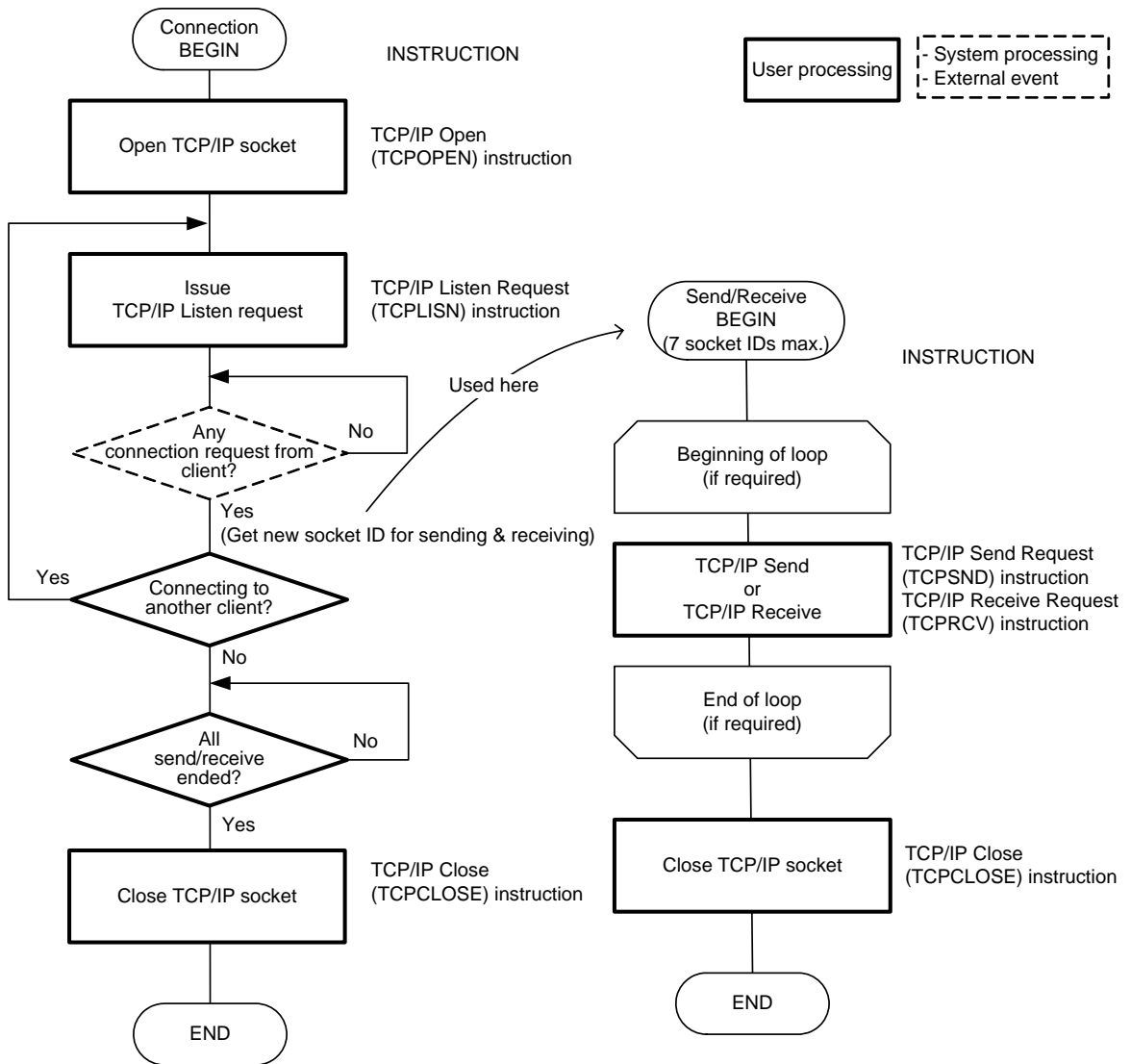
Figure 2.5.4   Flowchart for TCP/IP Socket Communications for Server

F0207.VSD
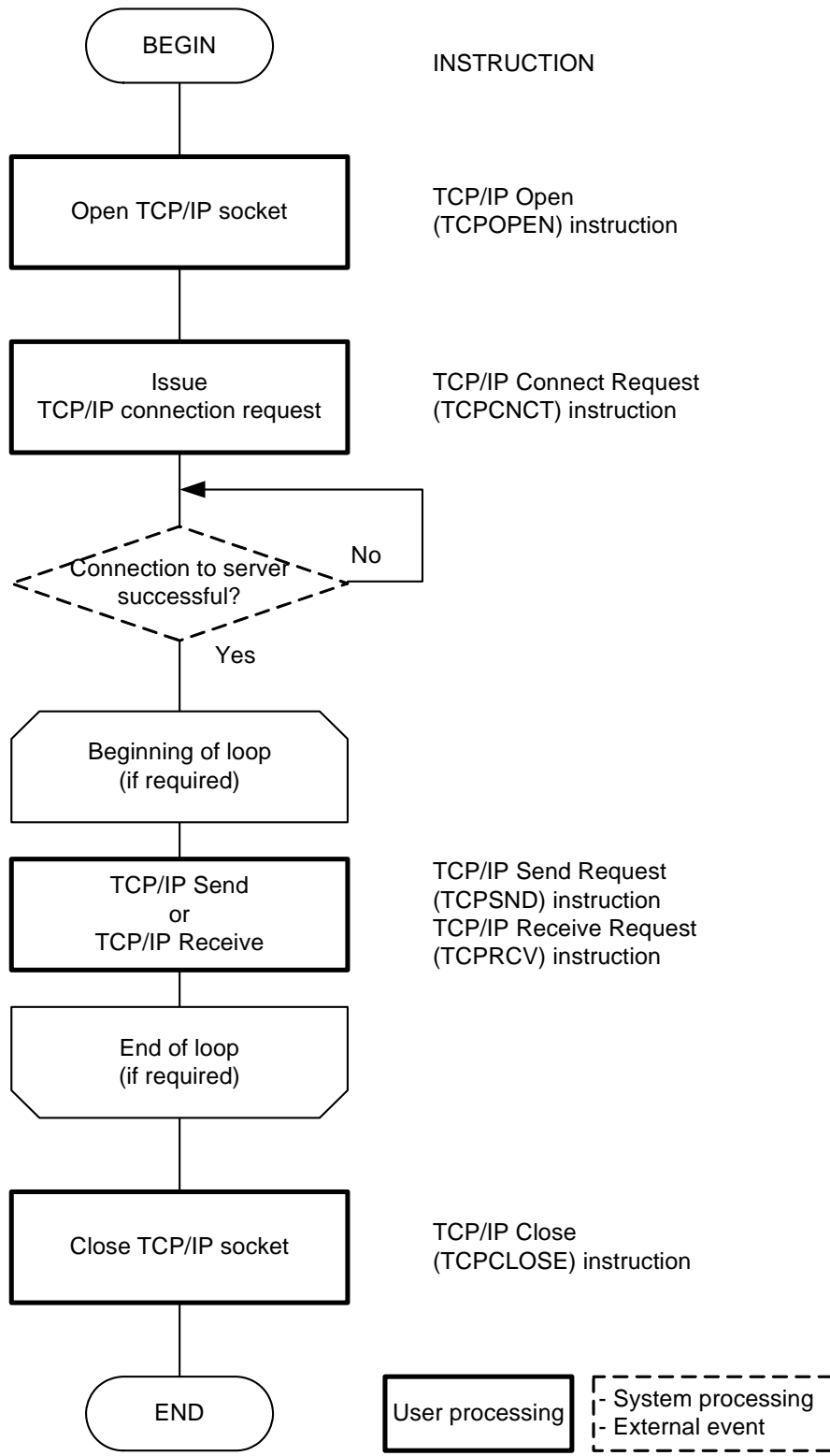
Figure 2.5.5   Flowchart for TCP/IP Socket Communications for Client

■ Preparing for TCP/IP Socket Communications

The preparation for TCP/IP socket communications differs for a server and a client.

● For Server (passive mode)

Instructions Used:

Table 2.5.5   Instructions Used for Preparation of TCP/IP Socket Communications (for server)

Instruction Name	Instruction Mnemonic	Description
TCP/IP Open	TCPOPEN	Opens a TCP/IP socket.
TCP/IP Listen Request	TCPLISN	Waits for a connection request from any client as a server, and establishes connection if a request is received.

Procedure

1. Execute a TCP/IP Open (TCPOPEN) instruction to open a TCP/IP socket. If execution is successful, system resource required for communications is secured and a socket ID is returned as its status.

2. Execute a TCP/IP Listen Request (TCPLISN) instruction to wait for connection requests from clients, specifying the socket ID returned in step 1.

3. When a connection request is received from a client, the TCP/IP Listen Request (TCPLISN) instruction establishes a connection, and if successful, the result signal is held to ON for one scan period and a new socket ID for sending and receiving is returned as its status.

After connection is established, sending and receiving with the client is carried out via the socket ID returned in step 3. The socket ID used in step 2 remains valid, and the same socket (that is, same port number) can be used to listen for another connection request from a different client by re-executing step 2.

Up to 8 TCP/IP sockets can be open concurrently at any one time. Subtracting the one socket used for listening, that means up to 7 concurrent client connections are allowed.

● **For Client (active mode)**

Instructions Used:

Table 2.5.6   Instructions Used for Preparation of TCP/IP Socket Communications (for server)

Instruction Name	Instruction Mnemonic	Description
TCP/IP Open	TCPOPEN	Opens a TCP/IP socket.
TCP/IP Connect Request	TCPCNCT	Issues a connection request to a server as a client, and connects to the server if permitted to do so.

Procedure
1. Execute a TCP/IP Open (TCPOPEN) instruction to open a TCP/IP socket. If execution is successful, system resource required for communications is secured and a socket ID is returned in status.

2. Execute a TCP/IP Connect Request (TCPCNCT) instruction to request for connection to a server, specifying the socket ID opened in step 1, destination, etc.

3. If the server accepts the connection request, a connection is established and ready for communications. At the same time, the TCP/IP Connect Request (TCPCNCT) instruction turns on the result signal to notify execution completion.

After connection is established, sending and receiving with the server is carried out via the socket ID returned in step 1.

■ Send Procedure for TCP/IP Socket Communications

This procedure sends data stored in devices via an open socket by executing a TCP/IP Send Request (TCPSND) instruction.
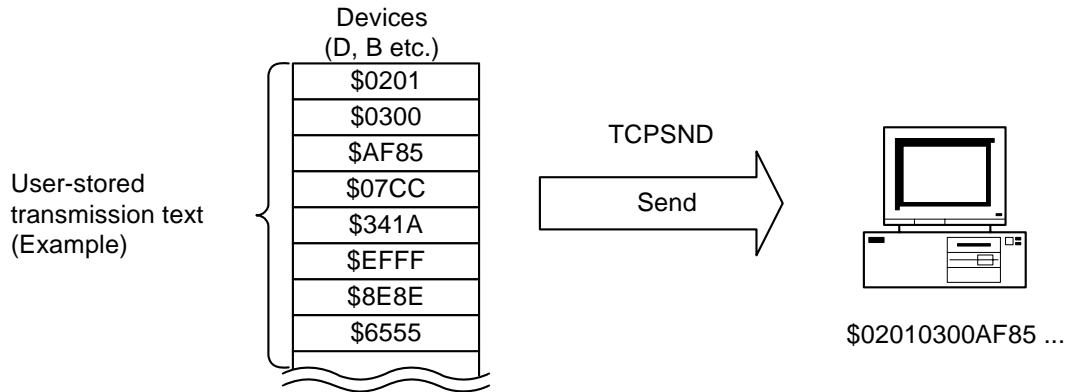
F0209.VSD

Figure 2.5.6   Send Procedure for TCP/IP Socket

● Instructions Used

Table 2.5.7   Instructions Used in Send Procedure for TCP/IP Socket

Instruction Name	Instruction Mnemonic	Description
TCP/IP Send Request	TCPSND	Sends data from a specified TCP/IP socket.

● Procedure

1. Execute a TCP/IP Send Request (TCPSND) instruction, specifying the socket ID, first device for send data, send data size, etc.

2. When transmission is completed, the TCP/IP Send Request (TCPSND) instruction holds result signal to ON to notify execution completion. You can check whether execution is successful by checking the returned status.

For a server machine, specify the socket ID returned as status by the TCP/IP Listen Request (TCPLISN) instruction in step 1. For a client machine, specify the socket ID returned as status by the TCP/IP Open (TCPOPEN) instruction in step 1.

■ Receive Procedure for TCP/IP Socket

This procedure receives data to a specified device via an open socket by executing a TCP/IP Receive Request (TCPRCV) instruction. The processing differs depending on whether receive data is present in the system control buffer when the request is issued (that is, when the instruction is executed).
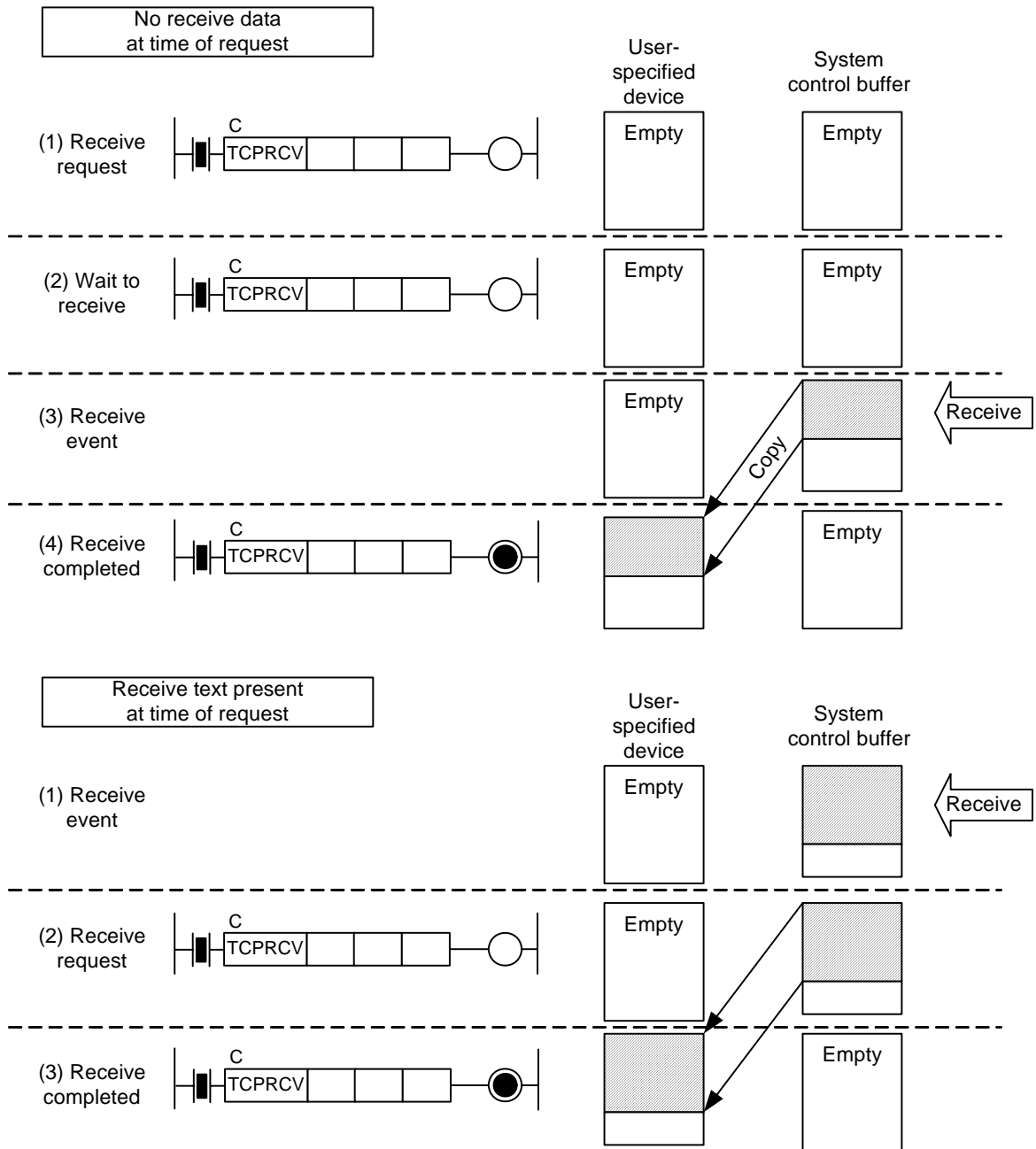
Figure 2.5.7   Receive Procedure for TCP/IP Socket

● Instructions Used

Table 2.5.8   Instructions Used in Receive Procedure for TCP/IP Socket

Instruction Name	Instruction Mnemonic	Description
TCP/IP Receive Request	TCPRCV	Receives data from a specified TCP/IP socket.

● **Procedure**

1. Execute a TCP/IP Receive Request (TCPRCV) instruction, specifying the socket ID, first device for storing received data, size of receive area, etc.

2. If received data is present in the system buffer, proceed to step 3. If there is no received data in the system buffer, wait for receive data to arrive within the timeout interval.

3. When receive processing is completed, the received data is stored to the specified device, and the TCP/IP Receive Request (TCPRCV) instruction turns the result signal to ON for one scan period to notify execution completion. You can check whether execution is successful by checking the returned status.

4. In TCP/IP, depending on route conditions, sometimes the required data size cannot be sent in one transmission so you should repeat steps 1 to 3 until the required data size has been received.

For a server machine, specify the socket ID returned in status by the TCP/IP Listen Request (TCPLISN) instruction in step 1. For a client machine, specify the socket ID returned in status by the TCP/IP Open (TCPOPEN) instruction in step 1.

Whether data received in the system buffer is removed when read depends on the buffer option specified in the TCP/IP Receive Request (TCPRCV) instruction. If normal mode (default) or auto increment mode is specified, received data is removed from the system buffer to create space for new data.

TIP

Specifying auto increment mode in a TCP/IP Receive Request (TCPRCV) instruction automatically increments the address for storing received data after each instruction execution so that the entire data received through multiple instruction executions can be stored contiguously to devices. For details, see the description for the TCP/IP Receive Request (TCPRCV) instruction.

■ Ending TCP/IP Socket Communications

● Instructions Used

Table 2.5.9   Instructions Used for Ending TCP/IP Socket Communications

Instruction Name	Instruction Mnemonic	Description
TCP/IP Close	TCPCLOSE	Closes a specified TCP/IP socket ID.

● Procedure

1. Execute a TCP/IP Socket Close (TCPCLOSE) instruction, specifying a socket ID.

When a socket is closed, system resource allocated for the socket ID is released for use by new TCP/IP open instructions.

■ Canceling TCP/IP Socket Communications

● Instructions Used

None

● Procedure

1. Set the input condition of the executing TCP/IP socket instruction to OFF.

You can cancel the execution of a continuous type application instruction including a TCP/IP socket instruction by turning off its input condition. However, when instruction processing actually terminates is indefinite and may even be after sending or processing has been completed. Depending on the state of the destination, cancellation may take a long time to complete.

SEE ALSO

For details on cancellation, see "● Canceling Execution of Continuous Type Application Instructions" in Subsection 2.6.1, "Using Socket Instructions".

2.5.3　　Precautions about Socket Communications

The precautions about socket communications are described below in separate sections for UDP/IP and TCP/IP.

■ Precautions about UDP/IP Socket Communications

● Size of receive area and data overflow

In UDP/IP, one send execution always corresponds to one receive execution and the entire data is sent in a single block known as a packet. This is known as a datagram model.

If the specified receive area size is smaller than a sent packet, the excess received data is discarded because each packet is expected to be retrieved in one receive execution.

Therefore, you should always specify a receive area size larger than the maximum expected packet size. Programming will be straightforward if you design the user buffer used for receiving to be as large as the maximum packet size.

● Missing packets

The UDP/IP protocol does not include flow control. Furthermore, its internal receive buffer for a socket is designed to keep packets that fit within its size. As a result, if multiple packets are transmitted consecutively but data receiving to device by the ladder program cannot catch up, old packets may be overwritten so that not all transmitted packets are received.

In principle, if missing packets are not allowed, TCP/IP sockets should be used in place of UDP/IP sockets. Alternatively, the application must be programmed to deal with missing packets by, say, inserting a serial number in packets on the sending end and requesting for retransmission if missing numbers are detected on the receiving end.

2-22

■ Precautions about TCP/IP Socket Communications

● Handling fragments when receiving

In the IP protocol, large size data is segmented before transmission to accommodate channel restrictions. This process is known as fragmentation and the data segments are known as fragments.

Fragmentation in TCP/IP communications is not transparent to a user application, which is responsible for handling fragmented data. As each receive execution receives only a fragment of the transmitted data, the number of send executions and the number of receive executions may not tally. When seen from a user application, the data received for a receive request may only be a part of the entire data transmitted from the source. To ensure that all transmitted data is received, a user application must check the received size and issue multiple receive requests as required.

● Recognizing end of received data

In TCP/IP, a user application is responsible for recognizing end of received data. This is because TCP/IP handles sent data as contiguous data regardless of the original send request units. This kind of data processing is known as a byte stream. Thus, it is the responsibility of the user application to recognize the end of received data in the receive buffer for individual receive requests.

● Flow control when receiving

In TCP/IP, when the receive buffer is full, its flow control puts the sending end in wait state. Therefore, when receiving data exceeding the receive buffer size, a user application cannot receive all data using one receive request. The user application should receive the arrived data, clear the buffer, and repeat until all data has been received.

2.6   Socket Instructions

This section describes how to use socket instructions, followed by their specifications.

2.6.1   Using Socket Instructions

■ Continuous Type Application Instructions

Execution of many socket instructions cannot be completed within one scan period. To avoid affecting control processing, a processing request is issued at the time of instruction execution but the time-consuming actual processing is carried out in the background. Such instructions are known as "continuous type application instructions."
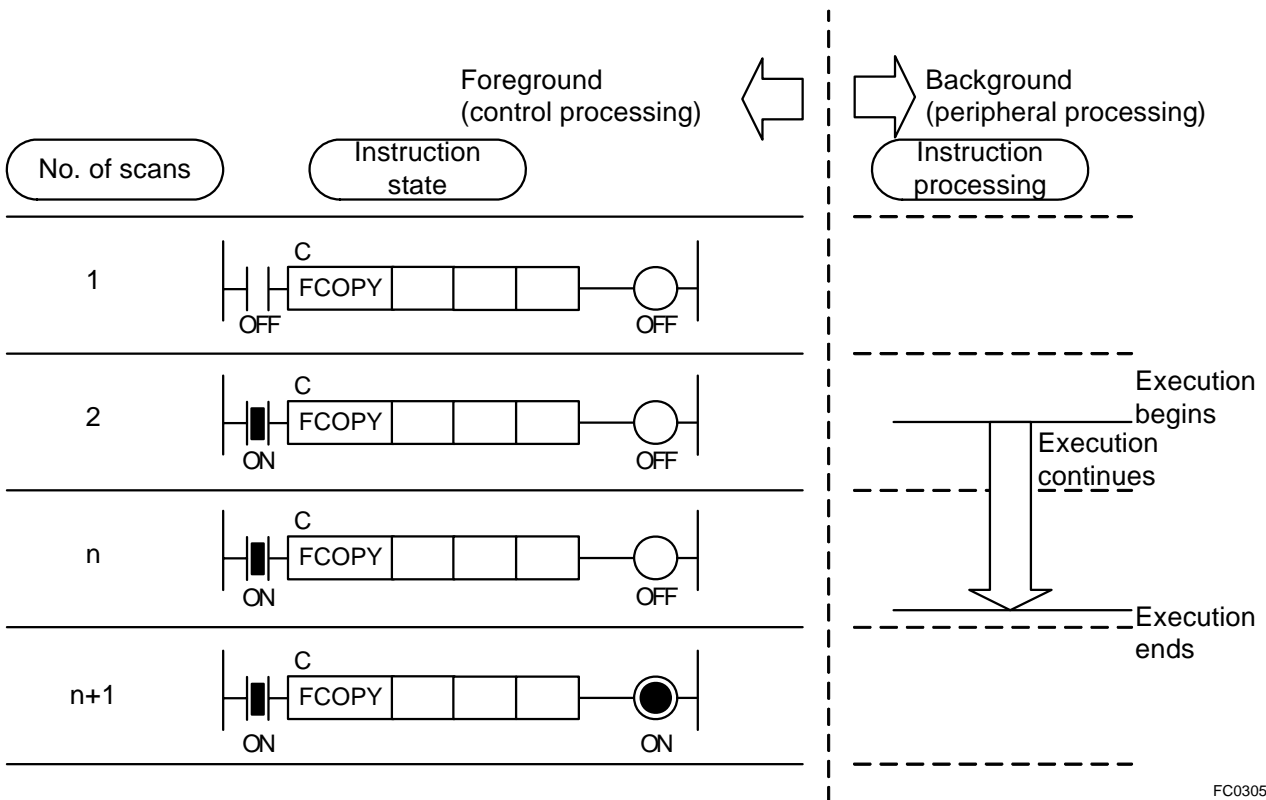
Figure 2.6.1   Concept of Continuous Type Application Instruction

● **Operation of Continuous Type Application Instructions**

This subsection describes the operation of a continuous type application instruction. In the description, the term "input condition" refers to the ON/OFF state of the circuit connection line immediately preceding the continuous type application instruction.

- To execute the instruction:

  Change its input condition from OFF to ON.
- To continue instruction execution:

  Hold its input condition in ON state.
- When instruction execution completes:

  The result signal (on the circuit line connected to the output (right) end of the instruction) is held to ON for one scan period. A user program can check the completion of a continuous type application instruction by monitoring an OUT instruction or some other output-type instruction placed on the output end of the instruction.
- To re-execute the instruction after it has completed execution:

  Turn off and again turn on its input condition. The condition must be held in OFF state for at least 1 scan period.
- To cancel (abort) instruction execution:

  Turn off its input condition during instruction execution. The result signal is held to ON for one scan period. However, the background instruction processing does not end immediately. For more details, see "● Canceling Execution of Continuous Type Application Instructions" later in this subsection.

Table 2.6.1   Operation of Continuous Type Application Instructions

Instruction State of Preceding Scan	Input Condition of Preceding Scan	Input Condition of Current Scan	Transition of Instruction State in Current Scan	Result Signal of Current Scan
Stopped	OFF	ON	Execute	OFF
		OFF	Stopped	OFF
Execute	ON	ON	Continue Execution	OFF
			Execution Completed[*1]	ON for 1 scan
		OFF	Cancelled	ON for 1 scan
Execution Completed[*1]	ON	ON	Execution Completed	OFF
		OFF	Stopped	OFF
Cancelled	OFF	ON	Start execution	OFF
		OFF	Stopped	OFF

*1: The transition to 'Execution Completed' state is independent of the input condition, and is triggered by completion of background instruction processing.

● **Operation Result of Continuous Type Application Instructions**

Continuous type application instructions output two types of operation result at the end of instruction execution. A user program determines the completion of instruction execution using the result signal, and checks whether execution is successful using the status.

Table 2.6.2   Operation Result of Continuous Type Application Instructions

Operation Result	Description
Result signal	At the end of instruction execution, the result signal is held to ON for one scan. The result signal is OFF at other times. A user program determines whether instruction execution has completed by checking the ON/OFF state of the result signal.
Status	Regardless of whether instruction execution is successful, a status value is stored in a user-specified device. Some devices may store other return values in addition to the status so the status has a multi-word table structure. If an error status is returned, a user program should perform application error processing such as retry processing.
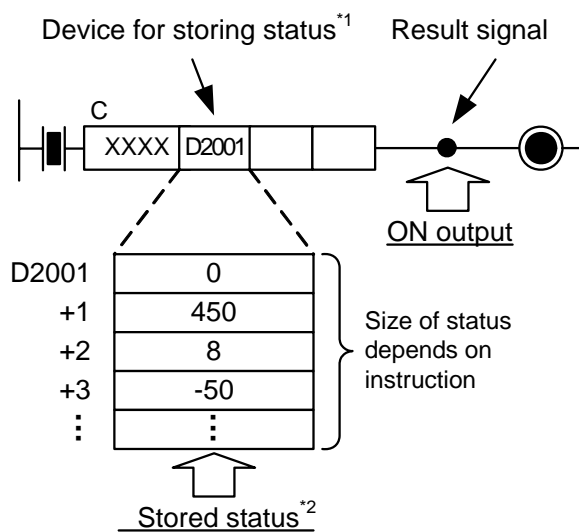
*1: D2001 is used as an example for illustration purpose.
*2: This is an example of stored status values.

FC0306.VSD

Figure 2.6.2   Operation Result Output of Continuous Type Application Instructions

● **Error Processing of Continuous Type Application Instructions**

If instruction execution ends normally, a zero or positive integer is stored in status. If execution ends in error, a negative integer is stored in status.

A user program should read the execution result status and perform whatever error processing (e.g. retry) as appropriate if an error status is returned.

Even if an error status is returned, the module does not store an error code in a special register, write to the system log (error log), turn on the ALM LED or ERR LED, or switch the operating mode.
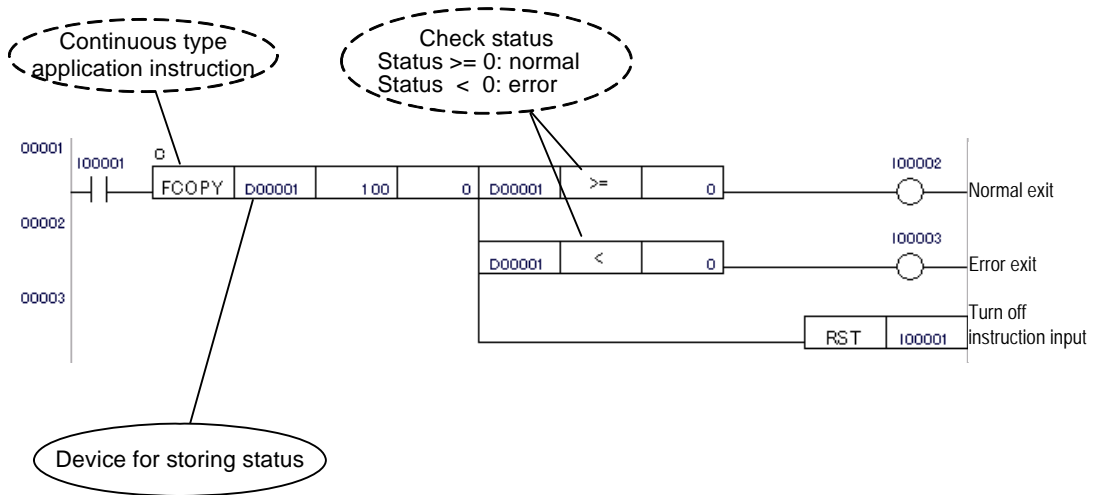
FC0307.VSD

Figure 2.6.3   Error Processing of Continuous Type Application Instructions

● **Error Status of Continuous Type Application Instructions**

The table below shows the error status codes of continuous type application instructions.

Table 2.6.3   Continuous Type Application Instruction Status (timeout-related (-1xxx), non-error-related (-2xxx), exclusive control related (-3xxx))

Category	Continuous Type Application Instruction Status		
	Value	Name	Description
Timeout	-1000	Instruction Timeout	Processing failed to end within the timeout interval specified by an instruction parameter.
	-1001	Internal Communication Timeout	No response was received within the internal communication timeout interval. The following timeout interval can be defined by a user as a CPU property. - FTP Client Network Timeout
Non-error	-2000	End of File Detected	End of file was detected during processing.
	-2001	No Match Found	No match was found.
	-2002	Disconnected by Remote Node	Connection was terminated by the remote node. Check the status of the remote node. This status is also returned if high network load causes data loss.
	-2003	Specified Size/Times Processed	Processing has been completed for the specified data size or iterations. - The size of data received by a TCP/IP Receive Instruction (TCPRCV instruction) reaches the specified receive area size.
	-2004	Block Size Error	Data size is smaller than the specified block size.
Exclusive control	-3001	Repeated Use of Function	A function or resource that disallows repeated use was used repeatedly. - Repeated execution of FTP client instruction - Repeated execution of file operation instruction or disk operation instruction - Repeated use of file ID or socket ID
	-3003	Write-prohibit Destination	A write attempt to a destination was unsuccessful because: - the destination was being accessed - the destination is a directory - the destination is read-only
	-3004	Repeated Write Mode	An attempt was made to open a file, which is already open in Write (Append) mode.
	-3005	Internal Resource Depleted	Internal resource is temporarily depleted. To resolve the problem, retry later. If the problem persists, consider reducing processing load. - FA-M3 internal resource - Protocol stack internal resource

Table 2.6.4   Continuous Type Application Instruction Status (network-related (-5xxx))

Category	Continuous Type Application Instruction Status		
	Value	Name	Description
Network	-5000	Connection Error	Error was detected during connection.
	-5001	Unknown Destination	The destination was not found.
	-5002	Buffer Overflow	Send/receive buffer used by socket instructions has overflowed.
	-5030	FTP User Authentication Failure	Access was denied by FTP server's user authentication process.
	-5031	FTP Password Authentication Failure	Access was denied by FTP server's password authentication process.
	-5032	FTP Command Sequence Error	FTP client processing could not continue because a reply received from the FTP server was out of sequence. This error may be due to repeated cancel operations or bad line quality.
	-5421	FTP Negative Reply 421	FTP server returns a negative reply. The last three digits of this error code (positive value) represent the reply code received from the FTP server.[1]
	-5425	FTP Negative Reply 425	
	-5426	FTP Negative Reply 426	
	-5450	FTP Negative Reply 450	
	-5451	FTP Negative Reply 451	
	-5452	FTP Negative Reply 452	
	-5500	FTP Negative Reply 500	
	-5501	FTP Negative Reply 501	
	-5502	FTP Negative Reply 502	
	-5503	FTP Negative Reply 503	
	-5504	FTP Negative Reply 504	
	-5530	FTP Negative Reply 530	
	-5532	FTP Negative Reply 532	
	-5550	FTP Negative Reply 550	
	-5551	FTP Negative Reply 551	
	-5552	FTP Negative Reply 552	
	-5553	FTP Negative Reply 553	

*1: For details on the meaning of each reply code, see the official FTP specification (RFC959). Note that the causes and meanings of reply codes may vary with individual FTP server implementations.

Table 2.6.5   Continuous Type Application Instruction Status (file system related (-6xxx))

Category	Continuous Type Application Instruction Status		
	Value	Name	Description
File system	-6000	Duplicate Filename	Specified destination filename already exists
	-6002	Insufficient Space	There is insufficient space on the storage media. Or, number of files or directories exceeded maximum limit.
	-6004	Memory Card Not Installed	Processing is not allowed because no memory card is installed.
	-6005	Memory Card Not Mounted	Processing is not allowed because no memory card is mounted.
	-6006	Protection Switch is ON	Processing is not allowed because the card protection switch is enabled.
	-6007	File System Failure	Processing could not continue because a file system failure was detected or the file system is not in FAT16 format. Reformat the disk in FAT16 format, or replace the memory card. This status may be returned occasionally when there is insufficient space on the storage media.
	-6008	Memory Card Failure	Processing could not continue because a memory card failure was detected. Replace the memory card.
	-6009	Unknown Write Error	An error of unknown cause was detected during write processing. Reformat disk to FAT16 format, or replace the memory card.
	-6010	FLS Processing Sequence Error	Executions of FLSFIRST, FLS and FLSFIN instructions were out of sequence.
	-6011	File Interpretation Error	The NULL byte was detected during interpretation of a text file.

Table 2.6.6   Continuous Type Application Instruction Status (General Instruction (-9xxx))

Category	Continuous Type Application Instruction Status		
	Value	Name	Description
General Instructions	-9000	Cancel Request Issued	A cancel request was issued. Check the resource relay to determine when cancellation is completed.
	-9010	Resource Not Opened	The specified file ID or socket ID is not open. Execute an Open instruction for the ID.
	-9011	Resource Depleted	- No more unused socket ID or file ID is available. Check the resource relay. - An attempt was made to run multiple FTP clients. Concurrent execution of FTP clients is not allowed.
	-9012	Resource Released by External Factor	Processing could not continue because a user has caused the resource relay to be turned off so writing to the resource relay is prohibited. This error may occur if the SD memory card is unmounted when a file is open.
	-9013	Function Not Started	- A function required for processing is not running. - FTP client is not running. Execute an FTPOPEN instruction.
	-9014	Invalid Device Access	An attempt was made to access an invalid device number. Check index modification, indirect designation, data size and status size.
	-9015	Data Processing Error	The requested processing could not continue because of invalid data.
	-9020	Security Error	The specified password or keyword is incorrect.
	-9021	CPU Property ROM Write Error	An attempt to write CPU property data to the internal ROM failed.
	-9999	Internal Error	Internal error was detected.

Table 2.6.7   Continuous Type Application Instruction Status (parameter error related (-1xxxx))

Category	Continuous Type Application Instruction Status		
	Value	Name	Description
Parameter Error	-10xxx	Parameter Error	The specified parameter is invalid. The last 3 digits of the error code indicate the position of the invalid instruction parameter and its offset from the beginning of the table in words if the parameter is a table. Status: -10 △□□ △: Parameter number (1 to 3) □□: Offset in table (00 to 99)

	-12xxx	Invalid Pathname	The specified pathname is invalid. This error is generated if path interpretation failed because the specified file pathname violated a syntax rule.
			The third digit of the error code indicates the location of the invalid parameter.
			
			Status: -12 △□□
			△: 1 to 3 : Text parameter number
			        4 : CPU property
			        9 : Unknown type
			□□: System reserved (currently 00)

	-13xxx	Pathname Object Not Found	The object designated by the pathname is not found. This error is generated if the specified pathname contains an invalid file or directory. For instance, "\RAMDISK\MYDIR" is specified but there is no directory named "MYDIR" on the RAM disk.
			This error may also be generated if a wildcard is specified but no match is found.
			The third digit of the error code indicates the location of the invalid parameter.
			
			Status: -13 △□□
			△: 1 to 3 : Text parameter number
			        4 : CPU property
			        9 : Unknown type
			□□: System reserved (currently 00)

	-15xxx	Invalid String Length	The string length parameter is invalid.
			This error is generated if the string length exceeds the maximum limit, or if NULL is specified for a parameter that does not allow a NULL value.
			The third digit of the error code indicates the location of the invalid parameter.
			
			Status: -15 △□□
			△: 1 to 3 : Text parameter number
			        4 : CPU property
			        9 : Unknown type
			□□: System reserved (currently 00)

● Canceling Execution of Continuous Type Application Instructions

Execution of a continuous type application instruction can be cancelled by turning off its input condition during execution. When a falling edge is detected in the input condition, the result signal is immediately held to ON to notify termination of execution, and a Cancel Request Issued status code (-9000) is stored in the instruction status.

However, note that despite notification of instruction termination, background instruction processing is not yet terminated. Instead, a cancellation request is issued to background processing, and a few seconds may be required to complete the termination.

If the same continuous type application instruction is executed before background processing cancellation is completed, resource competition occurs and an exclusive control related error will be generated. To avoid this, you should include the resource relay in the input condition of a continuous type application instruction.

TIP

Just as with instruction cancellation, in the event of an instruction timeout (error code: -1000), background instruction processing continues to run for a short while. Therefore, it is also necessary in this case to incorporate exclusive control in the program using resource relays.

SEE ALSO

For details on resource relays, see "■ Resource Relays" later in this subsection.

⚠ **CAUTION**

When the input of a continuous type application instruction is turned off, the instruction immediately returns a Cancel Request Issued status and terminates execution. However, actual background processing such as background communications is not terminated immediately. To check for termination of actual processing, check that the associated resource relay is turned off.

● **Precautions When Executing Continuous Type Application Instructions**

By nature, continuous type application instructions require multiple scans to complete processing, and thus should not be executed only once but must be executed repeatedly until execution completes.

The table below shows the precautions when executing continuous type application instructions from different program types.

Table 2.6.8 Precautions when Executing Continuous Type Application Instructions

Program Type	Precaution
Ladder block (execute-all-blocks mode)	None
Ladder block (execute-specified-blocks mode)	Executing an Inactivate Block (INACT) instruction during execution of a continuous type application instruction forces cancellation of instruction processing.
Sensor control block	Continue execution of a sensor control block until the execution of a continuous type application instruction ends. If you stop a sensor control block before instruction execution ends, instruction processing cannot be completed.
I/O interrupt routine	Use of continuous type application instructions in I/O interrupt routines is not allowed.
Subroutine	Repeat a subroutine call until the execution of a continuous type application instruction ends. If you stop subroutine call before instruction execution ends, instruction processing cannot be completed.
Macro and input macro	Repeat a macro call until the execution of a continuous type application instruction ends. If you stop macro call before execution of continuous type application instruction ends, instruction processing cannot be completed. Calling a macro containing an executing continuous type application instruction from a different location in the program is not allowed.

⚠ **CAUTION**

- A continuous type application instruction will not execute correctly if it is executed in only one scan.

- Do not execute the same continuous type application instruction more than once within the same scan using macros. Repeat execution using FOR-NEXT instruction or JMP instruction is also disallowed.

● **Restrictions for Inserting Continuous Type Application Instructions**

There are some restrictions for inserting continuous type application instructions in a ladder diagram. Placing a continuous type application instruction in an invalid location generates a program syntax error in WideField2.

The figure below illustrates some locations where continuous type application instructions cannot be inserted.
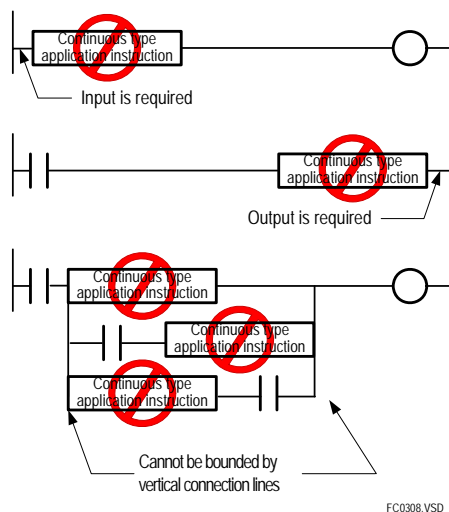
Figure 2.6.4 Restrictions for Inserting Continuous Type Application Instructions

● **Online Edit of Continuous Type Application Instructions**

Do not online edit a circuit containing an executing continuous type application instruction. If an executing continuous type application instruction is edited online, instruction processing will be forcedly terminated and re-executed using the modified parameters (including text parameters). In this case, the result signal of the continuous type application instruction will not be held to ON for 1 scan to indicate end of instruction processing.

Even if parameter values are not modified during online edit, a Repeated Use of Function error (status code -3001) may still be generated during re-execution depending on the status of the resource.

⚠ **CAUTION**

Before performing online edit of a circuit containing continuous type application instructions, check to ensure that no continuous type application instruction is running.

■ Resource Relays

Resource relays are special relays for preventing competition between continuous type application instructions. A resource relay indicates the status of a resource, which is subject to exclusive control. Resources include file IDs socket IDs, functions and instructions.

By inserting a resource relay in the input condition of a continuous type application instruction, you can prevent errors due to resource competition. In particular, resource relays are required for checking for completion of cancellation processing or instruction timeout processing in user applications where cancellation request for a continuous type application instruction, or timeout (-1000) may occur.

● **Resource Relays (related to socket instructions)**

Table 2.6.9   Resource Relays (related to socket instructions)

Category	Continuous Type Application Instruction Resource Relays		
No.	Name	Function	Description
M1028	No Unused UDP Socket	No unused UDP socket is available.	Turns on when all UDP/IP sockets are in use.
M1029	No Unused TCP Socket	No unused TCP socket is available.	Turns on when all TCP/IP sockets are in use.
M1105 to M1120	Socket Open	Socket is open.	Each socket ID is associated with one special relay. The relay for a socket ID turns on while the socket ID is open. When the relay for a socket ID is OFF, the socket ID cannot be used. This is a read-only relay. Do not write to it.
M1121 to M1136	Socket Busy	Socket is busy.	Each socket ID is associated with one special relay. The relay for a socket ID turns on during execution of any socket instruction using the socket ID. When the relay for a socket ID is ON, no other socket communication instruction using the same socket ID can be executed except for concurrent execution of sending and receiving. This is a read-only relay. Do not write to it.
M1073 to M1088	Socket Sending	Socket is performing send processing.	Each socket ID is associated with one special relay. The relay for a socket ID turns on during send processing of the socket. When the relay for a socket ID is ON, no send request is allowed for the same socket ID. This is a read-only relay. Do not write to it.
M1089 to M1104	Socket Receiving	Socket is performing receive processing.	Each socket ID is associated with one special relay. The relay for a socket ID turns on during receive processing of the socket. When the relay for a socket ID is ON, no receive request is allowed for the same socket ID. This is a read-only relay. Do not write to it.

■ Text Parameter

Some file system instructions use text parameters as instruction parameters. A text parameter value can be stored using the Text Parameter (TPARA) instruction. The Text Parameter (TPARA) instruction must be executed before an instruction that requires text parameters.

● Text Parameter (TPARA)

This instruction is used to specify a text parameter required by some continuous type application instructions.

Table 2.6.10   Text Parameter

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required? Yes	Input Condition Required? No	Step Count	Processing Unit	Carry
Application Instruction	–	Text Parameter	TPARA	⊣ TPARA ⊢	✓	–	5	8 bit	–

■ Parameter

Text Parameter ⊣ TPARA | n | s1 | s2 | s3 ⊢

n   :   Text parameter number (W) (1-4)

s1 :   Device storing character string 1 (W)

(Up to 255 characters, terminated by a NULL character)

s2 :   Device storing character string 2 (W)

(Up to 255 characters, terminated by a NULL character)

s3 :   Device storing character string 3 (W)

(Up to 255 characters, terminated by a NULL character)

■ Available Devices

Table 2.6.11   Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	W	Z	R	V	Constant	Index Modification	Indirect Designation, Pointer P
n									✓	✓	✓		✓	✓	✓	Yes	Yes
s1									✓	✓	✓		✓	✓	✓	Yes	Yes
s2									✓	✓	✓		✓	✓	✓	Yes	Yes
s3									✓	✓	✓		✓	✓	✓	Yes	Yes

Note: See Section 1.15, "Restrictions on Devices Used as Instruction Parameters" of "Sequence CPU – Instructions" (IM34M6P12-03E)

■ Function

This instruction is used to specify text parameters required by some continuous type application instructions. You should specify the text parameter number according to the text parameter number of the instruction requiring the text parameter.
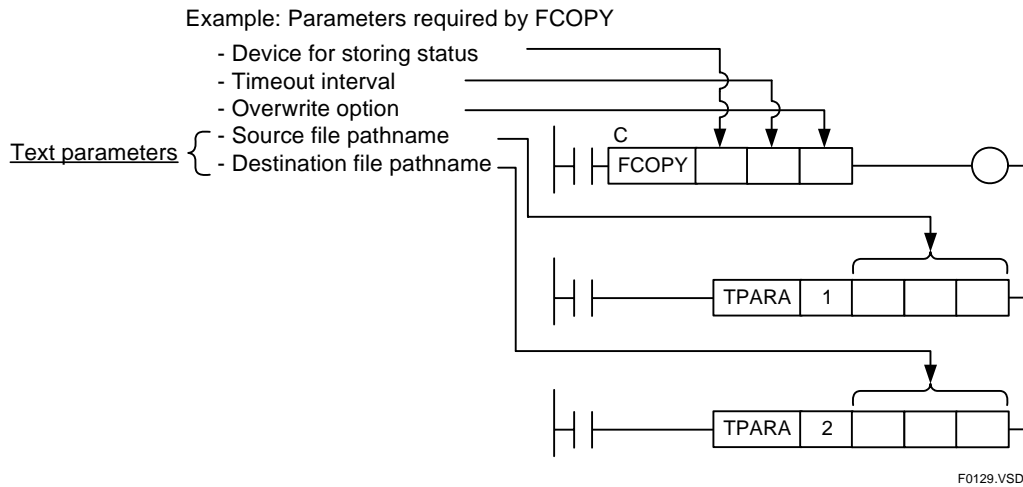
Figure 2.6.5   Text Parameter Number (TPARA instruction must be executed before continuous type application instruction)

The Text Parameter instruction must be executed to set up a text parameter before an instruction requiring the text parameter. Text parameters are stored in the system text parameter area. An instruction requiring a text parameter reads the text parameter from the text parameter area when it begins execution (at the rising edge of the input).

One text parameter area is provided for all continuous type application instructions executing in the normal scan, and another area is provided for all continuous type application instructions executing in the sensor control block. Therefore, normal blocks and the sensor control block do not compete for the text parameter area but continuous type application instructions sharing each area do compete. You should store text parameter value before each instruction execution to ensure proper execution.
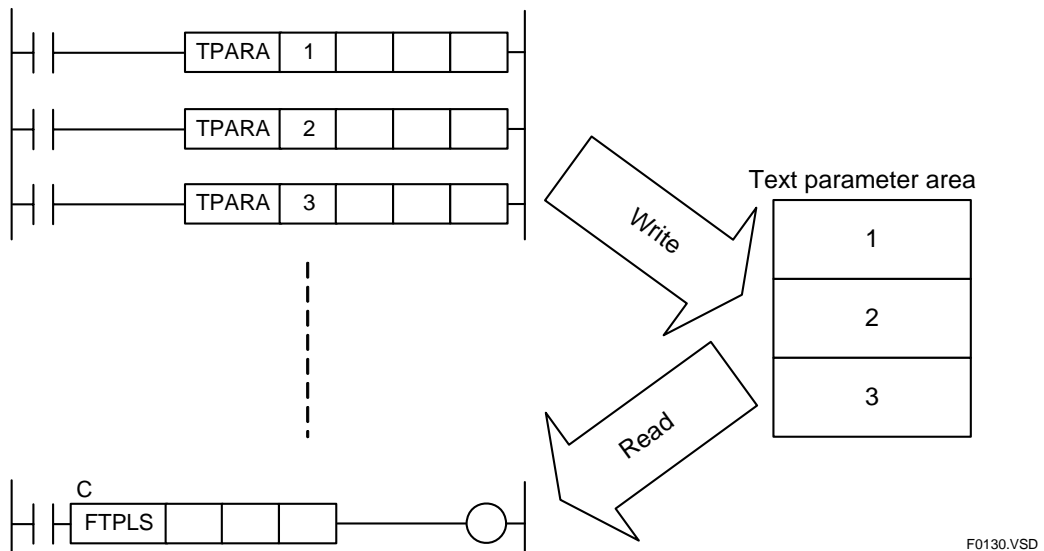
Figure 2.6.6   Text Parameter Area

This instruction performs character string concatenation. It concatenates strings A to C (see next figure) into one text parameter n (n=1 to 3). This string concatenation feature enables user programs to define smaller string units and increase reuse of defined string constants.

Specify NULL for all non-required strings A to C. Using a zero constant value in an instruction parameter is equivalent to specifying a NULL value.
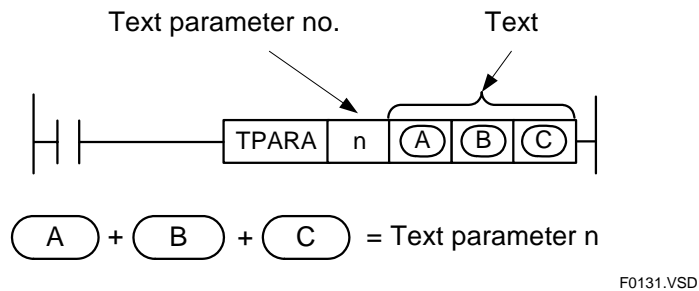
Figure 2.6.7   One Text Parameter

Each text parameter can contain up to 255 characters. The individual lengths and combined length of strings A to C must not exceed 255 characters.

TIP

Using string concatenation, you can specify as text parameter various string combinations such as string A, string B, string C, string (A+C), string (A+B), etc. Note that all unused parameters must be specified as NULL.

■ Programming Example
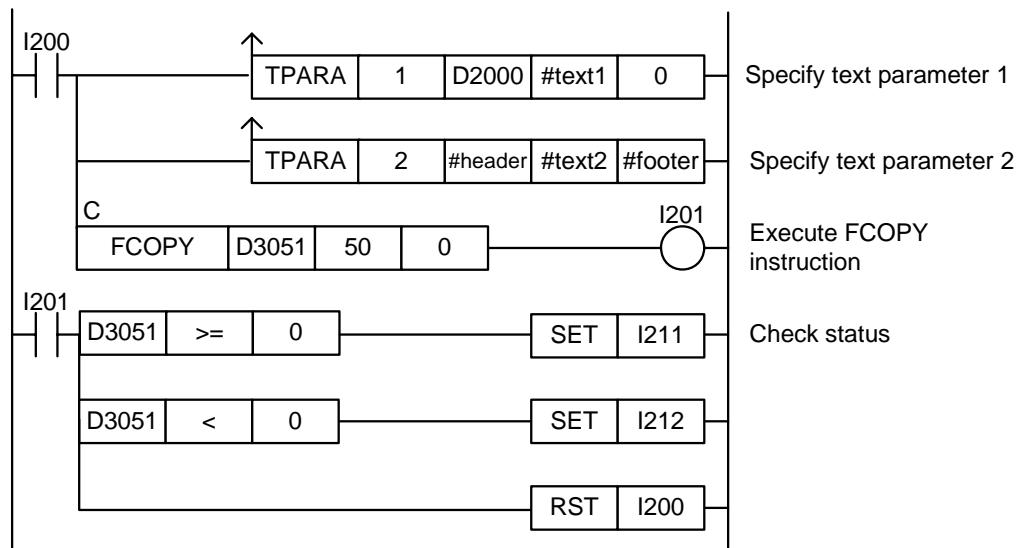
Figure 2.6.8   Example of a Text Parameter Program

This sample code sets up text parameter 1 and text parameter 2, which are to be passed to a Copy File instruction (FCOPY).

The string stored in devices starting with D2000 and the string defined by constant name #text1 are concatenated to become text parameter 1. The three strings defined by constant names #header, #text2 and #footer are concatenated to become text parameter 2.

2.6.2 List of Socket Instructions

Table 2.6.12   List of Socket Instructions

Instruction Group	Service Name	Ladder Instruction	Description
UDP/IP communications preparation instructions	UDP/IP Open	UDPOPEN	Opens a UDP/IP socket to enable communications.
	UDP/IP Close	UDPCLOSE	Closes a UDP/IP socket.
UDP/IP send and receive instructions	UDP/IP Send Request	UDPSND	Sends data from a specified UDP/IP socket.
	UDP/IP Receive Request	UDPRCV	Receives data from a specified UDP/IP socket.
TCP/IP communications preparation instructions	TCP/IP Open	TCPOPEN	Opens a TCP/IP socket.
	TCP/IP Close	TCPCLOSE	Closes a TCP/IP socket.
	TCP/IP Connection Request	TCPCNCT	Issues a connection request to a server as a client, and connects to the server if permitted to do so.
	TCP/IP Listen Request	TCPLISN	Waits for a connection request from any client as a server, and establishes connection if a request is received.
TCP/IP send and receive instructions	TCP/IP Send Request	TCPSND	Sends device data from a specified TCP/IP socket.
	TCP/IP Receive Request	TCPRCV	Writes data received from a specified TCP/IP socket to device.

2.6.3 Socket Instruction Specifications

This subsection describes the specification of each socket instruction.

2.6.3.1 UDP/IP Communications Preparation Instructions

Table 2.6.13   List of UDP/IP Communications Preparation Instructions

Service Name	Ladder Instruction	Description
UDP/IP Open	UDPOPEN	Opens a UDP/IP socket to enable communications.
UDP/IP Close	UDPCLOSE	Closes a UDP/IP socket.

■ UDP/IP Open (UDPOPEN)

Opens a UDP/IP socket to enable communications.

Table 2.6.14   UDP/IP Open

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required? Yes	No	Step Count	Processing Unit	Carry
Continuous type application instruction	–	UDP/IP Open	UDPOPEN	C ─ UDPOPEN	✓	–	6	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see "■ Continuous Type Application Instructions" in Subsection 2.6.1, "Using Socket Instructions".

Parameter

UDP/IP Open ─ C | UDPOPEN | ret | n1 | n2 ─
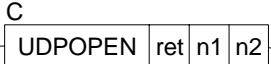
Table 2.6.15   Parameters

Parameter	Description
ret[1]	Device for storing return status (W)
n1	Timeout interval (W) [0 = infinite, 1-32767 (x100 ms)]
n2	My port number (W) [1-65535][2][3]

[1]: ret (status) is table data. For details on the return status (ret), see "Status (Return Value)".
[2]: Do not specify my port number as 12289, 12290, 12291, 12305 or 12307 as these numbers are used by the higher-level link service and remote programming service.
[3]: Word data is handled as an unsigned decimal or hexadecimal number.

Status (Return Value)

Table 2.6.16   Status (Return Value)

Offset (word)		Description
ret	ret+0	0, > 0 SOCKET ID (W) [0-7] (socket is opened successfully)
		< 0 Error status

SEE ALSO

For more details on error status, see "● Error Status of Continuous Type Application Instructions" of "■ Continuous Type Application Instructions" in Subsection 2.6.1, "Using Socket Instructions".

■ UDP/IP Close (UDPCLOSE)

Closes a UDP/IP socket. Once a socket is closed, no more sending or receiving is allowed via the socket unless and until the socket is reopened.

Table 2.6.19   UDP/IP Close

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Pro-cessing Unit	Carry
					Yes	No			
Continuous type application instruction	–	UDP/IP Close	UDPCLOSE	C ─UDPCLOSE ─	✓	–	6	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see "■ Continuous Type Application Instructions" in Subsection 2.6.1, "Using Socket Instructions".

Parameter

UDP/IP Close ─ C UDPCLOSE | ret | n1 | n2 ─

Table 2.6.20   Parameters

Parameter	Description
ret*1	Device for storing return status (W)
n1	Timeout interval (W) [0 = infinite, 1-32767 (x100 ms)]
n2	Socket ID (W) [0-7]

*1: ret (status) is table data. For details on the return status (ret), see "Status (Return Value)".

Status (Return Value)

Table 2.6.21   Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit
		< 0	Error status

SEE ALSO

For more details on error status, see "● Error Status of Continuous Type Application Instructions" of "■ Continuous Type Application Instructions" in Subsection 2.6.1, "Using Socket Instructions".

Available Devices

Table 2.6.22   Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	W	Z	R	V	Constant	Index Modification	Indirect Designation, Pointer P
ret									✓	✓	✓		✓	✓		Yes	Yes
n1									✓	✓	✓		✓	✓	✓	Yes	Yes
n2									✓	✓	✓		✓	✓	✓	Yes	Yes

Note: See Section 1.15, "Restrictions on Devices Used as Instruction Parameters" of "Sequence CPU – Instructions" (IM34M6P12-03E)

Resource Relays

Table 2.6.23   Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
	M1028	No Unused UDP Socket	Execute UDPCLOSE instruction for a socket ID only if its corresponding Socket Busy, Socket Sending and Socket Receiving relays are all off.
	M1029	No Unused TCP Socket	
	M1105 to M1120	Socket Open	
✓	M1121 to M1136	Socket Busy	
✓	M1073 to M1088	Socket Sending	
✓	M1089 to M1104	Socket Receiving	

Function

Closes a UDP/IP socket. Once a socket is closed, no more sending or receiving is allowed via the socket.

⚠ CAUTION

Issuing multiple close requests for the same socket is not allowed.

Programming Example
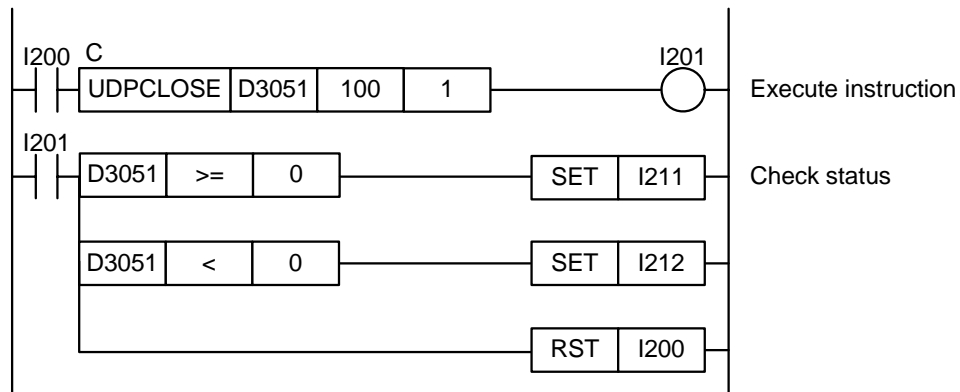
Figure 2.6.10   UDP/IP Close Sample Program

This sample code closes a UDP/IP socket associated with socket ID 1. It specifies the timeout interval as 100 (10 s).

The table below shows the returned status data (ret), assuming normal exit.

Device	Value	Table Parameter
ret = D3051	0	Status

2.6.3.2   UDP/IP Send and Receive Instructions
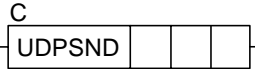
Table 2.6.24   List of UDP/IP Send and Receive Instructions

Service Name	Ladder Instruction	Description
UDP/IP Send Request	UDPSND	Sends data from a specified UDP/IP socket.
UDP/IP Receive Request	UDPRCV	Receives data from a specified UDP/IP socket.

■ UDP/IP Send Request (UDPSND)

Sends data stored in a specified device using UDP/IP communications.

Table 2.6.25   UDP/IP Send Request

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required? Yes	No	Step Count	Pro–cessing Unit	Carry
Continuous type application instruction	–	UDP/IP Send Request	UDPSND	C —[UDPSND ⬚ ⬚ ⬚]—	✓	–	6	8 bit	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see "■ Continuous Type Application Instructions" in Subsection 2.6.1, "Using Socket Instructions".

Parameter

UDP/IP Send Request   C —[UDPSND | ret | t | s]—
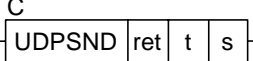
Table 2.6.26   Parameters

Parameter		Description
ret [1]		Device for storing return status (W)
t	t+0	Timeout interval (W) [0 = infinite, 1-32767 (x100 ms)]
	t+1	Socket ID (W) [0-7]
	t+2	Size of send data (W) [0-2048 (bytes)] [2]
	t+3	Socket destination (W) [[3] -1 = IP address and port no. (designated by t+4 to t+6) 1-16 = Socket address setting no. in CPU properties]
	t+4	Destination IP address low (W) [$0000-$FFFF] [3]
	t+5	Destination IP address high (W) [$0000-$FFFF] [3]
	t+6	Destination port no. (W) [1-65535] [4]
s		First device of send data (W)

*1: ret (status) is table data. For details on the return status (ret), see "Status (Return Value)".
*2: 1472 bytes max. for broadcast communications.
*3: Do not specify destination IP address as 0.0.0.0. Otherwise, the operation or error status will be indefinite.
*4: Word data is handled as an unsigned decimal or hexadecimal number.

Status (Return Value)

Table 2.6.27   **Status (Return Value)**

Offset (word)			Description
ret	ret+0	> 0	Sent data size [1-2048 (bytes)]
		< 0	Error status

SEE ALSO

For more details on error status, see "● Error Status of Continuous Type Application Instructions" of "■ Continuous Type Application Instructions" in Subsection 2.6.1, "Using Socket Instructions".

Available Devices

Table 2.6.28   **Available Devices**

Device Parameter	X	Y	I	E	L	M	T	C	D	B	W	Z	R	V	Con-stant	Index Modification	Indirect Designation, Pointer P
ret									✓	✓	✓		✓	✓		Yes	Yes
t									✓	✓	✓		✓	✓		Yes	Yes
s									✓	✓	✓		✓	✓	#	Yes	Yes

Note: See Section 1.15, "Restrictions on Devices Used as Instruction Parameters" of "Sequence CPU – Instructions" (IM34M6P12-03E)

\#:   You can specify the constant name of a constant definition but not a normal constant.

Resource Relays

Table 2.6.29   **Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition**

Add to Input Condition	Number	Name	Usage
	M1028	No Unused UDP Socket	Execute UDPSND instruction for a socket ID only if both its corresponding Socket Busy and Socket Sending relays are OFF.
	M1029	No Unused TCP Socket	
	M1105 to M1120	Socket Open	
✓	M1121 to M1136	Socket Busy	
✓	M1073 to M1088	Socket Sending	
	M1089 to M1104	Socket Receiving	

Function

Sends data stored in a specified device using UDP/IP communications.

You can either specify the destination using a socket address setting number defined in the socket address setup of CPU properties, or specify an IP address and port number directly as instruction parameters. In the latter case, set the socket destination parameter as -1.

For the socket ID parameter, specify the socket ID returned by the UDP/IP Open (UDPOPEN) instruction executed earlier.

Broadcast Transmission

If UDP broadcast is enabled in the socket setup of CPU properties, you can perform broadcast transmission by setting the lowest byte of the destination IP address to 255. For example, if the network address is 192.168.0.xxx, specify 192.168.0.255 to perform broadcast transmission. Any attempt at broadcast transmission when UDP broadcast is disabled in CPU properties will generate an unknown destination error (error code: -5001).

A target network node may fail to receive a broadcast transmission if it is configured to ignore broadcast transmission or a different IP address is defined as the broadcast address. For details, check with the network administrator.

A maximum send data size of 1472 bytes is allowed for broadcast transmission.

⚠ CAUTION

- Concurrent send requests for the same socket are not allowed but concurrent execution of a send request and a receive request is allowed.
- If you specify a socket address setting number with a defined hostname in CPU properties in the instruction, performance will be affected by the time required for DNS resolution.
- The nature of the protocol is such that the instruction will exit normally even if the link is down (e.g. the cable is not connected or the hub is switched off).

Programming Example

I200		BMOV	D0051	D2001	7		Set up parameter table t
C	UDPSND	D3051	D2001	B1025	I201		Execute instruction
I201	D3051	>=	0	SET	I211		Check status
	D3051	<	0	SET	I212		
				RST	I200		
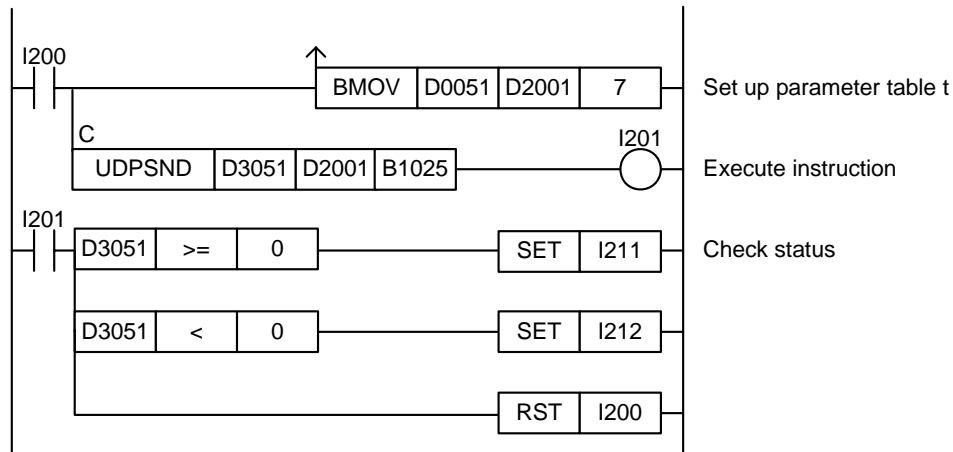
Figure 2.6.11   UDP/IP Send Request Sample Program

This sample code sends to the node associated with socket destination number 4, 200 bytes of data stored in device starting from device B1025.

It specifies ret(=D3051), t(=D2001) and s(=B1025) with t set up as shown in the table below.

Device	Value	Table Parameter
t = D2001	600	Timeout interval (=60 s)
D2002	1	Socket ID (= 1)
D2003	200	Size of send data (= 200 bytes)
D2004	4	Socket destination no. (= 4)
D2005	0	Destination IP address (not required for this example)
D2006		
D2007	0	Destination port no. (not required for this example)

The table below shows the returned status data (ret), assuming normal exit.

Device	Value	Table Parameter
ret = D3051	200	Status (sent data size)

■ UDP/IP Receive Request (UDPRCV)

Stores data received from a UDP/IP socket to a specified device.

Table 2.6.30   UDP/IP Receive Request

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Pro–cessing Unit	Carry
					Yes	No			
Continuous type application instruction	–	UDP/IP Receive Request	UDPRCV	C ─ UDPRCV ┌─┬─┬─┐ ─	✓	–	6	8 bit	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see "■ Continuous Type Application Instructions" in Subsection 2.6.1, "Using Socket Instructions".

Parameter

C
UDP/IP Receive Request   ─ UDPRCV │ret│ t │ d ─

Table 2.6.31   Parameters

Parameter		Description
ret[*1]		Device for storing return status (W)
t	t+0	Timeout interval (W) [0 = infinite, 1-32767 (x100 ms)]
	t+1	Socket ID (W) [0-7]
	t+2	Size of receive area (W) [0-4096 (bytes)]
	t+3	Append NULL option (W) [0=no; 1=yes]
	t+4	Buffer option (W) [[*2] 0=Delete packet in receive buffer after retrieval 1=Keep packet in receive buffer after retrieval 2=Check packet size of receive buffer (without receive processing)]
d		First device for storing received data (W)

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".
*2: Buffer option 0 is recommended to avoid buffer overflow.

Status (Return Value)

Table 2.6.32   Status (Return Value)

Offset (word)			Description
ret	ret+0	> 0	Received data size [1-4096 (bytes)] [*5]
		< 0	Error status
	ret+1		CPU properties socket address setting search result (W) [1-16 = Match for both IP address and port no.[*1] 101-116 = Match for IP address only[*2] -1 = No match[*3]]
	ret+2		Sender IP address low (W) [$0000-$FFFF]
	ret+3		Sender IP address high (W) [$0000-$FFFF]
	ret+4		Sender port number (W) [0-65535] [*4]

*1: If a match is found for the IP address and port number of the sender in the socket address setup of CPU properties, the corresponding setting number is returned.
*2: If a match is found for only the IP address of the sender in the socket address setup of CPU properties, the corresponding setting number + 100 is returned.
*3: If no match is found for the IP address of the sender in the socket address setup of CPU properties, -1 is returned.
*4: Word data is handled as an unsigned decimal or hexadecimal number.
*5: Received data size includes any NULL byte appended according to the Append NULL option.

Available Devices

Table 2.6.33 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	W	Z	R	V	Con–stant	Index Modification	Indirect Designation, Pointer P
ret									✓	✓	✓		✓	✓		Yes	Yes
t									✓	✓	✓		✓	✓		Yes	Yes
d									✓	✓	✓		✓	✓		Yes	Yes

Note: See Section 1.15, "Restrictions on Devices Used as Instruction Parameters" of "Sequence CPU – Instructions" (IM34M6P12-03E)

Resource Relays

Table 2.6.34 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
	M1028	No Unused UDP Socket	
	M1029	No Unused TCP Socket	Execute UDPRCV instruction for a socket ID only if both its corresponding Socket Busy and Socket Receiving relays are OFF.
	M1105 to M1120	Socket Open	
✓	M1121 to M1136	Socket Busy	
	M1073 to M1088	Socket Sending	
✓	M1089 to M1104	Socket Receiving	

Function

Stores data received from a UDP/IP socket to a specified device. For the socket ID parameter, specify the socket ID returned by the UDP/IP Open (UDPOPEN) instruction executed earlier. The size of the data stored to device is returned in status.

You can specify whether to append a NULL byte behind received data using the Append Null Option.

If no data is received in the buffer when the instruction is executed, the instruction waits for data to arrive. However, if the buffer option parameter is set to 2 (= check packet size of receive buffer), the instruction completes execution without entering wait state even if no data has been received. If a timeout interval is specified and no data is received within the specified time, the instruction exits from wait state, holds the result signal to ON and returns a timeout error in status.

If the buffer option parameter is 0, the data packet in the receive buffer is discarded after retrieval regardless of the size of receive area.

Broadcast transmission
If broadcast transmission is enabled in the socket setup of CPU properties, the instruction receives broadcast packets.

> ⚠ **CAUTION**
>
> - Concurrent receive requests for the same socket are not allowed but concurrent execution of a send request and a receive request is allowed.
> - You should specify the buffer option as 0 (= delete packet after retrieval) unless there is a special reason not to do so. Specifying a non-zero buffer option means that the receive buffer is not emptied and this may result in buffer overflow.
> - If you specify a socket address setting number with a defined hostname in CPU properties in the instruction, performance will be affected by the time required for DNS resolution.

Programming Example
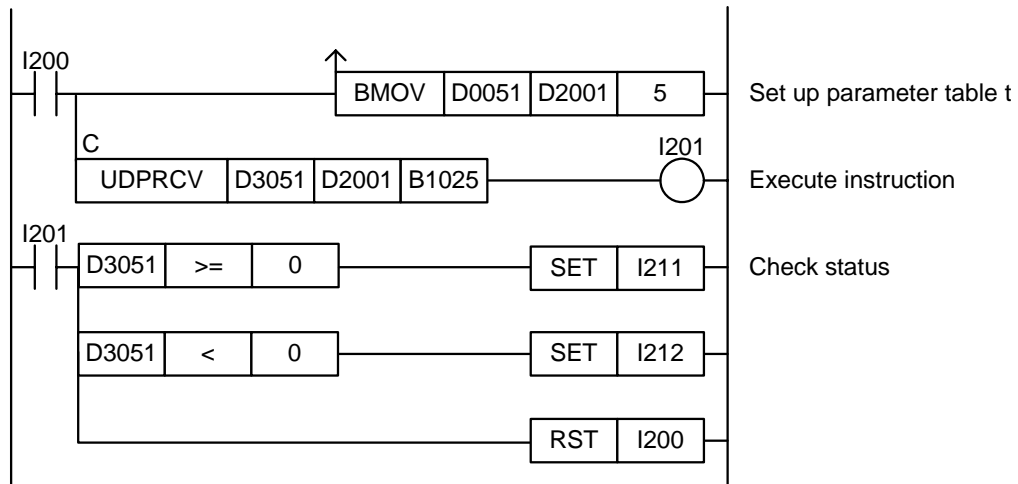
Figure 2.6.12   UDP/IP Receive Request Sample Program

This sample code receives data from the UDP/IP socket associated with socket ID 4, and stores the received data to device, starting from device B1025.

It specifies ret(=D3051), t(=D2001) and d(=B1025), with t set up as shown in the table below.

Device	Value	Table Parameter
t = D2001	6000	Timeout interval (= 600 s)
D2002	4	Socket ID (=4)
D2003	2048	Size of receive area (= 2048 bytes)
D2004	0	Append NULL option (= No)
D2005	0	Buffer option (= Delete packet after retrieval (recommended))

The table below shows an example of the returned status data (ret), assuming that a 520-byte packet is received from an IP address (192.168.0.67:10456), which is not registered in the socket address setup of CPU properties.

Device	Value	Table Parameter
ret = D3051	520	Status
D3052	-1	Sender socket address setting number
D3053	$0043	Sender IP address
D3054	$C0A8	
D3055	10456	Sender port number

2.6.3.3　TCP/IP Communications Preparation Instructions

Table 2.6.35　　TCP/IP Communications Preparation Instructions

Service Name	Ladder Instruction	Description
TCP/IP Open	TCPOPEN	Opens a TCP/IP socket.
TCP/IP Close	TCPCLOSE	Closes a TCP/IP socket.
TCP/IP Connection Request	TCPCNCT	Issues a connection request to a server as a client, and connects to the server if permitted to do so.
TCP/IP Listen Request	TCPLISN	Waits for a connection request from any client as a server, and establishes connection if a request is received.

■ TCP/IP Open (TCPOPEN)

Opens a TCP/IP socket.

Table 2.6.36　　TCP/IP Open

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Processing Unit	Carry
					Yes	No			
Continuous type application instruction	–	TCP/IP Open	TCPOPEN	C ─┤ TCPOPEN ├─	✓	–	5	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see "■ Continuous Type Application Instructions" in Subsection 2.6.1, "Using Socket Instructions".
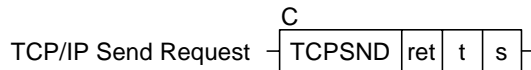
Parameter

C
TCP/IP Open　─┤ TCPOPEN │ ret │ n ├─

Table 2.6.37　　Parameters

Parameter	Description
ret[1]	Device for storing return status (W)
n	Timeout interval (W) [0 = infinite, 1-32767 (x100 ms)]

*1: ret (status) is table data. For details on the return status (ret), see "Status (Return Value)".

Status (Return Value)

Table 2.6.38　　Status (Return Value)

Offset (word)	Description	
ret	> 0	Socket ID (W) [8-15] (socket is successfully opened)
	< 0	Error status

SEE ALSO

For more details on error status, see "● Error Status of Continuous Type Application Instructions" of "■ Continuous Type Application Instructions" in Subsection 2.6.1, "Using Socket Instructions".

Available Devices

Table 2.6.39   Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	W	Z	R	V	Con-stant	Index Modification	Indirect Designation, Pointer P
ret									✓	✓	✓		✓	✓		Yes	Yes
n									✓	✓	✓		✓	✓	✓	Yes	Yes

Note: See Section 1.15, "Restrictions on Devices Used as Instruction Parameters" of "Sequence CPU – Instructions" (IM34M6P12-03E)

Resource Relays

Table 2.6.40   Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
	M1028	No Unused UDP Socket	
✓	M1029	No Unused TCP Socket	Execute TCPOPEN instruction only if the No Unused TCP Socket relay is OFF.
	M1105 to M1120	Socket Open	
	M1121 to M1136	Socket Busy	
	M1073 to M1088	Socket Sending	
	M1089 to M1104	Socket Receiving	

Function

Opens a TCP/IP socket. Opening a socket secures system resources required for communications to enable execution of the TCP/IP Connect Request (TCPCNCT) instruction or the TCP/IP Listen Request (TCPLISN) instruction.

Up to 8 TCP/IP sockets can be open concurrently at any one time. The socket ID is automatically allocated a value from 8 to 15, but not necessarily in any order.

If execution is successful, this instruction returns a socket ID in status, which is to be used in subsequent TCP/IP Connect Request (TCPCNCT) instructions and TCP/IP Listen Request (TCPLISN) instructions. When connected as a client (after executing a TCPCNCT instruction), the same socket ID can also be used in TCP/IP send and receive instructions.

⚠ CAUTION

The allocated socket ID can be any value between 8 and 15. Note that it does not start from 0.

Programming Example
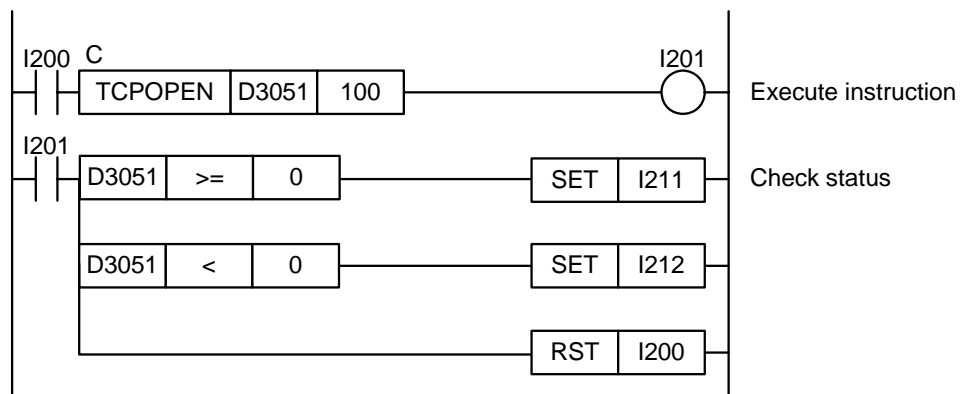
Figure 2.6.13   TCP/IP Open Sample Program

This sample code opens a TCP/IP socket. It specifies the timeout interval as 100 (10 s).

The table below shows the returned status data (ret), assuming normal exit.

Device	Value	Table Parameter
ret = D3051	8	Status (socket ID)

■ TCP/IP Close (TCPCLOSE)

Closes a TCP/IP socket. Once a socket is closed, the socket ID can no longer be used.

Table 2.6.41   TCP/IP Close

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required? Yes	Input Condition Required? No	Step Count	Pro–cessing Unit	Carry
Continuous type application instruction	–	TCP/IP Close	TCPCLOSE	C ⌐TCPCLOSE ⌐	✓	–	6	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see "■ Continuous Type Application Instructions" in Subsection 2.6.1, "Using Socket Instructions".
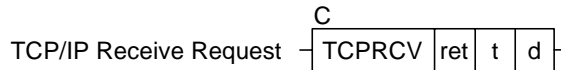
Parameter

TCP/IP Close   C ⌐TCPCLOSE | ret | n1 | n2 ⌐

Table 2.6.42   Parameters

Parameter	Description
ret[*1]	Device for storing return status (W)
n1	Timeout interval (W) [0 = infinite, 1-32767 (x100 ms)]
n2	Socket ID (W) [8-15]

*1: ret (status) is table data. For details on the return status (ret), see "Status (Return Value)".

Status (Return Value)

Table 2.6.43   Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit
		< 0	Error status

SEE ALSO

For more details on error status, see "● Error Status of Continuous Type Application Instructions" of "■ Continuous Type Application Instructions" in Subsection 2.6.1, "Using Socket Instructions".

Available Devices

Table 2.6.44   Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	W	Z	R	V	Con-stant	Index Modification	Indirect Designation, Pointer P
ret									✓	✓	✓		✓	✓		Yes	Yes
n1									✓	✓	✓		✓	✓	✓	Yes	Yes
n2									✓	✓	✓		✓	✓	✓	Yes	Yes

Note: See Section 1.15, "Restrictions on Devices Used as Instruction Parameters" of "Sequence CPU – Instructions" (IM34M6P12-03E)

Resource Relays

Table 2.6.45 **Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition**

Add to Input Condition	Number	Name	Usage
	M1028	No Unused UDP Socket	Execute TCPCLOSE instruction for a socket ID only if its corresponding Socket Busy, Socket Sending and Socket Receiving relays are all off.
	M1029	No Unused TCP Socket	
	M1105 to M1120	Socket Open	
✓	M1121 to M1136	Socket Busy	
✓	M1073 to M1088	Socket Sending	
✓	M1089 to M1104	Socket Receiving	

Function

Closes a TCP/IP socket. Once a socket is closed, the socket ID can no longer be used.

⚠ CAUTION

- Issuing multiple close requests for the same socket is not allowed.
- Depending on the state of the destination, cancellation may sometimes take a long time to complete.

Programming Example
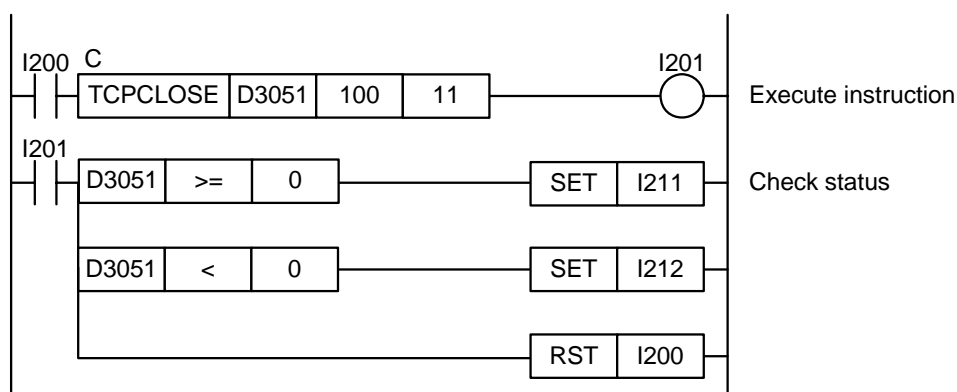
Figure 2.6.14 **TCP/IP Close Sample Program**

This sample code closes a TCP/IP socket associated with socket ID 11. It specifies the timeout interval as 100 (10 s).

The table below shows the returned status data (ret), assuming normal exit.

Device	Value	Table Parameter
ret = D3051	0	Status

■ TCP/IP Connect Request (TCPCNCT)

Issues a connection request to a TCP/IP server (a node which is waiting for connection or, in other words, listening), and establishes a connection if permitted to do so.

Table 2.6.46   TCP/IP Connect Request

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Pro–cessing Unit	Carry
					Yes	No			
Continuous type application instruction	–	TCP/IP Connect Request	TCPCNCT	C ─[TCPCNCT　　　]─	✓	–	5	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see "■ Continuous Type Application Instructions" in Subsection 2.6.1, "Using Socket Instructions".

Parameter

TCP/IP Connect Request ─ C [TCPCNCT | ret | t] ─
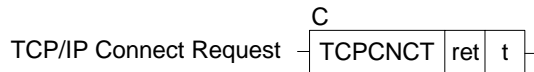
Table 2.6.47   Parameters

Parameter		Description
ret[*1]		Device for storing return status (W)
t	t+0	Timeout interval (W) [0 = infinite, 1-32767 (x100 ms)]
	t+1	Socket ID (W) [8-15]
	t+2	Socket destination (W) [[*2] -1 = IP address and port no. (designated by t+3 to t+5) 1-16 = Socket address setting no. in CPU properties]
	t+3	Destination IP address low (W) [$0000-$FFFF] [*2]
	t+4	Destination IP address high (W) [$0000-$FFFF] [*2]
	t+5	Destination port no. (W) [1-65535] [*3]

*1: ret (status) is table data. For details on the return status (ret), see "Status (Return Value)".
*2: Do not specify destination IP address as 0.0.0.0. Otherwise, a parameter error status will be returned.
*3: Word data is handled as an unsigned decimal or hexadecimal number.

Status (Return Value)

Table 2.6.48   Status (Return Value)

Offset (word)			Description
ret	ret+0	0	Normal exit
		< 0	Error status

SEE ALSO

For more details on error status, see "● Error Status of Continuous Type Application Instructions" of "■ Continuous Type Application Instructions" in Subsection 2.6.1, "Using Socket Instructions".

Available Devices

Table 2.6.49   **Available Devices**

Device Parameter	X	Y	I	E	L	M	T	C	D	B	W	Z	R	V	Con-stant	Index Modification	Indirect Designation, Pointer P
ret									✓	✓	✓		✓	✓		Yes	Yes
t									✓	✓	✓		✓	✓		Yes	Yes

Note: See Section 1.15, "Restrictions on Devices Used as Instruction Parameters" of "Sequence CPU – Instructions" (IM34M6P12-03E)

Resource Relays

Table 2.6.50   **Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition**

Add to Input Condition	Number	Name	Usage
	M1028	No Unused UDP Socket	
	M1029	No Unused TCP Socket	Execute TCPCNCT instruction for a socket ID only if its corresponding Socket Busy relay is OFF.
	M1105 to M1120	Socket Open	
✓	M1121 to M1136	Socket Busy	
	M1073 to M1088	Socket Sending	
	M1089 to M1104	Socket Receiving	

Function

Prepares for communications as a client by issuing a connection request to a TCP/IP server (a node which is waiting for connection or, in other words, listening), and establishing a connection if permitted to do so.

You can either specify the destination using a socket address setting number defined in the socket address setup of CPU properties, or specify an IP address and port number directly as instruction parameters.

⚠ CAUTION

- Issuing multiple connection requests for the same socket is not allowed.
- Issuing a connection request to the address of a node itself is not allowed. Doing so will generate an unknown destination error (error code: -5001).
- If a connection error code (-5000) or unknown destination error code (-5001) is returned in status, you must execute the TCP/IP Close (TCPCLOSE) instruction. By nature of a general protocol stack, the socket ID used transits to an invalid state.
- Sometimes communications may fail even after a TCP/IP Connect Request (TCPCNCT) exits normally. This may happen if the server side develops an error such as depletion of available sockets after it returns a successful reply to a connection request but before it successfully establishes a transmission channel.

Programming Example
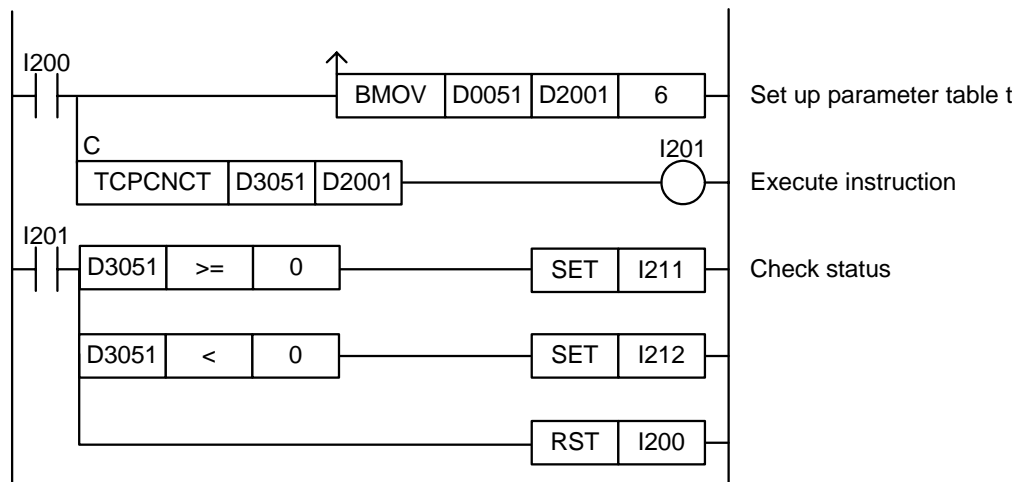
Figure 2.6.15   TCP/IP Connect Request Sample Program

This sample code issues a connection request by directly specifying 192.168.0.6:20677 as the destination address in the instruction.

It specifies ret(=D3051) and t(=D2001) with t set up as shown in the table below.

Device	Value	Table Parameter
t = D2001	600	Timeout interval (=60 s)
D2002	12	Socket ID (= 12)
D2003	-1	Socket destination no.(=direct designation)
D2004	$0006	Destination IP address (= 192.168.0.6)
D2005	$C0A8	
D2006	20677	Destination port no. (= 20677)

The table below shows the returned status data (ret), assuming normal exit.

Device	Value	Table Parameter
ret =D3051	0	Status

■ TCP/IP Listen Request (TCPLISN)

Waits for connection request from any TCP/IP client, and establishes connection if a request is received.

Table 2.6.51   TCP/IP Listen Request

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Pro–cessing Unit	Carry
					Yes	No			
Continuous type application instruction	–	TCP/IP Listen Request	TCPLISN	C ─[TCPLISN]─	✓	–	5	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see "■ Continuous Type Application Instructions" in Subsection 2.6.1, "Using Socket Instructions".

Parameter

TCP/IP Listen Request   C ─[TCPLISN | ret | t]─

Table 2.6.52   Parameters

Parameter		Description
ret[1]		Device for storing return status (W)
t	t+0	Timeout interval (W) [0 = infinite, 1-32767 (x100 ms)]
	t+1	Socket ID (W) [8-15]
	t+2	My port number (W) [1-65535][2][3]

*1: ret (status) is table data. For details on the return status (ret), see "Status (Return Value)".
*2: Do not specify my port number as 12289, 12290, 12291, 12305 or 12307 as these numbers are used by the higher-level link service and remote programming service.
*3: Word data is handled as an unsigned decimal or hexadecimal number.

Status (Return Value)

Table 2.6.53   Status (Return Value)

	Offset (word)	Description	
ret	ret+0	0	Normal exit
		< 0	Error status
	ret+1	New socket ID for sending and receiving (W) [8-15]	
	ret+2	CPU properties socket address setting search result (W) [1-16 = Match for both IP address and port no.[1] 101-116 = Match for IP address only[2] -1 = No match[3]]	
	ret+3	Source IP address low (W) [$0000-$FFFF]	
	ret+4	Source IP address high (W) [$0000-$FFFF]	
	ret+5	Source port number (W) [1-65535][4]	

*1: If a match is found for the IP address and port number of the source in the socket address setup of CPU properties, the corresponding setting number is returned.
*2: If a match is found for only the IP address of the source in the socket address setup of CPU properties, the corresponding setting number + 100 is returned.
*3: If no match is found for the IP address of the source in the socket address setup of CPU properties, -1 is returned.
*4: Word data is handled as an unsigned decimal or hexadecimal number.

SEE ALSO

For more details on error status, see "● Error Status of Continuous Type Application Instructions" of "■ Continuous Type Application Instructions" in Subsection 2.6.1, "Using Socket Instructions".

Available Devices

Table 2.6.54   Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	W	Z	R	V	Constant	Index Modification	Indirect Designation, Pointer P
ret									✓	✓	✓		✓	✓		Yes	Yes
t									✓	✓	✓		✓	✓		Yes	Yes

Note: See Section 1.15, "Restrictions on Devices Used as Instruction Parameters" of "Sequence CPU – Instructions" (IM34M6P12-03E)

Resource Relays

Table 2.6.55   Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
	M1028	No Unused UDP Socket	Execute TCPLISN instruction for a socket ID only if both the No Unused TCP Socket relay and its corresponding Socket Busy relay are OFF.
✓	M1029	No Unused TCP Socket	
	M1105 to M1120	Socket Open	
✓	M1121 to M1136	Socket Busy	
	M1073 to M1088	Socket Sending	
	M1089 to M1104	Socket Receiving	

Function

Prepares for communications as a server. Waits for connection request from any TCP/IP client, and establishes connection if a request is received.

When the instruction successfully establishes a connection with a client, it returns a new socket ID in status. The new socket ID is to be used for subsequent sending and receiving. The socket ID specified as a parameter of this instruction is not used for sending and receiving and therefore can be reused to listen for a connection request from a different client by re-executing this instruction. In other words, the same socket (=same port number) can be used to listen for connection requests from multiple clients. After connection, data can be sent to and received from multiple clients.

CAUTION

- Issuing multiple connection requests for the same socket is not allowed.
- When sending data to and receiving data from TCP/IP clients, use the socket ID returned in status by this instruction, but not the socket ID that is specified as a parameter of this instruction.

Programming Example
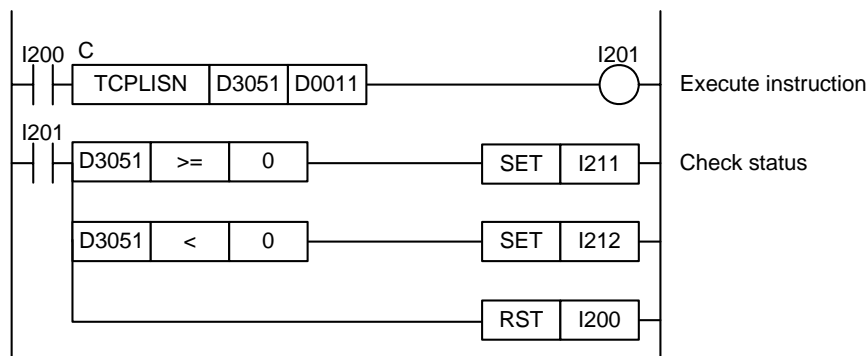
Figure 2.6.16   TCP/IP Connect Request Sample Program

This sample code listens for a connection request from any client. It assumes that the required parameter values (timeout interval, the socket ID to be used by the TCPLISN instruction, my port number) are already stored in the device area starting from device D0011. It specifies D3051 as the first device for storing the returned status and new socket ID for sending and receiving.

The table below shows an example of the returned status data (ret), assuming that the instruction exited normally after processing a connection request from a peer (192.168.0.9: 10456), which is registered as socket address setting 3 in CPU properties.

Device	Value	Table Parameter
ret = D3051	0	Status
D3052	12	New socket ID for sending and receiving
D3053	3	Socket address setting no. of source
D3054	$0009	Source IP address
D3055	$C0A8	
D3056	10456	Source port number

2.6.3.4 TCP/IP Send and Receive Instructions

Table 2.6.56   List of TCP/IP Send and Receive Instructions

Service Name	Ladder Instruction	Description
TCP/IP Send Request	TCPSND	Sends device data from a specified TCP/IP socket.
TCP/IP Receive Request	TCPRCV	Writes data received from a specified TCP/IP socket to device.

■ TCP/IP Send Request (TCPSND)

Sends data stored in a specified device using TCP/IP communications.

Table 2.6.57   TCP/IP Send Request

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required? Yes	Input Condition Required? No	Step Count	Pro–cessing Unit	Carry
Continuous type application instruction	–	TCP/IP Send Request	TCPSND	C –[TCPSND]–	✓	–	6	8 bit	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see "■ Continuous Type Application Instructions" in Subsection 2.6.1, "Using Socket Instructions".

Parameter

C
TCP/IP Send Request –[TCPSND | ret | t | s]–

Table 2.6.58   Parameters

Parameter		Description
ret[1]		Device for storing return status (W)
t	t+0	Timeout interval (W) [0 = infinite, 1-32767 (x100 ms)]
	t+1	Socket ID (W) [8-15]
	t+2	Size of send data (W) [0-2048 (bytes)]
s		First device of send data (W)

*1: ret (status) is table data. For details on the return status (ret), see "Status (Return Value)".

Status (Return Value)

Table 2.6.59   Status (Return Value)

Offset (word)		Description	
ret	ret+0	> 0	Sent data size [1-2048 (bytes)]
		< 0	Error status

SEE ALSO

For more details on error status, see "● Error Status of Continuous Type Application Instructions" of "■ Continuous Type Application Instructions" in Subsection 2.6.1, "Using Socket Instructions".

Available Devices

Table 2.6.60   Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	W	Z	R	V	Con-stant	Index Modification	Indirect Designation, Pointer P
ret									✓	✓	✓		✓	✓		Yes	Yes
t									✓	✓	✓		✓	✓		Yes	Yes
s									✓	✓	✓		✓	✓	#	Yes	Yes

Note: See Section 1.15, "Restrictions on Devices Used as Instruction Parameters" of "Sequence CPU – Instructions" (IM34M6P12-03E)

#:   You can specify the constant name of a constant definition but not a normal constant.

Resource Relays

Table 2.6.61   Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
	M1028	No Unused UDP Socket	Execute TCPSND instruction for a socket ID only if both its corresponding Socket Busy relay and Socket Sending relay are OFF.
	M1029	No Unused TCP Socket	
	M1105 to M1120	Socket Open	
✓	M1121 to M1136	Socket Busy	
✓	M1073 to M1088	Socket Sending	
	M1089 to M1104	Socket Receiving	

Function

Sends data stored in a specified device using TCP/IP communications.

Specify the destination as a socket ID. For a client, specify the socket ID returned by the TCP/IP Open (TCPOPEN) instruction. For a server, specify the socket ID returned by the TCP/IP Listen Request (TCPLISN) instruction.

⚠ CAUTION

Concurrent send requests for the same socket are not allowed but concurrent execution of a send request and a receive request is allowed.

Programming Example
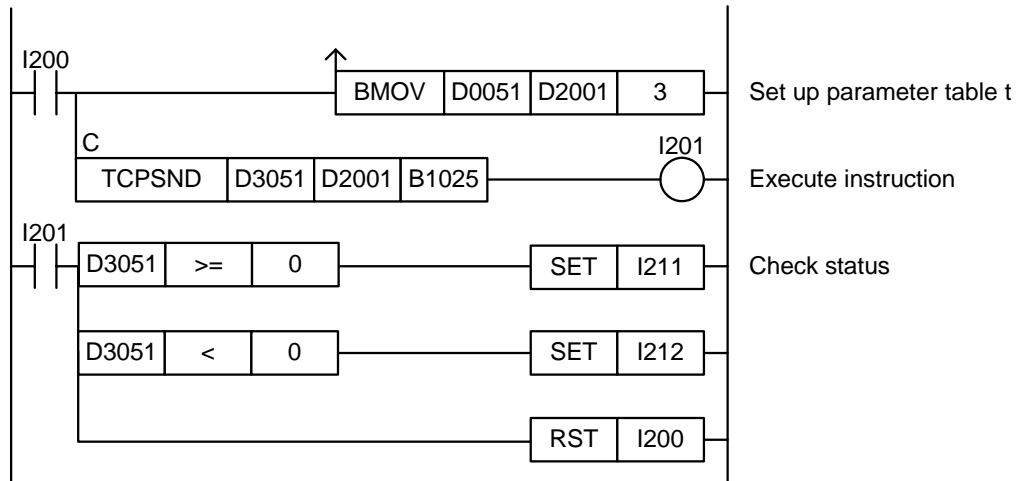
Figure 2.6.17   TCP/IP Send Request Sample Program

This sample code sends to the node connected to TCP/IP socket ID 12, 212 bytes of data stored in device starting from device B1025.

It specifies ret(=D3051), t(=D2001) and s(=B1025) with t set up as shown in the table below.

Device	Value	Table Parameter
t = D2001	600	Timeout interval (=60 s)
D2002	12	Socket ID (= 12)
D2003	212	Size of send data (= 212 bytes)

The table below shows the returned status data (ret), assuming normal exit.

Device	Value	Table Parameter
ret = D3051	212	Status (sent data size)

■ TCP/IP Receive Request (TCPRCV)

Stores data received from a TCP/IP socket to a specified device.

Table 2.6.62   TCP/IP Receive Request

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Pro–cessing Unit	Carry
					Yes	No			
Continuous type application instruction	–	TCP/IP Receive Request	TCPRCV	C ─┤TCPRCV ├─	✓	–	6	8 bit	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see "■ Continuous Type Application Instructions" in Subsection 2.6.1, "Using Socket Instructions".

Parameter

TCP/IP Receive Request ─┤ C
TCPRCV │ret│ t │ d ├─

Table 2.6.63   Parameters

Parameter		Description
ret[1]		Device for storing return status (W)
t	t+0	Timeout interval (W) [0 = infinite, 1-32767 (x100 ms)]
	t+1	Socket ID (W) [8-15]
	t+2	Size of receive area (W) [0-2048 (bytes)]
	t+3	Append NULL option (W) [2] [0=no; 1=yes]

	t+4	Buffer option (W) [[3]
		   0 = Delete read data in receive buffer after retrieval
		      (normal mode)
		   1 = Keep read data in receive buffer after retrieval
		   2 = Check data size in receive buffer
		   3 = Delete data in receive buffer without retrieval
		   4 = Delete read data in receive buffer after retrieval
		      (Auto increment mode)
		]

d		First register for receive area (W)

[1]: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".
[2]: Any NULL byte appended according to the Append NULL option should be included in the size of receive area.
[3]: Buffer options 0 and 4 are recommended to avoid buffer overflow.

Status (Return Value)

Table 2.6.64   Status (Return Value)

Offset (word)			Description
ret	ret+0	> 0	Received data size [1-2048 (bytes)][1]
		< 0	Error status

[1]: Received data size includes any NULL byte appended according to Append NULL option.

SEE ALSO

For more details on error status, see "● Error Status of Continuous Type Application Instructions" of "■ Continuous Type Application Instructions" in Subsection 2.6.1, "Using Socket Instructions".

Available Devices

Table 2.6.65   Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	W	Z	R	V	Con-stant	Index Modification	Indirect Designation, Pointer P
ret									✓	✓	✓		✓	✓		Yes	Yes
t									✓	✓	✓		✓	✓		Yes	Yes
d									✓	✓	✓		✓	✓		Yes	Yes

Note: See Section 1.15, "Restrictions on Devices Used as Instruction Parameters" of "Sequence CPU – Instructions" (IM34M6P12-03E)

Resource Relays

Table 2.6.66   Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
	M1028	No Unused UDP Socket	Execute TCPRCV instruction for a socket ID only if both its corresponding Socket Busy relay and Socket Receiving relay are OFF.
	M1029	No Unused TCP Socket	
	M1105 to M1120	Socket Open	
✓	M1121 to M1136	Socket Busy	
	M1073 to M1088	Socket Sending	
✓	M1089 to M1104	Socket Receiving	

Function

Stores data received from a TCP/IP socket to a specified device. After execution, the size of data stored to device is returned in status. You can specify whether to append a NULL byte behind received data using the Append Null Option.

If no data is received in the buffer when the instruction is executed, the instruction waits for data to arrive. However, if the buffer option parameter is set to 2 (= check data size in receive buffer), the instruction completes execution without entering wait state even if no data has been received. If a timeout interval is specified and no data is received within the specified time, the instruction exits from wait state, holds the result signal to ON and returns a timeout error in status.

Auto increment mode
Specifying 4 for the buffer option parameter selects auto increment mode in which the instruction automatically increments parameter d (first device for receive area) by the received data size so that the entire data received through multiple instruction executions can be stored contiguously to devices.

Using auto increment mode
Auto increment mode must be used together with normal mode. Specify normal mode for the first execution of TCPRCV and specify auto increment mode for the second and subsequent executions. This way, the received data will be stored contiguously to device.

Do not modify the value of parameter d (first device of receive area) for the second and subsequent executions as the byte offset is computed automatically by the instruction.

Canceling auto increment mode

To reset the byte offset to zero, you can either:

- Execute the instruction by specifying any value other than 4 for the buffer option;
- Execute the instruction with a modified value of parameter d (first address of receive area)

⚠ CAUTION

- Concurrent receive requests for the same socket are not allowed but concurrent execution of a send request and a receive request is allowed.
- Selecting a buffer option that does not delete data in the receive buffer may lead to buffer overflow.
- If auto increment mode is specified and the byte offset (= accumulated received size) exceeds the specified size of receive area parameter, a "Specified Size/Times Processed" (-2003) status is returned.

Programming Example
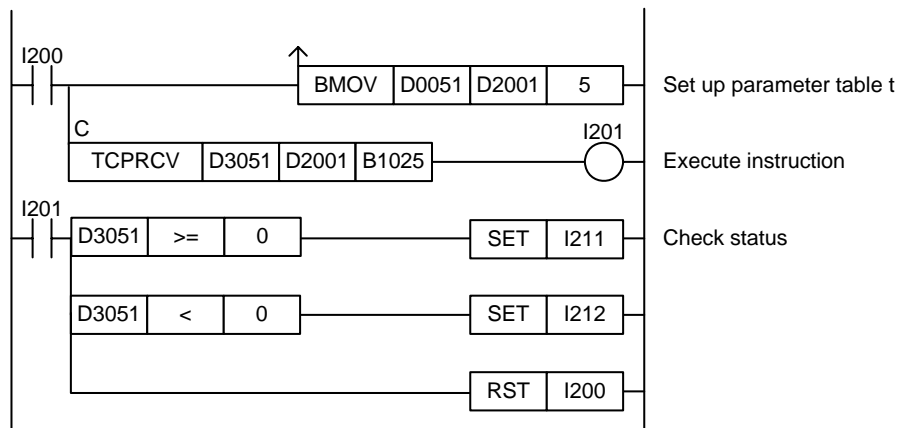
Figure 2.6.18   TCP/IP Receive Request Sample Program

This sample code waits for data from the node connected to TCP/IP socket ID 12, and stores the received data to device, starting from device B1025.

It specifies ret(=D3051), t(=D2001) and d(=B1025), with t set up as shown in the table below.

Device	Value	Table Parameter
t = D2001	6000	Timeout interval (= 600 s)
D2002	12	Socket ID (=12)
D2003	2048	Size of receive area (= 2048 bytes)
D2004	0	Append NULL option (= No)
D2005	0	Buffer option (= Normal mode (recommended))

The table below shows the returned status data (ret), assuming normal exit after receiving 608 bytes of data.

Device	Value	Table Parameter
ret = D3051	608	Status (received data size)

2.7   Socket Communications Sample Program

This section describes a sample program for the socket communications function. These sample programs are intended to help a user better understand the instruction specifications and are not intended to be used directly in user applications.

■ List of Sample Programs

The table below shows the file structure of a sample program provided for the socket communications function. Files for the sample program are automatically copied to the respective folders shown below when WideField2 is installed.

Table 2.7.1   Sample Program Components and Location

Sample Program Name	Component	Location
UDP/IP echo server	Project	~\Fam3pjt\CPUSample\F3SP66\UECHOC\UECHOC.YPJT ~\Fam3pjt\CPUSample\F3SP66\UECHOS\UECHOS.YPJT
	CPU Properties	~\Fam3pjt\CPUSample\F3SP66\UECHOC\UECHOC.YPRP ~\Fam3pjt\CPUSample\F3SP66\UECHOS\UECHOS.YPRP
TCP/IP echo server	Project	~\Fam3pjt\CPUSample\F3SP66\TECHOC\TECHOC.YPJT ~\Fam3pjt\CPUSample\F3SP66\TECHOS\TECHOS.YPJT
	CPU Properties	~\Fam3pjt\CPUSample\F3SP66\TECHOC\TECHOC.YPRP ~\Fam3pjt\CPUSample\F3SP66\TECHOS\TECHOS.YPRP

Note: "~" in the "Location" column denotes the folder where WideField2 is installed.

2.7.1 UDP/IP Echo Server

■ Function and Usage

This is a sample program for a UDP/IP echo server. It consists of two components, namely the client and the echo server. Two F3SP66-4S modules and one hub for building an Ethernet network are required for running the program.

The client sends 2048 (the size can be customized by modifying the constant definition) bytes of data to the echo server, and then receives a reply from the echo server.

The echo server returns the 2048 bytes of data received from the client back to the client with no modification.

There are two ways to specify a destination in a UDPSND instruction -- either by referring to an address setting number in CPU properties or by directly specifying an address in the instruction parameter table. This sample program uses the latter approach.

■ Structure of Sample Program

● List of Instructions Used

The table below shows the main ladder instructions used in the sample program.

Table 2.7.2   List of Socket Instructions Used (for client)

Ladder Instruction Mnemonic	Purpose
UDPOPEN	Opens a UDP socket to be used by the client.
UDPCLOSE	Closes the UDP socket no longer needed by the client.
UDPSND	Sends data to the echo server.
UDPRCV	Receives data from the echo server.

Table 2.7.3   List of Socket Instructions Used (for echo server)

Ladder Instruction Mnemonic	Purpose
UDPOPEN	Opens a UDP socket to be used by the echo server.
UDPCLOSE	Closes the UDP socket no longer needed by the echo server.
UDPSND	Sends data to the client.
UDPRCV	Receives data from the client.

● List of Special Relays Used

The table below lists the main special relays used in the sample program.

Table 2.7.4   List of Special Relays Used (for client, echo server)

Name of Special Relay	No. of Special Relay	Function
No Unused UDP Socket	M1028	Checks for an unused UDP socket.
Socket Busy	M1121 to M1136	Checks that the socket is not busy.
Socket Sending	M1073 to M1088	Checks that the socket is not sending data.
Socket Receiving	M1089 to M1104	Checks that the socket is not receiving data.
Always On	M0033	Used in Always On circuit.
1 Scan ON at Program Start	M0035	Turns on for one scan after program starts execution
US1 LED Lit	M0125	Turns on US1 LED
US2 LED Lit	M0127	Turns on US2 LED

● **Project**

The table below shows the content of the WideField2 project containing the sample program.

Table 2.7.5   Project Content (for Client)

Name	Component	Description	
UECHOC	Configuration	SP66 configuration with default setup. You can also use a F3SP67-6S provided you change the CPU type in the configuration.	
	Blocks	Total of blocks	1
		Block 1	MAIN (client)
	Macros	Total number of macros	1
		Macro 1	DTMAKE (macro for creating send data)
	Constant definition	#MYPORT	My port number
		#TGT_IP	IP address of destination
		#TGT_PT	Port number of destination
		#SNDSIZE	Size of send data
		#MAXSIZE	Maximum packet size for receiving
		Others, 7 definitions in total	

Table 2.7.6   Project Content (for Echo Server)

Name	Component	Description	
UECHOS	Configuration	SP66 configuration with default setup. You can also use a F3SP67-6S provided you change the CPU type in the configuration.	
	Blocks	Total number of blocks	1
		Block 1	MAIN (accepts connection request from client)
	Macros	Total number of macros	0
	Constant definition	#MYPORT	My port number
		#MAXSIZE	Max. size for receiving
		#E_5000	Error code (connection error)
		#TOUT_A	Instruction timeout interval
		#TOUT_B	Instruction timeout interval

● **CPU Properties**

The table below shows the content of the CPU property file of the sample program.

You can run the sample program with the default values but you may need to modify some property values to match the user environment.

Table 2.7.7   CPU Properties (for Client)

File Name	Required Setup for Execution of Sample Program	
UECHOC.YPRP	Ethernet setup	Specify the IP address and subnet mask to match the network environment. If you are configuring a local network for the sample program, you can run the sample program using the default values. The sample program uses the following default values: - ETHER_MY_IPADDRESS = 192.168.0.2 - ETHER_SUBNET_MASK = 255.255.255.0
	Socket setup	You can run the sample program using the default values.
	Socket address setup	No setup is required because the sample program specifies the destination directly in the instruction parameter table.

Table 2.7.8   CPU Properties (for Echo Server)

File Name	Required Setup for Execution of Sample Program	
UECHOS.YPRP	Ethernet setup	Specify the IP address and subnet mask to match the network environment. If you are configuring a local network for the sample program, you can run the sample program using the default values. The sample program uses the following default values: - ETHER_MY_IPADDRESS = 192.168.0.3 - ETHER_SUBNET_MASK = 255.255.255.0

● **Files**

This sample program uses no data file.

◼ Ladder Program Listing

The figure on the following pages shows the ladder program listing. For details on the purpose of individual devices used in the ladder program, see the I/O comments of the block tag name definition.

The macro listing is omitted as the macro is not directly related to socket communications.

● Project (UECHOC) Block (MAIN)

```
00001  Sample Program for Socket Communications – UDP/IP Echo Server (Client)
00002  Function: Sends data of specified size to node of specified socket addr
00003  setting no., and subsequently receives an echo reply.
00004  (1) Initialize devices
00005  M00035                                                    L
       1 scan ON                                          MOV        1   I00001
                                                                         Beginning
                                                                         of sample
                                                                         program
00006                                                      L
                                                          MOV        0   I00034
                                                                         UDPOPEN
                                                                         failed
00007                                                  RSET       0  B00001   1024
                                                                         Buffer for
                                                                         sending
00008                                                  RSET       0  B01025   1024
                                                                         Buffer for
                                                                         receiving
00009                                                  RSET       0  D00001    128
                                                                         Socket no.
                                                                         (UDPOPEN
                                                                         status)

00010  (2) Open UDP/IP socket
00011  I00001      M01028                                     ↑              Check for unused socket
       Beginning   No Unused                              SET        I00002
       of sample   UDP Socket                                        Execute
       program                                                       UDPOPEN
00012  I00002      C                                                 I00003   [UDPOPEN]
       Execute    UDPOPEN  D00001  #TOUT_A  #MYPORT                   UDPOPEN
       UDPOPEN             Socket no.                                 execution
                          (UDPOPEN                                    completed
                          status)
00013  I00003                                             RST        I00001   Check status
       UDPOPEN                                                       Beginning
       execution                                                     of sample
       completed                                                     program
00014                                                      RST        I00002
                                                                     Execute
                                                                     UDPOPEN
00015       D00001      >=          0                      SET        I00008   -> OK
            Socket no.                                               UDPSND
            (UDPOPEN                                                 resource
            status)                                                 check
00016       D00001      <           0                      SET        I00034   -> NG
            Socket no.                                               UDPOPEN
            (UDPOPEN                                                 failed
            status)

00017  (3) Send
00018  I00008      M01121    M01073                        SET        I00009   <res. chk>Busy/sending
       UDPSND    ↑V001       ↑V001                                   Execute
       resource  Socket Busy  Socket                                 UDPSND
       check                  Sending
00019  I00009                                          MOV  #TOUT_B   D00002   Set parameter
       Execute                                                       UDPSND:
       UDPSND                                                        Timeout
                                                                     interval
00020                                                  MOV  D00001    D00003
                                                            Socket no.  UDPSND:
                                                            (UDPOPEN    Socket no.
                                                            status)
00021                                                  MOV  #SNDSIZE  D00004
                                                                     UDPSND:
                                                                     Size of send
                                                                     data
00022                                                  MOV       -1   D00005   Direct designation
                                                                     UDPSND:
                                                                     Destination
00023                                                  L
                                                       MOV  #TGT_IP   D00006   Destination IP address
                                                                     UDPSND:
                                                                     Destination
                                                                     IP addr. low
00024                                                  MOV  #TGT_PT   D00008   Destination port no.
                                                                     UDPSND:
                                                                     Destination
                                                                     port no.
00025                                           ↑M
                                           DTMAKE   D00031   B00001   #SNDSIZE  Create send data
                                                    Seed for  Buffer for
                                                    creating  sending
                                                    data
00026                                                      ↑
                                                          INC       D00031
                                                                     Seed for
                                                                     creating
                                                                     data
00027  I00009      C                                                 I00010   [UDPSND]
       Execute    UDPSND  D00009   D00002   B00001                   UDPSND
       UDPSND             UDPSND   Timeout   sending                 execution
                          status   interval                         completed
00028  I00010                                             RST        I00008   Check status
       UDPSND                                                       UDPSND
       execution                                                    resource
       completed                                                    check
00029                                                      RST        I00009
                                                                     Execute
                                                                     UDPSND
00030       #SNDSIZE     =         D00009                  SET        I00011   -> OK
                                   UDPSND                            UDPRCV
                                   status                            resource
                                                                    check
00031       #SNDSIZE     <>        D00009                  SET        I00035   -> NG
                                   UDPSND                            UDPSND
                                   status                            failed
```

F0211.VSD

Figure 2.7.1   UDP/IP Sample Program Listing: Client MAIN (1/2)

Figure 2.7.2   UDP/IP Sample Program Listing: Client MAIN (2/2)

● **Project (UECHOS) Block (MAIN)**

```
00001  Sample Program for Socket Communications – UDP/IP Echo Server (Server)
00002  Function: Returns data received from client without modification
00003  (1) Initialize devices
00004   M00035                                                    L  MOV        1      I00001
        1 scan ON                                                                      Beginning
                                                                                       of sample
                                                                                       program

00005                                                             L  MOV        0      I00034
                                                                                       UDPOPEN
                                                                                       failed

00006                                                   RSET         0    B00001       1024
                                                                                       Buffer for
                                                                                       sending
                                                                                       and receiv...

00007                                                   RSET         0    D00001       128
                                                                                       Socket no.
                                                                                       (UDPOPEN
                                                                                       status)

00008  (2) Open UDP/IP socket
00009   I00001   M01028                                           ↑   SET        I00002    Check for unused socket
        Beginning  No Unused                                                       Execute
        of sample  UDP Socket                                                      UDPOPEN
        program

00010   I00002   C  UDPOPEN   D00001   #TOUT_A   #MYPORT                            I00003    [UDPOPEN]
        Execute              Socket no.                                            UDPOPEN
        UDPOPEN              (UDPOPEN                                               execution
                            status)                                                completed

00011   I00003                                                        RST       I00001    Check status
        UDPOPEN                                                                  Beginning
        execution                                                               of sample
        completed                                                               program

00012                                                                 RST       I00002
                                                                                Execute
                                                                                UDPOPEN

00013      D00001      >=         0                                   SET       I00008    ->OK
        Socket no.                                                              UDPRCV
        (UDPOPEN                                                                resource
        status)                                                                 check

00014      D00001      <          0                                   SET       I00034    ->NG
        Socket no.                                                              UDPOPEN
        (UDPOPEN                                                                failed
        status)

00015  (3) Receive
00016   I00008   M01121   M01089                                      SET       I00009    <res chk>Busy/
        UDPRCV   Socket Busy  Socket                                           Execute             Receiving
        resource         Receiving                                            UDPRCV
        check

00017   I00009                                            MOV   #TOUT_B   D00010   Set parameters
        Execute                                                          UDPRCV:
        UDPRCV                                                           Timeout
                                                                         interval

00018                                                     MOV   D00001   D00011
                                                               Socket no.  UDPRCV:
                                                               (UDPOPEN   Socket no.
                                                               status)

00019                                                     MOV   #MAXSIZE  D00012
                                                                         UDPRCV:
                                                                         Receive
                                                                         area size

00020                                                     MOV        0    D00013
                                                                         UDPRCV:
                                                                         Append
                                                                         NULL opti...

00021                                                     MOV        0    D00014
                                                                         UDPRCV:
                                                                         Buffer
                                                                         option

00022   I00009   C  UDPRCV   D00015   D00010   B00001                     I00010    [UDPRCV]
        Execute           UDPRCV    UDPRCV:   Buffer for                  UDPRCV
        UDPRCV            status (size) Timeout  sending                   execution
                                   interval  and receiv...                completed

00023   I00010                                                        RST       I00008    Check status
        UDPRCV                                                                  UDPRCV
        execution                                                              resource
        completed                                                              check

00024                                                                 RST       I00009
                                                                                Execute
                                                                                UDPRCV

00025      D00015      >=         1                                   SET       I00012    ->OK
        UDPRCV                                                                  UDPSND
        status (size)                                                          resource
                                                                               check

00026      D00015      <=         0    D00015      >     #F_5000       SET       I00011    ->Continue
        UDPRCV                     UDPRCV                                       Re-execute
        status (size)              status (size)                               UDPRCV

00027                    D00015      <=     #F_5000                   SET       I00012    ->NG
                       UDPRCV                                                   UDPCLOSE
                       status (size)                                           resource
                                                                               check

00028                                                                 SET       I00035
                                                                                UDPRCV
                                                                                failed

00029   I00011   I00010                                               RST       I00011    Re-execute UDPRCV
        Re-execute UDPRCV                                                       Re-execute
        UDPRCV   execution                                                     UDPRCV
                completed

00030                                                                 SET       I00008
                                                                                UDPRCV
                                                                                resource
                                                                                check
```

F0213.VSD

Figure 2.7.3   UDP/IP Sample Program Listing: Echo Server MAIN (1/2)

00031 **(4) Send**
00032 | I00012 M01121 M01073 ──────────────────────── SFT I00013 <res chk>Busy/Sending
UDPSND SocketBusy Socket Execute
resource Sending UDPSND
check

00033 | I00013 ──────────────────────── MOV #TOUT_B D00002 Set parameter
Execute UDPSND:
UDPSND Timeout
interval

00034 MOV D00001 D00003
Socket no. UDPSND:
(UDPOPEN Socket no.
status)

00035 MOV D00015 D00004 Received size
UDPRCV UDPSND:
status (size) Size of send
data

00036 MOV -1 D00005 Direct designation
UDPSND:
Destination

00037 L MOV D00017 D00006 Destination IP address
Sender IP UDPSND:
addr L Destination
(UDPRCV ... IP addr. low

00038 MOV D00019 D00008 Destination port no.
Sender port UDPSND:
no. Destination
(UDPRCV ... port no.

00039 | I00013 C UDPSND D00009 D00002 B00001 ──── I00014 [UDPSND]
Execute UDPSND Timeout Buffer for UDPSND
UDPSND status interval sending execution
and receiv... completed

00040 | I00014 ──────────────────────── RST I00012 Check status
UDPSND UDPSND
execution resource
completed check

00041 RST I00013
Execute
UDPSND

00042 | D00015 = D00009 ──── SFT I00015 ->OK
UDPRCV UDPSND Return to
status (size) status UDPRCV

00043 | D00015 <> D00009 #F_5000 < D00009 ──── SFT I00015 ->Continue
UDPRCV UDPSND UDPSND Return to
status (size) status status UDPRCV

00044 | #F_5000 >= D00009 ──── SFT I00017 ->NG
UDPSND UDPCLOSE
status resource
check

00045 SFT I00036
UDPSND
failed

00046 | I00015 I00014 ──────────────────────── RST I00015 Re-execute UDPRCV
Return to UDPSND Return to
UDPRCV execution UDPRCV
completed

00047 SFT I00008
UDPRCV
resource
check

00048 **(5) Close UDP/IP socket**
00049 | I00017 M01121 M01073 M01089 ──── SFT I00018 <res chk>Busy/Send/
UDPCLOSE SocketBusy Socket Socket Execute Rcv
resource Sending Receiving UPPCLOSE
check

00050 | I00018 C UDPCLOSE D00030 #TOUT_B D00001 ──── I00019 [TCPCLOSE]
Execute UDPCLOSE Socket no. UPPCLOSE
UPPCLOSE status (UDPOPEN execution
status) completed

00051 | I00019 ──────────────────────── RST I00017 Check status
UPPCLOSE UDPCLOSE
execution resource
completed check

00052 RST I00018
Execute
UPPCLOSE

00053 | D00030 = 0 ──── SFT I00020 ->OK
UDPCLOSE End of
status sample
program

00054 | D00030 < 0 ──── SFT I00038 ->NG
UDPCLOSE TCPCLOSE
status failed

00055 **(E) Display error**
00056 | I00034 ──────────────────────── M00125 Turn on US1 LED if error
UDPOPEN US1 LED Lit
failed

00057 | I00035
UDPRCV
failed

00058 | I00036
UDPSND
failed

00059 | I00038
TCPCLOSE
failed

F0214.VSD

Figure 2.7.4   UDP/IP Sample Program Listing: Echo Server MAIN (2/2)

2.7.2 TCP/IP Echo Server

■ Function and Usage

This is a sample program for a TCP/IP echo server. It consists of two projects, namely the client and the echo server. Two F3SP66-4S modules and one hub for building an Ethernet network are required for running the program.

The client sends 2048 (the size can be customized by modifying the constant definition) bytes of data to the echo server, and then receives a reply from the echo server.

The echo server returns the 2048 bytes of data received from the client back to the client with no modification. The echo server passes a processing request to a child block each time it receives a connection request from a client. It can support echo processing for up to 7 clients.

■ Structure of Sample Program

● List of Instructions Used

The table below shows the main ladder instructions used in the sample program.

Table 2.7.9   List of Socket Instructions Used (for client)

Ladder Instruction Mnemonic	Purpose
TCPOPEN	Opens a TCP socket to be used by the client.
TCPCLOSE	Closes the TCP socket no longer needed by the client.
TCPCNCT	Issues a request to connect to the echo server.
TCPSND	Sends data to the echo server.
TCPRCV	Receives data from the echo server.

Table 2.7.10   List of Socket Instructions Used (for echo server)

Ladder Instruction Mnemonic	Purpose
TCPOPEN	Opens a socket to be used by the echo server in a TCPLISN instruction to listen for connection requests from the client.
TCPCLOSE	Closes the socket used by the echo server in a TCPLISN instruction.
TCPLISN	Listens for any connection request from the client, and establishes connection if a request is received. When connection is established, a new socket ID for used in data receiving and sending with the client is returned.
TCPSND	Sends data to the client using the new socket ID returned by TCPLISN.
TCPRCV	Receives data from the client using the new socket ID returned by TCPLISN.

● List of Special Relays Used

The table below lists the main special relays used in the sample program.

Table 2.7.11   List of Special Relays Used (for client, echo server)

Name of Special Relay	No. of Special Relay	Function
No Unused TCP Socket	M1029	Checks for an unused TCP socket.
Socket Busy	M1121 to M1136	Checks that the socket is not busy.
Socket Sending	M1073 to M1088	Checks that the socket is not sending data.
Socket Receiving	M1089 to M1104	Checks that the socket is not receiving data.
Always On	M0033	Used in Always On circuit.
1 Scan ON at Program Start	M0035	Turns on for one scan after program starts execution
US1 LED Lit	M0125	Turns on US1 LED
US2 LED Lit	M0127	Turns on US2 LED

● **Project**

The table below shows the content of the WideField2 project containing the sample program.

Table 2.7.12   Project Content (for Client)

Name	Component		Description
TECHOC	Configuration		SP66 configuration with default setup. You can also F3SP67-6S provided you change the CPU type in the configuration.
	Blocks	Total number of blocks	1
		Block 1	MAIN (client)
	Macros	Total number of macros	1
		Macro 1	DTMAKE (macro for creating send data)
	Constant definition	#TGT_NO	For specifying TCPNCNT destination Socket address setting no. of CPU properties
		#SNDSIZE	Size of send data
		#N_MODE	TCPRCV buffer option Normal mode
		#A_MODE	TCPRCV buffer option Auto increment mode
		#TOUT_A	Instruction timeout interval
		Others, 7 definitions in total	

Table 2.7.13   Project Content (for Echo Server)

Name	Component		Description
TECHOS	Configuration		SP66 configuration with default setup except for local device setup. You can also F3SP67-6S provided you change the CPU type in the configuration.
	Blocks	Total number of blocks	8
		Block 1	MAIN (accepts connection requests from clients)
		Blocks 2 to 8	CHILD (child process for processing sending and receiving)
	Macros	Total number of macros	0
	Constant definition	#PORT	Port no. used by TCPLISN for listening
		#TOUT_A	Instruction timeout interval
		#TOUT_B	Instruction timeout interval
		#E_2002	Error code (Disconnected by remote node)
		#E_5000	Error code (connection error)

● **CPU Properties**

The table below shows the content of the CPU property file of the sample program. You can run the sample program with the default values but you may need to modify some property values to match the user environment.

Table 2.7.14   CPU Properties (for Client)

File Name		Required Setup for Execution of Sample Program
TECHOC.YPRP	Ethernet setup	Specify the IP address and subnet mask to match the network environment. If you are configuring a local network for the sample program, you can run the sample program using the default values. The sample program uses the following default values: - ETHER_MY_IPADDRESS = 192.168.0.2 - ETHER_SUBNET_MASK = 255.255.255.0
	Socket setup	You can run the sample program using the default values.
	Socket address setup	Specify the IP address of the echo server or the port number used by the echo server in the TCPLISN instruction. The default values are the same as the default values on the echo server end. The sample program uses the following default values: - SOCKET_PORT_1 = 1024 - SOCKET_ADR_IP_1 = 192.168.0.3

Table 2.7.15   CPU Properties (for Echo Server)

File Name		Required Setup for Execution of Sample Program
TECHOS.YPRP	Ethernet setup	Specify the IP address and subnet mask to match the network environment. If you are configuring a local network for the sample program, you can run the sample program using the default values. The sample program uses the following default values: - ETHER_MY_IPADDRESS = 192.168.0.3 - ETHER_SUBNET_MASK = 255.255.255.0

● **Files**

This sample program uses no data file.

■ Ladder Program Listing

The figure on the following pages shows the ladder program listing. For details on the purpose of individual devices used in the ladder program, see the I/O comments of the block tag name definition.

The macro listing is omitted as the macro is not directly related to socket communications.

● **Project (TECHOC) Block (MAIN)**

Figure 2.7.5   TCP/IP Sample Program Listing: Client MAIN (1/3)

00030 **(4) Send**

00031 I00008 | M01121 | M01073   SFT   I00009   <res. chk>Busy/sending
TCPSND resource check | Socket Busy | Socket Sending   Execute TCPSND

00032 I00009   MOV   #TOUT_B   D00010   Set parameters
Execute TCPSND   0   TCPSND parameter table

00033   MOV   D00001   D00010   1
Socket no. (TCPOPEN status)   TCPSND parameter table

00034   MOV   #SNDSIZE   D00010   2
TCPSND parameter table

00035   M   DTMAKE   D00019   B00001   #SNDSIZE   Create send data
Seed for creating data   Buffer for sending

00036   INC   D00019
Seed for creating data

00037 I00009   C   TCPSND   D00009   D00010   B00001   I00010   [TCPSND]
Execute TCPSND   TCPSND status   TCPSND parameter table   Buffer for sending   TCPSND execution completed

00038 I00010   RST   I00008   Check status
TCPSND execution completed   TCPSND resource check

00039   RST   I00009
Execute TCPSND

00040 #SNDSIZE   =   D00009   SFT   I00011   ->OK
TCPSND status   TCPRCV resource check

00041 #SNDSIZE   <>   D00009   SFT   I00036   ->NG
TCPSND status   TCPSND failed

00042 **(5) Receive**

00043 I00011 | M01121 | M01089   SFT   I00012   <res chk>Busy/Receiving
TCPRCV resource check | Socket Busy | Socket Receiving   Execute TCPRCV

00044 I00012   MOV   #TOUT_C   D00014   Set parameters
Execute TCPRCV   0   TCPRCV parameter table

00045   MOV   D00001   D00014   1
Socket no. (TCPOPEN status)   TCPRCV parameter table

00046   MOV   #SNDSIZE   D00014   2
TCPRCV parameter table

00047   MOV   0   D00014   3
TCPRCV parameter table

00048 0   =   D00013   MOV   #N_MODE   D00014   Normal mode (first time)
TCPRCV status   4   TCPRCV parameter table

00049 0   <>   D00013   MOV   #A_MODE   D00014   Auto-increment mode (aft)
TCPRCV status   4   TCPRCV parameter table

00050 I00012   C   TCPRCV   D00013   D00014   B01025   I00013   [TCPRCV]
Execute TCPRCV   TCPRCV status   TCPRCV parameter table   Buffer for receiving   TCPRCV execution completed

00051 I00013   RST   I00011   Check status
TCPRCV execution completed   TCPRCV resource check

00052   RST   I00012
Execute TCPRCV

00053 D00013   <   0   SFT   I00037   ->NG
TCPRCV status   TCPRCV failed

00054 D00013   >=   0   D00013   <   #SNDSIZE   SFT   I00014   ->Continue
TCPRCV status   TCPRCV status   Continue receive

00055 D00013   =   #SNDSIZE   MOV   0   D00013   ->Complete
TCPRCV status   TCPRCV status

00056   SFT   I00017
TCPCLOSE resource check

00057 I00014 | I00013   RST   I00014
Continue receive | TCPRCV execution completed   Continue receive

00058   SFT   I00011
TCPRCV resource check

F0216.VSD

Figure 2.7.6   TCP/IP Sample Program Listing: Client MAIN (2/3)

Figure 2.7.7   TCP/IP Sample Program Listing: Client MAIN (3/3)

● **Project (TECHOS) Block (MAIN)**

```
00001 Sample Program for Socket Communications – TCP/IP Echo Server (Server)
00002 Function: Returns data received from client without modification
00003 (1) Initialize
00004 M00035                                                          MOV        1    /I00001
      1 scan ON                                                                       Check for
                                                                                      unused
                                                                                      socket
00005                                                                 MOV        0    /I00034
                                                                                      TCPOPEN
                                                                                      failed
00006                                                                 MOV        0    /I00001
                                                                                      Run child
                                                                                      process (1)
00007                                                        RSET          0  /D00001        32
                                                                              Unique
                                                                              child ID
00008                                                        RSET          0  /D00001        32
                                                                              TCPOPEN:
                                                                              timeout
                                                                              interval
00009 (2) Open socket for use by TCPLISN
00010 /I00001     M01029                                             SET        /I00002   Check for unused socket
      Check for   No Unused                                                     Execute
      unused      TCP Socket                                                    TCPOPEN
      socket
00011 /I00002   | TCPOPEN | /D00002 | #TOUT_A |                                 /I00003   [TCPOPEN]
      Execute     TCPOPEN    Socket no.                                         TCPOPEN
      TCPOPEN     status     (TCPOPEN                                           execution
                             status)                                           completed
00012 /I00003                                                        RST        /I00001   Check status
      TCPOPEN                                                                    Check for
      execution                                                                 unused
      completed                                                                 socket
00013                                                                RST        /I00002
                                                                                Execute
                                                                                TCPOPEN
00014 | /D00002 |  >=  | 0 |                                         SET        /I00004   -> OK
      Socket no.                                                                TCPLISN
      (TCPOPEN                                                                  resource
      status)                                                                  check
00015                                                    MOV   /D00002   V00001
                                                              Socket no.  Socket no.
                                                              (TCPOPEN    index (for
                                                              status)     TCPLISN)
00016 | /D00002 |  <  | 0 |                                         SET        /I00034   -> NG
      Socket no.                                                                TCPOPEN
      (TCPOPEN                                                                  failed
      status)
00017 (3) Wait for connection request from client
00018 /I00004     M01029      M01121                                SET        /I00005   <res chk> Unused/Busy
      TCPLISN     No Unused   Socket Busy                                       Execute
      resource    TCP Socket                                                   TCPLISN
      check
00019 /I00005                                               MOV     100   /D00009   Set parameters
      Execute                                                             TCPLISN:
      TCPLISN                                                             instruction
                                                                         timeout
00020                                                       MOV  /D00002  /D00009
                                                               Socket no.  TCPLISN:
                                                               (TCPOPEN    instruction
                                                               status)     timeout
00021                                                       MOV   #PORT   /D00009
                                                                          TCPLISN:
                                                                          instruction
                                                                          timeout
00022 /I00005   | TCPLISN | /D00003 | /D00009 |                                 /I00006   [TCPLISN]
      Execute     TCPLISN    TCPLISN:                                           TCPLISN
      TCPLISN     status     instruction                                       execution
                             timeout                                            completed
00023                                                                INC        /D00012   Tally listen count
                                                                                TCPLISN
                                                                                exeuction
                                                                                count (L)
00024 /I00006                                                        RST        /I00004   Check status
      TCPLISN                                                                    TCPLISN
      execution                                                                 resource
      completed                                                                 check
00025                                                                RST        /I00005
                                                                                Execute
                                                                                TCPLISN
00026 | /D00003 |  =  | 0 |                                                     /I00008   -> OK
      TCPLISN                                                                    Execute
      status                                                                    child
                                                                                creation
00027 | /D00003 | < | 0 | | /D00003 | > | #F_5000 |                 SET        /I00007   -> Continue
      TCPLISN         TCPLISN                                                   Execute
      status          status                                                   TCPLISN –
                                                                                retry
00028              | /D00003 | <= | #F_5000 |                       SET        /I00035   -> NG
                     TCPLISN                                                    TCPLISN
                     status                                                     failed
00029 /I00007     /I00006                                           RST        /I00007   Process retry
      Execute     TCPLISN                                                      Execute
      TCPLISN –   execution                                                    TCPLISN –
      retry       completed                                                     retry
00030                                                                SET        /I00004
                                                                                TCPLISN
                                                                                resource
                                                                                check
```

F0218.VSD

Figure 2.7.8   TCP/IP Sample Program Listing: Echo Server MAIN (1/2)

Figure 2.7.9   TCP/IP Sample Program Listing: Echo Server MAIN (2/2)

● **Project (TECHOS) Block (CHILD)**

F0220.VSD

Figure 2.7.10   TCP/IP Sample Program Listing: Echo Server CHILD (1/2)
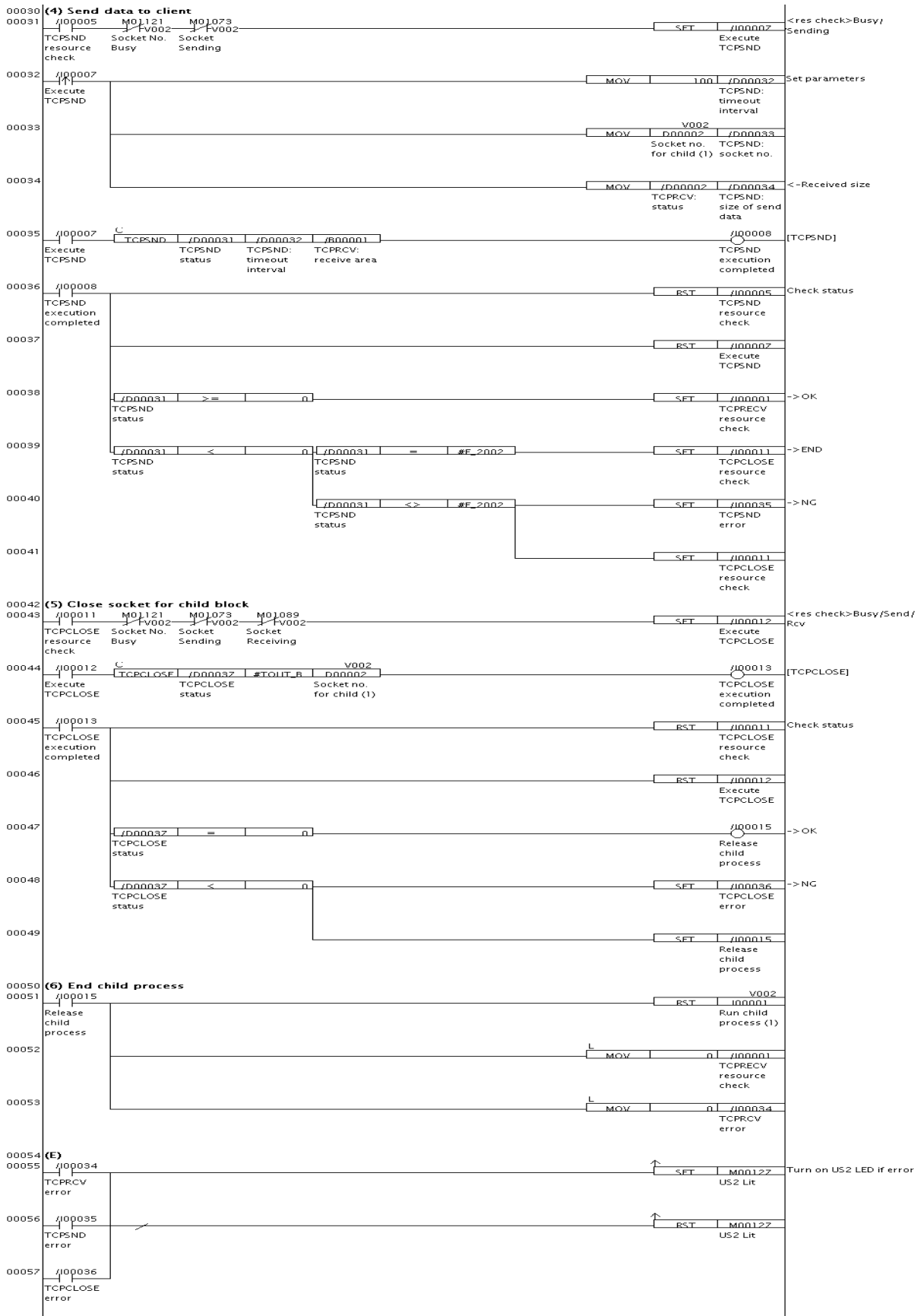
Figure 2.7.11   TCP/IP Sample Program Listing: Echo Server CHILD (2/2)

3. FTP Function

This chapter describes the FTP function.

SEE ALSO

The FTP function uses the SD memory card, RAM disk and file system. For details, see Part C, "Storage Functions" of "Sequence CPU – Functions (for F3SP66-4S, F3SP67-6S)" (IM34M6P14-01E)

3.1 Overview of FTP Function

This section gives a general overview of FTP as a standard file transfer protocol, followed by an overview of the FTP function of the CPU module.

3.1.1 Description of FTP

FTP (File Transfer Protocol) is a widely-used protocol for file transfer and disk operation. FTP is commonly used for file transfer between PCs and workstations as it is supported by many software applications and C-language libraries.

A machine running FTP acts as an FTP server and an FTP client.

- FTP Client

  An FTP client initiates a request for file transfer or disk operation. An FTP client sends an FTP command to an FTP server in the form of a request, to which the FTP server returns a response.

- FTP Server

  An FTP server accepts file transfer or disk operation requests. When an FTP client sends an FTP command as a request to the FTP server, the FTP server prepares the requested file to be sent, or performs the requested disk operation, and then sends a response to the FTP client.

FTP client

Request
Example:
Request to send file

Response
Example:
Send requested file

FTP server
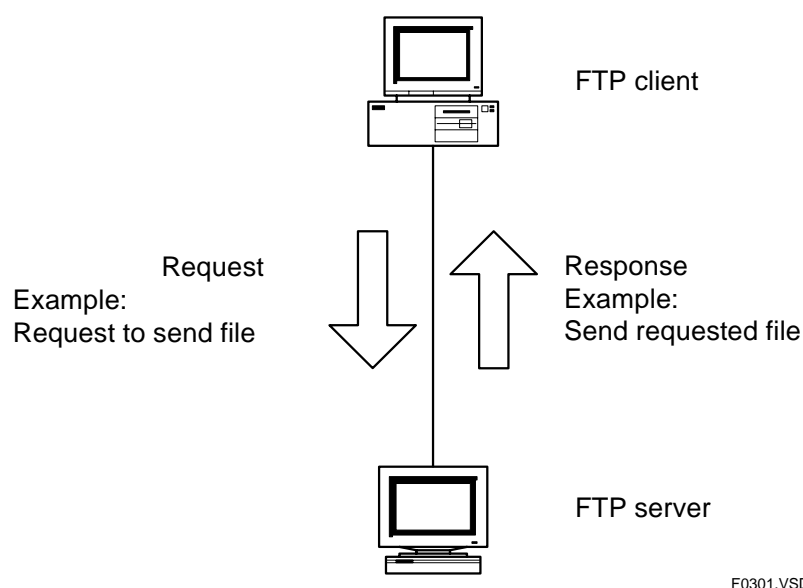
F0301.VSD

Figure 3.1.1   FTP Client and Server Connection

3.1.2   FTP Functions Supported by the Module

● **FTP client**

The CPU module supports FTP client functions. FTP commands are issued using special ladder instructions.

● **FTP server**

The CPU module supports FTP server functions. It performs file transfer or disk operations in response to requests from FTP clients such as PCs or other CPU modules.

● **Virtual directory commands**

Virtual directory commands are extended FTP server functions of the CPU Module. By coding a command as the file pathname of an FTP command, the module can be made to perform various operations. For instance, using FTP, the module can be made to automatically convert data in a transferred file and store the data to devices, or to automatically convert device data into a file and send it, or to load or save a project.

These commands are known as virtual directory commands because they are specified as file pathnames that do not actually exist on disk.
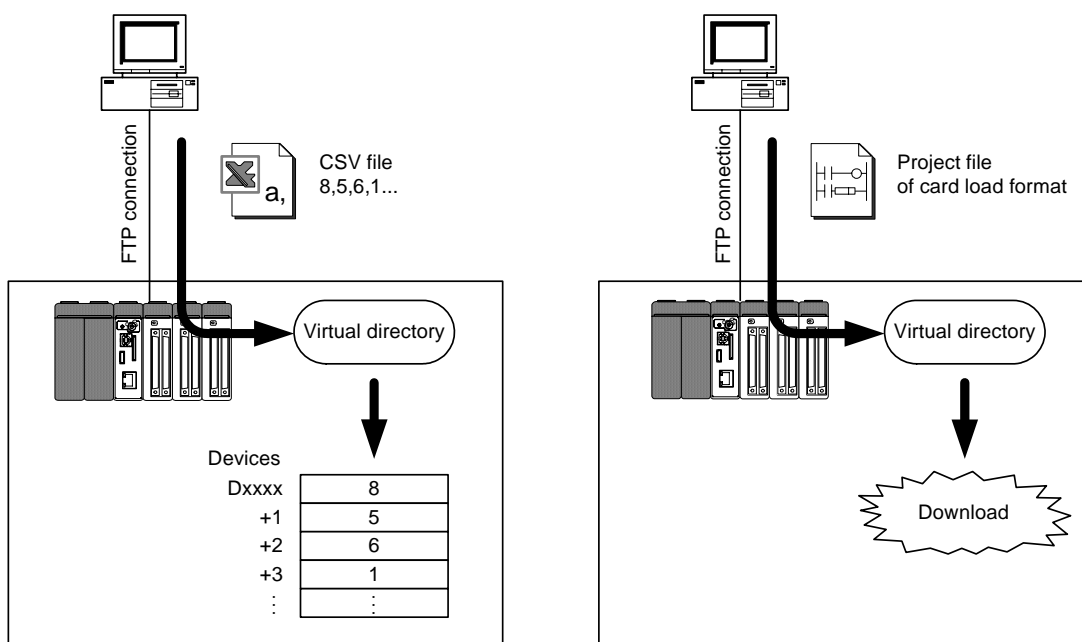
Figure 3.1.2   Concept of Virtual Directory Command

3-3

3.2 FTP Network Configurations and Access Methods

This section describes possible network configurations for using FTP. It also describes how to establish FTP client connections and specify files in FTP commands for each of the network configurations.

3.2.1 FTP Connection on Ethernet

FTP connections can be established between machines over an Ethernet network.

The table below lists the possible client-server machine combinations for FTP connection over Ethernet.

Table 3.2.1   Possible Client-Server Combinations for FTP Connection on Ethernet

FTP Client Machine	FTP Server Machine
PC	FA-M3
FA-M3	PC
FA-M3	FA-M3

Note: -   The term "PC" here refers to any machine other than the sequence CPU module, which is the subject of this manual.
-    The term "FA-M3" here refers to the sequence CPU module, which is the subject of this manual.

The term "FTP client unit" here refers to an FA-M3 unit installed with a CPU module which is running as an FTP client. The term "FTP server unit" here refers to an FA-M3 unit installed with a CPU module which is running as an FTP server.

■ FTP Connection with PC as Client and FA-M3 as Server

We describe here how to establish an FTP connection with a PC running as FTP client and an FA-M3 running as FTP server, as well as how to specify files in FTP commands in such a configuration.
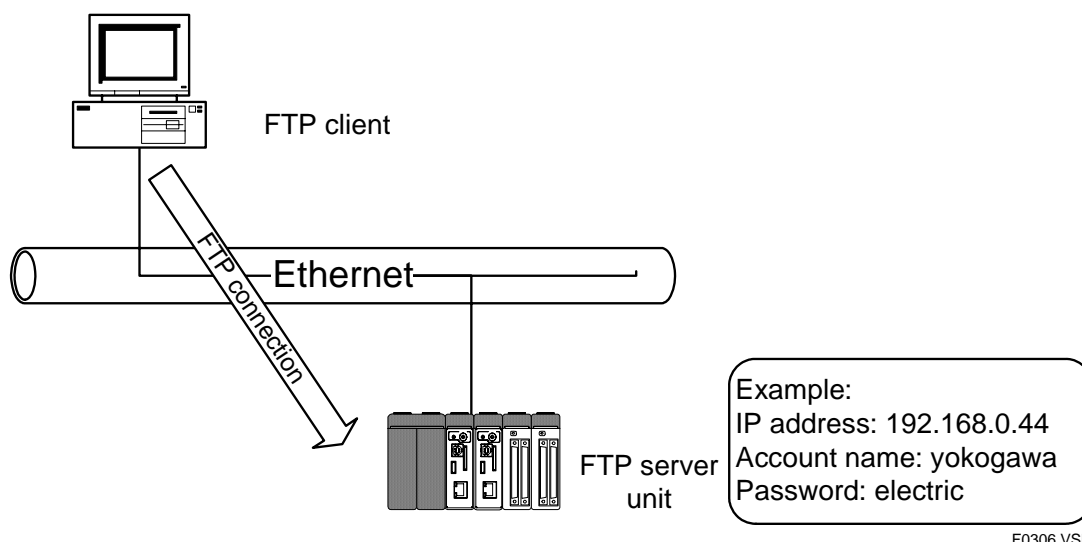

```
FTP client

FTP connection        Ethernet

FTP server
unit

Example:
IP address: 192.168.0.44
Account name: yokogawa
Password: electric

F0306.VSD
```

Figure 3.2.1   FTP Connection with PC as Client and FA-M3 as Server

● **How to establish FTP connection**

Specify a host for connection:
Specify the IP address or hostname of a CPU module, which is mounted on the FTP server unit and connected to the Ethernet. Specify the FTP server port number in the FTP client setup if the default value of 21 is not appropriate.

Example: To establish an FTP connection from a PC to an FA-M3 having IP address 192.168.0.44, enter:

```
open 192.168.0.44
```

Enter FTP account name:
Enter the FTP server account name of the CPU module, which is mounted on the FTP server unit and connected to the Ethernet.

Example: Assuming that the FTP server account name is "yokogawa", enter:

```
User: yokogawa
```

Enter FTP password:
Enter an FTP server password for the CPU module, which is mounted on the FTP server unit and connected to the Ethernet.

Example: Assuming the password is "electric", enter "electric". For security reasons, the entered password is normally masked as shown below.

```
Password: ********
```

● **How to specify files in FTP commands**

Specify the pathname of a file on the CPU module to be accessed.

■ FTP Connection with FA-M3 as Client and PC as Server

We describe here how to establish an FTP connection with an FA-M3 running as FTP client and a PC running as FTP server, as well as how to specify files in FTP commands in such a configuration.

FTP client unit

Ethernet

FTP server

Example:
IP address: 192.168.0.44
Account name: yokogawa
Password: electric

F0307.VSD
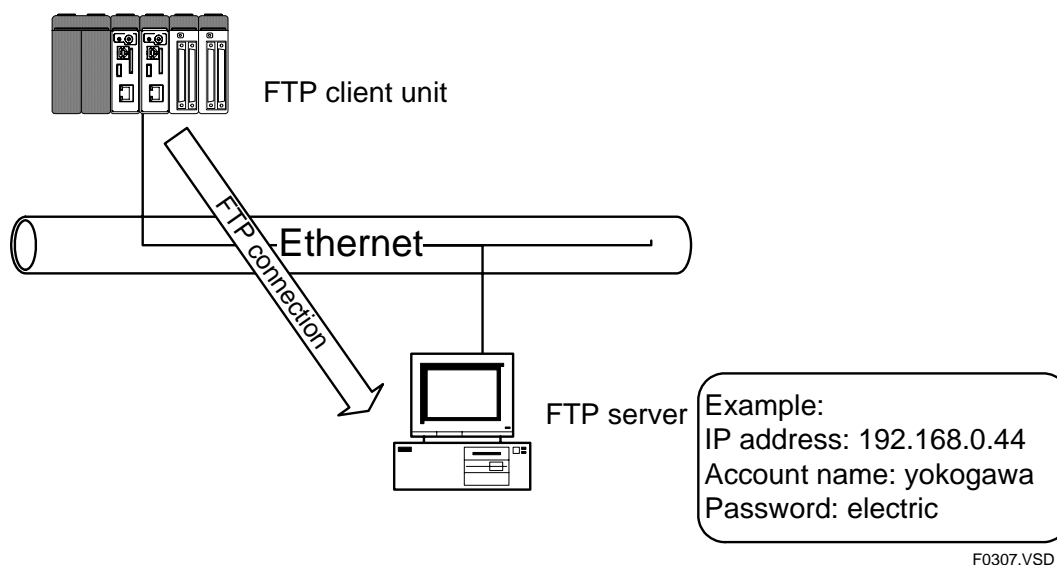
Figure 3.2.2   FTP Connection with FA-M3 as Client and PC as Server

● How to establish FTP connection

Specify a host for connection:
Specify the IP address or hostname of the PC. Specify the FTP server port number in the FTP client setup of CPU properties if the default value of 21 is not appropriate.

Example: To establish an FTP connection from an FA-M3 to a PC having IP address 192.168.0.44, specify:

```
Destination FTP server/IP address (CPU properties) = 192.168.0.44
```

Enter FTP account name:
Enter the FTP server account name of the PC.

Example: Assuming that the FTP server account name is "yokogawa", enter:

```
Destination FTP server/account (CPU properties) = yokogawa
```

Enter FTP password:
Enter the FTP server password of the PC.

Example: Assuming the password is "electric", specify:

```
Destination FTP server/password (CPU properties) = electric
```

● How to specify files in FTP commands

Specify the file pathname relative to the FTP server home directory of the PC.

■ FTP Connection with FA-M3 as Client and FA-M3 as Server

We describe here how to establish an FTP connection with an FA-M3 running as FTP client and another FA-M3 running as FTP server, as well as how to specify files in FTP commands in such a configuration.

Example:
IP address: 192.168.0.44
Account name: yokogawa
Password: electric

F0308.VSD
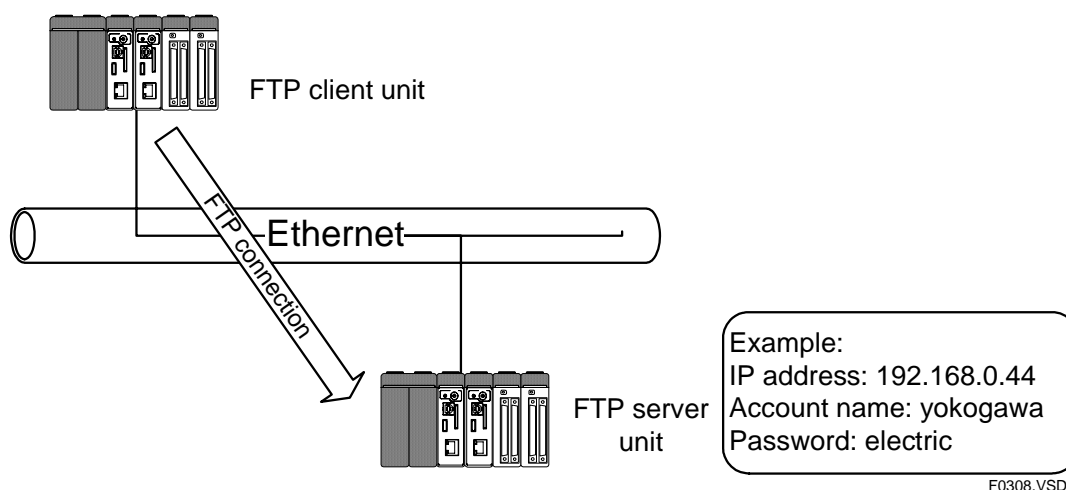
Figure 3.2.3   FTP Connection with FA-M3 as Client and FA-M3 as Server

● How to establish FTP connection

Specify a host for connection:
Specify the IP address or hostname of the FA-M3 running as FTP server. Specify the FTP server port number in the FTP client setup of CPU properties if the default value of 21 is not appropriate.

Example: To establish an FTP connection from an FA-M3 running as FTP client to another FA-M3 running as FTP server and having IP address 192.168.0.44, specify:

```
Destination   FTP   server/IP   address   (CPU   properties)   =
192.168.0.44
```

Enter FTP account name:
Enter the FTP server account name of the FA-M3 running as FTP server.

Example: Assuming that the FTP server account name is "yokogawa", specify:

```
Destination FTP server/account (CPU properties) = yokogawa
```

Enter FTP password:
Enter the FTP server password of the FA-M3 running as FTP server.

Example: Assuming the password is "electric", specify:

```
Destination FTP server/password (CPU properties) = electric
```

● How to specify files in FTP commands

Specify the pathname of a file on the CPU module to be accessed.

3.3 FTP Client

This section describes FTP client functions of the module.

3.3.1 FTP Client Specifications

■ General Specifications

Table 3.3.1   FTP Client Specifications

Item	Specification
Number of FTP clients	1
User interface	Special ladder instructions

■ Instruction List

Table 3.3.2   FTP Commands (Ladder Instructions) Supported by the Module

Ladder Instruction Name	Mnemonic	Function
FTP Client Open	FTPOPEN	Runs FTP client, and sends account name and password for connection to an FTP server.
FTP Client Quit	FTPQUIT	Disconnects from an FTP server and exits from FTP client.
FTP Client Put File	FTPPUT	Transfers a file to the FTP server.
FTP Client Put Unique File	FTPPUTU	Transfers a file to the FTP server to be stored with a unique filename determined by FTP server.
FTP Client Append File	FTPAPEND	Transfers a file to the FTP server to be appended to a specified file on the FTP server.
FTP Client Get File	FTPGET	Gets a file from the FTP server.
FTP Client Change Directory	FTPCD	Changes the remote current directory on the FTP server.
FTP Client Change Local Directory	FTPLCD	Changes the local current directory on the FTP client.
FTP Client Current Directory Info	FTPPWD	Gets information about the current directory of the FTP server.
FTP Client Get File List	FTPLS	Gets file information from the FTP server.
FTP Client Delete File	FTPDEL	Deletes one or more files on the FTP server.
FTP Client Rename File	FTPREN	Renames a file on the FTP server.
FTP Client Make Directory	FTPMKDIR	Creates a directory on the FTP server.
FTP Client Remove Directory	FTPRMDIR	Deletes a directory on the FTP server.
FTP Client Representation Type	FTPTYPE	Selects ASCII or binary representation for FTP data transfer. The initial value is binary.

■ Special Relays and Special Registers

● Special relays

The table below lists special relays related to FTP client functions.

Table 3.3.3   Special Relays (related to FTP Client Functions)

Category	FTP Client Resource Relays		
No.	Name	Function	Description
M1027	FTP Client Busy	An FTP client instruction is being executed.	This relay turns on during execution of any FTP client instruction. When the relay is ON, no other FTP client instruction can be executed. By inserting this relay in the input condition of an FTP client instruction, you can prevent inadvertent duplicate execution. This is a read-only relay. Do not write to it.

● Special Registers

There are no special registers related to FTP client functions.

3.3.2 FTP Client Setup

This subsection describes how to configure the FTP client function.

■ Basic Setup

The table below shows basic setup required for the FTP client function before use.

Table 3.3.4   Basic Setup for FTP Client Function

Name of Setup	Type of Setup	SEE ALSO[1]
Ethernet setup	CPU Properties	A9.5.2, "Ethernet Setup"
FTP client address setup	CPU Properties	"■ FTP Client Address Setup" of A9.5.5, "FTP Client Setup"

[1]: For details on individual setup items, see "Sequence CPU – Functions (for F3SP66-4S, F3SP67-6S)" (IM34M6P14-01E).

● Ethernet setup

Ethernet setup configures the CPU module for joining an Ethernet network.

- Minimally, you must specify the IP address and subnet mask. If you set the subnet mask to "0.0.0.0", the default mask for the class of the IP address is used.
- To access another network via a gateway, you must define the default gateway address.
- To access other network nodes by hostname, you must define the DNS related settings (DNS server, my hostname, domain name, domain suffixes)

● FTP client address setup

FTP client setup defines the IP address, hostname, account name (user name) and password of one or more destination FTP servers. The FTP Client Open (FTPOPEN) instruction performs connection processing using this information.

■ Optional Setup

The FTP client function may be configured as required before use.

Table 3.3.5   Optional Setup for FTP Client

Name of Setup	Type of Setup	SEE ALSO[1]
FTP client setup	CPU properties	"■ FTP Client Setup" of A9.5.5, "FTP Client Setup"

[1]: For details on individual setup items, see "Sequence CPU – Functions (for F3SP66-4S, F3SP67-6S)" (IM34M6P14-01E).

● FTP client setup

You can define the response timeout interval (FTPC_NETACK_TOUT) for TCP/IP communications, which is the layer below FTP in the network protocol. In a high-load, low-speed communications environment, internal communications timeout errors (error code: -1001) may be reported during execution of FTP client instructions. Lengthening the timeout interval may resolve the problem in such situations. This timeout interval also determines the minimum time required to detect the absence of a remote node.

3.3.3 Using FTP Client

This subsection describes how to use the FTP client function.

Instruction Used:

```
            ┌──────────┐
            │  BEGIN   │
            └──────────┘
                 │
                 ▼
        ┌──────────────────┐
        │  Run FTP client  │         FTP Client Open (FTPOPEN) instruction
        └──────────────────┘
                 │
                 ▼
          Connection to FTP    No
          server successful? ──────┐
                 │ Yes            (loop back)
                 ▼
        Beginning of loop
          (if required)
                 │
                 ▼
        ┌──────────────────┐
        │   - Put file     │         FTP Client Put File (FTPPUT) instruction
        │   - Get file     │         FTP Client Get File (FTPGET) instruction
        │ - Get file       │         FTP Client Get File List (FTPLS) instruction
        │   information    │         etc.
        │      etc.        │
        └──────────────────┘
                 │
                 ▼
          End of Loop
          (if required)
                 │
                 ▼
        ┌──────────────────┐
        │Terminate FTP     │         FTP Client Quit (FTPQUIT) instruction
        │client            │
        └──────────────────┘
                 │
                 ▼
        ┌──────────────────┐
        │Disconnect from   │
        │FTP server        │
        └──────────────────┘
                 │
                 ▼
            ┌──────────┐
            │   END    │
            └──────────┘
```

| User processing | - System processing / - External event |

F0317.VSD

Figure 3.3.1   FTP Client Procedure

■ Executing and Terminating FTP Client

● Executing FTP Client

The FTP Client Open (FTPOPEN) instruction establishes an FTP connection to the FTP server if the execution exits normally.

● Terminating FTP Client

The FTP Client Quit (FTPQUIT) instruction disconnects an FTP connection with the FTP server if the execution exits normally.

● Restrictions on FTP Client Execution

Concurrent execution of multiple FTP clients is not allowed. To connect to a different FTP server, stop the active FTP client, change the destination and start FTP client again.

■ Specifying Destination for Connection to FTP Server

Define one or more FTP server destinations using FTP Client Address setup of CPU Properties. You may then select one of these destinations by specifying its setting number as an instruction parameter of the FTP Client Open (FTPOPEN) instruction.

SEE ALSO

- For details on FTP client address setup of CPU properties, see Subsection 3.3.2, "FTP Client Setup."

■ Executing FTP Commands

FTP commands can be executed using special FTP client ladder instructions.

SEE ALSO

For details on FTP client instructions, see Section 3.4, "FTP Client Instructions."

3.4 FTP Client Instructions

This section briefly describes FTP client instructions.

3.4.1 Using FTP Client Instructions

■ Continuous type Application Instructions

All FTP client instructions are continuous type application instructions.

Continuous type application instructions perform background processing that spans multiple scans. When instruction execution is completed, the result signal (on the circuit line connected to the output end of the instruction) is held to ON for one scan period. Furthermore, a status indicating whether execution is successful is stored in a device specified as an instruction parameter.

SEE ALSO

For details on continuous type application instructions, see "■ Continuous Type Application Instructions" of Section 2.6.1, "Using Socket Instructions."

■ Resource Relays

Resource relays are special relays for preventing competition between continuous type application instructions. A resource relay indicates the release status of a resource subject to exclusive control. Resources include file IDs, socket IDs, functions and instructions.

By inserting a resource relay in the input condition of a continuous type application instruction, you can prevent errors due to resource competition. In particular, resource relays are required for checking for completion of cancellation processing or instruction timeout processing in user applications where cancellation request for a continuous type application instruction, or timeout (-1000) may occur.

● Resource Relays (related to FTP Client instructions)

Table 3.4.1   Resource Relays (related to FTP Client instructions)

Category	Continuous Type Application Instruction Resource Relays		
No.	Name	Function	Description
M1027	FTP Client Busy	FTP client instruction is executing.	Turns on during execution of any FTP client instruction. Execution of any other FTP client instruction is not allowed while this relay is ON. This relay can be inserted in the input condition of FTP client instructions to prevent inadvertent repeat executions. This is a read-only relay. Do not write to it.

■ Text Parameter

Some continuous type application instructions require text parameters to be specified in addition to the usual instruction parameters. A text parameter can be specified using the Text Parameter (TPARA) instruction.

SEE ALSO

For details on text parameters, see "■ Text Parameter" in Section 2.6.1, "Using Socket Instructions".

■ Handling of File Pathname

● Drive Name

A drive name is a disk identifier. On a Windows PC, a disk identifier is typically represented in the form of "C:\" or "D:\". The CPU module has two types of disks, namely, RAM disk and SD memory card, which are assigned the following directory names:

RAM disk   : \RAMDISK

SD memory card   : \CARD1

TIP

- The '\' prefix in a drive name indicates the "root directory."

- Each directory or file in the root directory is coded after the drive name, separated by a backslash character ('\').

● Relative pathname and absolute pathname

Both relative pathnames and absolute pathnames can be used in FTP client instructions. Relative pathnames are pathnames relative to the current directory.

● Specifying abc.txt using an absolute pathname
\RAMDISK\MYDIR\abc.txt

● Change current directory to:
\RAMDISK\MYDIR

● Specifying abc.txt using a relative pathname:
abc.txt

Specifying the same file

$(\text{Current directory}) + (\text{Relative pathname}) = (\text{Absolute pathname})$

FC0312.VSD

Figure 3.4.1   Relative Pathname and Absolute Pathname

● Current directory (local)

The local current directory here refers to the current directory on the disk on the CPU module executing the FTP client instruction.

FTP client instructions use a common current directory value, which applies only to FTP client instructions, and is independent of the current directory of file system instructions and card batch file functions.

The current directory defaults to "\RAMDISK" when FTP client is started.

To change the current directory, use the FTP Client Change Local Directory (FTPLCD) instruction.

You may not specify a directory below pathname "\VIRTUAL".

TIP

Deleting or moving a directory designated as the current directory does not generate an error. However, you should redefine the current directory in this case.

● **Current directory (remote)**

The remote current directory here refers to the current directory on the disk of the connected FTP server.

The management of the current directory on the FTP server follows the specifications of the FTP server. To change the current directory on the FTP server, use the FTP Client Change Directory (FTPCD) instruction.

SEE ALSO

For details on the handling of the current directory by the CPU module when it is running as an FTP server, see "● Current Directory" of "■ Handling of File Pathname" of Subsection 3.6.3, "Using FTP Server."

● **Root directory**

The root directory is the directory at the top of the file hierarchy. It is represented by pathname "\". A user may change directory to the root directory, display the current directory, and get file information about the root directory. However, creating a file or directory in the root directory is not allowed.

If you execute the Get File List command for the root directory, the drive name of the module is displayed.

3.4.2 List of FTP Client Instructions

Table 3.4.2   FTP Commands Supported by the CPU Module Running as an FTP Client

Ladder Instruction Name	Mnemonic	Function
FTP Client Open	FTPOPEN	Runs FTP client, and sends account name and password for connection to an FTP server.
FTP Client Quit	FTPQUIT	Disconnects from an FTP server and exits from FTP client.
FTP Client Put File	FTPPUT	Transfers a file to the FTP server.
FTP Client Put Unique File	FTPPUTU	Transfers a file to the FTP server to be stored with a unique filename determined by FTP server.
FTP Client Append File	FTPAPEND	Transfers a file to the FTP server to be appended to a specified file on the FTP server.
FTP Client Get File	FTPGET	Gets a file from the FTP server.
FTP Client Change Directory	FTPCD	Changes the remote current directory on the FTP server.
FTP Client Change Local Directory	FTPLCD	Changes the local current directory on the FTP client.
FTP Client Current Directory Info	FTPPWD	Gets information about the current directory of the FTP server.
FTP Client Get File List	FTPLS	Gets file information from the FTP server.
FTP Client Delete File	FTPDEL	Deletes one or more files on the FTP server.
FTP Client Rename File	FTPREN	Renames a file on the FTP server.
FTP Client Make Directory	FTPMKDIR	Creates a directory on the FTP server.
FTP Client Remove Directory	FTPRMDIR	Deletes a directory on the FTP server.
FTP Client Representation Type	FTPTYPE	Selects ASCII or binary representation for FTP data transfer. The initial value is binary.

3.5 FTP Client Instruction Specifications

This section describes the specifications of FTP client instructions.

3.5.1 FTP Client Open (FTPOPEN)

Runs FTP client and connects to an FTP server.

Table 3.5.1 FTP Client Open

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Processing Unit	Carry
					Yes	No			
Continuous type application instruction	–	FTP Client Open	FTPOPEN	C ─┤ FTPOPEN ☐☐☐ ├─	✓	–	6	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see "■ Continuous Type Application Instructions" in Section 2.6.1, "Using Socket Instructions".

■ Parameter

FTP Client Open ─┤ C
FTPOPEN │ ret │ n1 │ n2 ├─

Table 3.5.2 Parameters

Parameter	Description
ret[*1]	Device for storing return status (W)
n1	Timeout interval (W) [1-32767(×100 ms), 0 = longest(2147483647 ms)]
n2	FTP client address setting no. (w)[1-4][*2]

*1:   ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".
*2:   Do not specify 0.0.0.0 for the Destination IP address in FTP client address setup.

■ Status (Return Value)

Table 3.5.3 Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit
		< 0	Error status

SEE ALSO

For more details on error status, see "● Error Status of Continuous Type Application Instructions" of "■ Continuous Type Application Instructions" in Section 2.6.1, "Using Socket Instructions".

■ Available Devices

Table 3.5.4   Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	W	Z	R	V	Con-stant	Index Modification	Indirect Designation, Pointer P
ret									✓	✓	✓		✓	✓		Yes	Yes
n1									✓	✓	✓		✓	✓	✓	Yes	Yes
n2									✓	✓	✓		✓	✓	✓	Yes	Yes

Note: See Section 1.15, "Restrictions on Devices Used as Instruction Parameters" of "Sequence CPU – Instructions" (IM34M6P12-03E)

■ Resource Relays

Table 3.5.5   Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

| Add to Input Condition	Number	Name	Usage
✓	M1027	FTP Client Busy	Execute the instruction only if the FTP Client Busy relay is OFF.

■ Function

Runs FTP client and connects to an FTP server. If connection is successful, the FTP client is ready to send and receive files. The FTP server must also be running at the destination.

You can select the destination FTP server by specifying a setting number (1-4) of FTP client address setup for instruction parameter n2. In the FTP client address setup, specify one or more FTP server destinations (IP address or hostname), along with port number, account name and password.

SEE ALSO

For details on FTP client address setup, see "FTP Client Address Setup" of Subsection A9.5.5, "FTP Client Setup" of "Sequence CPU – Functions (for F3SP66-4S, F3SP67-6S)" (IM34M6P14-01E)

To change the destination when an FTP client is running, terminate the FTP client using the FTP Client Quit (FTPQUIT) instruction and re-execute the FTPOPEN instruction.

The port number used by the FTP client itself is automatically assigned by the system.

⚠ **CAUTION**

Only one FTP client service can be running on a CPU module at any one time.

■ Programming Example
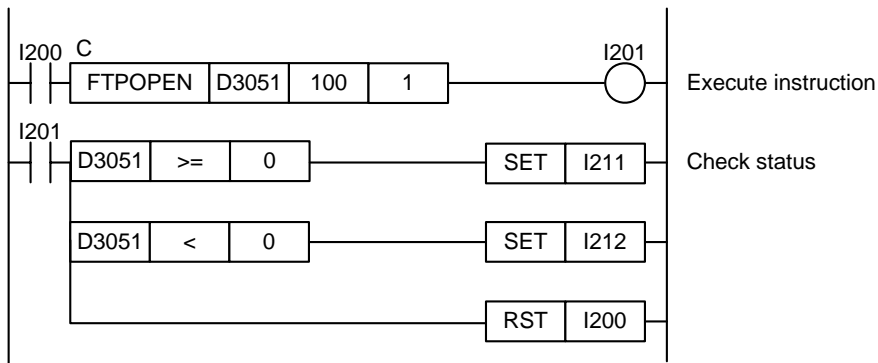
Figure 3.5.1 Example of an FTP Client Open Program

This sample code connects to the FTP server designated by FTP client address setting number 1. The timeout interval is set to 100 (10 s).

The table below shows the returned status data (ret), assuming normal exit.

Device	Value	Table Parameter
ret = D3051	0	Status

3.5.2   FTP Client Quit (FTPQUIT)

Disconnects an FTP client (CPU module) started by the FTP Client Open (FTPOPEN) instruction from its connected FTP server, and terminates the FTP client service.

Table 3.5.6   FTP Client Quit

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Processing Unit	Carry
					Yes	No			
Continuous type application instruction	–	FTP Client Quit	FTPQUIT	C ⊢ FTPQUIT	✓	–	5	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see "■ Continuous Type Application Instructions" in Section 2.6.1, "Using Socket Instructions".

■ Parameter

FTP Client Quit   C ⊢ FTPQUIT | ret | n ⊢

Table 3.5.7   Parameters

Parameter	Description
ret[*1]	Device for storing return status (W)
n	Timeout interval (W) [1-32767(×100 ms), 0 = longest(2147483647 ms)]

*1:   ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

■ Status (Return Value)

Table 3.5.8   Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit
		< 0	Error status

SEE ALSO

For more details on error status, see "● Error Status of Continuous Type Application Instructions" of "■ Continuous Type Application Instructions" in Section 2.6.1, "Using Socket Instructions".

■ Available Devices

Table 3.5.9   Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	W	Z	R	V	Constant	Index Modification	Indirect Designation, Pointer P
ret									✓	✓	✓		✓	✓		Yes	Yes
n									✓	✓	✓		✓	✓	✓	Yes	Yes

Note: See Section 1.15, "Restrictions on Devices Used as Instruction Parameters" of "Sequence CPU – Instructions" (IM34M6P12-03E)

■ Resource Relays

Table 3.5.10   Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
✓	M1027	FTP Client Busy	Execute the instruction only if the FTP Client Busy relay is OFF.

■ Function

Disconnects an FTP client (CPU module) started by the FTP Client Open (FTPOPEN) instruction from its connected FTP server, and terminates the FTP client service.

⚠ CAUTION

Depending on the status of the remote node, termination may take a long time.

■ Programming Example
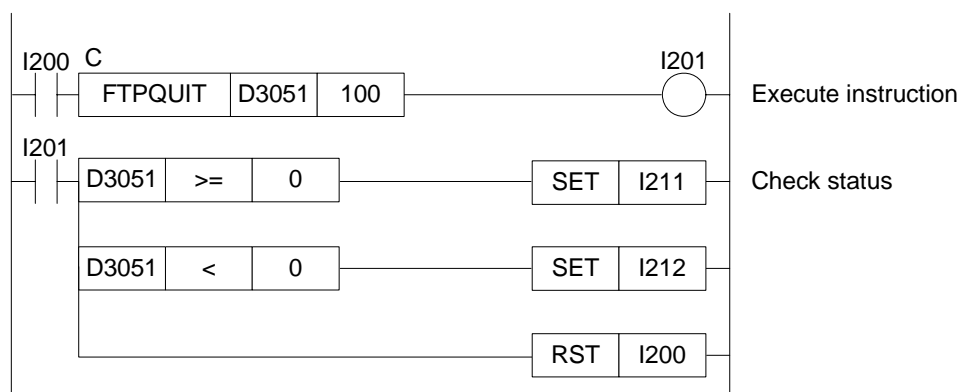
Figure 3.5.2   Example of an FTP Client Quit Program

This sample program terminates an FTP client. The timeout interval is set to 100 (10 s).

The table below shows the returned status data (ret), assuming normal exit.

Device	Value	Table Parameter
ret = D3051	0	Status

3.5.3 FTP Client Put File (FTPPUT)

Sends a file stored on the disk of the CPU module to an FTP server.

Table 3.5.11 FTP Client Put File

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Pro-cessing Unit	Carry
					Yes	No			
Continuous type application instruction	–	FTP Client Put File	FTPPUT	C ─┤ FTPPUT ├─	✓	–	5	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see "■ Continuous Type Application Instructions" in Section 2.6.1, "Using Socket Instructions".

■ Parameter

C
FTP Client Put File ─┤ FTPPUT │ ret │ n ├

Table 3.5.12 Parameters

Parameter	Description
ret[*1]	Device for storing return status (W)
n	Timeout interval (W) [1-32767(×100 ms), 0 = longest(2147483647 ms)]

*1:   ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

Table 3.5.13 Text Parameters

	Parameter	Description
1	s	Source file pathname [*2]
2	d	Destination file pathname [*1*2]

*1:   If the value is NULL, the sent data will be stored in the current directory of the FTP server with the same filename as the source filename.
*2:   If a wildcard pattern is specified for the source file pathname s, the destination file pathname d must be a directory.

SEE ALSO

Specify text parameters using the Text Parameter (TPARA) instruction. For details on text parameters and the Text Parameter (TPARA) instruction, see "■ Text Parameter" in Section C2.6.1, "Using Socket Instructions".

■ Status (Return Value)

Table 3.5.14 Status (Return Value)

Offset (word)			Description
ret	ret+0	> 0	Number of files sent (W)[1-32767]
		< 0	Error status

SEE ALSO

For more details on error status, see "● Error Status of Continuous Type Application Instructions" of "■ Continuous Type Application Instructions" in Section 2.6.1, "Using Socket Instructions".

■ Available Devices

Table 3.5.15   Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	W	Z	R	V	Con-stant	Index Modification	Indirect Designation, Pointer P
ret									✓	✓	✓		✓	✓		Yes	Yes
n									✓	✓	✓		✓	✓	✓	Yes	Yes

Note: See Section 1.15, "Restrictions on Devices Used as Instruction Parameters" of "Sequence CPU – Instructions" (IM34M6P12-03E)

■ Resource Relays

Table 3.5.16   Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
✓	M1027	FTP Client Busy	Execute the instruction only if the FTP Client Busy relay is OFF.

■ Function

Sends a file stored on the disk of the CPU module to the FTP server.

Multiple files can be sent by including wildcard characters ('*', '?') in the file name. In such situations, even if an error occurs at the FTP server end during file transfer, processing of un-transferred files continues.

At the end of transfer, the number of transferred files is returned and stored in Status.

⚠ CAUTION

This instruction cannot be executed concurrently with other FTP client instructions.

■ Programming Example
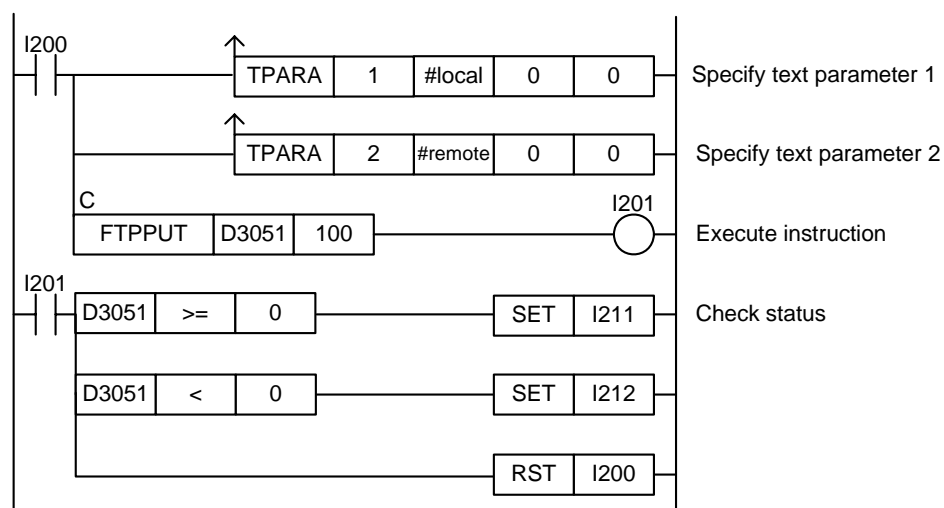
Figure 3.5.3   Example of an FTP Client Put File Program

This sample code transfers a file on the FTP client with file pathname defined by constant name "#local" to the FTP server file pathname defined by constant name "#remote". The timeout interval is set to 100 (10 s).

The table below shows the returned status data (ret), assuming normal exit.

Device	Value	Table Parameter
ret = D3051	1	Status

3.5.4   FTP Client Put Unique File (FTPPUTU)

Sends a file on the module disk to the FTP server to be stored with a unique filename.

Table 3.5.17   FTP Client Put Unique File

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Processing Unit	Carry
					Yes	No			
Continuous type application instruction	–	FTP Client Put Unique File	FTPPUTU	C ⎕FTPPUTU⎕⎕⎕⎕	✓	–	6	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see "■ Continuous Type Application Instructions" in Section 2.6.1, "Using Socket Instructions".

■ Parameter

FTP Client Put Unique File — C ⎕FTPPUTU⎕ ret ⎕ n1 ⎕ n2 ⎕

Table 3.5.18   Parameters

Parameter	Description
ret[1]	Device for storing return status (W)
n1	Timeout interval (W) [1-32767(×100 ms), 0 = longest(2147483647 ms)]

n2	Filename return option (W) [
	    0 = Filename is not returned.
	    1 = Filename is returned.
	]

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

Table 3.5.19   Text Parameters

	Parameter	Description
1	s1	Source file pathname
2	s2	(Reserved)[1]

*1: Always specify NULL for this system-reserved parameter.

SEE ALSO

Specify text parameters using the Text Parameter (TPARA) instruction. For details on text parameters and the Text Parameter (TPARA) instruction, see "■ Text Parameter" in Section 2.6.1, "Using Socket Instructions".

■ Status (Return Value)

Table 3.5.20   Status (Return Value)

Offset (word)		Description	
ret	ret+0	> 0	Number of files sent (W)[1]
		< 0	Error status
	ret+1 -17	Destination file name determined by FTP server (0-32 characters) Appended with a trailing NULL character.	

SEE ALSO

For more details on error status, see "● Error Status of Continuous Type Application Instructions" of "■ Continuous Type Application Instructions" in Section 2.6.1, "Using Socket Instructions".

■ Available Devices

Table 3.5.21 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	W	Z	R	V	Con-stant	Index Modification	Indirect Designation, Pointer P
ret									✓	✓	✓		✓	✓		Yes	Yes
n1									✓	✓	✓		✓	✓	✓	Yes	Yes
n2									✓	✓	✓		✓	✓	✓	Yes	Yes

Note: See Section 1.15, "Restrictions on Devices Used as Instruction Parameters" of "Sequence CPU – Instructions" (IM34M6P12-03E)

■ Resource Relays

Table 3.5.22 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
✓	M1027	FTP Client Busy	Execute the instruction only if the FTP Client Busy relay is OFF.

■ Function

Sends from the module to the FTP server a file, which is to be stored in the current directory of the FTP server with a unique filename automatically determined by the FTP server according to its specifications. The original filename of the sent file on the FTP client is ignored during file naming.
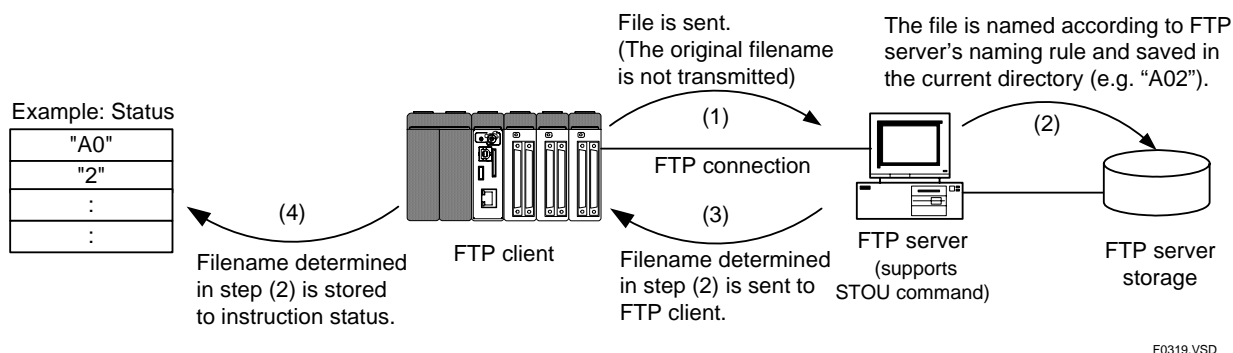
Figure 3.5.4 Sending a File to FTP Server

The destination filename on the FTP server can be returned as status data using the Filename Return Option parameter. The filename is returned without its pathname. The maximum filename length that can be returned is 32 characters and any characters exceeding the limit will be discarded.

Always specify NULL for the system-reserved s2 text parameter.

Wildcard characters must not be used with this instruction.

TIP

Filename extraction processing of the module follows the RFC1123 specifications. It will work correctly even if the reply from the FTP server does not contain the "FILE:" string. In this case, the module extracts and outputs the last word from the reply text, which therefore may include other characters preceding the filename. (The reply from the IIS of Microsoft Windows does not contain the "FILE:" string so the last word will be extracted as the filename.)

> ⚠ **CAUTION**
>
> - This instruction cannot be executed concurrently with other FTP client instructions.
> - Wildcard characters must not be used with this instruction.

■ Programming Example
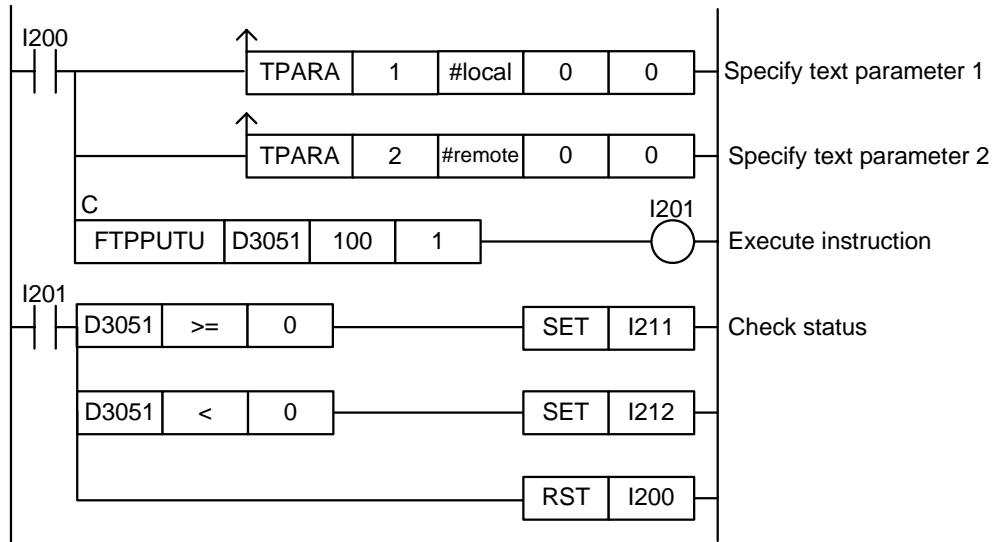
Figure 3.5.5   Example of an FTP Client Put Unique File Program

This sample code uses the FTPPUTU instruction to send a file on the FTP client with file pathname defined by constant name "#local" to be stored in the current directory of the FTP server with a unique name. The timeout interval is set to 100 (10 s); the Filename Return Option is set to 1.

```
#local = "\ramdisk\mydir\myfile.csv"
```

The table below shows the returned status, assuming normal exit and a destination filename on the FTP server of "A000004.tmp".

Device	Value	Table Parameter
ret = D3051	1	Status
D3052	"A0"	File name determined by the FTP server
D3053	"00"	(A000004.tmp)
D3054	"00"	
D3055	"4."	
D3056	"tm"	
D3057	"p"+NULL	

3.5.5　FTP Client Append File (FTPAPEND)

Sends a file on the module disk to be appended to a specified file on the FTP server.

Table 3.5.23　FTP Client Append File

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Processing Unit	Carry
					Yes	No			
Continuous type application instruction	–	FTP Client Append File	FTPAPEND	C ┤ FTPAPEND □ ├	✓	–	5	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see "■ Continuous Type Application Instructions" in Section 2.6.1, "Using Socket Instructions".

■ Parameter

FTP Client Append File ┤ C FTPAPEND | ret | n ├

Table 3.5.24　Parameters

Parameter	Description
ret[1]	Device for storing return status (W)
n	Timeout interval (W) [1-32767(×100 ms), 0 = longest(2147483647 ms)]

*1:　ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

Table 3.5.25　Text Parameters

	Parameter	Description
1	s	Source file pathname
2	d	Destination file pathname[1]

*1:　If the value is NULL, the sent data will be stored in the current directory of the FTP server with the same filename as the source filename.

SEE ALSO

Specify text parameters using the Text Parameter (TPARA) instruction. For details on text parameters and the Text Parameter (TPARA) instruction, see "■ Text Parameter" in Section 2.6.1, "Using Socket Instructions".

■ Status (Return Value)

Table 3.5.26　Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit
		< 0	Error status

SEE ALSO

For more details on error status, see "● Error Status of Continuous Type Application Instructions" of "■ Continuous Type Application Instructions" in Section 2.6.1, "Using Socket Instructions".

◼ Available Devices

Table 3.5.27   Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	W	Z	R	V	Con-stant	Index Modification	Indirect Designation, Pointer P
ret									✓	✓	✓		✓	✓		Yes	Yes
n									✓	✓	✓		✓	✓	✓	Yes	Yes

Note: See Section 1.15, "Restrictions on Devices Used as Instruction Parameters" of "Sequence CPU – Instructions" (IM34M6P12-03E)

◼ Resource Relays

Table 3.5.28   Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
✓	M1027	FTP Client Busy	Execute the instruction only if the FTP Client Busy relay is OFF.

◼ Function

Sends a file on the module disk to be appended to a specified file on the FTP server.

If the specified destination filename exists on the FTP server, the sent file is appended to the existing file. Otherwise, this instruction behaves the same way as the FTP Client Put File (FTPPUT) instruction.
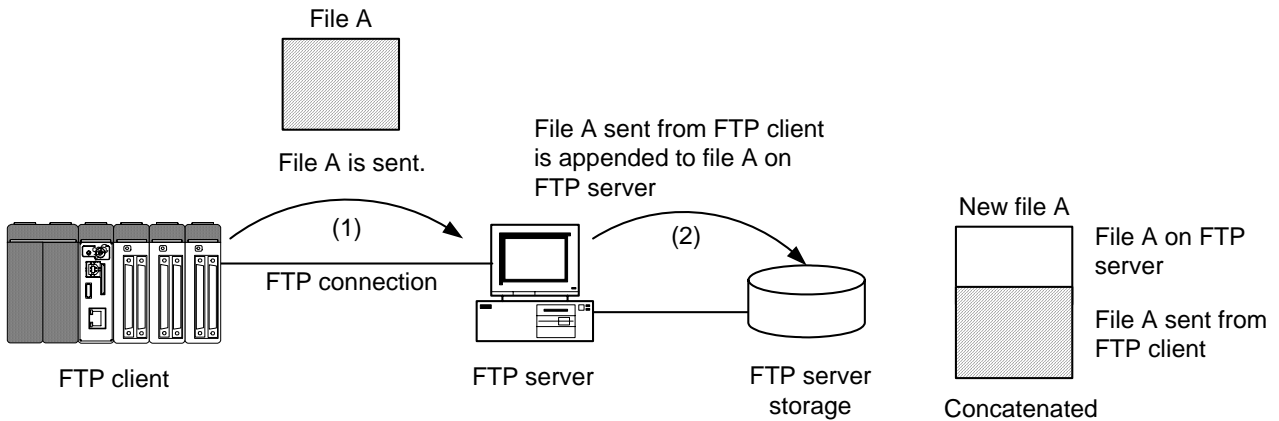
F0320.VSD

Figure 3.5.6   Appending a File to a Specified File on the FTP Server

⚠ CAUTION

- This instruction cannot be executed concurrently with other FTP client instructions.
- Wildcard characters must not be used with this instruction.

■ Programming Example
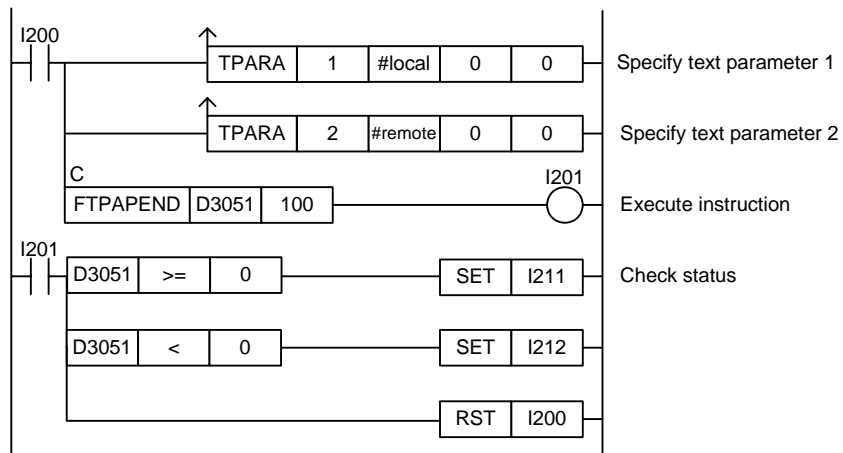
Figure 3.5.7   Example of an FTP Client Append File Program

This sample code uses the FTPAPEND instruction to send a file on the FTP client designated by constant name "#local" to be appended to a file on the FTP server designated by constant name "#remote". The timeout interval is set to 100 (10 s).
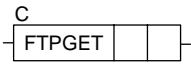
The table below shows the returned status data (ret), assuming normal exit.

Device	Value	Table Parameter
ret = D3051	0	Status

3.5.6 FTP Client Get File (FTPGET)

Gets a file from the FTP server and saves it to the disk of the module.
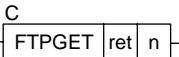
Table 3.5.29 FTP Client Get File

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Processing Unit	Carry
					Yes	No			
Continuous type application instruction	–	FTP Client Get File	FTPGET	C ⊢FTPGET [ ][ ]⊣	✓	–	5	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see "■ Continuous Type Application Instructions" in Section 2.6.1, "Using Socket Instructions".

■ Parameter

FTP Client Get File ⊢ C FTPGET | ret | n ⊣

Table 3.5.30 Parameters

Parameter	Description
ret[*1]	Device for storing return status (W)
n	Timeout interval (W) [1-32767(×100 ms), 0 = longest(2147483647 ms)]

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

Table 3.5.31 Text Parameters

	Parameter	Description
1	s	Source file pathname [*2]
2	d	Destination file pathname [*1*2]

*1: If the value is NULL, the received data will be stored in the current directory of the FTP client with the same filename as the source filename.
*2: If a wildcard pattern is specified for the source file pathname s, the destination file pathname d must be a directory.

SEE ALSO

Specify text parameters using the Text Parameter (TPARA) instruction. For details on text parameters and the Text Parameter (TPARA) instruction, see "■ Text Parameter" in Section 2.6.1, "Using Socket Instructions".

■ Status (Return Value)

Table 3.5.32 Status (Return Value)

Offset (word)			Description
ret	ret+0	> 0	Number of files received (W) [1-32767]
		< 0	Error status

SEE ALSO

For more details on error status, see "● Error Status of Continuous Type Application Instructions" of "■ Continuous Type Application Instructions" in Section 2.6.1, "Using Socket Instructions".

■ Available Devices

Table 3.5.33 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	W	Z	R	V	Con-stant	Index Modification	Indirect Designation, Pointer P
ret									✓	✓	✓		✓	✓		Yes	Yes
n									✓	✓	✓		✓	✓	✓	Yes	Yes

Note: See Section 1.15, "Restrictions on Devices Used as Instruction Parameters" of "Sequence CPU – Instructions" (IM34M6P12-03E)

■ Resource Relays

Table 3.5.34 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
✓	M1027	FTP Client Busy	Execute the instruction only if the FTP Client Busy relay is OFF.

■ Function

Gets a file from the FTP server and saves it to the disk of the module.

Multiple files can be retrieved by including wildcard characters ('*', '?') in the file name. In such situations, even if an error occurs at the FTP server end during file transfer, processing of un-transferred files continues.

At the end of transfer, the number of transferred files is returned and stored in Status.

⚠ CAUTION

This instruction cannot be executed concurrently with other FTP client instructions.

■ Programming Example
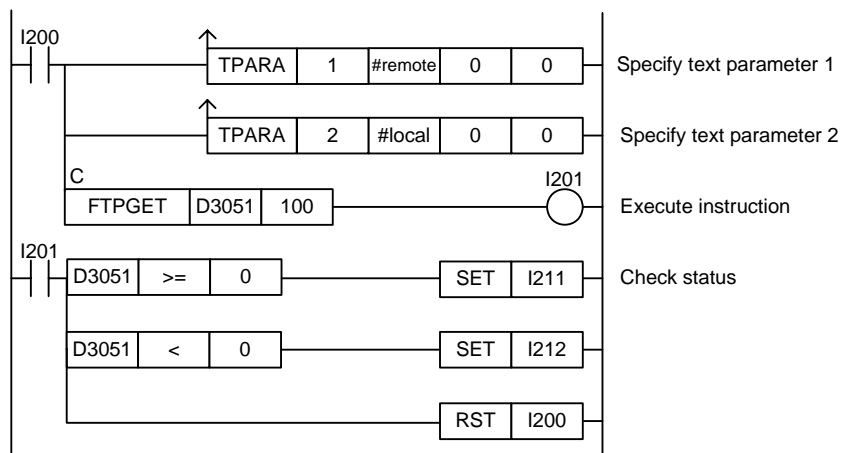
Figure 3.5.8 Example of an FTP Client Get File Program

This sample code gets a file on the FTP server with file pathname defined by constant name "#remote" and saves it to the FTP client file pathname defined by constant name "#local". The timeout interval is set to 100 (10 s).
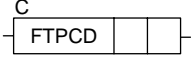
The table below shows the returned status data (ret), assuming normal exit.

Device	Value	Table Parameter
ret = D3051	1	Status

3.5.7   FTP Client Change Directory (FTPCD)

Changes the remote current directory on the FTP server.
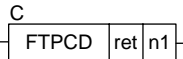
Table 3.5.35   FTP Client Change Directory

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Pro- cessing Unit	Carry
					Yes	No			
Continuous type application instruction	–	FTP Client Change Directory	FTPCD	C ─┤ FTPCD │ │ ├─	✓	–	5	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see "■ Continuous Type Application Instructions" in Section 2.6.1, "Using Socket Instructions".

■ Parameter

FTP Client Change Directory   C
─┤ FTPCD │ ret │ n1 ├─

Table 3.5.36   Parameters

Parameter	Description
ret[*1]	Device for storing return status (W)
n1	Timeout interval (W) [1-32767(×100 ms), 0 = longest(2147483647 ms)]

*1:   ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

Table 3.5.37   Text Parameters

Parameter		Description
1	n2	New current directory pathname

SEE ALSO

Specify text parameters using the Text Parameter (TPARA) instruction. For details on text parameters and the Text Parameter (TPARA) instruction, see "■ Text Parameter" in Section 2.6.1, "Using Socket Instructions".

■ Status (Return Value)

Table 3.5.38   Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit
		< 0	Error status

SEE ALSO

For more details on error status, see "● Error Status of Continuous Type Application Instructions" of "■ Continuous Type Application Instructions" in Section 2.6.1, "Using Socket Instructions".

■ Available Devices

Table 3.5.39 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	W	Z	R	V	Con-stant	Index Modification	Indirect Designation, Pointer P
ret									✓	✓	✓		✓	✓		Yes	Yes
n1									✓	✓	✓		✓	✓	✓	Yes	Yes

Note: See Section 1.15, "Restrictions on Devices Used as Instruction Parameters" of "Sequence CPU – Instructions" (IM34M6P12-03E)

■ Resource Relays

Table 3.5.40 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
✓	M1027	FTP Client Busy	Execute the instruction only if the FTP Client Busy relay is OFF.

■ Function

Changes the remote current directory on the FTP server.

⚠ CAUTION

This instruction cannot be executed concurrently with other FTP client instructions.

■ Programming Example
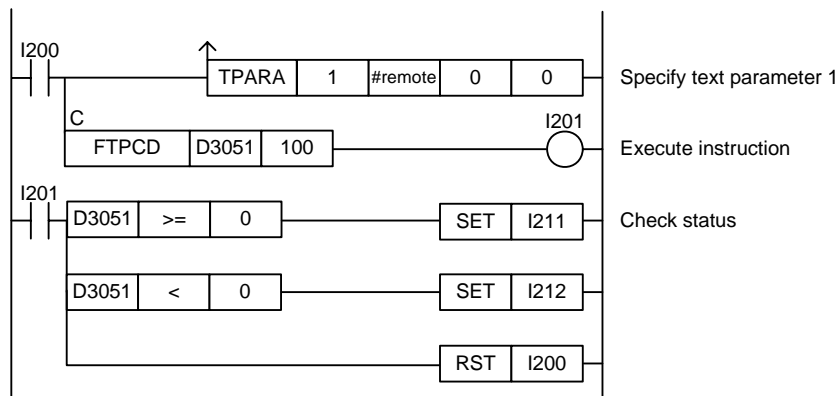
Figure 3.5.9 Example of an FTP Client Change Directory Program

This sample code changes the current directory on the FTP server to the directory defined by constant name #remote. The timeout interval is set to 100 (10 s).

The table below shows the returned status data (ret), assuming normal exit.

Device	Value	Table Parameter
ret = D3051	0	Status

3.5.8　FTP Client Change Local Directory (FTPLCD)

Changes the local current directory on the FTP client.

Table 3.5.41　FTP Client Change Local Directory

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Processing Unit	Carry
					Yes	No			
Continuous type application instruction	–	FTP Client Change Local Directory	FTPLCD	C ─┤ FTPLCD │ │ ├─	✓	–	5	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see "■ Continuous Type Application Instructions" in Section 2.6.1, "Using Socket Instructions".

■ Parameter

C
FTP Client Change Local Directory ─┤ FTPLCD │ ret │ n1 ├─
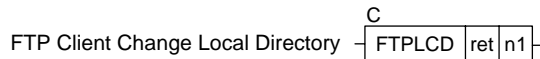
Table 3.5.42　Parameters

Parameter	Description
ret[*1]	Device for storing return status (W)
n1	Timeout interval (W) [1-32767(×100 ms), 0 = longest(2147483647 ms)]

*1:　ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

Table 3.5.43　Text Parameters

Parameter		Description
1	n2	New local current directory pathname

SEE ALSO

Specify text parameters using the Text Parameter (TPARA) instruction. For details on text parameters and the Text Parameter (TPARA) instruction, see "■ Text Parameter" in Section 2.6.1, "Using Socket Instructions".

■ Status (Return Value)

Table 3.5.44　Status (Return Value)

Offset (word)		Description
ret	0	Normal exit
	< 0	Error status

SEE ALSO

For more details on error status, see "● Error Status of Continuous Type Application Instructions" of "■ Continuous Type Application Instructions" in Section 2.6.1, "Using Socket Instructions".

■ Available Devices

Table 3.5.45   Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	W	Z	R	V	Con-stant	Index Modification	Indirect Designation, Pointer P
ret									✓	✓	✓		✓	✓		Yes	Yes
n1									✓	✓	✓		✓	✓	✓	Yes	Yes

Note: See Section 1.15, "Restrictions on Devices Used as Instruction Parameters" of "Sequence CPU – Instructions" (IM34M6P12-03E)

■ Resource Relays

Table 3.5.46   Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
✓	M1027	FTP Client Busy	Execute the instruction only if the FTP Client Busy relay is OFF.

■ Function

Changes the local current directory on the FTP client. The local current directory defaults to "\RAMDISK" when FTP client is started.

Changing the local current directory of the FTP client does not affect the current directories of other processing systems (e.g. current directory of the file system instruction group) as the current directories are independent of each other.

■ Programming Example
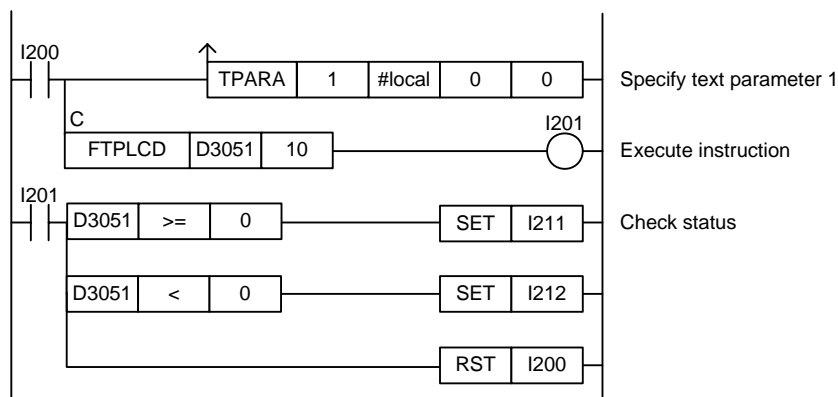
Figure 3.5.10   Example of an FTP Client Change Local Directory Program

This sample code changes the current directory on the FTP client to the directory defined by constant name #local. The timeout interval is set to 10 (1 s).

The table below shows the returned status data (ret), assuming normal exit.

Device	Value	Table Parameter
ret = D3051	0	Status

3.5.9 FTP Client Current Directory Info (FTPPWD)

Gets information about the current directory of the FTP server.

Table 3.5.47   FTP Client Current Directory Info

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Pro-cessing Unit	Carry
					Yes	No			
Continuous type application instruction	–	FTP Client Current Directory Info	FTPPWD	C ─┤ FTPPWD ⬚⬚⬚ ├─	✓	–	6	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see "■ Continuous Type Application Instructions" in Section 2.6.1, "Using Socket Instructions".

■ Parameter

FTP Client Current Directory Info ─┤ C FTPPWD | ret | t | d ├─

Table 3.5.48   Parameters

Parameter		Description
ret[1]		Device for storing return status (W)
t	t+0	Timeout interval (W) [1-32767(×100 ms), 0 = longest(2147483647 ms)]
	t+1	Max. returned words (W) [1-65]
d		Destination device (W)

*1:   ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

■ Status (Return Value)

Table 3.5.49   Status (Return Value)

Offset (word)			Description
ret	ret+0	0	Normal exit
		< 0	Error status

SEE ALSO

For more details on error status, see "● Error Status of Continuous Type Application Instructions" of "■ Continuous Type Application Instructions" in Section 2.6.1, "Using Socket Instructions".

■ Available Devices

Table 3.5.50   Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	W	Z	R	V	Con-stant	Index Modification	Indirect Designation, Pointer P
ret									✓	✓	✓		✓	✓		Yes	Yes
t									✓	✓	✓		✓	✓		Yes	Yes
d									✓	✓	✓		✓	✓		Yes	Yes

Note: See Section 1.15, "Restrictions on Devices Used as Instruction Parameters" of "Sequence CPU – Instructions" (IM34M6P12-03E)

■ Resource Relays

Table 3.5.51   Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
✓	M1027	FTP Client Busy	Execute the instruction only if the FTP Client Busy relay is OFF.

■ Function

Gets information about the current directory of the FTP server. The returned file pathname is a text string of maximum 127 bytes, with a NULL byte appended at the end. If the returned file pathname exceeds the specified maximum number of returned words, the excess bytes are discarded, and a data processing error code (-9015) is stored in status.

■ Programming Example
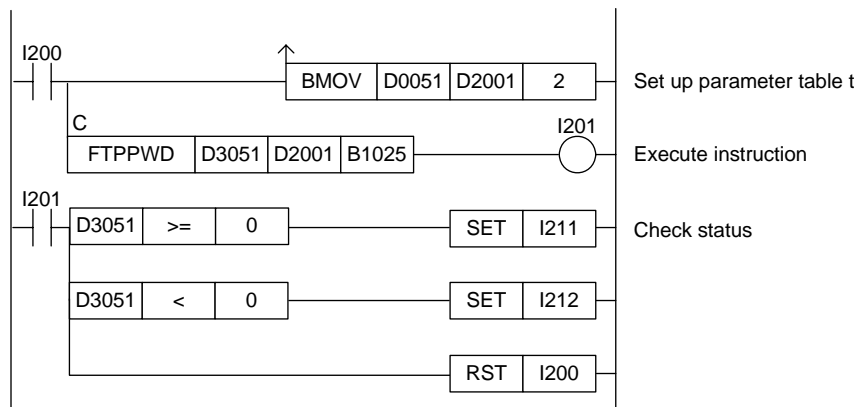
Figure 3.5.11   Example of an FTP Client Current Directory Info Program

This sample code gets information about the current directory of the connected FTP server, and stores the current directory pathname to device B1025.

It specifies ret(=D3051), t(=D2001) and d(=B1025), with t set up as follows.

Device	Value	Table Parameter
t = D2001	100	Timeout interval (= 10 s)
D2002	65	Maximum returned words (= 65 words)

The table below shows the returned status data (ret), assuming normal exit.

Device	Value	Table Parameter
ret = D3051	0	Status

The table below shows a sample output of current directory information.

Device	Value	Table Parameter
d = B1025	"C: "	Current directory information (C: \MYDATA)
B1026	"\M"	
B1027	"YD"	
B1028	"AT"	
B1029	"A"+ NULL	

3.5.10 FTP Client Get File List (FTPLS)

Gets detailed information about a specified directory or file on the FTP server.

Table 3.5.52　FTP Client Get File List

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Pro-cessing Unit	Carry
					Yes	No			
Continuous type application instruction	–	FTP Client Get File List	FTPLS	C ─| FTPLS | | |─	✓	–	5	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see "■ Continuous Type Application Instructions" in Section 2.6.1, "Using Socket Instructions".

■ Parameter

C
FTP Client Get File List ─| FTPLS | ret | n |─

Table 3.5.53　Parameters

Parameter	Description
ret [1]	Device for storing return status (W)
n	Timeout interval (W) [1-32767(×100 ms), 0 = longest(2147483647 ms)]

*1:　ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

Table 3.5.54　Text Parameters

	Parameter	Description
1	s	Target directory pathname [1]
2	d	Output file pathname
3	n2	"ls" command option [2]

*1:　Specify a NULL value to get information about the current directory.
*2:　Prefix the command option parameter value with a hyphen ('-'). For more details about the "ls" command options, see the specifications of the FTP server. If an option is specified, the target directory pathname 's' parameter is ignored and information about the current directory is returned.

SEE ALSO

Specify text parameters using the Text Parameter (TPARA) instruction. For details on text parameters and the Text Parameter (TPARA) instruction, see "■ Text Parameter" in Section 2.6.1, "Using Socket Instructions".

■ Status (Return Value)

Table 3.5.55　Status (Return Value)

Offset (word)			Description
Ret	ret+0	0	Normal exit
		< 0	Error status

SEE ALSO

For more details on error status, see "● Error Status of Continuous Type Application Instructions" of "■ Continuous Type Application Instructions" in Section 2.6.1, "Using Socket Instructions".

■ Available Devices

Table 3.5.56 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	W	Z	R	V	Con-stant	Index Modification	Indirect Designation, Pointer P
ret									✓	✓	✓		✓	✓		Yes	Yes
n									✓	✓	✓		✓	✓	✓	Yes	Yes

Note: See Section 1.15, "Restrictions on Devices Used as Instruction Parameters" of "Sequence CPU – Instructions" (IM34M6P12-03E)

■ Resource Relays

Table 3.5.57 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
✓	M1027	FTP Client Busy	Execute the instruction only if the FTP Client Busy relay is OFF.

■ Function

Gets a list of the names of files and directories contained in a FTP server directory designated by the target pathname ('s') parameter. The returned information is output in text format to a file designated by the output file pathname ('d') parameter. Internally, this instruction executes the "NLST" FTP command.

You can specify options for the "NLST" command as a parameter of this FTPLS instruction. For instance, specifying a command option of "-l" returns the file attribute, creation date and other information in addition to the file name. Beware, however, that if you specify an option, the source directory pathname ('s') parameter is ignored, and information of the current directory is always returned.

The table below lists common "NLST" options used. The FTP server function of this module supports only the "-l" option when the module is running as an FTP server.

Table 3.5.58 Examples of "NLST" command options

Option	Description
-l	Returns list output containing file size, creation date and other additional information.
-t	Returns list output sorted in descending order of date.
-tr	Returns list output sorted in ascending order of date.
-F	Appends a '/' identifier behind directory names.
-tF	Returns list output sorted in descending order of date, and appends a '/' identifier behind directory names.

Note: Supported "ls" command options vary with individual FTP server implementations so some of the options described above may be unavailable.

⚠ CAUTION

- The operation and implementation of the "NLST" options is according to the specifications of an individual FTP server.

- This instruction cannot be executed concurrently with other FTP client instructions.

■ Programming Example
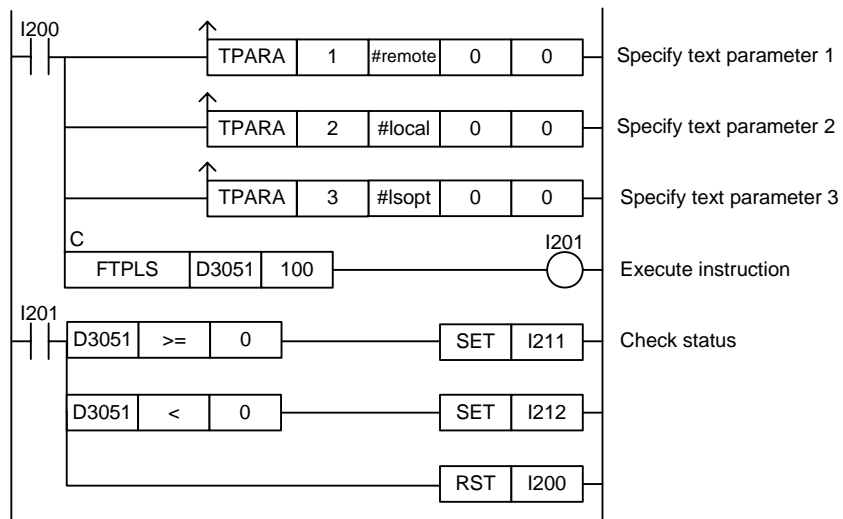
Figure 3.5.12 Example of an FTP Client Get File List Program

This sample code gets file information for the current directory of the FTP server as constant name #remote is assigned the null string. The returned information is output to the file pathname defined by constant name #local. "NLST" option string defined by constant name #lsopt is included as an instruction parameter. The timeout interval is set to 100 (10 s).

```
#remote = ""
#local = "\ramdisk\filestat.txt"
#lsopt = "-l"
```

The table below shows the returned status data (ret), assuming normal exit.

Device	Value	Table Parameter
ret = D3051	0	Status

3.5.11 FTP Client Delete File (FTPDEL)

Deletes one or more specified files on the FTP server.

Table 3.5.59 FTP Client Delete File

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required? Yes	No	Step Count	Pro-cessing Unit	Carry
Continuous type application instruction	–	FTP Client Delete File	FTPDEL	C ┤ FTPDEL ├	✓	–	5	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see "■ Continuous Type Application Instructions" in Section 2.6.1, "Using Socket Instructions".

■ Parameter

FTP Client Delete File   ┤ C FTPDEL │ ret │ n ├

Table 3.5.60 Parameters

Parameter	Description
ret[*1]	Device for storing return status (W)
n	Timeout interval (W) [1-32767(×100 ms), 0 = longest(2147483647 ms)]

*1:   ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

Table 3.5.61 Text Parameters

Parameter		Description
1	d	Target file pathname

SEE ALSO

Specify text parameters using the Text Parameter (TPARA) instruction. For details on text parameters and the Text Parameter (TPARA) instruction, see "■ Text Parameter" in Section 2.6.1, "Using Socket Instructions".

■ Status (Return Value)

Table 3.5.62 Status (Return Value)

Offset (word)			Description
ret	ret+0	> 0	Number of deleted files (W) [1-32767]
		< 0	Error status

SEE ALSO

For more details on error status, see "● Error Status of Continuous Type Application Instructions" of "■ Continuous Type Application Instructions" in Section 2.6.1, "Using Socket Instructions".

■ Available Devices

Table 3.5.63   Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	W	Z	R	V	Con-stant	Index Modification	Indirect Designation, Pointer P
ret									✓	✓	✓		✓	✓		Yes	Yes
n									✓	✓	✓		✓	✓	✓	Yes	Yes

Note: See Section 1.15, "Restrictions on Devices Used as Instruction Parameters" of "Sequence CPU – Instructions" (IM34M6P12-03E)

■ Resource Relays

Table 3.5.64   Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
✓	M1027	FTP Client Busy	Execute the instruction only if the FTP Client Busy relay is OFF.

■ Function

Deletes one or more specified files on the FTP server.

Multiple files can be deleted by including wildcard characters ('*', '?') in the file name. In such situations, even if an error occurs at the FTP server end during file deletion, processing of undeleted files continues.

The number of deleted files is returned and stored in Status.

⚠ CAUTION

This instruction cannot be executed concurrently with other FTP client instructions.

■ Programming Example
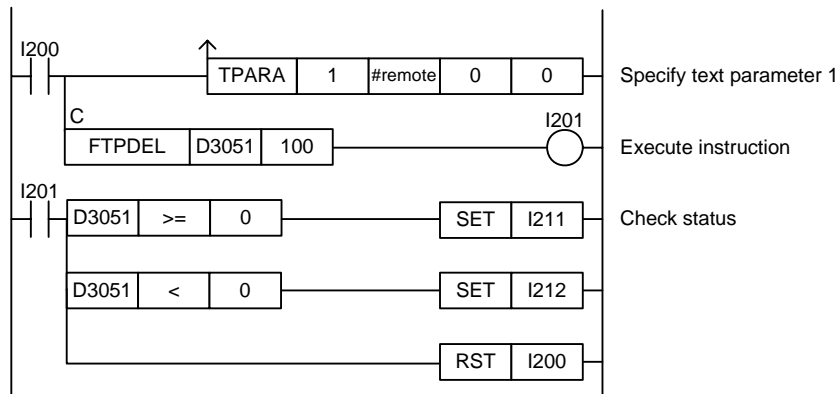
Figure 3.5.13   Example of an FTP Client Delete File Program

This sample code deletes the file on the FTP server with pathname defined by constant name #remote. The timeout interval is set to 100 (10 s).

The table below shows the returned status data (ret), assuming normal exit.

Device	Value	Table Parameter
ret = D3051	1	Status

3.5.12   FTP Client Rename File (FTPREN)

Renames a file on the FTP server.

Table 3.5.65   FTP Client Rename File

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Pro-cessing Unit	Carry
					Yes	No			
Continuous type application instruction	–	FTP Client Rename File	FTPREN	C ─┤ FTPREN □ □ ├─	✓	–	5	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see "■ Continuous Type Application Instructions" in Section 2.6.1, "Using Socket Instructions".

■ Parameter

FTP Client Rename File ─┤ C FTPREN │ ret │ n ├─

Table 3.5.66   Parameters

Parameter	Description
ret[*1]	Device for storing return status (W)
n	Timeout interval (W) [1-32767(×100 ms), 0 = longest(2147483647 ms)]

*1:   ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

Table 3.5.67   Text Parameters

	Parameter	Description
1	s	Old file pathname
2	d	New file pathname

SEE ALSO

Specify text parameters using the Text Parameter (TPARA) instruction. For details on text parameters and the Text Parameter (TPARA) instruction, see "■ Text Parameter" in Section 2.6.1, "Using Socket Instructions".

■ Status (Return Value)

Table 3.5.68   Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit
		< 0	Error status

SEE ALSO

For more details on error status, see "● Error Status of Continuous Type Application Instructions" of "■ Continuous Type Application Instructions" in Section 2.6.1, "Using Socket Instructions".

■ Available Devices

Table 3.5.69 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	W	Z	R	V	Constant	Index Modification	Indirect Designation, Pointer P
ret									✓	✓	✓		✓	✓		Yes	Yes
n									✓	✓	✓		✓	✓	✓	Yes	Yes

Note: See Section 1.15, "Restrictions on Devices Used as Instruction Parameters" of "Sequence CPU – Instructions" (IM34M6P12-03E)

■ Resource Relays

Table 3.5.70 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
✓	M1027	FTP Client Busy	Execute the instruction only if the FTP Client Busy relay is OFF.

■ Function

Renames a file on the FTP server.

⚠ CAUTION

This instruction cannot be executed concurrently with other FTP client instructions.

■ Programming Example
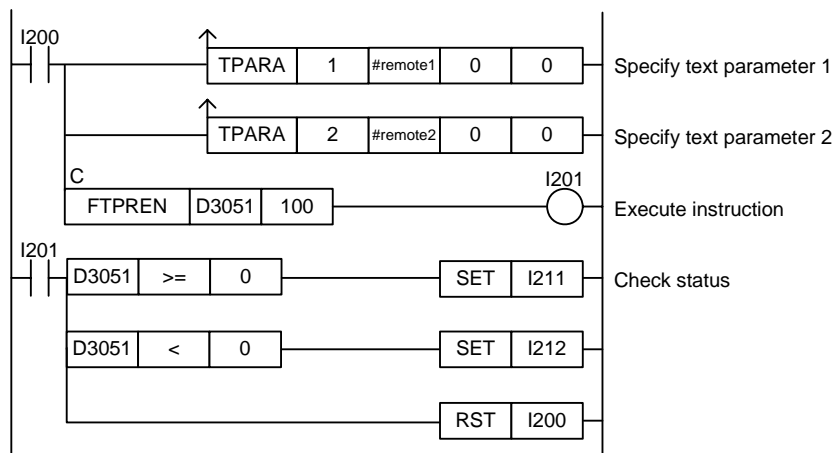
Figure 3.5.14 Example of an FTP Client Rename File Program

This sample code renames an FTP server file designated by constant name #remote1 to the new name defined by constant name #remote2. The timeout interval is set to 100 (10 s).

The table below shows the returned status data (ret), assuming normal exit.

Device	Value	Table Parameter
ret = D3051	0	Status

3.5.13 FTP Client Make Directory (FTPMKDIR)

Creates a directory on the FTP server.

Table 3.5.71 FTP Client Make Directory

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Processing Unit	Carry
					Yes	No			
Continuous type application instruction	–	FTP Client Make Directory	FTPMKDIR	C ⊢ FTPMKDIR ⊢	✓	–	5	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see "■ Continuous Type Application Instructions" in Section 2.6.1, "Using Socket Instructions".

■ Parameter

FTP Client Make Directory ⊢ C / FTPMKDIR | ret | n ⊢

Table 3.5.72 Parameters

Parameter	Description
ret[*1]	Device for storing return status (W)
n	Timeout interval (W) [1-32767(×100 ms), 0 = longest(2147483647 ms)]

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

Table 3.5.73 Text Parameters

Parameter		Description
1	d	Pathname of directory to be created

SEE ALSO

Specify text parameters using the Text Parameter (TPARA) instruction. For details on text parameters and the Text Parameter (TPARA) instruction, see "■ Text Parameter" in Section 2.6.1, "Using Socket Instructions".

■ Status (Return Value)

Table 3.5.74 Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit
		< 0	Error status

SEE ALSO

For more details on error status, see "● Error Status of Continuous Type Application Instructions" of "■ Continuous Type Application Instructions" in Section 2.6.1, "Using Socket Instructions".

■ Available Devices

Table 3.5.75 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	W	Z	R	V	Con-stant	Index Modification	Indirect Designation, Pointer P
ret									✓	✓	✓		✓	✓		Yes	Yes
n									✓	✓	✓		✓	✓	✓	Yes	Yes

Note: See Section 1.15, "Restrictions on Devices Used as Instruction Parameters" of "Sequence CPU – Instructions" (IM34M6P12-03E)

■ Resource Relays

Table 3.5.76 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
✓	M1027	FTP Client Busy	Execute the instruction only if the FTP Client Busy relay is OFF.

■ Function

Creates a directory on the FTP server.

⚠ CAUTION

This instruction cannot be executed concurrently with other FTP client instructions.

■ Programming Example
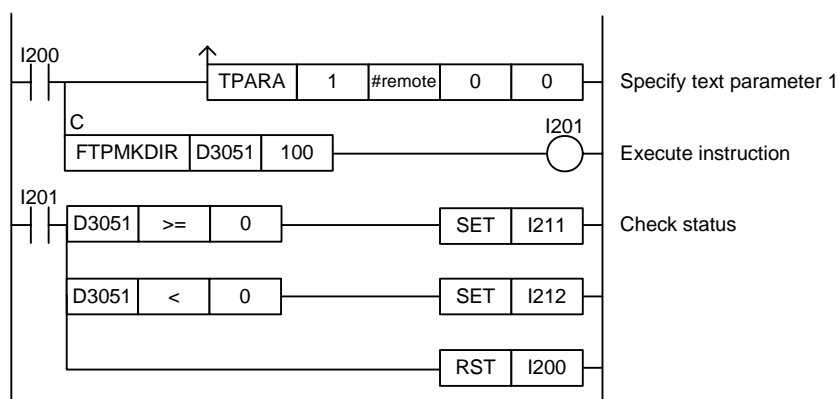
Figure 3.5.15 Example of an FTP Client Make Directory Program

This sample code creates a new directory on the FTP server according to the directory pathname defined by constant name #remote. The timeout interval is set to 100 (10 s).

The table below shows the returned status data (ret), assuming normal exit.

Device	Value	Table Parameter
ret = D3051	0	Status

3.5.14   FTP Client Remove Directory (FTPRMDIR)

Deletes a specified directory on the FTP server.

Table 3.5.77   FTP Client Remove Directory

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required? Yes	Input Condition Required? No	Step Count	Pro- cessing Unit	Carry
Continuous type application instruction	–	FTP Client Remove Directory	FTPRMDIR	C ─┤FTPRMDIR│ │├─	✓	–	5	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see "■ Continuous Type Application Instructions" in Section 2.6.1, "Using Socket Instructions".

■ Parameter

FTP Client Remove Directory   ─┤ C FTPRMDIR │ret│ n ├─

Table 3.5.78   Parameters

Parameter	Description
ret[*1]	Device for storing return status (W)
n	Timeout interval (W) [1-32767(×100 ms), 0 = longest(2147483647 ms)]

*1:   ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

Table 3.5.79   Text Parameters

Parameter		Description
1	d	Pathname of directory to be deleted

SEE ALSO

Specify text parameters using the Text Parameter (TPARA) instruction. For details on text parameters and the Text Parameter (TPARA) instruction, see "■ Text Parameter" in Section 2.6.1, "Using Socket Instructions".

■ Status (Return Value)

Table 3.5.80   Status (Return Value)

Offset (word)		Description
ret	0	Normal exit
	< 0	Error status

SEE ALSO

For more details on error status, see "● Error Status of Continuous Type Application Instructions" of "■ Continuous Type Application Instructions" in Section 2.6.1, "Using Socket Instructions".

■ Available Devices

Table 3.5.81   Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	W	Z	R	V	Constant	Index Modification	Indirect Designation, Pointer P
ret									✓	✓	✓		✓	✓		Yes	Yes
n									✓	✓	✓		✓	✓	✓	Yes	Yes

Note: See Section 1.15, "Restrictions on Devices Used as Instruction Parameters" of "Sequence CPU – Instructions" (IM34M6P12-03E)

■ Resource Relays

Table 3.5.82   Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
✓	M1027	FTP Client Busy	Execute the instruction only if the FTP Client Busy relay is OFF.

■ Function

Deletes a specified directory on the FTP server.

> ⚠ **CAUTION**
>
> This instruction cannot be executed concurrently with other FTP client instructions.

■ Programming Example

| I200 | | | ↑ | TPARA | 1 | #remote | 0 | 0 | | Specify text parameter 1 |

```
        I200                    ↑
        ─┤ ├──────────────────  TPARA   1   #remote   0   0        Specify text parameter 1

               C                                          I201
               FTPRMDIR   D3051   100                     ( )      Execute instruction

        I201
        ─┤ ├─    D3051   >=   0              SET   I211            Check status

                 D3051   <    0              SET   I212

                                             RST   I200
```
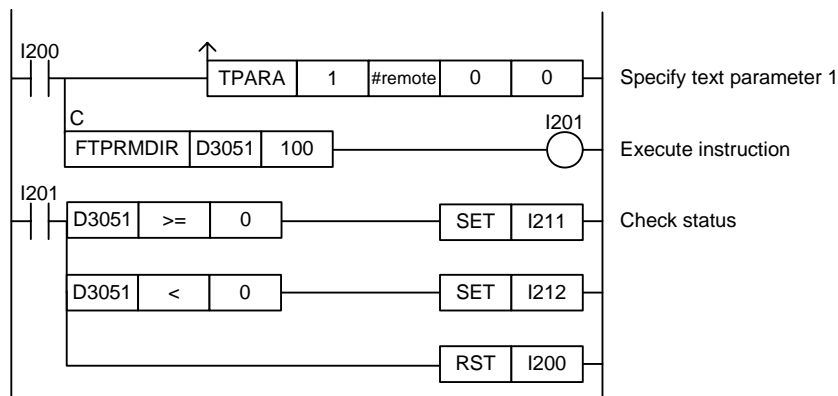
Figure 3.5.16   Example of an FTP Client Remove Directory Program

This sample code deletes from the FTP server the directory designated by the directory pathname defined by constant name #remote. The timeout interval is set to 100 (10 s).

The table below shows the returned status data (ret), assuming normal exit.

Device	Value	Table Parameter
ret = D3051	0	Status

3.5.15 FTP Client Representation Type (FTPTYPE)

Selects ASCII or binary representation for FTP data transfer.

Table 3.5.83 FTP Client Representation Type

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Processing Unit	Carry
					Yes	No			
Continuous type application instruction	–	FTP Client Representation Type	FTPTYPE	C ─┤FTPTYPE │ │ ├─	✓	–	6	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see "■ Continuous Type Application Instructions" in Section 2.6.1, "Using Socket Instructions".

■ Parameter

FTP Client Representation Type ─┤ C
FTPTYPE │ ret │ n1 │ n2 ├─

Table 3.5.84 Parameters

Parameter	Description
ret[1]	Device for storing return status (W)
n1	Timeout interval (W) [1-32767(×100 ms), 0 = longest(2147483647 ms)]
n2	Representation type (W)[0 = ASCII, 1 = binary]

[1]: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

■ Status (Return Value)

Table 3.5.85 Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit
		< 0	Error status

SEE ALSO

For more details on error status, see "● Error Status of Continuous Type Application Instructions" of "■ Continuous Type Application Instructions" in Section 2.6.1, "Using Socket Instructions".

■ Available Devices

Table 3.5.86 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	W	Z	R	V	Constant	Index Modification	Indirect Designation, Pointer P
ret									✓	✓	✓		✓	✓		Yes	Yes
n1									✓	✓	✓		✓	✓	✓	Yes	Yes
n2									✓	✓	✓		✓	✓	✓	Yes	Yes

Note: See Section 1.15, "Restrictions on Devices Used as Instruction Parameters" of "Sequence CPU – Instructions" (IM34M6P12-03E)

■ Resource Relays

Table 3.5.87   Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
✓	M1027	FTP Client Busy	Execute the instruction only if the FTP Client Busy relay is OFF.

■ Function

Selects ASCII or binary representation for FTP data transfer. The representation type defaults to binary when FTP client is started.

In binary representation data transfer, data in files are transferred as is. In general, binary representation can be used for any file format. Both text files (e.g. files with filename extensions ".txt", ".ypjc" and ".yprp") and binary files (e.g. files with filename extensions ".bin", ".pdf", ".doc" and ".jpg") can be sent using binary representation.

ASCII representation is used for sending text files when newline code conversion is required. At transmission, CRLF and CR characters are transmitted without change but LF characters are converted to CRLF. Beware that specifying ASCII representation for transferring a binary file will result in invalid data due to conversion processing.

⚠ CAUTION

- This instruction cannot be executed concurrently with other FTP client instructions.
- If the contents of the source file and destination file are unexpectedly different, check whether the problem arose because ASCII representation was specified. If no newline conversion is required, specify binary representation for FTP transfer of all file formats.

■ Programming Example
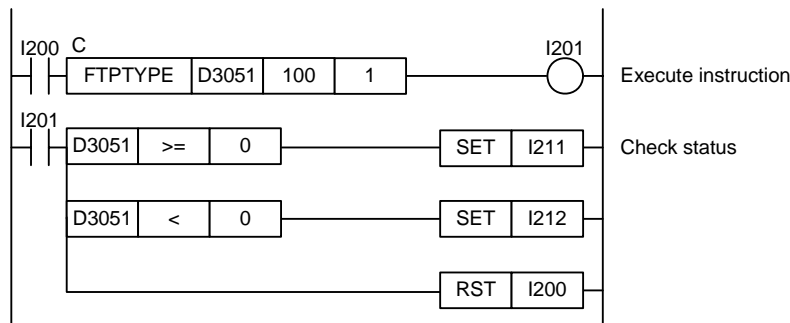
Figure 3.5.17   Example of an FTP Client Representation Type Program

This sample code switches to binary representation for data transfer to the connected FTP server. The timeout interval is set to 100 (10 s).

The table below shows the returned status data (ret), assuming normal exit.

Device	Value	Table Parameter
ret = D3051	0	Status

3.6 FTP Server

This section describes the FTP server function of the module.

3.6.1 FTP Server Specifications

The table below shows the specifications of the FTP server function.

Table 3.6.1   FTP Server Specifications

Item	Specifications
Number of connected clients	4
Home directory	\RAMDISK
Port number	21 (default value)
Maximum command length	256 bytes
Maximum file pathname length[*1]	256 bytes
Security function	- Login password - FTP sever log - Automatic disconnection upon multiple command buffer overruns

*1:   In the case of a relative pathname, the length limit is applied after conversion to absolute pathname.

The table below lists the commands supported by the FTP server function. The command names use telnet command notation. Command mapping on the FTP client end (get, put, etc.) is according to specifications of individual clients.

Table 3.6.2   Commands Supported by FTP Server (telnet command notation)

Command	Function	Specification
USER	User name	
PASS	User password	
CWD	Change current directory	
XCWD	Change current directory	
CDUP	Change to parent directory	
REIN	Reinitialize server state	
QUIT	Terminate service	
PORT	Specify data connection port	
PASV	Set server in passive mode	
TYPE	Specify file format	ASCII, binary
STRU	Specify file structure	Only F-File is supported.
MODE	Specify transfer mode	Only S-Stream is supported.
RETR	Retrieve a file copy	
STOR	Store file to server	
APPE	Append file	

STOU	Store file with unique name	- Specifications
		   RFC1123
		- Naming rule
		   User name.nnn (where nnn=000 to 999)

REST	Resend	
RNFR	Rename from	Used together with RNTO
RNTO	Rename to	Used together with RNFR
DELE	Delete file on server	
RMD	Remove directory	
XRMD	Remove directory	
MKD	Create directory	
XMKD	Create directory	
PWD	Return current directory	
XPWD	Display current directory	
LIST	Display file list	
NLST	Display file list	
NOOP	No operation	
HELP	Display help	

3.6.2   FTP Server Setup

This subsection describes how to configure the FTP server function before use.

■ Basic Setup

The table below shows required setup for the FTP server function before use.

Table 3.6.3   Basic Setup for FTP Server Function

Name of Setup	Type of Setup	SEE ALSO [*1]
Ethernet setup	CPU properties	A9.5.2, "Ethernet Setup"

*1:   For details on individual setup items, see "Sequence CPU – Functions (for F3SP66-4S, F3SP67-6S)" (IM34M6P14-01E).

● Ethernet setup

Minimally, you must specify the IP address and subnet mask. If you set the subnet mask to "0.0.0.0", the default mask for the class of the IP address is used.

■ Optional Setup

The FTP server function may be configured as required.

Table 3.6.4   Optional Setup for FTP Server Function

Name of Setup	Type of Setup	SEE ALSO [*1]
FTP server setup	CPU properties	A9.5.6, "FTP Server Setup"
Network filter setup	CPU properties	A9.5.8, "Network Filter Setup"
Function removal	Configuration	A9.2.12, "Function Removal"

*1:   For details on individual setup items, see "Sequence CPU – Functions (for F3SP66-4S, F3SP67-6S)" (IM34M6P14-01E).

● FTP server setup

You may use FTP server setup of CPU properties to modify default values for the following items related to the FTP server function.

- FTP server/my port no. (default value=21)
- FTP server/maximum connections (default value=4)
- FTP server/password (default value="fam3@")
- FTP server/log (default value=Yes)
- FTP server/anonymous login enable (default value=Enabled)
  When anonymous login is enabled, a user may log in successfully using any password by specifying the username as "anonymous".
- FTP server/interval timeout (default value=3600 s)
  If the FTP server receives no request from an FTP client within the specified time, it terminates the connection with the FTP client.
- FTP server/network timeout (default value=60 s)
  You can define the response timeout interval for TCP/IP communications, which is the layer below FTP in the network protocol.   In a high-load, low-speed communications environment, an internal communications timeout error (error code: -1001) may be reported during execution of an FTP client instruction. Lengthening this timeout interval may solve the problem in such situations.

● Network filter setup

You may perform network filter setup to restrict the IP addresses connecting to the FTP server. By default, connections from all IP addresses are allowed. This setting affects all functions using the CPU built-in 10BASE-T/100BASE-TX connector such as socket communications function and remote programming service.

3.6.3 Using FTP Server

This subsection describes how to use the FTP server function.

■ Starting and Stopping FTP Server

● Starting FTP Server

The FTP server function is automatically executed at module startup (power on or module reset). The FTP server setup of CPU properties is read when the FTP server is started.

● Stopping FTP Server

You cannot actually terminate the FTP server but you can stop the FTP server from accepting requests from FTP clients by executing the FTP Server Stop Request Service (FTPSTOP) instruction. To resume acceptance of requests from FTP clients by the FTP server, execute the FTP Server Run Request Service (FTPSRUN) instruction.

When you remove the FTP server using Function Removal of Configuration, it does not actually terminate the FTP server function but stops the FTP server from acceptance requests from FTP clients.

TIP

Even if you have removed the FTP server using Function Removal of Configuration, you can resume acceptance of requests from FTP clients by the FTP server by executing the FTP Server Run Request Service (FTPSRUN) instruction.

■ Connecting with FTP Clients

To connect to the FTP server from a remote FTP client, execute an open command from the remote FTP client. If the remote client is an FA-M3 unit, you can connect to the FTP server by executing the FTP Client Open (FTPOPEN) instruction.

SEE ALSO

- For details on how to specify the connection destination, see section 3.2, "FTP Network Configurations and Access Methods."
- For details on how to use the FTP client function of the module, see Section 3.3, "FTP Client."

■ Handling of File Pathname

● Relative pathname and absolute pathname

Both relative pathnames and absolute pathnames can be used with the FTP server function of the module. Relative pathnames are pathnames relative to the current directory.

● Specifying abc.txt using an absolute pathname
  \RAMDISK\MYDIR\abc.txt
● Change current directory to:
  \RAMDISK\MYDIR
● Specifying abc.txt using a relative pathname:
  abc.txt

Specifying the same file

(Current directory) + (Relative pathname) = (Absolute pathname)

FC0312.VSD

Figure 3.6.1 Relative Pathname and Absolute Pathname

● **Home directory**

The home directory of the FTP server is "\RAMDISK".

● **Current directory**

The current directory of the FTP server defaults to the home directory ("\RAMDISK") of the FTP server when an FTP connection is established with a remote FTP client. To change the current directory of the FTP server, execute a change current directory (cd) command from the remote FTP client. If the remote FTP client is an FA-M3 unit, you can execute the FTP Client Change Directory (FTPCD) instruction.

You may not specify a directory below pathname "\VIRTUAL" for the current directory.

3.6.4   FTP Server Log

Logs all accesses to and all responses from the FTP server.

■ FTP Server Log Specifications

● FTP server log information

The table below lists the comma-delimited fields that form a log record.

Table 3.6.5   FTP Server Log Items

Field	Description
Log number	"xxx" (where xxx=000 to 188) A running number. A smaller number indicates an older log record.
Date	"yy/mm/dd" (yy=year, mm=month, dd=day) The date when the log record is created.
Time	"hh: mm: ss" (hh=hour, mm=minute, ss=second) The time when the log record is created.
FTP client IP address	"xxx.xxx.xxx.xxx" (where xxx=000 to 255) IP address of the remote FTP client
Transmission direction	"-->" (transmission from FTP client to FTP server) "<--" (transmission from FTP server to FTP client) Direction of message transmission
FtpServer	An additional code generated to facilitate determination of transmission direction
Account name	Login account name Characters beyond the length limit of 10 characters are not logged.
Message	Transmitted FTP command at TELNET command level.

● Time when information is output to FTP server log

The FTP function outputs a log record to the FTP server log when it receives a request from the FTP client and when it sends a response to the FTP client.

Up to 189 log records are allowed. When the limit is exceeded, the oldest log record is overwritten.

● Special relays and special registers

No special relays or special registers are related to the FTP server log function.

■ FTP Server Log Setup

● Basic Setup

The FTP server log function requires no basic setup before use.

● Optional Setup

Table 3.6.6   Optional Setup for FTP Server Log Function

Name of Setup	Type of Setup	SEE ALSO[1]
FTP Server Setup	CPU properties	A9.5.6, "FTP Server Setup"

[1]: For details on individual setup items, see "Sequence CPU – Functions (for F3SP66-4S, F3SP67-6S)" (IM34M6P14-01E).

To generate no FTP server log, set the FTP Server/Log property of FTP Server Setup of CPU Properties to 'No'. By default, FTP server log is generated.

■ Using FTP Server Log

● Starting FTP server log output

FTP server log output is automatically started if enabled in CPU properties.

● Getting FTP server log

You can get the FTP server log as a text file using smart access functions.

Table 3.6.7   Methods for Getting FTP Server Log

Function	Outline	SEE ALSO
Rotary switch function	You can save the FTP server log to the SD memory card using the rotary switch.	B1.4.5, "Module Info" [1]
Card batch command function	You can save the FTP server log to a directory on the module by executing a card batch file.	B2.8.2.3, "Get Log (LOG)" [1]
Virtual directory function	You can get the FTP server log from a remote FTP client by executing a virtual directory command on the module.	3.7.7.3, "Get Log (LOG)"

[1]: For details on individual items, see "Sequence CPU – Functions (for F3SP66-4S, F3SP67-6S)" (IM34M6P14-01E).

● Stopping FTP server log output

You can stop FTP server log output using FTP server setup of CPU properties.

SEE ALSO

For details, see "■ FTP Server Log Setup" described earlier.

● Clearing FTP server log

The FTP server log is cleared automatically at module startup (power on or reset).

3.6.5　FTP Server Instructions

FTP server instructions can be executed to suspend or resume the FTP server request service, which accepts requests from remote FTP clients.

3.6.5.1　FTP Server Run Request Service (FTPSRUN)

This FTP server instruction resumes the FTP server request service, which accepts requests from FTP clients, if the service had been suspended by a FTP Server Stop Request Service (FTPSSTOP) instruction or by Function Removal of configuration.

Table 3.6.8　FTP Server Run Request Service

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required? Yes	No	Step Count	Processing Unit	Carry
Continuous type application instruction	–	FTP Server Run Request Service	FTPSRUN	C ⊢ FTPSRUN ⊢	✓	–	5	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see "■ Continuous Type Application Instructions" in Section 2.6.1, "Using Socket Instructions".

■ Parameter

C
FTP Server Run Request Service ⊢ FTPSRUN | ret | n ⊢

Table 3.6.9　Parameters

Parameter	Description
ret[1]	Device for storing return status (W)
n	Timeout interval (W) [1-32767(×100 ms), 0 = longest(2147483647 ms)]

[1]: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

■ Status (Return Value)

Table 3.6.10　Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit
		< 0	Error status

SEE ALSO

For more details on error status, see "● Error Status of Continuous Type Application Instructions" of "■ Continuous Type Application Instructions" in Section 2.6.1, "Using Socket Instructions".

■ Available Devices

Table 3.6.11 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	W	Z	R	V	Constant	Index Modification	Indirect Designation, Pointer P
ret									✓	✓	✓		✓	✓		Yes	Yes
n									✓	✓	✓		✓	✓	✓	Yes	Yes

Note: See Section 1.15, "Restrictions on Devices Used as Instruction Parameters" of "Sequence CPU – Instructions" (IM34M6P12-03E)

■ Resource Relays

None. Ensure that there is no repeated execution of this instruction in your program.

■ Function

Execution of this instruction is normally not required as FTP server is automatically started at power on or module reset. This FTP server instruction, however, can be used to resume the FTP server request service, which accepts requests from FTP clients, if the service had been suspended by a FTP Server Stop Request Service (FTPSSTOP) instruction or by Function Removal of configuration.

⚠ CAUTION

Processing of this instruction is always completed even in the presence of a timeout or cancellation.

■ Programming Example
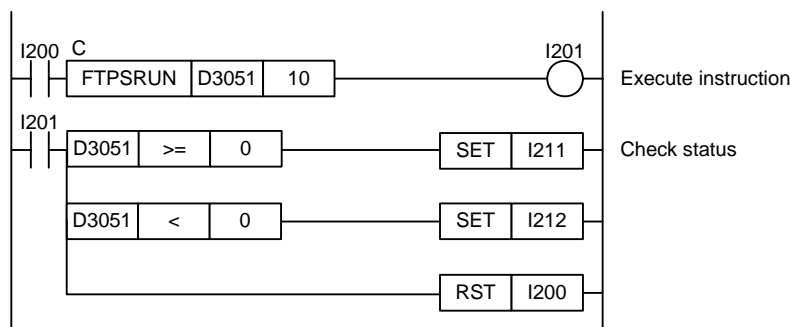
Figure 3.6.2 Example of an FTP Server Run Request Service Program

This sample code resumes the FTP server request service, which had been suspended by a FTP Server Stop Request Service (FTPSSTOP) instruction. The timeout interval is set to 10 (1 s).

The table below shows the returned status data (ret), assuming normal exit.

Device	Value	Table Parameter
ret = D3051	0	Status

3.6.5.2 FTP Server Stop Request Service (FTPSSTOP)

This FTP server instruction suspends the FTP server request service, which accepts requests from FTP clients.

Table 3.6.12   FTP Server Stop Request Service

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Processing Unit	Carry
					Yes	No			
Continuous type application instruction	–	FTP Server Stop Request Service	FTPSSTOP	C ⊣ FTPSSTOP ⊢	✓	–	5	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see "■ Continuous Type Application Instructions" in Section 2.6.1, "Using Socket Instructions".

■ Parameter

FTP Server Stop Request Service ⊣ C FTPSSTOP | ret | n ⊢
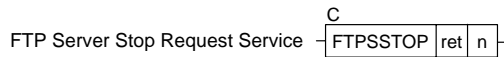
Table 3.6.13   Parameters

Parameter	Description
ret[*1]	Device for storing return status (W)
n	Timeout interval (W) [1-32767(×100 ms), 0 = longest(2147483647 ms)]

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

■ Status (Return Value)

Table 3.6.14   Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit
		< 0	Error status

SEE ALSO

For more details on error status, see "● Error Status of Continuous Type Application Instructions" of "■ Continuous Type Application Instructions" in Section 2.6.1, "Using Socket Instructions".

■ Available Devices

Table 3.6.15   Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	W	Z	R	V	Constant	Index Modification	Indirect Designation, Pointer P
ret									✓	✓	✓		✓	✓		Yes	Yes
n									✓	✓	✓		✓	✓	✓	Yes	Yes

Note: See Section 1.15, "Restrictions on Devices Used as Instruction Parameters" of "Sequence CPU – Instructions" (IM34M6P12-03E)

■ Resource Relays

None. Ensure that there is no repeat execution in your program.

◼ Function

This FTP server instruction suspends the FTP server request service, which accepts requests from FTP clients. If a request from an FTP client is being processed when this instruction is executed, processing of the request will still be carried through to the end.

⚠ CAUTION

Processing of this instruction is always completed even in the event of a timeout or cancellation.

◼ Programming Example
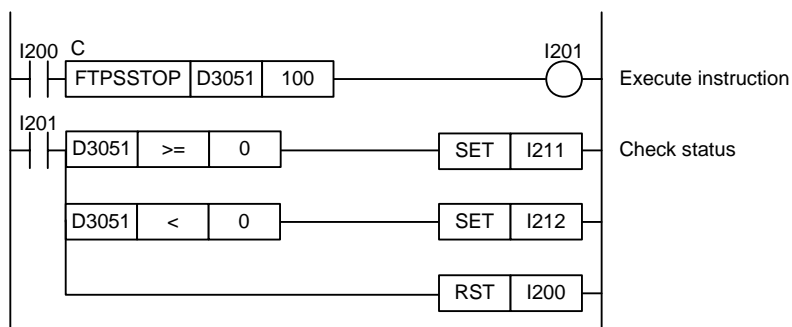
Figure 3.6.3   Example of an FTP Server Stop Request Service Program

This sample program suspends the FTP server request service. The timeout interval is set to 100 (10 s).

The table below shows the returned status data (ret), assuming normal exit.

Device	Value	Table Parameter
ret = D3051	0	Status

3.7 Virtual Directory Commands

This section describes virtual directory commands.

3.7.1 Overview of Virtual Directory Commands

This subsection gives an overview of virtual directory commands.

■ Overview of Virtual Directory Commands

● What are virtual directory commands?

Virtual directory commands are extended FTP server functions of the module. By coding a command as the file pathname of an FTP put or get command, the module can be made to perform various operations. For instance, using FTP, the module can be made to automatically convert data in a transferred file and store the data to devices, or to automatically convert device data into a file and send it, or to load or save a project.
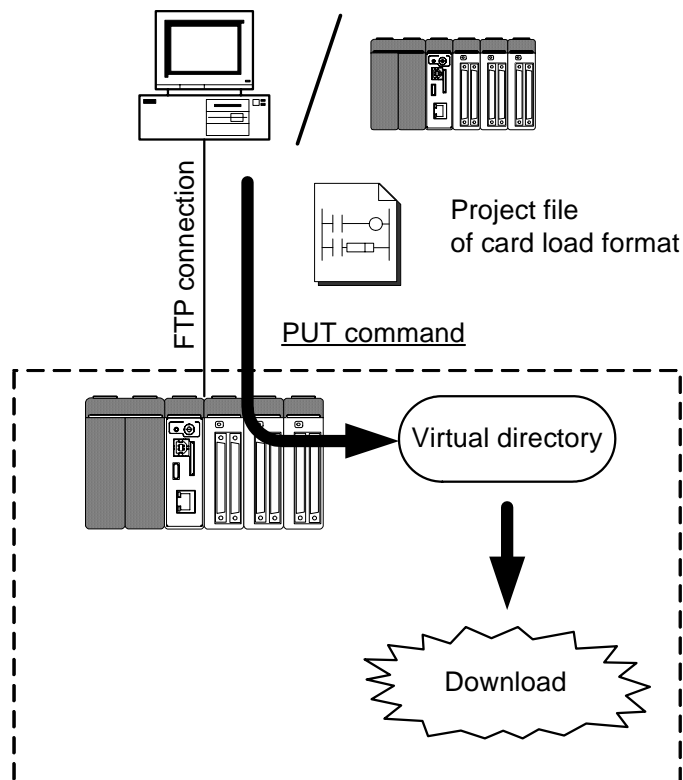
F0322.VSD

Figure 3.7.1   Concept of Virtual Directory Command (PUT)
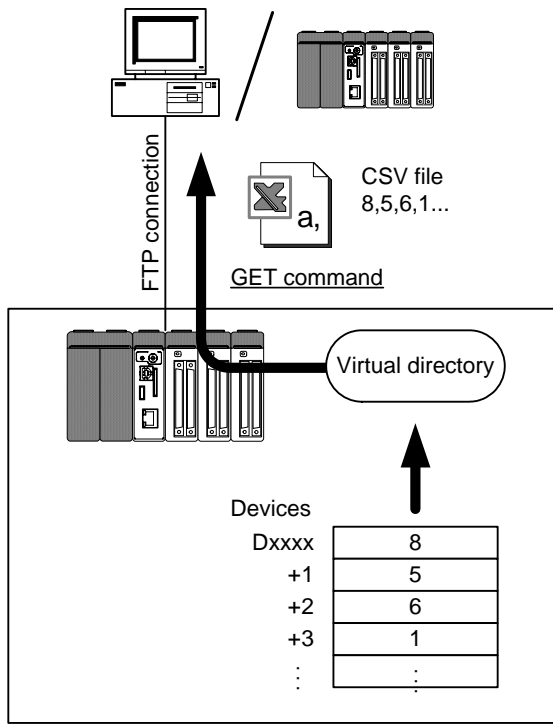
F0323.VSD

Figure 3.7.2   Concept of Virtual Directory Command (GET)

● Support for device access using FTP

Both bit (relay) based and word based device access commands are provided. Virtual directory commands enable complex processing that would require other protocols in the past to be carried using only FTP. Some examples of such processing are transmission of a processing request trigger after sending a recipe file and polling the operating status of an application program.
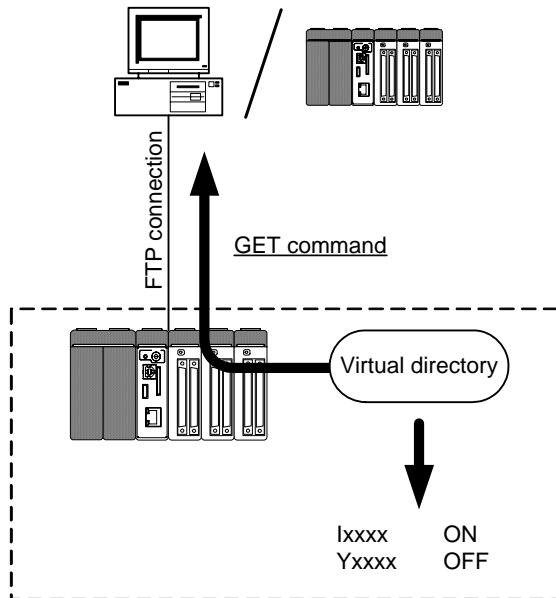
F0324.VSD

Figure 3.7.3   Setting a Relay using a Virtual Directory Command

■ List of Virtual Directory Commands

The table below lists the available virtual directory commands. The following figure shows the corresponding virtual directory structure.

Table 3.7.1   List of Virtual Directory Commands

Command Group	Function Name	Command Name	Description
File/device conversion & transfer commands	Convert CSV File to Device	F2DCSV	Converts a CSV formatted file into device data.
	Convert Device to CSV File	D2FCSV	Gets device data after it has been converted into a CSV formatted file.
	Convert Binary File to Device	F2DBIN	Converts a binary file into device data.
	Convert Device to Binary File	D2FBIN	Gets device data after it has been converted into a binary file.
Device access commands	Bit access	BRD	Reads bits.
		BWR	Writes bits.
		BFL	Writes bits of the same data.
	Word access	WRD	Reads words.
		WWR	Writes words.
		WFL	Writes words of the same data.
Maintenance commands	Load Project	LOAD	Loads project or CPU property data into the internal ROM of the module.
	Save Project	SAVE	Gets project or CPU property data from the internal ROM of the module.
	Get Log	LOG	Gets the system log or FTP server log in text format.
	CPU Info	CPUINFO	Gets CPU Status (operating mode, alarm status, rotary switch status, and card mount status).
	Application Info	APINFO	Gets system information (project information, I/O setup information).
	Run Mode	RUN	Switches the operating mode to Run mode.
	Stop Mode	STOP	Switches the operating mode to Stop mode.
	Activate Block	ACT	Activates a specified block.
	Inactivate Block	INACT	Inactivates a specified block.
	Reset CPU	CPURESET	Resets CPU.
	Clear Alarms	ALMCLEAR	Clears all alarms.
	Help	HELP	Gets help information on virtual directory commands.
File and disk operation commands	Unmount [1]	UNMOUNT	Unmounts a memory card.
Card batch file execution commands	Run Card Batch File	BATGO	Executes a specified card batch file.

*1:   You can use FTP commands for file deletion, directory creation and other file operations.
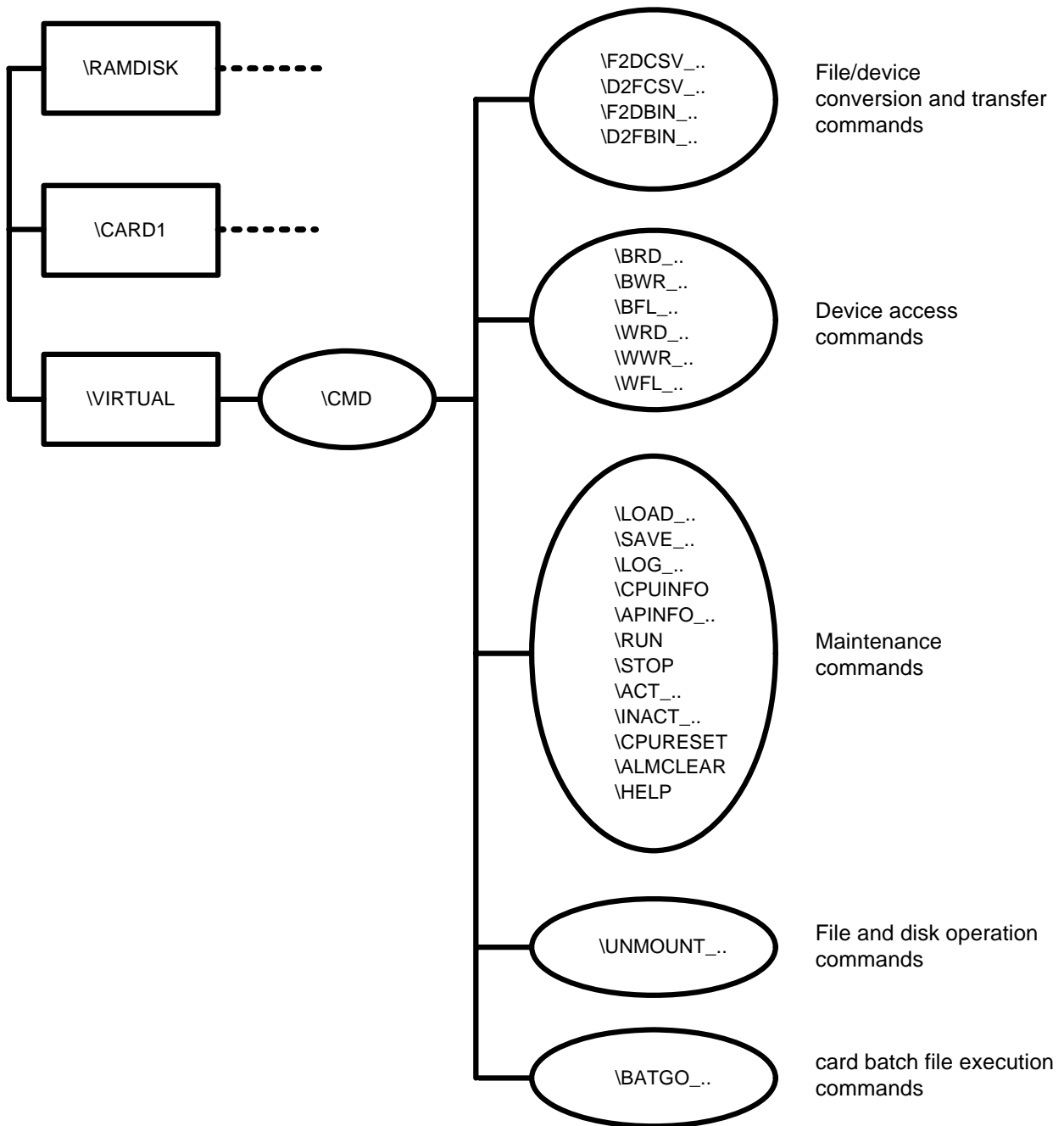
F0390.VSD

Note: - The directory structure below "\VIRTUAL" cannot be accessed using "ls" or other FTP commands for getting file or directory information.
- The "_.." suffix to a command indicates that the command has required parameters.
- FTP LIST, NLST (or FTPLS instruction of the module) commands cannot be used to get directory or file information of a virtual directory (a directory below "\VIRTUAL").

Figure 3.7.4 Directory Structure of Virtual Directory Commands

3.7.2　Virtual Directory Command Setup

This subsection describes how to configure virtual directory commands before use.

■ Basic Setup

The virtual directory command function requires some basic setup before use.

● FTP server setup

Virtual directory commands run on the FTP server so FTP server setup must be performed.

If a timeout error (error code SE05) is reported during execution of a virtual directory command having a long processing time (e.g. a Run Batch File (BATGO) command), lengthening the FTP server network timeout (FTPS_NETACK_TOUT) interval may solve the problem.

SEE ALSO

For details on FTP server setup, see Subsection 3.6.2, "FTP Server Setup."

■ Optional Setup

Virtual directory commands may be configured as required before use.

Table 3.7.2　Optional Setup for Virtual Directory Commands

Name of Setup	Type of Setup	SEE ALSO[1]
Function removal	Configuration	A9.2.12, "Function Removal"
FTP client setup	CPU properties	A9.5.5, "FTP Client Setup"

[1]: For details on individual setup items, see "Sequence CPU – Functions (for F3SP66-4S, F3SP67-6S)" (IM34M6P14-01E).

● Function removal

To disable all virtual directory commands, remove the virtual directory function using function removal of configuration.

● FTP client setup

If an internal timeout error (error code: -1001) is returned as instruction status to a FA-M3 unit running as FTP client, adjusting the network timeout on the FTP client end using FTP client setup may solve the problem.

3.7.3 Using Virtual Directory Commands

This subsection describes how to use virtual directory commands.

- Preparing for the use of virtual directory commands
- Executing virtual directory commands
- Checking execution results of virtual directory commands
- Virtual directory command syntax
- Error reply messages of virtual directory commands
- Relationship between Virtual directory command Execution and EXE LED status

The flowchart on the next page illustrates the procedure for using virtual directory commands.

BEGIN

Run FTP client — Run FTP client on PC, FA-M3, etc.

DESCRIPTION

Connect to
FTP server — Connect to target FA-M3 FTP server using OPEN command or instruction.

Server
connection successful? — No / Yes

Beginning of loop
(If required)

Issue
virtual directory command

Issue virtual directory command using PUT command or GET command

- If PUT/GET command returns an OK status, it means that the virtual directory command is successful.
- If PUT/GET command returns an error status, it means that the virtual directory command is not successful.

PUT or GET
successful? — Yes / No

Get or check
response file — If execution fails, check the response file for error details. If no error is found in response file, it indicates an error generated by FTP itself.

End of loop
(if required)

Terminate
FTP client — Terminate FTP client, and disconnect from FTP server using QUIT command or instruction.

Disconnect
from FTP server

END

User processing

- System processing
- External event

F0325.VSD

Figure 3.7.5 Procedure for Executing Virtual Directory Commands

■ Preparing for the Use of Virtual Directory Commands

● Run FTP server

Virtual directory commands run on the FTP server function of the module. By default, the FTP server function of the module is automatically started at module startup (power on or reset).

SEE ALSO

For details on using and configuring the FTP server function, see Section 3.6, "FTP Server."

⚠ CAUTION

Virtual directory commands are proprietary extended FTP server functions of the module so they can be used only if the FTP server machine is the module and not a PC. There is, however, no restriction on the FTP client machine, which can be the module, a PC or some other machine.

● Prepare FTP client

Virtual directory commands are issued by an FTP client. Prepare an FTP client application or FTP client library, which is to be executed on the module or a PC acting as the FTP client. In addition, a command prompt on the PC is useful for checking execution outcome.

● Connect to FTP server (the module) from FTP client

Establish an FTP connection from the FTP client to the FTP server (the module).

SEE ALSO

For details on how to connect to the FTP server, see Subsection 3.6.3, "Using FTP Server."

■ Executing Virtual Directory Commands

An execution request for a virtual directory command is issued by coding the virtual directory command in the file pathname of an FTP put or get command, and sending the FTP command from the FTP client to the FTP server (the module).

● Issuing a virtual directory command using a put command

You can invoke various functions of the module running as a remote FTP server by coding the appropriate virtual directory command in the destination file pathname of a put command. The example below shows a put command for sending a project file, which is to be loaded on the module, from the PC to the module.

put myprj.ypjc \VIRTUAL\CMD\LOAD_.ypjc

put   File to be sent   Virtual directory command
       using put

A file on an FTP client can be processed according to a virtual directory command by putting the file on the FTP server.
This sample command loads a project of card load format named "myprj.ypjc" into the internal ROM of the CPU module.

F0326.VSD

Figure 3.7.6   Issuing a Virtual Directory Command using a put Command

● Issuing a virtual directory command using a get command

You can invoke various functions on the module running as a remote FTP server by coding the appropriate virtual directory command in the source file pathname of a get command. The get command can be used from a PC to get a project file from the module and save it on the PC. The get command can also be used to switch the operating mode, as well as read from or write to a specified device and perform other operations which do not require getting a file.

get \VIRTUAL\CMD\SAVE_pass_.ypjc myprj.ypjc

get   Virtual directory command   Source file pathname

You can get information (file) or perform processing on the module running as FTP server from an FTP client by "getting" a virtual directory command.
This sample command gets a project stored in the internal ROM of the FTP server and saves it as a project file of card load format named "myprj.ypjc" in the current directory of the FTP client.

F0327.VSD
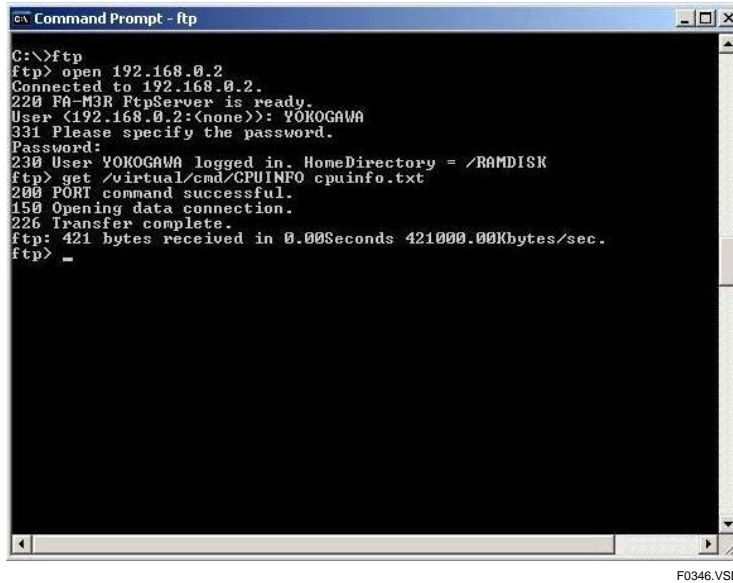
Figure 3.7.7   Issuing a Virtual Directory Command using a get Command

■ Checking Execution Results of Virtual Directory Commands

You can check the execution result of a virtual directory command by checking the reply of the put command or get command and checking the response file.

● Successful execution of virtual directory command

If a virtual directory command execution is successful, the put command or get command exits normally.

F0346.VSD
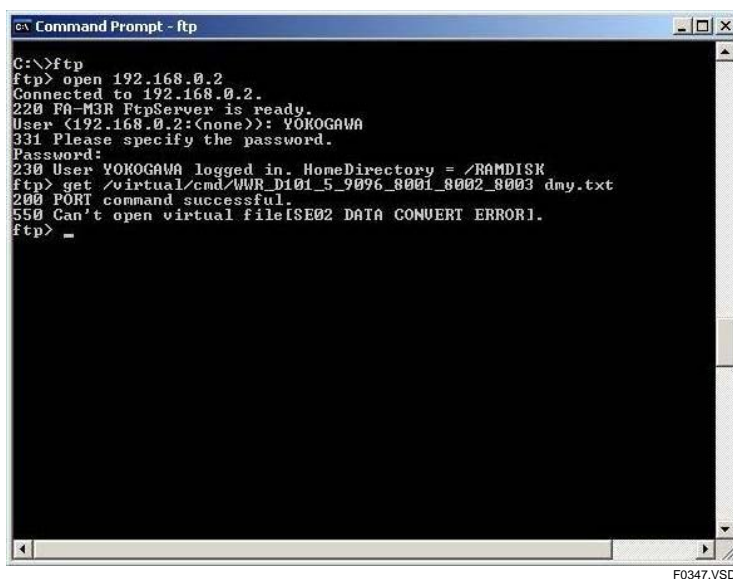
Figure 3.7.8   Successful Execution of Virtual Directory Command

● Unsuccessful execution of virtual directory command

If a virtual directory command execution is unsuccessful, the put command or get command exits with error.

When an error occurs, the FTP server returns a "550 Can't open virtual file [<detailed error code>]" error message if it recognizes the virtual directory command, and returns a "550 Can't open file" error message if otherwise.

F0347.VSD

Figure 3.7.9   Unsuccessful Execution of Virtual Directory Command

In addition, a response file describing the detailed error code is generated with the following file pathname for each FTP server account name.

```
\VIRTUAL\CMD\RES_<account name>.res
```

TIP

- The response file contains information for the most recent error, and is overwritten each time an error occurs. Therefore, if a put command or get command exits with error, you should check the response file before issuing the next virtual directory command.

- Two FTP connections established using the same account name will have the same response file name, which will pose a problem when checking detailed error codes.

■ Virtual Directory Command Syntax

A virtual directory command consists of a common part, a command part, a parameter part and a file part.

The table below describes each part of a virtual directory command.

Table 3.7.3   Virtual Directory Command Structure

Command Part	Description	Syntax
Common	Specify an identifier denoting a virtual directory command.	\VIRTUAL\CMD
Command	Code the command name of a virtual directory command.	<command name> Specify a command name.
Parameters	Specify parameters (arguments) of the command, using an underscore ('_') character as delimiter between the first parameter and the command part, as well as between parameters. The number of parameters varies with command.	<parameters> Specify command arguments. Only alphanumeric characters are allowed.
File	Specify file(s). See the description of individual commands for details as it is command-dependent.	<file name> Specify a file name.

The common part is delimited from the command part using the backslash ('\') character, which is the same character used for delimiting subdirectories.

Some examples of virtual directory commands are shown below.

```
>put data012.csv \VIRTUAL\CMD\F2DCSV_D101_-1_0_2_1_0_0_128

>get \VIRTUAL\CMD\D2FCSV_D101_2_128_0_6_1_0_0_4 data012.csv
```

■ Error Reply Messages of Virtual Directory Commands

Table 3.7.4 Error Reply Messages of Virtual Directory Commands

Reply Message	Error Code	Description
PARAMETER ERROR	SE01	Invalid parameter
DATA CONVERT ERROR	SE02	Input data could not be converted to the specified format.
DEVICE BOUNDARY VALUE EXCEEDED.	SE03	An attempt was made to access a write-prohibit area.
MULTI CPU ERROR	SE04	CPU number is invalid or no response was received from the target CPU.
TIMEOUT ERROR	SE05	Internal timeout has occurred.
FILE SYSTEM ERROR	SE10	Processing could not continue because a file system failure was detected. Reformat the disk in FAT16 format, or replace the memory card.
INVALID FILE	SE11	File data is invalid or could not be interpreted.
NO FILE ERROR	SE12	File or directory was not found. Or, no match was found for the specified wildcard pattern.
FILE OPEN ERROR	SE13	An attempt was made to open a file which is already opened in Write or Append mode.
FILE EXIST ERROR	SE14	Specified destination file already exists. Or, a directory could not be deleted because there are files in the directory.
FILE PERMISSION ERROR	SE15	A write attempt to a destination was unsuccessful because: - the destination was being accessed; - the destination is a directory; or - the destination is read-only.
NOEMPTY ERROR	SE16	No free space is available on the disk. Or, the number of files or directories exceeded the system limit.
NO CARD ERROR	SE17	Processing is not allowed because no memory card is inserted.
CARD UNMOUNT ERROR	SE18	Processing is not allowed because no memory card is mounted.
CARD PROTECT ERROR	SE19	Processing is not allowed because the protection switch is ON.
CARD ERROR	SE20	Processing could not continue because a memory card failure was detected. Replace the memory card.
SECURITY ERROR	SE21	Security password mismatch
RUN MODE ERROR	SE22	Processing is not allowed in Run or Debug mode.
STOP MODE ERROR	SE23	Processing is not allowed in Stop mode.
CHANGE MODE ERROR	SE24	Operating mode change is not allowed. This may be because online edited changes are being written to the CPU module or because of some other reason.
PROGRAM EXECUTION MODE ERROR	SE25	Block activation is not allowed in execute-all-blocks mode.
INVALID BLOCK NAME	SE26	The specified block was not found.
FUNCTION DELETION	SE27	The function is removed in the configuration.
FTPSERVER ERROR	SE30	Processing could not continue due to an FTP server error.

■ Relationship between Virtual Directory Command Execution and EXE LED Status

The table below shows the relationship between the execution status of a virtual directory command and the status of the EXE LED.

Unlike the rotary switch function and card batch file function, the EXE LED does not blink when execution of a virtual directory command exits with error.

Table 3.7.5 Relationship between Virtual Directory Command Execution and EXE LED Status

State of Virtual Directory Command	State of LED
Running	- Lit
Normal exit	- Off
Error exit	

3.7.4   Virtual Directory Command Specifications

Detailed specifications of the virtual directory commands are described in separate subsections by command group.

- File/device conversion & transfer commands
- Device access commands
- Maintenance commands
- File operation and disk operation commands
- Card batch file execution commands

⚠ CAUTION

Virtual directory commands can handle file data of up to 2064384 bytes (about 2 megabytes). This capacity is shared by all FTP clients connected to the FTP server so the size limit for each virtual directory command is smaller when multiple virtual directory commands are executed concurrently.

A NOEMPTY ERROR (SE16) is generated if the size limit is exceeded.

3.7.5   File/Device Conversion & Transfer Commands

Table 3.7.6   List of File/Device Conversion & Transfer Commands

Instruction Name	Command Name	Function
Convert CSV File to Device	F2DCSV	Converts a CSV formatted file into device data.
Convert Device to CSV File	D2FCSV	Converts device data into a CSV formatted file.
Convert Binary File to Device	F2DBIN	Converts a binary file into device data.
Convert Device to Binary File	D2FBIN	Converts device data into a binary file.

Table 3.7.7   Terminology Description for File/Device Conversion & Transfer Commands

Term	Description
Binary file	A file containing binary data, with no delimiters.
CSV formatted file	A text file in which ASCII coded data elements are delimited by comma (,) characters or TAB characters. A CSV file can be displayed directly in Excel. Conversely, an Excel file can be converted to a CSV formatted file with some limitations. A newline is also considered as a delimiter. Beware that a newline and a contiguous delimiter character is treated as one field.
Field	A field is one data element in a CSV formatted file.
Record	A record (one line) in a CSV formatted file is delimited by a newline code. One record contains 1 to n fields.

3.7.5.1　Convert CSV File to Device (F2DCSV)

Converts data in CSV formatted file to binary data and writes the data to contiguous devices.

■ FTP Command Used

put

■ Syntax

Table 3.7.8　Command Specifications

Command Part		Syntax
Common		\VIRTUAL\CMD
Command [*1]		\F2DCSV
Parameters [*1]	1	First device for writing (7 ASCII characters max.) [device name][*2]
	2	Number of fields to be read (7 ASCII characters max.) [-1 = until file end 0 - 4194304 (if device unit = bit) 0 - 524288 (if device unit = byte) 0 - 262144 (if device unit = word) 0 - 131072 (if device unit = long word)]
	3	Field representation type (1 ASCII character) [0 = Decimal 1 = Hexadecimal 2 = Floating-point representation A ([-]d.dddd e[+/-]ddd form) 3 = Floating-point representation B ([-]dddd.dddd form)]
	4	Device unit (1 ASCII character) [0 = Bit 1 = Byte 2 = Word 3 = Long word]
	5	Sign extension (1 ASCII character) [0 = Pad with zeros 1 = Extend sign]
	6	Delimiter option (1 ASCII character) [0 = Comma (,) 1 = TAB]
	7	Newline option (1 ASCII character) [0 = CRLF 1 = LF]
	8	Write limit in words (6 ASCII characters max.) [1-262144 (words)]
File		CSV formatted file name

*1: Delimit the command and each parameter using an underscore ('_') character.
*2: The table below shows the supported devices.

Table 3.7.9　Supported Devices

	X	Y	I	E	L	M	T□	C□	D	B	W	Z	R	V	Con-stant	Index Modification	Indirect Designation, Pointer P
First device for writing		✓	✓	✓		✓	✓	✓	✓	✓		✓	✓			No	No

Note: -　TP = timer current value
-　CP = counter current value

Command Line:
```
PUT <file> <common><command>_<parameter(1)>_..._<parameter(8)>
```

■ Example

This sample command reads the CSV formatted file named "data012.csv" containing field data stored in decimal representation, and writes the number of required fields (128 words max.) as sign-extended word data to devices starting from B2001. It assumes that the file uses CRLF as newline.

```
>put data012.csv \VIRTUAL\CMD\F2DCSV_B2001_-1_0_2_1_0_0_128
```

An example of an FTP reply is shown below assuming the command was executed from a command prompt and exited with error.

```
200 PORT command successful.
150 Opening data connection.
550 Can't open virtual file[SE01 PARAMETER ERROR].
```

■ Reply

Table 3.7.10   Reply Messages

Reply Message	Code	Description
OK FIELD NUM. = xxxx	SE00	Normal exit "xxxx" indicates the number of fields processed.
Other messages	SE01,…	Error reply message

SEE ALSO

For details on error reply messages, see "■ Error Reply Messages of Virtual Directory Commands" of Subsection 3.7.3, "Using Virtual Directory Commands."

■ Function

Converts data in CSV formatted file to binary data and writes the data to contiguous devices.

- Text in decimal, hexadecimal or floating-point representation can be converted to device data. Floating-point representation is converted to IEEE single-precision floating-point representation.

  Decimal ("-128" to "255", "-32768" to "65535", "-2147483648" to "4294967295")

  Hexadecimal ("0x0" to "0xFFFFFFFF", "0" to "FFFFFFFF")

  Floating-point ([-]d.dddd e[+/-]ddd, [-]dddd.dddd, Infinite "-INF"/"+INF")
- Available device unit options are bit, byte, word and long word. You can also specify whether to perform sign extension.
- Available field delimiter options are the comma (,) and Tab characters.
- Comments can be included in the file. If a field begins with a double-quote ("), single-quote ('), two slashes (//), or a slash and an asterisk (/*), the instruction skips over all characters until it encounters a delimiter character or newline.
- Newline can be specified as CRLF (standard for Windows) or LF.

CSV formatted file

Device

```
 5, 8, 6, 30
-2, 0, 0, 8
...
```

F2DCSV

5	B2001+0
8	B2001+1
6	B2001+2
30	B2001+3
-2	B2001+4
0	B2001+5
0	B2001+6
8	B2001+7
:	:
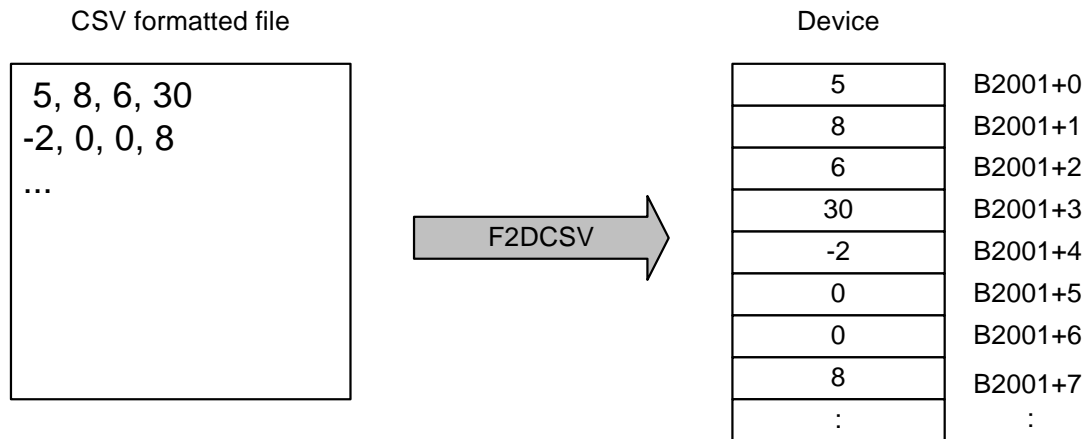
Note: Device numbers and conversion method shown are examples.

FB0212.VSD

Figure 3.7.10   CSV Formatted File to Device Conversion

TIP

Reading of File
- If end-of-file is encountered before the required number of fields is read, execution ends without error.
- A newline ends a record, and thus always ends a field.
- Within a field, any and all space characters preceding the data string are ignored, but any space character following the data string results in a field conversion error.
- '//' and other comment mark characters must always be coded at the beginning of a field. Otherwise, a conversion error will be generated.
- If NULL or other invalid binary code is encountered, execution ends with a file interpretation error.

Conversion Error and Interpretation Error
- If a conversion error is detected, 0 is written to the device. If a conversion error is detected in a field during conversion, an error is generated but processing continues.
- When the converted numeric value of a field exceeds the range of the device unit, a conversion error is generated.
- Non-numeric representation "NaN" of D2FCSV generates a conversion error.

Data Conversion and Writing to Device
- You can specify to pad with '0's or extend the sign when the converted value of a field is smaller than the size of the device unit.
- If the device unit is specified as bit, 0 is stored for a zero value while 1 is stored for any other value.
- If you specify the field representation type as floating-point representation, you must specify the device unit as long word.
- Writing to device spans multiple scan cycles.

3.7.5.2 Convert Device to CSV File (D2FCSV)

Converts device data to text and outputs a CSV formatted file.

■ FTP Command Used

get

■ Syntax

Table 3.7.11 Command Specifications

Command Part		Syntax
Common		\VIRTUAL\CMD
Command [1]		\D2FCSV
Parameters [1]	1	First device for reading (7 ASCII characters max.) [device name] [2]
	2	Device unit (1 ASCII character) [0 = bit 1 = byte 2 = word 3 = long word]
	3	Number of data units to be read (7 ASCII characters max.) [0 - 4194304 (if device unit = bit) 0 - 524288 (if device unit = byte) 0 - 262144 (if device unit = word) 0 - 131072 (if device unit = long word)]
	4	Field representation type (1 ASCII character) [0 = Decimal 1 = Hexadecimal 2 = Floating-point representation A ([-]d.dddd e[+/-]ddd form) 3 = Floating-point representation B ([-]dddd.dddd form)]
	5	Field length (2 ASCII characters max.) [0 = automatic (as required after conversion) 1-13 = fixed field length in characters]
	6	Field space handling (1 ASCII character) [[3] 0 = Pad with spaces 1 = Pad with zeros]
	7	Delimiter option (1 ASCII character) [0 = comma (,) 1 =TAB]
	8	Newline option (1 ASCII character [0 = CRLF 1 = LF]
	9	Newline insertion position (5 ASCII characters max.) [0 = Do not insert newline 1 - 32767 = Insert newline after n fields]
File		Output file name

[1]: Delimit the command and each parameter using an underscore ('_') character.
[2]: The table below shows the supported devices.
[3]: If the field length is specified as 0 (automatic), this parameter is ignored but a valid dummy value (say, 0) must still be specified.

Table 3.7.12 Supported Devices

	X	Y	I	E	L	M	T□	C□	D	B	W	Z	R	V	Con-stant	Index Modification	Indirect Designation, Pointer P
First device for reading	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		No	No

Note: - TP = timer current value; TS = timer preset value
      - CP = counter current value; CS = counter preset value

Command Line:
```
GET <common><command>_<parameter(1)>_..._<parameter(9)> <file>
```

■ Example

This sample command gets and stores device data starting from B2001 to a CSV formatted file in the current directory according to the conditions given below.

- Device unit   Word
- Number of data units to be read   128
- Field representation type   Decimal
- Field length   Automatic
- Field space handling   Pad with spaces
- Delimiter option   Comma
- Newline option   CRLF
- Newline insertion position   4

```
>get \VIRTUAL\CMD\D2FCSV_B2001_2_128_0_0_1_0_0_4 data012.csv
```

An example of an FTP reply is shown below assuming the command was executed from a command prompt and exited with error.
```
200 PORT command successful.
550 Can't open virtual file[SE01 PARAMETER ERROR].
```

■ Reply

Table 3.7.13 Reply Messages

Reply Message	Code	Description
OK DATA NUM. = xxxx	SE00	Normal exit "xxxx" indicates the number of fields processed.
Other messages	SE01,…	Error reply message

SEE ALSO

For details on error reply messages, see "■ Error Reply Messages of Virtual Directory Commands" of Subsection 3.7.3, "Using Virtual Directory Commands."

■ Function

Converts device data to text and outputs a CSV formatted file.

- Available device unit options for reading are bit, byte, word and long word.
- Device data can be converted to text in decimal, hexadecimal or floating-point representation after reading.
  Decimal ("0" to "1", "-128" to "127", "-32768" to "32767", "-2147483648" to "2147483647")
  Hexadecimal ("0" to "FFFFFFFF")
  Floating-point ([-]d.dddd e[+/-]ddd, [-]dddd.dddd, infinity "-INF" or "+INF", non-numeric "NaN")
- You can specify the field length in characters for text conversion.
- You can specify whether to pad with space characters or pad with zeros if the converted text is shorter than the specified field length.
- Available field delimiter options are the comma (,) and Tab characters.
- Newline can be specified as CRLF (standard for Windows) or LF.
- The number of fields in one record (from line beginning to line end) can be specified.
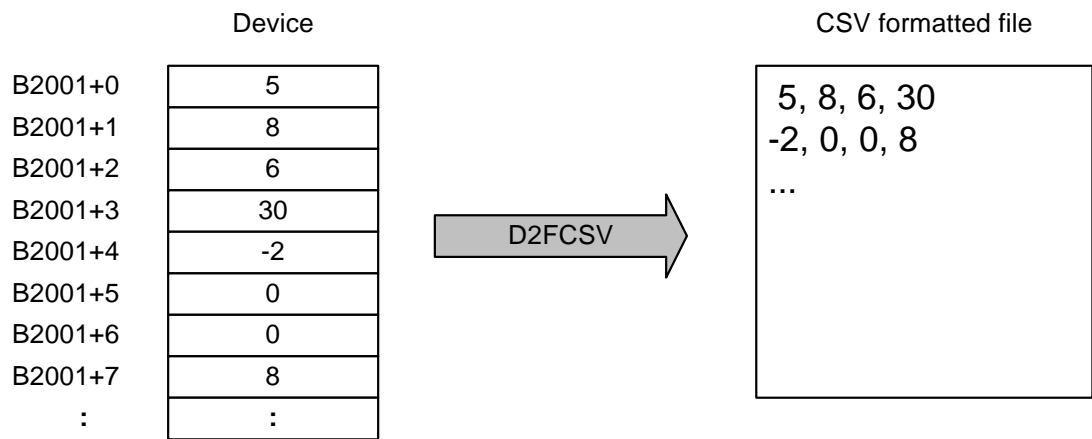
Note: Device numbers and conversion method shown are examples.

FB0213.VSD

Figure 3.7.11   Device to CSV Formatted File Conversion

TIP

Reading of Device Data
- Reading of data from devices spans multiple scan cycles.
- If you specify the field representation type as floating-point representation, you must specify the device unit as long word for devices storing IEEE single-precision floating-point numbers.

Conversion Error and Interpretation Error
- If a conversion error occurs, "ERR" is written to the field. If a conversion error is detected for a field during conversion, an error is generated but processing continues.
- If the converted text string is longer than the specified field length, a conversion error is generated.

Data Conversion
- If the field representation type is specified as decimal, the sign is included in the output digit count.
- In floating-point representation B, there are always 6 digits after the decimal point. Numeric values smaller than 0.000001 are rounded to 0.

3.7.5.3 Convert Binary File to Device (F2DBIN)

Converts data in binary file and writes the data to contiguous devices using the specified data unit.

■ FTP Command Used

put

■ Syntax

Table 3.7.14 Command Specifications

Command Part		Syntax
Common		\VIRTUAL\CMD
Command [*1]		\F2DBIN
Parameters [*1]	1	First device for writing (7 ASCII characters max.) [device name] [*2]
	2	Number of data units to be read (7 ASCII characters max.) [-1 = Until file end 0 - 4194304 (if device unit = bit) 0 - 524288 (if device unit = byte) 0 - 262144 (if device unit = word) 0 - 131072 (if device unit = long word)]
	3	Data unit (1 ASCII character) [1 = byte 2 = word 3 = long word]
	4	Device unit (1 ASCII character) [0 = bit 1 = byte 2 = word 3 = long word]
	5	Sign extension (1 ASCII character) [0 = Pad with zeros 1 = Extend sign]
	6	Write limit in words (6 ASCII characters max.) [1 - 262144 (words)]
File		Binary file name

*1: Delimit the command and each parameter using an underscore ('_') character.
*2: The table below shows the supported devices.

Table 3.7.15 Supported Devices

	X	Y	I	E	L	M	T▢	C▢	D	B	W	Z	R	V	Con-stant	Index Modification	Indirect Designation, Pointer P
First device for writing		✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓		No	No

Note: - TP = timer current value
- CP = counter current value

Command Line:
```
PUT <file> <common><command>_<parameter(1)>_..._<parameter(6)>
```

■ Example

This sample command reads all data (128 words max.) contained in the binary file named "data012.bin" in word units and writes the word data to devices starting from B2001. The sign extension parameter has no significance in this example.

```
>put data012.bin \VIRTUAL\CMD\F2DBIN_B2001_-1_2_2_1_128
```

An example of an FTP reply is shown below assuming the command was executed from a command prompt and exited with error.

```
200 PORT command successful.

150 Opening data connection.

550 Can't open virtual file[SE01 PARAMETER ERROR].
```

■ Reply

Table 3.7.16   Reply Messages

Reply Message	Code	Description
OK DATA NUM. = xxxx	SE00	Normal exit "xxxx" indicates the number of data units processed.
Other messages	SE01,…	Error reply message

SEE ALSO

For details on error reply messages, see "■ Error Reply Messages of Virtual Directory Commands" of Subsection 3.7.3, "Using Virtual Directory Commands."

■ Function

Converts data in binary file and writes the data to contiguous devices using the specified data unit.

Note: Device numbers and conversion method shown are examples.

FB0214.VSD

Figure 3.7.12   Binary File to Device Conversion

TIP

- If a conversion error occurs, 0 is written to the device. If a conversion error is detected during conversion, an error is generated but processing continues.
- If the specified data unit is larger than the specified device unit, a conversion error is generated.
- If the device unit is specified as bit, 0 is stored for a zero value while 1 is stored for any other value.
- If the specified data unit is smaller than the specified device unit, you can specify to pad with '0's or extend the sign.
- If end-of-file is encountered before the required number of data units are read, execution ends without error.
- Writing to device spans multiple scan cycles.

3.7.5.4   Convert Device to Binary File (D2FBIN)

Converts device data to a binary file.

■ FTP Command Used

get

■ Syntax

Table 3.7.17   Command Specifications

Command Part		Syntax
Common		\VIRTUAL\CMD
Command *1		\D2FBIN
Parameters *1	1	First device for reading (7 ASCII characters max.) [device name][2]

	2	Number of data units to be read (ASCII) [
		  0 -4194304 (if device data unit = bit)
		  0 - 524288 (if device data unit = byte)
		  0 - 262144 (if device data unit = word)
		  0 - 131072 (if device data unit = long word)
		]

	3	Device data unit (1 ASCII character) [
		  0 = bit
		  1 = byte
		  2 = word
		  3 = long word
		]

	4	File data unit (1 ASCII character) [
		  1 = byte
		  2 = word
		  3 = long word
		]

	5	Sign extension (1 ASCII character) [
		  0 = Pad with zeros
		  1 = Extend sign
		]

File		Output file name

*1:   Delimit the command and each parameter using an underscore ('_') character.
*2:   The table below shows the supported devices.

Table 3.7.18   Supported Devices

	X	Y	I	E	L	M	T□	C□	D	B	W	Z	R	V	Con-stant	Index Modification	Indirect Designation, Pointer P
First device to be read	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		No	No

Note: -   TP = timer current value; TS = timer preset value
      -   CP = counter current value; CS = counter preset value

Command Line:
  GET *<common><command>_<parameter(1)>_..._<parameter(5)> <file>*

■ Example

This sample command writes device data starting from B2001 to a binary file named "data012.bin" in the current directory according to the conditions given below.

- Number of data units to be read   128
- Device data unit   Word
- File data unit   Word
- Sign extension   Extend sign (has no significance in this example)

```
>get \VIRTUAL\CMD\D2FBIN_B2001_128_2_2_1 data012.bin
```

An example of an FTP reply is shown below assuming the command was executed from a command prompt and exited with error.

```
200 PORT command successful.
550 Can't open virtual file[SE01 PARAMETER ERROR].
```

■ Reply

Table 3.7.19 Reply Messages

Reply Message	Code	Description
OK DATA NUM. = xxxx	SE00	Normal exit "xxxx" indicates the number of data units processed.
Other messages	SE01,…	Error reply message

SEE ALSO

For details on error reply messages, see "■ Error Reply Messages of Virtual Directory Commands" of Subsection 3.7.3, "Using Virtual Directory Commands."

■ Function

Converts device data to a binary file.
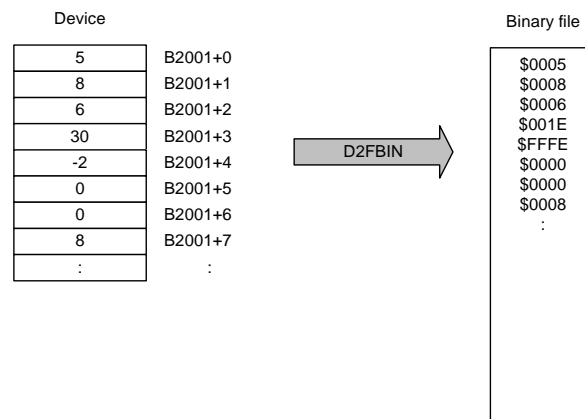
Note: Device numbers and conversion method shown are examples.

FB0215.VSD

Figure 3.7.13 Device Data to Binary File Conversion

TIP

- If a conversion error occurs, 0 is written to the file. If a conversion error is detected during conversion, an error is generated but processing continues.
- If the specified device unit is larger than the specified file unit, a conversion error is generated.
- If the specified device unit is smaller than the specified file unit, you can specify to pad with '0's or extend the sign.
- Reading of data from devices spans multiple scan cycles.

3.7.6   Device Access Commands

Table 3.7.20   List of Device Access Commands

Function Name	Command Name	Function
Bit access	BRD	Reads bits.
	BWR	Writes bits.
	BFL	Writes bits of the same data.
Word access	WRD	Reads words.
	WWR	Writes words.
	WFL	Writes words of the same data.

3.7.6.1   Bit Read (BRD)

Reads bit data from consecutive relay devices.

■ Syntax

Table 3.7.21   Command Specifications

Command Part		Syntax
Common		\VIRTUAL\CMD
Command [*1]		\BRD
Parameters [*1]	1	First device name (7 ASCII characters max.) [device name][*2]
	2	Number of bits (3 ASCII characters max.) [1-256]
File		Data file name

*1:   Delimit the command and each parameter using an underscore ('_') character.
*2:   The table below shows the supported devices.

Table 3.7.22   Supported Devices

	X	Y	I	E	L	M	T	C	D	B	W	Z	R	V	Con-stant	Index Modification	Indirect Designation, Pointer P
First device	✓	✓	✓	✓	✓	✓										No	No

Command Line:

```
GET <common><command>_<parameter(1)>_<parameter(2)> <file>
```

■ Example

This sample command reads 8 bits of data starting from device I223 and stores the data in binary representation in a text file named "res.txt."

```
>get \VIRTUAL\CMD\BRD_I223_8 res.txt
```

Assuming that devices I223 to I230 contain the values {0,1,0,1,1,1,1,1}, the contents of the data file will be:

```
01011111
```

An example of an FTP reply is shown below assuming the command was executed from a command prompt and exited with error.

```
200 PORT command successful.
550 Can't open virtual file[SE01 PARAMETER ERROR].
```

■ Reply

Table 3.7.23   Reply Messages

Reply Message	Code	Description
OK	SE00	Normal exit
Other messages	SE01,…	Error reply message

SEE ALSO

For details on error reply messages, see "■ Error Reply Messages of Virtual Directory Commands" of Subsection 3.7.3, "Using Virtual Directory Commands."

■ Function

Reads bit data from consecutive relay devices. The data read is output in binary representation to a text file. Character '1' (ASCII code=$31) is output to the file if a relay device is ON while character '0' (ASCII code=$30) is output if a relay device is OFF.

3.7.6.2   Bit Write (BWR)

Writes bit data into consecutive relay devices.

■ Syntax

Table 3.7.24   Command Specifications

Command Part		Syntax
Common		\VIRTUAL\CMD
Command *1		BWR
Parameters *1	1	First device name (7 ASCII characters max.) [device name]*2
	2	Number of bits (3 ASCII characters max.) [1-128]
	3	Write data (1 ASCII character x specified Number of bits) [string of '0's and '1's]
File		Dummy file name

*1: Delimit the command and each parameter using an underscore ('_') character.
*2: The table below shows the supported devices.

Table 3.7.25   Supported Devices

	X	Y	I	E	L	M	T	C	D	B	W	Z	R	V	Con-stant	Index Modification	Indirect Designation, Pointer P
First device		✓	✓	✓	✓											No	No

Command Line:
```
GET <common><command>_<parameter(1)>_..._<parameter(3)> <file>
```

■ Example

This sample command writes 8 bits of data ("00110001") to device, starting from device I223. It specifies the dummy file name as "dmy.txt".
```
>get \VIRTUAL\CMD\BWR_I223_8_00110001 dmy.txt
```

An example of an FTP reply is shown below assuming the command was executed from a command prompt and exited with error.
```
200 PORT command successful.
550 Can't open virtual file[SE01 PARAMETER ERROR].
```

■ Reply

Table 3.7.26   Reply Messages

Reply Message	Code	Description
OK	SE00	Normal exit
Other messages	SE01, …	Error reply message

SEE ALSO

For details on error reply messages, see "■ Error Reply Messages of Virtual Directory Commands" of Subsection 3.7.3, "Using Virtual Directory Commands."

■ Function

Writes bit data into consecutive relay devices. Specify a string of '1' and '0' characters in the parameters part to turn on or turn off each relay device accordingly.

3.7.6.3 Bit Fill (BFL)

Writes the same bit value into a specified number of consecutive relay devices.

■ Syntax

Table 3.7.27 Command Specifications

Command Part		Syntax
Common		\VIRTUAL\CMD
Command [*1]		BFL
Parameters [*1]	1	First device name (7 ASCII characters max.) [device name] [*2]
	2	Number of bits (3 ASCII characters max.) [1-256]
	3	Write data (1 ASCII character) [0 or 1]
File		Dummy file name

*1: Delimit the command and each parameter using an underscore ('_') character.
*2: The table below shows the supported devices.

Table 3.7.28 Supported Devices

	X	Y	I	E	L	M	T	C	D	B	W	Z	R	V	Con-stant	Index Modification	Indirect Designation, Pointer P
First device		✓	✓	✓	✓											No	No

Command Line:

```
GET <common><command>_<parameter(1)>_..._<parameter(3)> <file>
```

■ Example

This sample command writes bit value 0 to 160 relay devices, starting from device I223. It specifies the dummy file name as "dmy.txt".

```
>get \VIRTUAL\CMD\BFL_I223_160_0 dmy.txt
```

An example of an FTP reply is shown below assuming the command was executed from a command prompt and exited with error.

```
200 PORT command successful.
550 Can't open virtual file[SE01 PARAMETER ERROR].
```

■ Reply

Table 3.7.29 Reply Messages

Reply Message	Code	Description
OK	SE00	Normal exit
Other messages	SE01, …	Error reply message

SEE ALSO

For details on error reply messages, see "■ Error Reply Messages of Virtual Directory Commands" of Subsection 3.7.3, "Using Virtual Directory Commands."

■ Function

Writes the same bit value into a specified number of consecutive relay devices. Specify the write data as one digit in binary representation. Specify 1 or 0 in the parameters part to turn on or turn off the specified number of relay devices.

3.7.6.4   Word Read (WRD)

Reads word data from consecutive register devices.

■ Syntax

Table 3.7.30   Command Specifications

Command Part		Syntax
Common		\VIRTUAL\CMD
Command [*1]		WRD
Parameters [*1]	1	First device name (7 ASCII characters max.) [device name][*2]
	2	Number of words (2 ASCII characters max.) [01 to 64 in decimal representation]
File		Data file name

*1: Delimit the command and each parameter using an underscore ('_') character.
*2: The table below shows the supported devices.

Table 3.7.31   Supported Devices

	X	Y	I	E	L	M	T□	C□	D	B	W	Z	R	V	Constant	Index Modification	Indirect Designation, Pointer P
First device	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		No	No

Note: -   TP=timer current value; TS=timer preset value; TI=timer current value (count-up type)
-    CP=counter current value; CS=counter preset value; CI=counter current value (count-up type)

Command Line:
```
GET <common><command>_<parameter(1)>_<parameter(2)> <file>
```

■ Example

This sample command reads 4 words of data starting from register D101 and stores the data in hexadecimal representation in a text file named "regstat.txt."

```
>get \VIRTUAL\CMD\WRD_D101_4 regstat.txt
```

Assuming D101=$1, D102=$2, D103=$3, D104=$4, the content of the data file (regstat.txt) will be:

```
0001000200030004
```

An example of an FTP reply is shown below assuming the command was executed from a command prompt and exited with error.

```
200 PORT command successful.
550 Can't open virtual file[SE01 PARAMETER ERROR].
```

■ Reply

Table 3.7.32   Reply Messages

Reply Message	Code	Description
OK	SE00	Normal exit
Other messages	SE01, …	Error reply message

SEE ALSO

For details on error reply messages, see "■ Error Reply Messages of Virtual Directory Commands" of Subsection 3.7.3, "Using Virtual Directory Commands."

■ Function

Reads word data from consecutive register devices. The word data is output to a text file contiguously with 4 hexadecimal digits per data word.

3.7.6.5 Word Write (WWR)

Writes word data into consecutive register devices.

■ Syntax

Table 3.7.33 Command Specifications

Command Part		Syntax
Common		\VIRTUAL\CMD
Command *1		WWR
Parameters *1	1	First device name (7 ASCII characters max.) [device name][2]
	2	Number of words (2 ASCII characters max.) [01 to 32 in decimal representation]
	3	Write data (4 ASCII characters x specified number of words) [0000 to FFFF in hexadecimal representation (for the specified number of words)]
File		Dummy file name

*1: Delimit the command and each parameter using an underscore ('_') character.
*2: The table below shows the supported devices.

Table 3.7.34 Supported Devices

	X	Y	I	E	L	M	T□	C□	D	B	W	Z	R	V	Con-stant	Index Modification	Indirect Designation, Pointer P
First device		✓	✓	✓	✓		✓	✓	✓	✓	✓		✓	✓		No	No

Note: - TP = timer current value; TI = timer current value (count-up type)
- CP=counter current value; CI=counter current value (count-up type)

Command Line:
```
GET <common><command>_<parameter(1)>_..._<parameter(3)> <file>
```

■ Example

This sample command writes 4 words of data ($9096, $AA01, $0000, $8001) to device, starting from device D101. It specifies the dummy file name as "dmy.txt".

```
>get \VIRTUAL\CMD\WWR_D101_4_9096_AA01_0000_8001 dmy.txt
```

An example of an FTP reply is shown below assuming the command was executed from a command prompt and exited with error.

```
200 PORT command successful.
550 Can't open virtual file[SE01 PARAMETER ERROR].
```

■ Reply

Table 3.7.35 Reply Messages

Reply Message	Code	Description
OK	SE00	Normal exit
Other messages	SE01, …	Error reply message

SEE ALSO

For details on error reply messages, see "■ Error Reply Messages of Virtual Directory Commands" of Subsection 3.7.3, "Using Virtual Directory Commands."

■ Function

Writes word data into consecutive register devices. In the parameters part, specify the data to be written contiguously, with 4 hexadecimal digits per word, for the specified number of words.

3.7.6.6   Word Fill (WFL)

Writes the same word data into a specified number of consecutive register devices.

■ Syntax

Table 3.7.36   Command Specifications

Command Part		Syntax
Common		\VIRTUAL\CMD
Command *1		WFL
Parameters *1	1	First device name (7 ASCII characters max.) [device name] [2]
	2	Number of words (3 ASCII characters max.) [1 to 256 in decimal representation]
	3	Write data (4 ASCII character) [0000 to FFFF in hexadecimal representation]
File		Dummy file name

*1:   Delimit the command and each parameter using an underscore ('_') character.
*2:   The table below shows the supported devices.

Table 3.7.37   Supported Devices

	X	Y	I	E	L	M	T□	C□	D	B	W	Z	R	V	Con-stant	Index Modification	Indirect Designation, Pointer P
First device		✓	✓	✓	✓		✓	✓	✓	✓	✓		✓	✓		No	No

Note: -   TP = timer current value; TI = timer current value (count-up type)
     -   CP=counter current value; CI=counter current value (count-up type)

Command Line:
```
GET <common><command>_<parameter(1)>_..._<parameter(3)> <file>
```

■ Example

This sample command writes word value $0000 to 160 register devices, starting from device D101. It specifies the dummy file name as "dmy.txt".
```
>get \VIRTUAL\CMD\WFL_D101_160_0000 dmy.txt
```

An example of an FTP reply is shown below assuming the command was executed from a command prompt and exited with error.
```
200 PORT command successful.
550 Can't open virtual file[SE01 PARAMETER ERROR].
```

■ Reply

Table 3.7.38   Reply Messages

Reply Message	Code	Description
OK	SE00	Normal exit
Other messages	SE01, …	Error reply message

SEE ALSO

For details on error reply messages, see "■ Error Reply Messages of Virtual Directory Commands" of Subsection 3.7.3, "Using Virtual Directory Commands."

■ Function

Writes the same word data to a specified number of consecutive register devices. In the Parameters part, specify the data to be written as 4 hexadecimal digits.

3.7.7　Maintenance Commands

Table 3.7.39　List of Maintenance Commands

Function Name	Command Name	Function
Load Project	LOAD	Loads project or CPU properties.
Save Project	SAVE	Saves project or CPU properties.
Get Log	LOG	Gets various log files.
CPU Info	CPUINFO	Gets CPU module information.
Application Info	APINFO	Gets user application information.
Run Mode	RUN	Switches operating mode to Run mode.
Stop Mode	STOP	Switches operating mode to Stop mode.
Activate Block	ACT	Activates a specified block. You may also activate a sensor control block.
Inactivate Block	INACT	Inactivates a specified block. You may also inactivate a sensor control block.
Reset CPU	CPURESET	Resets the CPU.
Clear Alarms	ALMCLEAR	Clears all alarms, and returns a log of cleared alarms.
Help	HELP	Gets help information on card batch file commands.

3.7.7.1 Load Project (LOAD)

Loads a project file of card load format (file extension ".ypjc") or a CPU property file (file extension ".yprp") into the internal ROM of the module when used in a put command.

■ FTP Command Used

put

■ Syntax

Table 3.7.40 Command Specifications

Command Part	Syntax
Common	\VIRTUAL\CMD
Command [1]	\LOAD

Parameters [1]	Extension of file to be loaded (13 ASCII characters max.) [[2]
	   .ypjc = card load format project file name extension
	   xxxxxxxx.yprp = CPU property file name
	]

File	Card load format project file name (with file extension "ypjc") or CPU property file name (with file extension "yprp")

*1: Delimit the command and each parameter using an underscore ('_') character.
*2: "xxxxxxxx" can be any filename. This file name will be registered as the name of the CPU property data and the CPU property file name when the CPU property file is uploaded or saved. When loading a project, specify only ".ypjc" as even if a file name is specified, the project name used for creating the card load format project will be registered.

Command Line:
```
PUT <file> <common><command>_<parameter>
```

■ Example

The first sample command loads a project file of card load format named "myproj.ypjc"; The second sample command loads a CPU property file named "myprop.yprop".
```
>put myproj.ypjc \VIRTUAL\CMD\LOAD_.ypjc

>put myprop.yprp \VIRTUAL\CMD\LOAD_myprop.yprp
```

An example of an FTP reply is shown below assuming the command was executed from a command prompt and exited with error.
```
200 PORT command successful.

150 Opening data connection.

550 Can't open virtual file[SE01 PARAMETER ERROR].
```

■ Reply

Table 3.7.41 Reply Messages

Reply Message	Code	Description
OK	SE00	Normal exit
Other messages	SE01, …	Error reply message

SEE ALSO

For details on error reply messages, see "■ Error Reply Messages of Virtual Directory Commands" of Subsection 3.7.3, "Using Virtual Directory Commands."

■ Function

Loads a project file of card load format (file extension ".ypjc") or a CPU property file (file extension ".yprp") into the internal ROM of the module when used in a put command.

Operating mode dependency
Project loading is available only in Stop mode. CPU property loading is available in any operating mode.

The operating mode before command execution is retained after command execution.

Protection
If executable program protection is enabled for the project stored in the internal ROM, the project to be loaded must be protected by the same password for loading to succeed. Block protection alone has no effect on project loading.

If CPU property data stored in the internal ROM is protected with a keyword, the property file to be loaded must be set with the same keyword for loading to succeed.

If an error occurs
The operating mode right before execution remains in effect.

The table below shows the state of the data in the internal ROM in the event of an error.

Table 3.7.42   Internal ROM Contents after an Error

Error	Data of Internal ROM
PARAMETER ERROR	The data before execution is retained.
RUN MODE ERROR	
SECURITY ERROR	
INVALID FILE (CPU property file)	The CPU property data before execution is retained.
INVALID FILE (project file)	Project programs are cleared.
FTPSERVER ERROR	Unpredictable

3.7.7.2   Save Project (SAVE)

Gets project or CPU property data stored in the internal ROM of the module when used in a get command.

■ FTP Command Used

get

■ Syntax

Table 3.7.43   Command Specifications

Command Part		Syntax
Common		\VIRTUAL\CMD
Command [1]		\SAVE
Parameters [1]	1	Security password of executable program (8 ASCII characters max.) [2] or Security keyword for CPU properties (8 ASCII characters max.)

	2	Filename extension (5 ASCII characters)[
		        .ypjc = project file of card load format
		        .yprp = CPU property file
		]

File		Project file name (with extension "ypjc") or CPU property file name (with extension "yprp")

[1]:   Delimit the command and each parameter using an underscore ('_') character.
[2]:   This parameter cannot be omitted even if the executable program or CPU property data is not protected. In this case, you can specify any valid dummy text string.

Command Line:
```
GET <common><command>_<parameter(1)>_<parameter(2)> <file>
```

■ Example

This sample command gets and saves an unprotected project file of card load format to a file named "myprj.ypjc".
```
>get \VIRTUAL\CMD\SAVE_aaa_.ypjc myprj.ypjc
```

This sample command gets and saves a CPU property file protected with keyword "yokogawa" to a file named "myprop.yprp".
```
>get \VIRTUAL\CMD\SAVE_yokogawa_.yprp myprop.yprp
```

An example of an FTP reply is shown below assuming the command was executed from a command prompt and exited with error.
```
200 PORT command successful.
550 Can't open virtual file[SE01 PARAMETER ERROR].
```

■ Reply

Table 3.7.44   Reply Messages

Reply Message	Code	Description
OK	SE00	Normal exit
Other messages	SE01, …	Error reply message

SEE ALSO

For details on error reply messages, see "■ Error Reply Messages of Virtual Directory Commands" of Subsection 3.7.3, "Using Virtual Directory Commands."

■ Function

Gets project or CPU property data stored in the internal ROM of the module when used in a get command.

When used in a get command, this virtual directory command saves project or CPU property data stored in the internal ROM as a file in card load format or a CPU property file using the filename specified in the File part of the command.

You can specify whether to get a project file or CPU property file using the filename extension parameter. To get a project file, specify ".ypjc"; to get a CPU property file, specify ".yprp".

TIP

You may save the data using any filename but the project name remains the same as at the time of loading.

Protection

If executable program protection is enabled for the project stored in the internal ROM, you must specify a valid password as a command parameter. If the password is invalid, the command returns an error without saving the file.

Block protection, even if enabled, is ignored during saving.

If the CPU property data is protected with a keyword, you must specify a valid keyword as a command parameter for saving to succeed.

For security reasons, no keyword is output to the saved CPU property file.

3.7.7.3   Get Log (LOG)

Gets and saves log information as a text file.

■ FTP Command Used

get

■ Syntax

Table 3.7.45   Command Specifications

Command Part	Syntax
Common	\VIRTUAL\CMD
Command *1	\LOG

Parameters *1	Log type [
	    0 = system log
	    1 = FTP server log
	]

File	Log file name

*1: Delimit the command and each parameter using an underscore ('_') character.

Command Line:

```
GET <common><command>_<parameter> <file>
```

■ Example

This sample command saves the system log in a file named "cpu003.txt".

```
>get \VIRTUAL\CMD\LOG_0 cpu003.txt
```

An example of an FTP reply is shown below assuming the command was executed from a command prompt and exited with error.

```
200 PORT command successful.
550 Can't open virtual file[SE01 PARAMETER ERROR].
```

■ Reply

Table 3.7.46   Reply Messages

Reply Message	Code	Description
OK	SE00	Normal exit
Other messages	SE01, …	Error reply message

SEE ALSO

For details on error reply messages, see "■ Error Reply Messages of Virtual Directory Commands" of Subsection 3.7.3, "Using Virtual Directory Commands."

■ Function

Gets and saves log data in a text file. You can get the following types of log data:
- System log (messages logged for error events, power on/off events, etc.)
- FTP server log (execution log of the FTP server)

3.7.7.4   CPU Info (CPUINFO)

Gets and saves CPU information in a text file.

■ FTP Command Used

get

■ Syntax

Table 3.7.47   Command Specifications

Command Part	Syntax
Common	\VIRTUAL\CMD
Command	\CPUINFO
Parameters	–
File	CPU information file name

Command Line:
```
GET <common><command> <file>
```

■ Example

This sample command saves CPU information in a file named "mycpuinfo.txt".
```
>get \VIRTUAL\CMD\CPUINFO mycpuinfo.txt
```

An example of an FTP reply is shown below assuming the command was executed from a command prompt and exited with error.
```
200 PORT command successful.
550 Can't open virtual file[SE01 PARAMETER ERROR].
```

■ Reply

Table 3.7.48   Reply Messages

Reply Message	Code	Description
OK	SE00	Normal exit
Other messages	SE01, …	Error reply message

SEE ALSO

For details on error reply messages, see "■ Error Reply Messages of Virtual Directory Commands" of Subsection 3.7.3, "Using Virtual Directory Commands."

■ Function

Gets CPU information and saves it in a text file. The table below lists the returned CPU information.

Table 3.7.49　　Overview of CPU Information

CPU Information	Data[*1]	Data Range
Nameplate information	MODEL = *CPU type* SERIAL NO. = *Serial number* DATE = *Date of manufacture* MAC ID = *MAC address* FIRMWARE REV. = *Revision no.*	*CPU type* [F3SP66-4S, F3SP67-6S]. *Serial number* [3 alphanumeric characters and 6 numeric characters] *Date of manufacture* [YY/MM/DD] *MAC address* [12-digit hexadecimal number] *Revision no.* [starts with R00]
Operating mode	PROGRAM MODE= *Operating mode*	*Operating mode* [　0 = Stop mode 　1 = Run mode 　2 = Debug mode]
LED status	RDY LED = *LED status* RUN LED = *LED status* ALM LED = *LED status* ERR LED = *LED status* SD LED = *LED status* EXE LED = *LED status* US1 LED = *LED status* US2 LED = *LED status*	*LED status* [　0 = Off 　1 = Lit 　2 = Blinking]
MODE switch status	MODE SW = *MODE switch value*	*MODE switch value* [0 to F]
CARD1 mount status	CARD1 MOUNT STATUS = *Mount status*	*Mount status* [　0 = Unmounted 　1 = Mounted]
CARD1 free space	CARD1 FREE SPACE = *Free space*	*Free space* [bytes]
CARD1 capacity	CARD1 TOTAL SIZE = *Capacity*	*Capacity* [bytes]
RAM disk free space	RAMDISK FREE SPACE = *Free space*	*Free space* [bytes]
RAM disk capacity	RAMDISK TOTAL SIZE = *Capacity*	*Capacity* [bytes]
Alarm status	*Alarm name*	*Alarm name* (the same as that displayed by the alarm monitor)
Block activation status	*Block name 1* = *Activation status* 　　　　: *Block name n* = *Activation status*	*Block name* [up to 8 ASCII characters] *Activation status* [　0 = Inactive, 　1 = Active]

*1: The data range of each italicized item is given in the "Data Range" column.

3.7.7.5 Application Info (APINFO)

Gets application information (project information, configuration, I/O setup) in text format.

■ FTP Command Used

get

■ Syntax

Table 3.7.50   Command Specifications

Command Part	Syntax
Common	\VIRTUAL\CMD
Command	\APINFO
Parameters [*1]	Password of executable program (8 ASCII characters max.)
File	Application information file name

*1: This parameter cannot be omitted even if the executable program is not protected. In this case, you can specify any valid dummy text string.

Command Line:

```
GET <common><command>_<parameter> <file>
```

■ Example

This sample program gets and saves application information of a project whose executable program is not protected in a file named "myapr.txt".

```
>get \VIRTUAL\CMD\APINFO_aaa myapr.txt
```

An example of an FTP reply is shown below assuming the command was executed from a command prompt and exited with error.

```
200 PORT command successful.
550 Can't open virtual file[SE01 PARAMETER ERROR].
```

■ Reply

Table 3.7.51   Reply Messages

Reply Message	Code	Description
OK	SE00	Normal exit
Other messages	SE01, …	Error reply message

SEE ALSO

For details on error reply messages, see "■ Error Reply Messages of Virtual Directory Commands" of Subsection 3.7.3, "Using Virtual Directory Commands."

■ Function

Returns application information (project information, configuration, I/O setup) in text format. If the executable program of the project is protected, you must specify a valid password in the Parameters part of the command.

(1) Project information

The following project information is output in text format.

Table 3.7.52 System Information – Project Information

Project Information	Data[1]	Data Range
CPU model	CPU TYPE = *CPU type*	*CPU type* [up to 9 ASCII characters]
Name of stored project	PROJECT NAME = *Project name*	*Project name* [up to 8 ASCII characters]
Name of stored CPU property data	CPU PROPERTY NAME = *CPU property name*	*CPU property name* [up to 255 ASCII characters]
Number of stored program steps	PROGRAM STEP = *Number of steps*	*Number of steps* [0 to maximum limit for the module]
Number of component blocks	BLOCK NUM = *Number of blocks*	*Number of blocks* [1 to 1024]
Names of component blocks	BLOCK NAME 1 = *Name of block 1* : BLOCK NAME n = *Name of block n*	*Name of block n* [up to 8 ASCII characters] (n is between 1 and 1024)
Names of registered macros[2]	MACRO NAME 1 = *Name of macro 1* : MACRO NAME n = *Name of macro n*	*Name of macro n* [up to 8 ASCII characters] (n is between 1 and 256)

*1: The data range of each italicized item is given in the "Data Range" column.
*2: Information is not stored for unregistered macros.

(2) I/O setup information

I/O setup information for each slot of a unit is output as one set of comma-delimited elements consisting of type, number of X points, number of Y points and number of registers as shown below. I/O setup information is available for up to 8 units each consisting of up to 16 slots. Information for each unit is delimited by the newline code.

Table 3.7.53 System Information – I/O Setup Information

Unit	Sequence	Element	Format
UNIT0	1	Module type of SLOT1	4 ASCII characters
		Number of X points of SLOT1	2-digit decimal number
		Number of Y points of SLOT1	2-digit decimal number
		Number of registers of SLOT1	5-digit decimal number
		:	:
		Module type of SLOT16	4 ASCII characters
		Number of X points of SLOT16	2-digit decimal number
		Number of Y points of SLOT16	2-digit decimal number
		Number of registers of SLOT16	5-digit decimal number
UNIT1	2	Refer to the description for UNIT0.	Refer to the description for UNIT0.
UNIT2	3	Refer to the description for UNIT0.	Refer to the description for UNIT0.
UNIT3	4	Refer to the description for UNIT0.	Refer to the description for UNIT0.
UNIT4	5	Refer to the description for UNIT0.	Refer to the description for UNIT0.
UNIT5	6	Refer to the description for UNIT0.	Refer to the description for UNIT0.
UNIT6	7	Refer to the description for UNIT0.	Refer to the description for UNIT0.
UNIT7	8	Refer to the description for UNIT0.	Refer to the description for UNIT0.

(3) Configuration information

Configuration information is output in text format.

3.7.7.6 Run Mode (RUN)

Switches the operating mode to Run mode.

■ FTP Command Used

get

■ Syntax

Table 3.7.54   Command Specifications

Command Part	Syntax
Common	\VIRTUAL\CMD
Command	\RUN
Parameters	–
File	Dummy file name

Command Line:
```
GET <common><command> <file>
```

■ Example

This sample command switches the operating mode to Run mode.

```
>get \VIRTUAL\CMD\RUN dummy.txt
```

An example of an FTP reply is shown below assuming the command was executed from a command prompt and exited with error.

```
200 PORT command successful.
550 Can't open virtual file[SE01 PARAMETER ERROR].
```

■ Reply

Table 3.7.55   Reply Messages

Reply Message	Code	Description
OK	SE00	Normal exit
Other messages	SE01, …	Error reply message

SEE ALSO

For details on error reply messages, see "■ Error Reply Messages of Virtual Directory Commands" of Subsection 3.7.3, "Using Virtual Directory Commands."

■ Function

Switches the operating mode to Run mode. No error is generated even if the module is already in Run mode.

After command execution, a dummy file of zero byte is generated.

Switching to Run mode is not allowed while edited changes are being written to the CPU module in online edit mode.

3.7.7.7 Stop Mode (STOP)

Switches the operating mode to Stop mode.

■ FTP Command Used

get

■ Syntax

Table 3.7.56 Command Specifications

Command Part	Syntax
Common	\VIRTUAL\CMD
Command	\STOP
Parameters	–
File	Dummy file name

Command Line:
```
GET <common><command> <file>
```

■ Example

This sample command switches the operating mode to Stop mode.
```
>get \VIRTUAL\CMD\STOP dummy.txt
```

An example of an FTP reply is shown below assuming the command was executed from a command prompt and exited with error.
```
200 PORT command successful.
550 Can't open virtual file[SE01 PARAMETER ERROR].
```

■ Reply

Table 3.7.57 Reply Messages

Reply Message	Code	Description
OK	SE00	Normal exit
Other messages	SE01, …	Error reply message

SEE ALSO

For details on error reply messages, see "■ Error Reply Messages of Virtual Directory Commands" of Subsection 3.7.3, "Using Virtual Directory Commands."

■ Function

Switches the operating mode to Stop mode. No error is generated even if the module is already in Stop mode.

After command execution, a dummy file of zero byte is generated.

3.7.7.8   Activate Block (ACT)

Activates a specified block.

■ FTP Command Used

get

■ Syntax

Table 3.7.58   Command Specifications

Command Part	Syntax
Common	\VIRTUAL\CMD
Command [*1]	\ACT
Parameters [*1]	Block name (8 ASCII characters max.)
File	Dummy file name

*1:   Delimit the command and each parameter using an underscore ('_') character.

Command Line:
```
GET <common><command>_<parameter> <file>
```

■ Example

This sample command activates the block named "MAINBLK".
```
>get \VIRTUAL\CMD\ACT_MAINBLK dummy.txt
```

An example of an FTP reply is shown below assuming the command was executed from a command prompt and exited with error.
```
200 PORT command successful.
550 Can't open virtual file[SE01 PARAMETER ERROR].
```

■ Reply

Table 3.7.59   Reply Messages

Reply Message	Code	Description
OK	SE00	Normal exit
Other messages	SE01, …	Error reply message

SEE ALSO

For details on error reply messages, see "■ Error Reply Messages of Virtual Directory Commands" of Subsection 3.7.3, "Using Virtual Directory Commands."

■ Function

Activates a specified block. You may also activate a sensor control block. No error is generated even if the block is already running.

After command execution, a dummy file of zero byte is generated.

Block activation is not allowed in Stop mode and execute-all-blocks mode.

3.7.7.9   Inactivate Block (INACT)

Inactivates a specified block.

■ FTP Command Used

get

■ Syntax

Table 3.7.60   Command Specifications

Command Part	Syntax
Common	\VIRTUAL\CMD
Command [*1]	\INACT
Parameters [*1]	Block name (8 ASCII characters max.)
File	Dummy file name

*1:   Delimit the command and each parameter using an underscore ('_') character.

Command Line:

```
GET <common><command>_<parameter> <file>
```

■ Example

This sample command inactivates the block named "MOTION1".

```
>get \VIRTUAL\CMD\INACT_MOTION1 dummy.txt
```

An example of an FTP reply is shown below assuming the command was executed from a command prompt and exited with error.

```
200 PORT command successful.
550 Can't open virtual file[SE01 PARAMETER ERROR].
```

■ Reply

Table 3.7.61   Reply Messages

Reply Message	Code	Description
OK	SE00	Normal exit
Other messages	SE01, …	Error reply message

SEE ALSO

For details on error reply messages, see "■ Error Reply Messages of Virtual Directory Commands" of Subsection 3.7.3, "Using Virtual Directory Commands."

■ Function

Inactivates a specified block. You may also inactivate a sensor control block. No error is generated even if the block is not running.

After command execution, a dummy file of zero byte is generated.

Block inactivation is not allowed in Stop mode and execute-all-blocks mode.

3.7.7.10 Reset CPU (CPURESET)

Resets the CPU.

■ FTP Command Used

get

■ Syntax

Table 3.7.62 Command Specifications

Command Part	Syntax
Common	\VIRTUAL\CMD
Command	\CPURESET
Parameters	–
File	Reset information file name

Command Line:

```
GET <common><command> <file>
```

■ Example

This sample command resets the CPU and saves the reset information file in a file named "reset06.txt".

```
>get \VIRTUAL\CMD\CPURESET reset06.txt
```

An example of an FTP reply is shown below assuming the command was executed from a command prompt and exited with error.

```
200 PORT command successful.
550 Can't open virtual file[SE01 PARAMETER ERROR].
```

■ Reply

Table 3.7.63 Reply Messages

Reply Message	Code	Description
OK	SE00	Normal exit
Other messages	SE01, …	Error reply message

SEE ALSO

For details on error reply messages, see "■ Error Reply Messages of Virtual Directory Commands" of Subsection 3.7.3, "Using Virtual Directory Commands."

■ Function

Resets the CPU, and gets the reset information file.

The MODE switch value and timestamp at the time of reset are logged in the reset information file. The FTP connection is terminated after transfer of the reset information file.

⚠ CAUTION

Pay attention to the boot mode, which is determined by the MODE switch value, before resetting the CPU.

3.7.7.11   Clear Alarms (ALMCLEAR)

Clears all alarms.

■ FTP Command Used

get

■ Syntax

Table 3.7.64   Command Specifications

Command Part	Syntax
Common	\VIRTUAL\CMD
Command	\ALMCLEAR
Parameters	–
File	Cleared alarm log file name

Command Line:

```
GET <common><command> <file>
```

■ Example

This sample command clears all alarms, and gets a cleared alarm log file named "alm009.txt".

```
>get \VIRTUAL\CMD\ALMCLEAR alm009.txt
```

An example of an FTP reply is shown below assuming the command was executed from a command prompt and exited with error.

```
200 PORT command successful.
550 Can't open virtual file[SE01 PARAMETER ERROR].
```

■ Reply

Table 3.7.65   Reply Messages

Reply Message	Code	Description
OK	SE00	Normal exit
Other messages	SE01, …	Error reply message

SEE ALSO

For details on error reply messages, see "■ Error Reply Messages of Virtual Directory Commands" of Subsection 3.7.3, "Using Virtual Directory Commands."

■ Function

Clears all alarms. No error is generated even if no error is active when this command is executed. The command gets and saves a cleared alarm log file, which contains the following information:

- Error code
- Error message

Examples of Cleared Alarm Log File Content:
- If there was a momentary power failure:

    Momentary power failure, 02-0000
- If there was no active alarm:

    No error.

To check alarm information without clearing alarms, use the CPU Info (CPUINFO) command instead.

TIP

- If the cause of an alarm persists, it would appear as if this command has failed to clear the alarm. In this case, remove the cause of the alarm and re-execute the function.
- This command does not clear alarms related to I/O comparison error.

SEE ALSO

For details on the error messages, see Section B3.3, "Cleared Alarm Log Messages" of "Sequence CPU – Functions (for F3SP66-4S, F3SP67-6S)" (IM34M6P14-01E).

3.7.7.12 Help (HELP)

Gets and saves help information on virtual directory commands in a text file.

■ FTP Command Used

get

■ Syntax

Table 3.7.66   Command Specifications

Command Part	Syntax
Common	\VIRTUAL\CMD
Command	\HELP
Parameters	–
File	Help file name

Command Line:
```
GET <common><command> <file>
```

■ Example

This sample command gets and saves help information about virtual directory commands in a file named "vdchelp.txt".

```
>get \VIRTUAL\CMD\HELP vdchelp.txt
```

An example of an FTP reply is shown below assuming the command was executed from a command prompt and exited with error.

```
200 PORT command successful.
550 Can't open virtual file[SE01 PARAMETER ERROR].
```

■ Reply

Table 3.7.67   Reply Messages

Reply Message	Code	Description
OK	SE00	Normal exit
Other messages	SE01, …	Error reply message

SEE ALSO

For details on error reply messages, see "■ Error Reply Messages of Virtual Directory Commands" of Subsection 3.7.3, "Using Virtual Directory Commands."

■ Function

Gets and saves help information on virtual directory commands in a text file.

The help information includes a list of commands with their functional overview.

3.7.8　File Operation and Disk Operation Commands

Table 3.7.68　List of File Operation and Disk Operation Commands

Function Name	Command Name	Function
Unmount	UNMOUNT	Unmounts a memory card

TIP

You can use standard FTP commands (get, put, mkdir, etc.) to perform file transfer, directory creation, and other file operations.

3.7.8.1 Unmount (UNMOUNT)

Unmounts the memory card, which is inserted and mounted in the card slot.

■ FTP Command Used

get

■ Syntax

Table 3.7.69   Command Specifications

Command Part	Syntax
Common	\VIRTUAL\CMD
Command *1	\UNMOUNT
Parameters *1	Memory card slot number (1 ASCII character) [always 1]
File	Dummy file name

*1:   Delimit the command and each parameter using an underscore ('_') character.

Command Line:

```
GET <common><command>_<parameter> <file>
```

■ Example

This sample command unmounts memory card CARD1.

```
>get \VIRTUAL\CMD\UNMOUNT_1 dummy.txt
```

An example of an FTP reply is shown below assuming the command was executed from a command prompt and exited with error.

```
200 PORT command successful.
550 Can't open virtual file[SE01 PARAMETER ERROR].
```

■ Reply

Table 3.7.70   Reply Messages

Reply Message	Code	Description
OK	SE00	Normal exit
Other messages	SE01, …	Error reply message

SEE ALSO

For details on error reply messages, see "■ Error Reply Messages of Virtual Directory Commands" of Subsection 3.7.3, "Using Virtual Directory Commands."

■ Function

Unmounts the memory card, which is inserted and mounted in the card slot. A memory card in unmounted state can be safely removed, but does not allow access by programs or via FTP.

If the memory card is successfully unmounted with normal exit, the SD LED located on the front panel of the module turns off. Conversely, the SD LED is lit if the memory card is mounted.

After command execution, a dummy file of zero byte is generated.

The card slot number command parameter is fixed to 1.

3.7.9 Card Batch File Execution Commands

Table 3.7.71   List of Card Batch File Execution Commands

Function Name	Command Name	Function
Run Card Batch File	BATGO	Execute a specified card batch file.

3.7.9.1 Run Card Batch File (BATGO)

Executes a card batch file, which is stored on the memory card or RAM disk.

■ FTP Command Used

get

■ Syntax

Table 3.7.72   Command Specifications

Command Part		Syntax
Common		\VIRTUAL\CMD
Command *1		\BATGO
Parameters *1	1	Directory storing card batch file (7 ASCII characters max.) [Only "CARD1" or "RAMDISK" can be specified]
	2	Card batch file name (32 ASCII characters max.)
File		Standard output file name

*1:   Delimit the command and each parameter using an underscore ('_') character.

Command Line:
```
GET <common><command>_<parameter(1)>_<parameter(2)> <file>
```

■ Example

This sample command executes the card batch file named "mybat.bat", which is stored on the RAM disk.
```
>get \VIRTUAL\CMD\BATGO_ramdisk_mybat.bat dummy.txt
```

An example of an FTP reply is shown below assuming the command was executed from a command prompt and exited with error.
```
200 PORT command successful.
550 Can't open virtual file[SE01 PARAMETER ERROR].
```

■ Reply

Table 3.7.73   Reply Messages

Reply Message	Code	Description
OK	SE00	Normal exit
Other messages	SE01,…	Error reply message

SEE ALSO

For details on error reply messages, see "■ Error Reply Messages of Virtual Directory Commands" of Subsection 3.7.3, "Using Virtual Directory Commands."

■ Function

Executes a card batch file, which is stored on the memory card or RAM disk. You must store the card batch file directly below the "\CARD1" or "\RAMDISK" directory.

When the execution of the card batch file exits normally, the standard output file, which records the execution result of the card batch file, is returned. The standard output file is generated at the specified file pathname in card batch file format regardless of whether execution is successful.

SEE ALSO

For more details, see descriptions relating to the "Execution using a command" execution trigger in Chapter B2, "Card Batch File Function" of "Sequence CPU – Functions (for F3SP66-4S, F3SP67-6S) (IM34M6P14-01E)

3.8 FTP Function Sample Program

This section describes a sample program for the FTP function. This sample program is intended to help a user better understand the specifications and is not intended to be used directly in user applications.

■ List of Sample Programs

The table below shows the file structure of a sample program provided for FTP function. Files for the sample program are automatically copied to the respective folders shown below when WideField2 is installed.

Table 3.8.1   Sample Program Components and Location

Sample Program Name	Component	Location
FTP using Ethernet	Project	~\Fam3pjt\CPUSample\F3SP66\EFTP\EFTP.YPJT
	CPU Properties	~\Fam3pjt\CPUSample\F3SP66\EFTP\EFTPC.YPRP ~\Fam3pjt\CPUSample\F3SP66\EFTP\EFTPS.YPRP
	Files	~\Fam3pjt\CPUSample\F3SP66\EFTP\ftpput.txt

Note: " ~ " in the "Location" column denotes the folder where Wildefiled2 is installed.

3.8.1 FTP using Ethernet

■ Function and Usage

This is a sample program for FTP using Ethernet. Two F3SP66-4S modules and two SD memory cards are required to run the sample program.

Before running the sample program, copy the "ftpput.txt" file to the root directory of the SD memory card of the FTP client CPU module. The sample program performs the following processing:

1. Initializes devices.
2. Connects to FTP server.
3. Creates a directory on the FTP server.
   The directory to be created: (FTP server)\CARD1\FTPDIR
4. Changes the current directory to the newly created directory.
   Current directory: (FTP server)\CARD1\FTPDIR
5. Gets file information about the current directory.
   File information output filename: (FTP client)\CARD1\file_a.txt
6. Puts a file on the current directory.
   Source: (FTP client)\CARD1\ftpput.txt
   Destination: (FTP server)\CARD1\FTPDIR\yokogawa.txt
7. Gets a file from the current directory.
   Source: (FTP server)\CARD1\FTPDIR\yokogawa.txt
   Destination: (FTP client)\CARD1\ftpget.txt
8. Gets file information about the current directory.
   File information output filename: (FTP client)\CARD1\file_b.txt
9. Disconnects from FTP server.

Before running the sample program, copy the program components to their respective destinations as shown in the table below.

Table 3.8.2   Destinations for Copying Sample Program Components

Name of Unit	CPU No.	Role of CPU	Components to be Copied
FTP client unit	1	FTP client	EFTP.YPJT EFTPC.YPRP ftpput.txt
	2	–	–
	3	–	–
	4	–	–
FTP server unit	1	FTP server	EFTPS.YPRP
	2	–	–
	3	–	–
	4	–	–

■ Structure of Sample Program

● List of Instructions Used

The table below shows the main ladder instructions used in the sample program.

Table 3.8.3   List of File Operation and Disk Operation Instructions Used

Ladder Instruction Mnemonic	Purpose
FTPOPEN	Connects to FTP server.
FTPQUIT	Disconnects from FTP server.
FTPMKDIR	Creates directory on FTP server.
FTPCD	Changes current directory to directory created on FTP server.
FTPLS	Gets file information before and after putting file.
FTPPUT	Puts file on FTP server.
FTPGET	Gets file from FTP server.

● List of Special Relays Used

The table below lists the main special relays used in the sample program.

Table 3.8.4 List of Special Relays Used

Name of Special Relay	Special Relay No.	Function
FTP Client Busy	M1027	Checks whether any FTP client instruction is running.
Always ON	M0033	Used for Always on circuit
1 Scan ON at Program Start	M0035	Turns on for one scan after program starts execution
US1 LED Lit	M0125	Used for turning on US1 LED

● Project

The table below shows the content of the WideField2 project containing the sample program.

Table 3.8.5 Project Content

Name	Component	Description	
EFTP	Configuration	SP66 configuration with default setup. You can also use F3SP67-6S provided you change the CPU type in the configuration.	
	Blocks	Total number of blocks	1
		Block 1	MAIN
	Macros	Total number of macros	0
	Constant definition	#FTP_TGT	FTP client address setting number of CPU properties specified in the FTPOPEN instruction.
		#RDIR	Directory name (on FTP server end)
		#RFILE	Target file for PUT/GET (on FTP server end)
		#LDRIVE	Drive name (on FTP client end)
		#LDIR	Directory name (on FTP client end)
		#LFILE_P	Target file for PUT (on FTP client end)
		#LFILE_G	Target file for GET (on FTP client end)
		#LFILE_A	Output file name for FTPLS(1)
		#LFILE_B	Output file name for FTPLS(2)
		#LSOPT	"ls" command option
		Others, 15 definitions in total.	

● **CPU Properties**

The table below shows the content of the CPU property file of the sample program.

Table 3.8.6   CPU Properties (EFTPC.YPRP)

File Name	Required Setup for Execution of Sample Program	
EFTPC.YPRP	Ethernet setup	Specify the IP address and subnet mask to match the network environment. If you are configuring a local network for the sample program, you can run the sample program using the default values. The sample program uses the following default values: - ETHER_MY_IPADDRESS = 192.168.0.2 - ETHER_SUBNET_MASK = 255.255.255.0
	FTP client setup	You can run the sample program using the default values.
	FTP client address setup	Setting no. 1 defines the destination FTP server for the sample program with the following property values: - FTPC_SRV_ACCOUNT_1 = anonymous - FTPC_SRV_PASSWORD_1 = fam3@ - FTPC_SRV_PORT_1 = 21 - FTPC_SRV_IP_1 = 192.168.0.3

Table 3.8.7   CPU Properties (EFTPS.YPRP)

File Name	Required Setup for Execution of Sample Program	
EFTPS.YPRP	Ethernet setup	Specify the IP address and subnet mask to match the network environment. If you are configuring a local network for the sample program, you can run the program using the default values. The sample program uses the following default values: - ETHER_MY_IPADDRESS = 192.168.0.3 - ETHER_SUBNET_MASK = 255.255.255.0
	FTP server setup	You can run the sample program using the default values.

● **Files**

The table below lists the files used in the sample program.

Table 3.8.8   List of Files Used

File Name	Input/Output	Description
ftpput.txt	Input	File to be put on FTP server. Before running the sample program, store this file in the root directory of the SD memory card of the FTP client.
yokogawa.txt	Output	The sample program sends file "ftpput.txt" and stores it on the FTP server into this file.
ftpget.txt	Output	The sample program gets file "yokogawa.txt" from the FTP server and stores it into this file on the FTP client.
file_a.txt	Output	Output file for storing the file information before the put operation.
file_b.txt	Output	Output file for storing the file information after the put operation.

■ Ladder Program Listing

The figure on the following pages shows the ladder program listing. For details on the purpose of individual devices used in the ladder program, see the I/O comments of the block tag name definition.

● Project (EFTP) Block (MAIN)

```
00001 FTP Client Sample Program (Ethernet connection)
00002 Function: This program assumes that an FA-M3 is running as FTP server.
00003 It creates a directory, puts a file, gets a file and gets file info.
00004 (1) Initialize devices
00005  M00035                                                          RSET      0  D00001      7
                                                                                  FTPOPEN
                                                                                  status

00006                                                                   RSET      0  I00001      5
                                                                                  FTPOPEN
                                                                                  resource
                                                                                  check

00007                                                                   SET          I00001
                                                                                  FTPOPEN
                                                                                  resource
                                                                                  check

00008 (2) Establish FTP connection
00009 - Connect to FTP server using FTP client addr setup of CPU properties.
00010 - Setting 1 in sample CPU property file defines:
00011 Destination=192.168.0.3; account=anonymous; password=fam3@
00012  I00001     M01027                                             SET          I00002   <res chk>FTP Client
      FTPOPEN   FTP Client                                                       Execute            Busy
      resource  Busy                                                             FTPOPEN
      check

00013  I00002    C                                                              I00003   [FTPOPEN]
      Execute    FTPOPEN   D00001   #TOUT_R   #FTP_TGT                          FTPOPEN
      FTPOPEN              status                                               execution
                                                                                completed

00014  I00003                                                          RST          I00001   Check status
      FTPOPEN                                                                     FTPOPEN
      execution                                                                  resource
      completed                                                                  check

00015                                                                  RST          I00002
                                                                                  Execute
                                                                                  FTPOPEN

00016    D00001    =        0                                         SET          I00006   ->OK
      FTPOPEN                                                                     FTPMKDIR
      status                                                                     resource
                                                                                  check

00017    D00001   <>        0                                         SET          I00065   ->NG
      FTPOPEN                                                                     FTPOPEN
      status                                                                     failed

00018 (3) Create directory
00019 - If the directory to be created exists, error status -5550 is returned.
00020   As this does not affect the operation, it is handled as an OK case.
00021 - "FTPDIR" is created in root directory of memory card of the FTP server
00022  I00006     M01027                                             SET          I00007   <res chk>FTP Client
      FTPMKDIR  FTP Client                                                       Execute            Busy
      resource  Busy                                                             FTPMKDIR
      check

00023  I00007                                    TPARA      1  #RDIR      0      0 Set parameters
      Execute
      FTPMKDIR

00024  I00007    C                                                              I00008   [FTPMKDIR]
      Execute    FTPMKDIR  D00002   #TOUT_R                                    FTPMKDIR
      FTPMKDIR            status                                                execution
                                                                                completed

00025  I00008                                                          RST          I00006   Check status
      FTPMKDIR                                                                    FTPMKDIR
      execution                                                                  resource
      completed                                                                  check

00026                                                                  RST          I00007
                                                                                  Execute
                                                                                  FTPMKDIR

00027    D00002   >=        0                                         SET          I00011   ->OK
      FTPMKDIR                                                                    FTPCD
      status                                                                     resource
                                                                                  check

00028    D00002   <        0    D00002    =    #F_5550               SET          I00011   ->OK (already created)
      FTPMKDIR            FTPMKDIR                                               FTPCD
      status             status                                                 resource
                                                                                  check

00029                        D00002   <>    #F_5550                   SET          I00066   ->NG
                          FTPMKDIR                                               FTPMKDIR
                          status                                                 failed

00030                                                                  SET          I00036
                                                                                  FTPCLOSE
                                                                                  resource
                                                                                  check
```

F0348.VSD

Figure 3.8.1   FTP using Ethernet Sample Program Listing: MAIN (1/4)

```
00031 (4) Change current directory
00032 - Change current directory of FTP server to directory created earlier.
00033 - Change current directory to "\CARD1\FTPDIR" in this sample program.
```


Figure 3.8.2   FTP using Ethernet Sample Program Listing: MAIN (2/4)

00056 **(6) Put file**
00057 – You must store "ftpput.txt" on memory card of FTP client beforehand.
00058 – Put file "ftpput.txt" on current directory of FTP server.

00059 I00021   M01027   SET   I00022   <res chk>FTP Client Busy
FTPPUT   FTP Client   Execute
resource   Busy   FTPPUT
check

00060 I00022   TPARA   1   #LDRIVE   #LDIR   #LFILE_P   Set parameters
Execute
FTPPUT

00061   TPARA   2   #RFILE   0   0

00062 I00022   FTPPUT   D00005   #TOUT_B   I00023   [FTPPUT]
Execute   FTPPUT
FTPPUT   status   FTPPUT execution completed

00063 I00023   RST   I00021   Check status
FTPPUT   FTPPUT
execution   resource
completed   check

00064   RST   I00022
Execute
FTPPUT

00065   D00005   >=   0   SET   I00026   ->OK
FTPPUT   FTPGET
status   resource check

00066   D00005   <   0   SET   I00069   ->NG
FTPPUT   FTPPUT
status   failed

00067   SET   I00036
FTPCLOSE
resource
check

00068 **(7) Get file**
00069 – Get the file that was put on the FTP server earlier.
00070 – File is saved as "ftpget.txt" in root dir. of memory card of client.

00071 I00026   M01027   SET   I00027   <res chk>FTP Client Busy
FTPGET   FTP Client   Execute
resource   Busy   FTPGET
check

00072 I00027   TPARA   1   #RFILE   0   0   Set parameters
Execute
FTPGET

00073   TPARA   2   #LDRIVE   #LDIR   #LFILE_G

00074 I00027   FTPGET   D00006   #TOUT_B   I00028   [FTPGET]
Execute   FTPLS
FTPGET   status   FTPGET execution completed

00075 I00028   RST   I00026   Check status
FTPGET   FTPGET
execution   resource
completed   check

00076   RST   I00027
Execute
FTPGET

00077   D00006   >=   0   SET   I00031   ->OK
FTPLS   FTPLIST2
status   resource check

00078   D00006   <   0   SET   I00070   ->NG
FTPLS   FTPGET
status   failed

00079   SET   I00036
FTPCLOSE
resource
check

F0350.VSD

Figure 3.8.3   FTP using Ethernet Sample Program Listing: MAIN (3/4)

Figure 3.8.4   FTP using Ethernet Sample Program Listing: MAIN (4/4)

F0351.VSD

4. Higher-level Link Service (Personal Computer Link Function)

This chapter describes the Higher-level Link Service (Personal Computer Link Function).

4.1 Overview of Higher-level Link Service

The higher-level link service (also known as personal computer link function) enables a user to perform maintenance operations on the module (e.g. read/write devices and change the operating mode) from a monitor or a personal computer by sending processing requests as commands in a predefined format via a communications line. As no programming is required on the module end, this greatly simplifies creation of SCADA applications on the monitor or PC.

In this module, the higher-level link service can be accessed via its SIO port and/or the 10BASE-T /100BASE-TX connector located on its front panel.
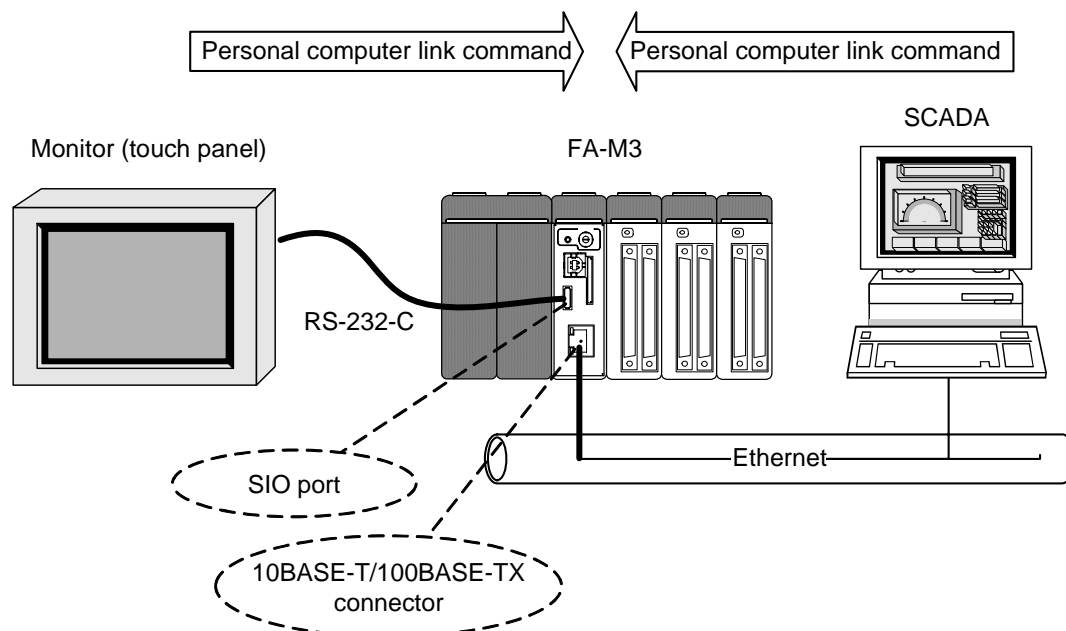
F0501.VSD

Figure 4.1.1   Example of Higher-level Link Service Configuration

TIP

This function is normally called "higher-level link service" when accessed via the 10BASE-T/100BASE-TX connector, but is normally called "personal computer (PC) link function" when accessed via the SIO port. The former term, namely, "higher-level link service" is adopted as a generic name for the function.

4.2 System Configurations for Higher-level Link Service

This section describes the available system configurations for Higher-level Link Service. In addition to system configurations using the built-in interfaces of the module, other available system configurations are also described.

● **Configuration for Higher-level Link Service via SIO Port**

In this system configuration, connection is via the SIO port located on the front panel of the module. A monitor or PC is connected using a dedicated RS-232-C monitor cable.
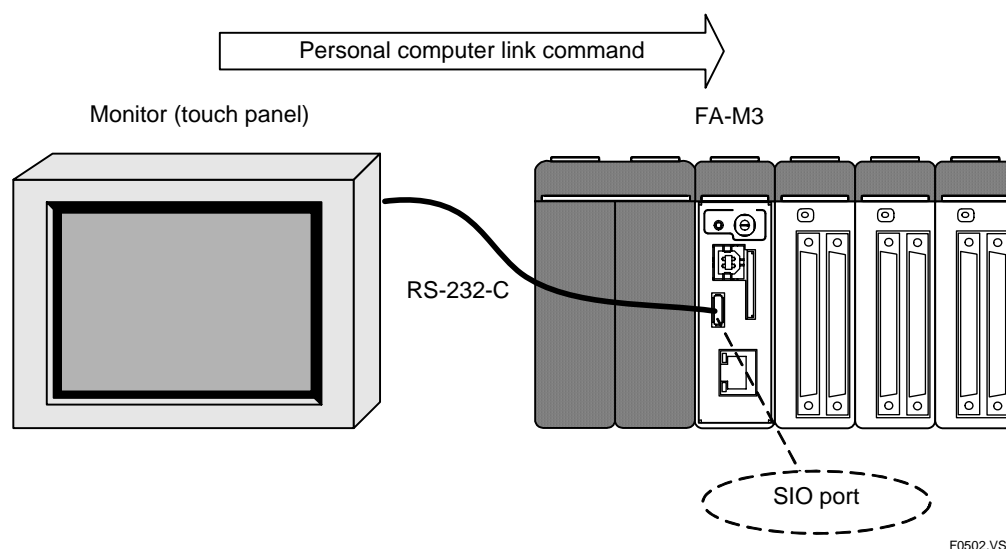
Figure 4.2.1   Configuration for Higher-level Link Service via SIO Port

Table 4.2.1   Supported Personal Computer Link Functions (via SIO Port)

Function	Description	Supported?
ASCII format personal computer link commands	Personal computer link function using ASCII format commands.	✓
Binary format personal computer link commands	Personal computer link function using binary format commands.	–
Event transmission function	Function for sending events from FA-M3 to a monitor or PC	–
Modem connection function	Function for providing higher-level link service via a modem and telephone line.	–
Write protection function	Function for prohibiting writing to devices using the personal computer link function.	✓

● Configuration for Higher-level Link Service via Ethernet

In this system configuration, connection is via the 10BASE-T/100BASE-TX connector located on the front panel of the module. A monitor or PC is connected using TCP/IP or UDP/IP communications protocol.
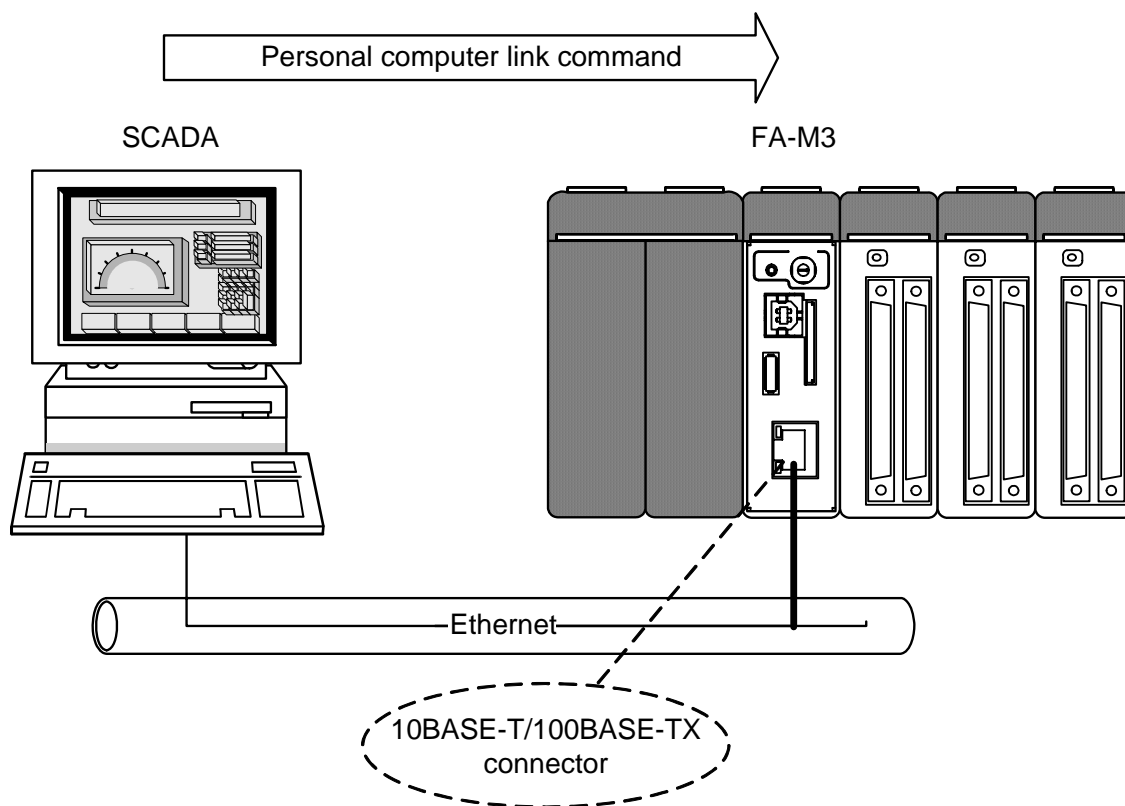
F0503.VSD

Figure 4.2.2   Configuration for Higher-level Link Service via Ethernet

Table 4.2.2   Supported Personal Computer Link Functions (via Ethernet)

Function	Description	Supported?
ASCII format personal computer link commands	Higher-level link service using ASCII format commands	✓
Binary format personal computer link commands	Higher-level link service using binary format commands	✓
Event transmission function	Function for sending events from FA-M3 to a monitor or PC	–
Modem connection function	Function for providing higher-level link service via a modem and telephone line	–
Write protection function	Function for prohibiting writing to devices using higher-level link service	✓

● **Configuration for Higher-level Link Service via Personal Computer Link Module**

In this system configuration, connection is via the communications connector of a Personal Computer Link Module (F3LC□□-□□). A monitor or PC is connected using RS-232-C, RS-422-A or RS-485 serial communications. Remote connection over a telephone line via a modem is also supported.
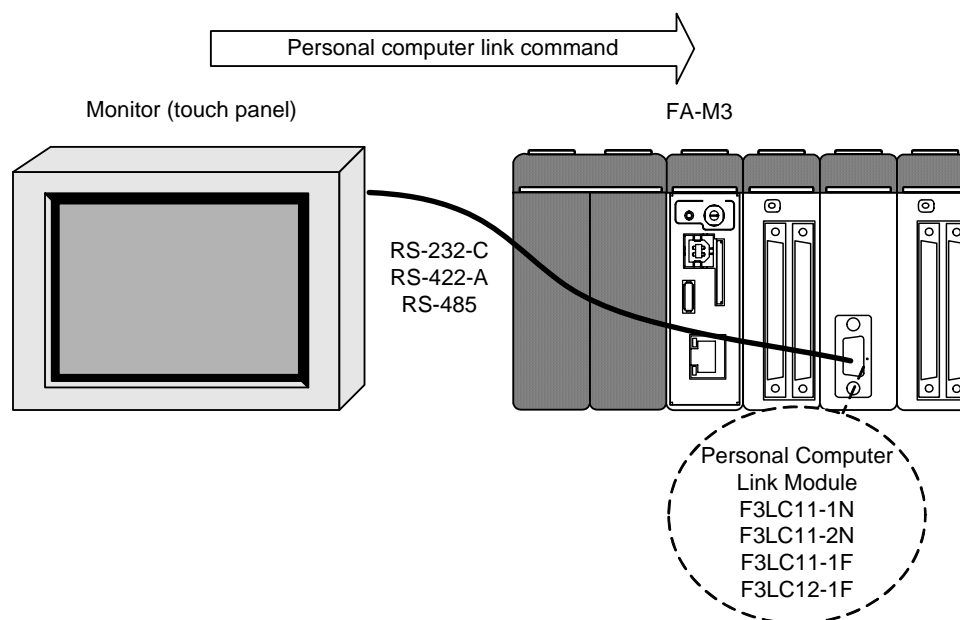
F0504.VSD

Figure 4.2.3   Configuration for Higher-level Link Service via Personal Computer Link Module

Table 4.2.3   Supported Higher-level Link Service (via Personal Computer Link Module)

Function	Description	Supported?
ASCII format personal computer link commands	Higher-level link service using ASCII format commands	✓
Binary format personal computer link commands	Higher-level link service using binary format commands	–
Event transmission function	Function for sending events from FA-M3 to a monitor or PC	✓
Modem connection function	Function for providing higher-level link service via a modem and telephone line	✓
Write protection function	Function for prohibiting writing to devices using higher-level link service	✓

SEE ALSO

For details on configuration using Personal Computer Link Module, see "Personal Computer Link Modules" (IM34M6H41-02E). This information is not included in this user's manual.

● **Configuration for Higher-level Link Service via Ethernet Interface Module**

In this system configuration, connection is via the 10BASE-T/100BASE-TX connector of an Ethernet Interface Module (F3LE□□-□T). A monitor or PC is connected using TCP/IP or UDP/IP communications protocol.

Table 4.2.4 Supported Higher-level Link Service (via Ethernet Interface Module)

Function	Description	Supported?
ASCII format personal computer link commands	Higher-level link service using ASCII format commands	✓
Binary format personal computer link commands	Higher-level link service using binary format commands	✓
Event transmission function	Function for sending events from FA-M3 to a monitor or PC	✓
Modem connection function	Function for providing higher-level link service via a modem and telephone line	–
Write protection function	Function for prohibiting writing to devices using higher-level link service	✓

Figure 4.2.4 Configuration for Higher-level Link Service via Ethernet Interface Module

SEE ALSO

For details on Ethernet Interface Module, see "Ethernet Interface Module" (IM34M6H24-01E and IM34M6H24-04E).

4.3 Personal Computer Link Function via SIO Port

This section describes the personal computer link function via SIO port.

4.3.1 Specifications

This subsection describes the personal computer link function via the SIO port in terms of the functional specifications, communication specifications and the cables used.

■ Functional Specifications

The table below shows the functional specifications of the personal computer link function via the SIO port.

Table 4.3.1   Supported Personal Computer Link Functions (via SIO Port)

Function	Description	Supported?
ASCII format personal computer link commands	Personal computer link function using ASCII format commands.	✓
Binary format personal computer link commands	Personal computer link function using binary format commands.	–
Event transmission function	Function for sending events from FA-M3 to a monitor or PC	–
Modem connection function	Function for providing higher-level link service via a modem and telephone line.	–
Write protection function	Function for prohibiting writing to devices using the personal computer link function.	✓

■ Communication Specifications

The table below shows the communication specifications of the personal computer link function via the SIO port.

Table 4.3.2   Communication Specifications of SIO Port (for Personal Computer Link Function)

Item	Description
Interface	EIA RS-232-C compliant
Transmission mode	Half-duplex transmission
Synchronization	Start-stop synchronization
Transmission rate (bps)	9600/19200/38400/57600/115200
Data format	Start bit : 1 bit
	Data length : 8 bits (fixed)
	Parity bit : None or Even
	Stop bit : 1 bit (fixed)
Error detection	Parity check
	Checksum : Yes/No
Control line (RS-232-C)	Not used
Xon/Xoff	Not used
Setup items	Transmission rate, data format, checksum, end character and protection
Protocol	Proprietary protocol
End character	Yes/No
Protection feature [*1]	Yes/No
Access range	All control data, upload/download program, Run/Stop program, read error log
Transmission distance	12 m max.
External connection	Dedicated monitor cable

*1: Enabling write protection prohibits writing to the FA-M3.

⚠ CAUTION

The SIO port of the module uses neither an RS-232-C control line nor Xon/Xoff characters. Therefore, a monitor or PC may fail to receive data correctly at high transmission rates.

The table below shows the available transmission rates and data formats.

Table 4.3.3   Available Transmission Rates and Data Formats

Transmission Rate (bps)	Data Bits	Parity	Stop Bit
9600	8 bits	Even	1 bit
9600	8 bits	None	1 bit
19200	8 bits	Even	1 bit
19200	8 bits	None	1 bit
38400	8 bits	Even	1 bit
38400	8 bits	None	1 bit
57600	8 bits	Even	1 bit
57600	8 bits	None	1 bit
115200	8 bits	Even	1 bit
115200	8 bits	None	1 bit

■ Cable

The table below lists dedicated cables for connecting a monitor or PC to the SIO port.

Table 4.3.4   Cables for SIO Port (Product Name: monitor cables)

Product Model	Specifications
KM21-2T	DOS/V compatible cable, 3 m long, (D-sub, 9-pin on PC end)
KM10-0S	D-sub 9-pin adapter cable

F0506.VSD
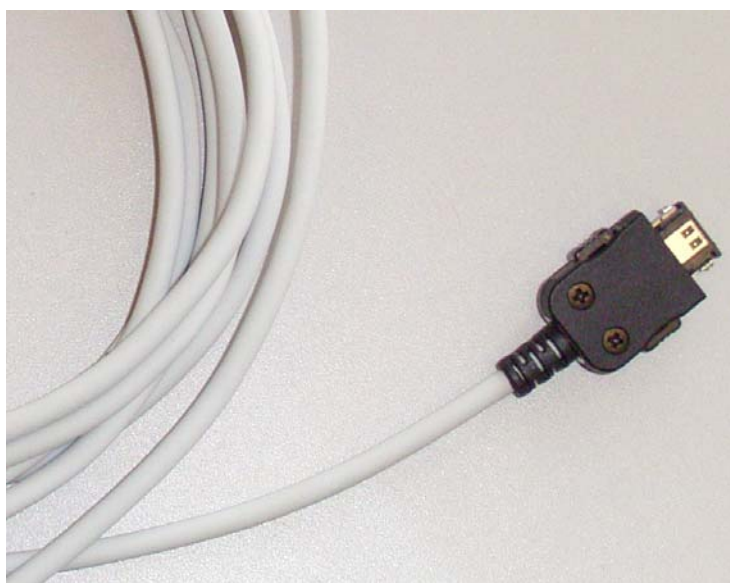
Figure 4.3.1   Monitor Cable (SIO port end)

TIP

For CE marking conformance of equipment using the personal computer link function, fit the monitor cable with a ferrite core.

Table 4.3.5   Examples of Ferrite Cores

Manufacturer	Product Series
Kitagawa Industries Co., Ltd.	RFC series
TDK Corporation	ZCAT series
Tokin Co., Ltd.	ESD-SR series

4.3.2   Communications Protocol

An overview of the personal computer link communications protocol is shown below.

The maximum text length that can be transmitted each time in a personal computer link command is 512 bytes.
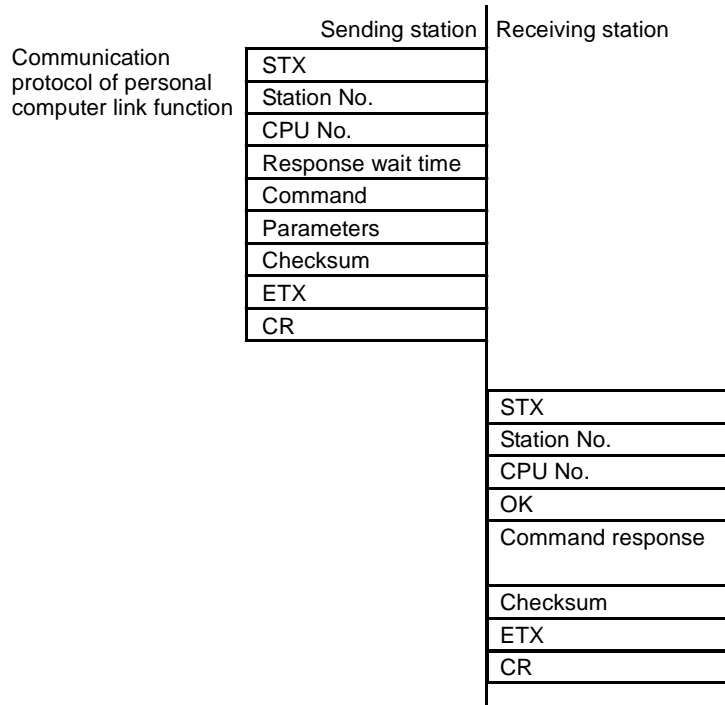
Figure 4.3.2   Personal Computer Link Communications Protocol

4.3.3   Commands and Responses

This subsection describes personal computer link commands and responses.

SEE ALSO

For details on commands and responses, see "Personal Computer Link Command" (IM34M6P41-01E).

■ Command Format and its Elements

The format of a command transmitted from a higher-level computer (or monitor) to the FA-M3 is shown below.

No. of Bytes	Element
1	STX code
2	Station No.
2	CPU No.
1	Response wait time
3	Command
Variable-length	Parameters
2	Checksum
1	ETX code
1	CR code

Checksum . . . . Required only if the configuration item "Checksum" is set to "Yes"

CR code . . . . Required only if the configuration item "End character" is set to "Yes"

F061106.VSD

Figure 4.3.3   Command Format and its Elements

Only uppercase alphabetic characters from A to Z (ASCII codes $41 to $5A in hexadecimal) are used in commands and responses.

The individual elements are described below.

● STX (Start of Text) Code

This control code identifies the beginning of text. The corresponding character code is $02.

● Station No.

The station number is fixed at 01 when the personal computer link function of the sequence CPU module is used.

● CPU No.

Identifies the target sequence CPU module or add-on CPU module for a command using a number from 01 to 04.

- 01: Sequence CPU module mounted in slot 1
- 02: Sequence CPU module mounted in slot 2
- 03: Sequence CPU module mounted in slot 3
- 04: Sequence CPU module mounted in slot 4

● **Response Wait Time**

You can specify the maximum waiting time (time delay of up to 600 ms) for a response following a command transmission. Set a longer wait time if the communication software running on the higher-level computer is, say, a BASIC interpreter. Specify this time using one character ('0' to 'F') as shown below.

Table 4.3.6   Response Wait Time

Character	Response Wait Time (ms)	Character	Response Wait Time (ms)
0	0	8	80
1	10	9	90
2	20	A	100
3	30	B	200
4	40	C	300
5	50	D	400
6	60	E	500
7	70	F	600
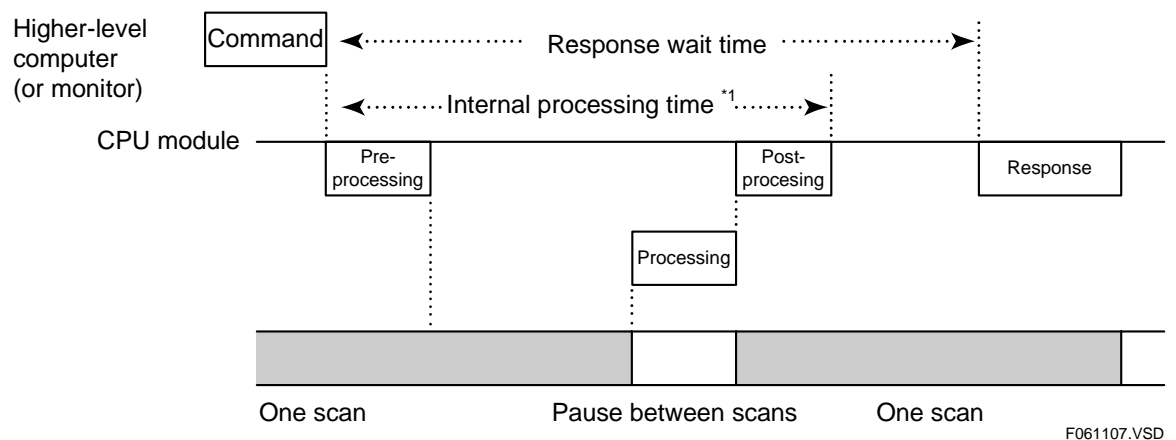
*1: Even if the response wait time is set to 0, processing will be delayed by the internal processing time.

Figure 4.3.4   Response Wait Time

● **Command**

Using three letters, specify the type of access, such as reading or writing, from a higher-level computer (or monitor) to the sequence CPU module.

● **Parameters**

These include device name, number of devices, data, etc. The actual parameters vary depending on the command used. Some commands require no parameters.

● **Checksum**

A checksum can be added to the transmission text for data validation. You can select whether to add a checksum in the program configuration.

If checksum is set to "Yes", a checksum must be appended to a command before transmission from the higher-level computer (or monitor) to the FA-M3. Moreover, a checksum is automatically appended to the response transmitted from FA-M3.

If checksum is set to "No", this element must not be appended to a command.

How the checksum is calculated is explained below.

- Add the ASCII codes of the characters following the STX character and preceding the checksum.

- Extract the low order byte of the sum and express its hexadecimal value as a character string (2 characters, 2 bytes) to obtain the checksum.

Transmission text (character string)

Range of checksum calculation

Checksum

STX	0	1	0	1	A	B	R	D	X	0	0	2	0	1	,	1	6	B	9	ETX	CR
02	30	31	30	31	41	42	52	44	58	30	30	32	30	31	2C	31	36	42	39	03	0D
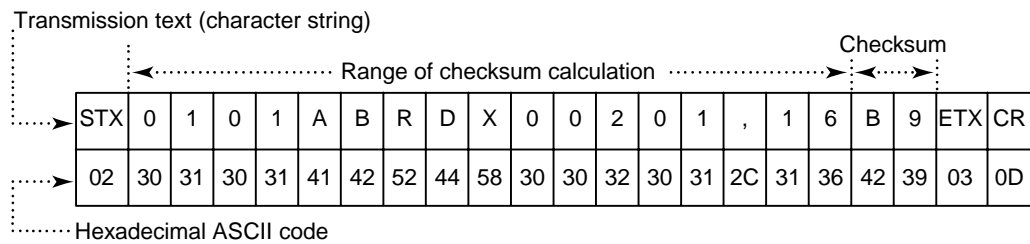
Hexadecimal ASCII code
The ASCII codes are added together as
30+31+30+31+41+42+52+44+58+30+30+32+30+31+2C+31+36=3B9 (in hexadecimal)
The checksum is the low-order byte ($B9) of the sum ($3B9 in hexadecimal),
expressed as a character string ("B9").

F061108.VSD

Figure 4.3.5   Checksum Calculation

● **ETX (End of Text) Code**

This control code identifies the end of text. The corresponding character code is $03.

● **CR (Carriage Return) Code**

This control code identifies the end. The corresponding character code is $0D (ASCII code 13 in decimal)

This control code is required only if the end character is set to "Yes" in the configuration.

■ Response Format and its Elements

The format of a response that is sent from the FA-M3 to a higher-level computer (or monitor) is shown here.

SEE ALSO

For details on individual elements and characters used, see "■ Command Format and its Elements" given earlier in this section.

● If communications is normal

No. of Bytes	Element
1	STX code
2	Station No.
2	CPU No.
2	OK
Variable-length	Command response
2	Checksum
1	ETX code
1	CR code

Appended to the response only if enabled accordingly in the configuration.

F061109.VSD

Figure 4.3.6   Response Format when Communications is Normal

If communications is successful, the character string "OK" and the command response are returned.

● If a communications error occurs

No. of Bytes	Element
1	STX code
2	Station No.
2	CPU No.
2	ER
2	EC 1
2	EC 2
3	Command
2	Checksum
1	ETX code
1	CR code

Appended to the response only enabled accordingly in the configuration.

F061110.VSD

Figure 4.3.7   Response Format when an Error Occurs

If a communications error occurs, the character string "ER" is returned along with two error codes (EC1 and EC2).

- EC1:   Error code
- EC2:   Detailed error code

In the communication failure is due to an error in the CPU number, the received 2-byte CPU number is returned. If the failure is due to an error in the station number, no response is returned.

If an ETX code in a command is not received, no response may be returned. If this happens, be sure to perform timeout processing on the higher-level computer or monitor.

■ Error Code in Response

When a communications error or a command error occurs, the module returns an "ER" character string and an error code as a response to the command.

The table below lists the error codes that may be included in a response.

Table 4.3.7   Error Code in a Response

Error Code (EC1)	Semantics	Possible Causes
01	CPU number error	- The CPU number is outside the range of 1 to 4.
02	Command error	- The command does not exist. - The command is not executable.
03	Device specification error	- The device name does not exist. [1] - A relay device is incorrectly specified for read/write access in word units.
04	Value outside the setting range	- Characters other than 0 and 1 are used for bit setting. [1] - Word setting is out of the valid range of 0000 to FFFF. - The specified starting position in a command, such as Load/Save, is out of the valid address range.
05	Data count out of range	- The specified bit count, word count, etc. exceeded the specifications range. [1] - The specified data count and the device parameter count, etc. do not match.
06	Monitor error	- Attempted to execute monitoring without having specified a monitor command (BRS, WRS).
08	Parameter error	- A parameter is invalid for a reason other than those given above. [1]
41	Communication error	- An error has occurred during communication. [1]
42	Checksum error	- Value of checksum differs. (Bit omitted or changed characters)
43	Internal buffer overflow	- The amount of data received exceeded stipulated value.
51	Timeout error	- No end-of-process response is returned from the CPU for reasons such as CPU power failure. (timeout)
52	CPU processing error	- The CPU has detected an error during processing. [1]
F1	Internal error	- A Cancel (PLC) command was issued during execution of a command other than a Load (PLD) or Save (PSV) command. - An internal error was detected.

*1: For details, see Table 4.3.8, "Detailed Error Code."

In the case of a parameter error, the number of the invalid parameter is stored in the detailed error code.

In the case of a communication error, detailed error information is stored in the detailed error code.

Table 4.3.8 Detailed Error Codes

Error code (EC1)	Semantics	Detailed Error Code (EC2)
03	Device specification error	Error parameter number, expressed in hexadecimal. (The number of the first parameter where an error has occurred, counting from the beginning of the parameters) (Example)
04	Value outside the setting range	
05	Data count out of range	(Example:)
08	Parameter error	
41	Communication error	
52	CPU processing error	1□: Self-diagnostic error 2□: Program error (including parameter error) 4□: Inter-CPU communication error 8□: Device access error 9□: Communication protocol error A□: Parameter error B□: Operating mode error, protected/exclusive access C□: Device/block specification error F□: Internal system error

For error codes 03, 04, 05, 08:

```
                       1   2   3  4  5    6    7 ← Parameter
 S                     ┌┐ ┌─┐    ┌─┐   ┌──┐       numbers
 T 0101ABRW 03 Y00501, 1, I0002, 0, | I0012, 1
 X                                    ↑
                       Erroneous device number
```

In this case, Error code EC1=03
                    Error code EC2=06.

For error code 41:

b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0

MSB   LSB

Each bit has the following meaning:

b_7: Reserved
b_6: Reserved
b_5: Framing error
b_4: Overrun error
b_3: Parity error
b_2: Reserved
b_1: Reserved
b_0: Reserved

Note: When the value of EC1 is other than those listed above, EC2 has no meaning.

■ List of Supported Devices

Use a comma (,) or a space ( ) to delimit parameters.

A device name is represented using six or seven characters (or bytes). Abbreviations may be used.

For example, X00201 can be abbreviated as X201 and V00002 as V02 or V2.

Example: To read data from 5 input relays of CPU 1, starting from input relay X00201 with a response wait time of 100 ms.

| S T X | 0 | 1 | 0 | 1 | A | B | R | D | X | 0 | 0 | 2 | 0 | 1 | , | 0 | 0 | 5 | | | |

Command   Parameters

Response wait time

CPU number

Station number (fixed to 01)   F061111.VSD
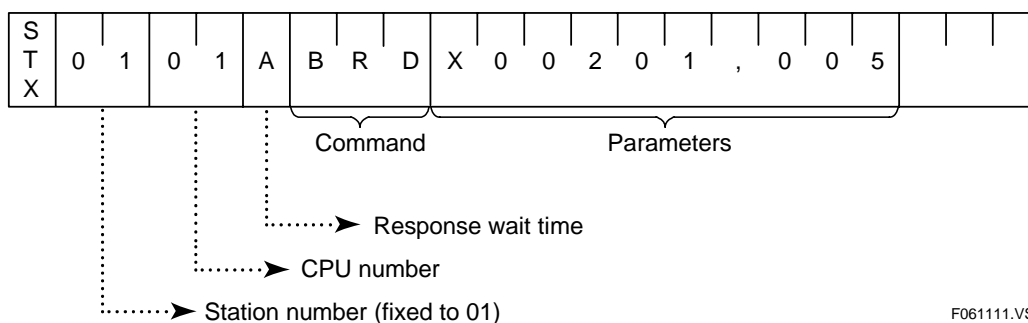
Figure 4.3.8 Reading Five Input Relays, Starting from X00201

Table 4.3.9 List of Supported Devices

Device name		Length	Read		Write	
			Bit	Word	Bit	Word
Relay Devices	Xnnnnn Input relay	6 bytes	✓	✓	–	–
	Ynnnnn Output relay	6 bytes	✓	✓	✓	✓
	Innnnn Internal relay	6 bytes	✓	✓	✓	✓
	Ennnnn (Extended) shared relay	6 bytes	✓	✓	✓	✓
	Lnnnnn Link relay	6 bytes	✓	✓	✓	✓
	Mnnnnn Special relay	6 bytes	✓	✓	✓[*5]	✓[*5]
	Txnnnn Timer	6 bytes	✓[*1]	✓[*2]	–	✓[*2]
	Cxnnnn Counter	6 bytes	✓[*1]	✓[*2]	–	✓[*2]
Word devices	Dnnnnn Data register	6 bytes	–	✓	–	✓
	Rnnnnn (Extended) shared register	6 bytes	–	✓	–	✓
	Vnnnnn Index register	6 bytes	–	✓	–	✓
	Bnnnnn[*3] File register	7 bytes	–	✓	–	✓
	Wnnnnn Link register	6 bytes	–	✓	–	✓
	Znnnnn Special register	6 bytes	–	✓	–	✓[*5]

*1: Specify:
- a time-out relay as TUnnnn
- an end-of-count relay as CUnnnn

*2: Specify:
- the current value of a countdown timer as TPnnnn
- the current value of a countdown counter as CPnnnn
- the current value of a count-up timer [3] as TInnnn
- the current value of a count-up counter [3] as CInnnn
- the preset value of a timer [4] as TSnnnn
- the preset value of a counter [4] as CSnnnn

*3: In the FA-M3, countdown timers and counters are provided for display on host personal computers.
Current value of count-up timer/counter = preset value – current value of countdown timer/counter.

*4: You may not use these preset values in word write commands.

*5: You may not use the BWR, BFL, WWR and WFL commands to write to the module. Use the BRW and WRW commands instead.

■ Precautions for Communications

- You should include timeout handling on the higher-level computer to handle situations where a response is not returned due to say, an incorrect station number specified in the command.

- If the personal computer link function is used to download a program, then you should not load another program from another source (personal computer link module, Ethernet interface module, etc.) at the same time. Otherwise, normal operation is not guaranteed.

- When writing to a shared device, the value may be immediately overwritten if another sequence CPU module is using the same device.

- If a power failure occurs when a monitor command is in use, it is necessary to set it again.

- The maximum text length that can be transmitted or received each time by the personal computer link function is 512 bytes. However, the maximum size that can be received by a higher-level computer may be limited to 256 bytes in some cases. In such cases, make sure that the response text length does not exceed 256 bytes by reducing the number of devices to be read.

4.3.4 Setup for Personal Computer Link Function via SIO Port

This subsection describes how to configure personal computer link function via SIO port before use.

■ Basic Setup

The table below shows the required setup for the personal computer link function via SIO port before use.

Table 4.3.10   Basic Setup for Personal Computer Link Function via SIO Port

Name of Setup	Type of Setup	SEE ALSO [1]
Set up Communication	Configuration	A9.2.7, "Set up Communication"

[1]: For details on individual setup items, see "Sequence CPU – Functions (for F3SP66-4S, F3SP67-6S)" (IM34M6P14-01E).

● Set up communication

Using communication setup, you must set the transmission rate and parity to match the device to be connected. Select also whether to enable checksum and end character.

■ Optional Setup

Personal computer link function via SIO port may be configured as required before use.

Table 4.3.11   Optional Setup for Personal Computer Link Function via SIO Port

Name of Setup	Type of Setup	SEE ALSO[1]
Function removal	Configuration	A9.2.12, "Function removal"

[1]: For details on individual setup items, see "Sequence CPU – Functions (for F3SP66-4S, F3SP67-6S)" (IM34M6P14-01E).

● Function removal

To disable the personal computer link function, remove the higher-level link service using function removal of configuration.

This setup affects higher-level link service via the SIO port or Ethernet of the CPU module. However, it does not affect higher-level link service via a personal computer link module or via an Ethernet interface module.

4.3.5   Using Personal Computer Link Function via SIO Port

This subsection describes how to use the personal computer link function via the SIO port.

■ Connecting to a Monitor

To connect to a monitor, perform the following steps:

1. Create screen data on a PC.
2. Transfer screen data to a monitor.
3. Match the communication setup of the monitor and the SIO port.
4. Connect the monitor and the SIO port. Begin monitoring.

SEE ALSO

For details on how to connect to a monitor, see the documentation of the monitor.

■ Connecting to a PC

● Connecting to a program created on a PC

To connect to a program on a PC, perform the following steps:

1. Create a communication program on a PC (using Visual Basic, etc.).
2. Match the communication setup of the communication program and the SIO port.
3. Connect the communication program and the SIO port. Begin monitoring.

● Connecting to a SCADA application

To connect to a SCADA application on a PC, perform the following steps:

1. Create screen data on a PC.
2. Match the communication setup of SCADA and the SIO port.
3. Connect SCADA and the SIO port. Begin monitoring.

SEE ALSO

For details on the application (Visual BASIC or SCADA) on the PC end, see the documentation for the application.

4.4   Higher-level Link Service via Ethernet

Higher-level link service via Ethernet is available for TCP/IP and UDP/IP protocols.

4.4.1   Specifications

The functional specifications and communications specifications of higher-level link service via Ethernet are described below.

■ Functional Specifications

The table below shows the functional specifications of the higher-level link service

Table 4.4.1   Supported Higher-level Link Service Functions (via Ethernet)

Function	Description	Supported?
ASCII format personal computer link commands	Higher-level link service using ASCII format commands	✓
Binary format personal computer link commands	Higher-level link service using binary format commands	✓
Event transmission function	Function for sending events from FA-M3 to a monitor or PC	–
Modem connection function	Function for providing higher-level link service via a modem and telephone line	–
Write protection function	Function for prohibiting writing to devices using higher-level link service	✓

■ Communications Specifications

The table below shows the communications specifications of the higher-level link service.

Table 4.4.2   Communications Specifications of Higher-level Link Service (via Ethernet)

Item	Specification
Communications protocol	TCP/IP or UDP/IP (configurable on port basis)
Maximum number of connections	TCP/IP = 8 UDP/IP = Non-connection type
Port number	12289 ($3001) = port A 12291 ($3003) = port B
Command data format	ASCII or binary (configurable on port basis)

4.4.2   Communications Protocol

In higher-level link service, the module returns one response to each personal computer link command received from a higher-level computer (monitor or PC).
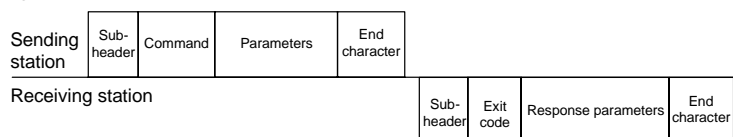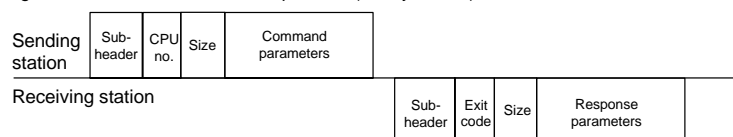
Figure 4.4.1   Communications Protocol

4.4.3　Data Frame

You can set the communications data format for the higher-level link service to either ASCII or binary. The selected data format applies to the entire higher-level link data frame.

The respective formats for the higher-level link data frame are shown below.

■ Data Frame in ASCII Format

Command
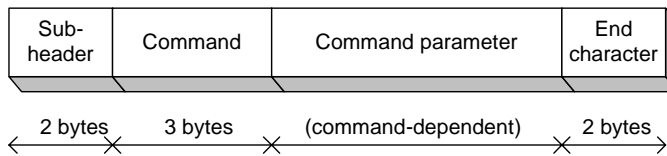
Response
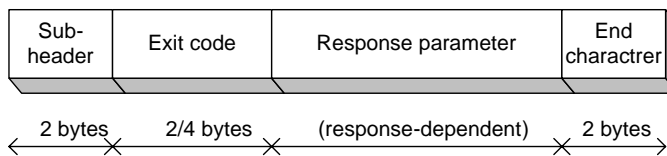
Figure 4.4.2　Data Frame in ASCII Format

● Subheader
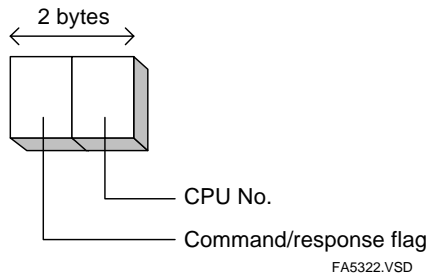
Figure 4.4.3　Subheader (ASCII)

Command/response Flag
Identifies a command or response using an ASCII character.

- "0" ($30): Command
- "1" ($31): Response

CPU No.
Identifies the target CPU module for a command by the slot where it is mounted using an ASCII character.

- "1" ($31): CPU module mounted in slot 1
- "2" ($32): CPU module mounted in slot 2
- "3" ($33): CPU module mounted in slot 3
- "4" ($34): CPU module mounted in slot 4

● **Command**

Identifies the type of request from a remote node.

SEE ALSO

For details on the commands, see "Personal Computer Link Command" (IM34M6P41-01E)

FA5323.VSD
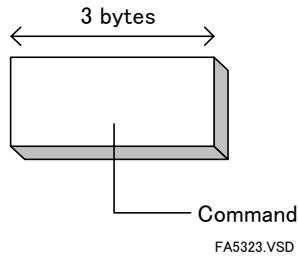
Figure 4.4.4   Command (ASCII)

● **Command Parameters**

This field contains device name, number of devices or other data. The actual parameters depend on the command. Some commands require no parameters.

SEE ALSO

For details on command parameters, see "Personal Computer Link Command" (IM34M6P41-01E).

● **Exit Code**

The result of the execution of a command is automatically appended to the response as an exit code.

SEE ALSO

For details on exit codes, see Subsection 4.4.4, "Exit Code and Detailed Error Code in Response."

Normal
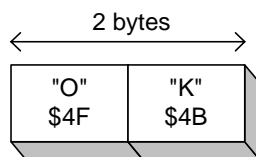
Error
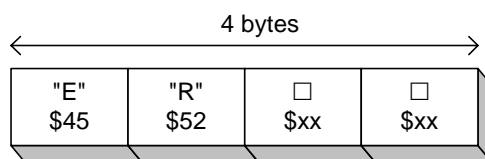
FA5324.VSD

Figure 4.4.5   Exit Codes (ASCII)

● **Response Parameters**

Normal Exit (exit code: "OK")
This field contains the response to a command. The actual parameters depend on the command. Some responses have no parameters.

Error Exit (exit code: "ER□□")

FA5325.VSD

5 bytes

Command
Detailed error code
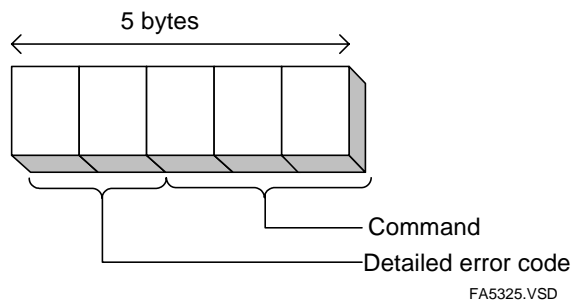
Figure 4.4.6   Response Parameters (ASCII)

Detailed Error Code

-   Valid only when the error code is "ER03," "ER04," "ER05," "ER08" or "ER52."

Command

-   The transmitted command is returned unchanged in the response.

● **End Characters**

FA5326.VSD

2 bytes

C_R $0D L_F $0A
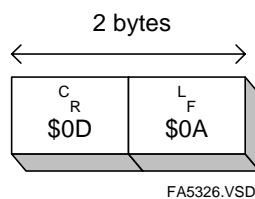
Figure 4.4.7   End Characters (ASCII)

A data frame must always be terminated with the $^C_R{}^L_F$ ($0D0A) characters.

Append these two characters to all commands.

These end characters are automatically appended to a response.

■ Data Frame in Binary Format

Command

Sub-header	CPU No.	Size	Command parameter
1 byte	1 byte	2 bytes	Size

Response
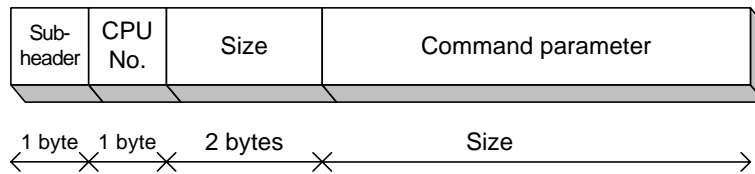
Sub-header	Exit code	Size	Response parameter
1 byte	1 byte	2 bytes	Size
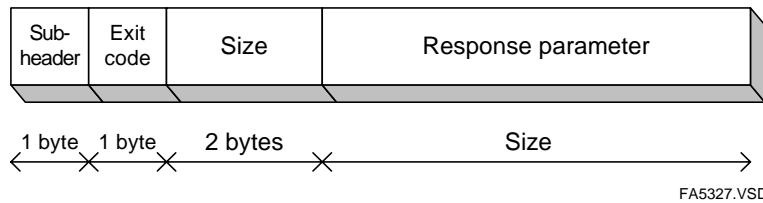
FA5327.VSD

Figure 4.4.8   Data Frame in Binary Format

● Subheader

1 byte

Command/response type

Command/response flag
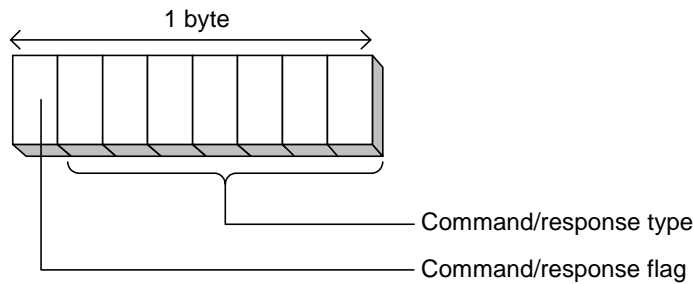
FA5328.VSD

Figure 4.4.9   Subheader (binary)

Command/response Flag
Identifies a command or a response with one bit.

- 0: Command
- 1: Response

Command/response Type
Indicates the type of request transmitted from a remote node.

SEE ALSO

For more on commands, see "Personal Computer Link Command" (IM34M6P41-01E).

● **CPU Number**

FA5329.VSD
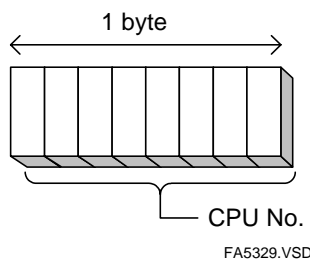
Figure 4.4.10   CPU Number (binary)

Identifies the target CPU module by the slot where it is mounted.

- $01: CPU module mounted in slot 1
- $02: CPU module mounted in slot 2
- $03: CPU module mounted in slot 3
- $04: CPU module mounted in slot 4

● **Exit Code**

The result of the execution of a command is automatically included in the response as an exit code.

SEE ALSO

For more details on exit codes, see Subsection 4.4.4, "Exit Code and Detailed Error Code in Response."

Exit code: If $00, ended normally.
If other than $00, ended in error.   FA532A.VSD
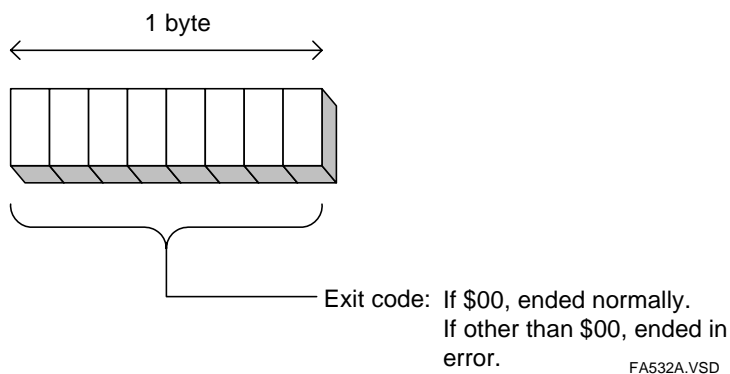
Figure 4.4.11   Exit Code (binary)

● **Size**

FA532B.VSD
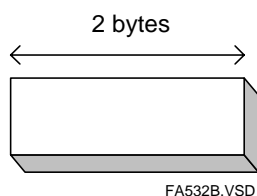
Figure 4.4.12   Size (binary)

Indicates the size of the command or response parameter field (in bytes).

If a frame has no parameter, the value is zero.

● **Command Parameters**

This field contains device names, number of points or other data. The actual parameters depend on the command. Some commands require no parameters.

SEE ALSO

For details on command parameters, see "Personal Computer Link Command" (IM34M6P41-01E).

● **Response Parameters**

Normal Exit (exit code: $00):
This field contains the response to a command. The actual parameters depend on the command. Some responses have no parameters.

SEE ALSO

For details on responses, see "Personal Computer Link Command" (IM34M6P41-01E).

Error Exit (exit code: non-zero)

FA532C.VSD
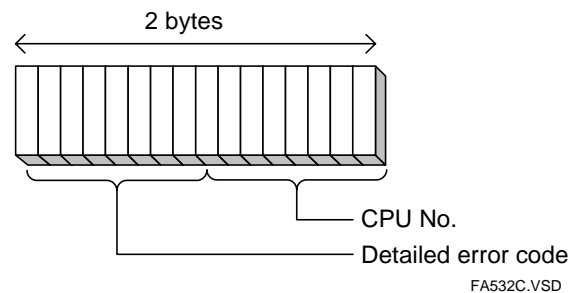
Figure 4.4.13   Response Parameters (binary)

Detailed Error Code (1 byte)

-    Valid only when the exit code is $03, $04, $05, $08 or $52.

SEE ALSO

For details on detailed error codes, see Subsection 4.4.4, "Exit Code and Detailed Error Code in Response."

CPU Number (1 byte)

-    The data contained in the transmitted command returned unchanged in the response.

4.4.4 Exit Code and Detailed Error Code in Response

This subsection describes the exit codes and detailed error codes, which may be included in a response.

■ Exit Code

The table below lists the exit codes that may be included in a response.

Table 4.4.3   Exit Codes

Exit Code ASCII	Exit Code Binary	Description	Possible Causes
"OK"	$00	Normal exit	
"ER01"	$01	CPU number error	The specified CPU number is not within the range of 1 to 4.
"ER02"	$02	Command error	The specified command does not exist or the command cannot be executed.
"ER03"	$03	Device specification error	The device does not exist.
"ER04"	$04	Value outside the setting range	A bit setting is neither 0 nor 1.
"ER05"	$05	Data count out of range	The number of bits or words specified exceeded the specifications range. Or, the number of parameters was different from the specified number of data items or devices.
"ER06"	$06	Monitor error	An attempt was made to run a monitor with no monitor specified.
"ER08"	$08	Parameter error	An invalid parameter, other than the above cases, is specified.
"ER51"	$51	Sequence CPU error	The sequence CPU module fails to respond within a specified time span (timeout).
"ER52"	$52	Sequence CPU processing error	An error was detected during CPU execution.

■ Detailed Error Code

If the exit code in a response is other than "OK" for ASCII format or $00 for binary format, the response parameter contains a detailed error code.

The detailed error code indicated is valid only if the exit code is "ER03", "ER04", "ER05", "ER08", or "ER52" for ASCII format; or $03, $04, $05, $08, or $52 for binary format. Otherwise, the value has no meaning.

Table 4.4.4   Detailed Error Codes

Exit Code ASCII	Exit Code Binary	Description	Detailed Error Code and Its Meaning		
"ER03"	$03	Device specification error	The detailed error codes are listed below. For ASCII format, the code is represented as a hexadecimal string.		
"ER04"	$04	Value outside setting range			
"ER05"	$05	Data count out of range			
"ER08"	$08	Parameter error			
"ER52"	$52	Sequence CPU processing error	ASCII	Binary	Meaning
			"1□"	$1□	Self-diagnostics error
			"2□"	$2□	Program error (including parameter error)
			"4□"	$4□	Inter-CPU communications error
			"8□"	$8□	Device access error
			"9□"	$9□	Command error
			"A□"	$A□	Parameter error
			"B□"	$B□	Operation mode error
			"C□"	$C□	Parameter error
			"F□"	$F□	System error

Note: "□" denotes an indeterminate number or character.

4.4.5   Specifying Devices in Commands

This section describes how to address a device of a sequence CPU module in a command.

■ List of Supported Devices

The table below lists the devices of a sequence CPU module that are accessible using commands.

Table 4.4.5   List of Supported Devices

Devices		Read Command		Write Command	
		By bit	By word	By bit	By word
Bit Devices	Input relay	✓	✓	×	×
	Output relay	✓	✓	✓	✓
	Internal relay	✓	✓	✓	✓
	Shared relay	✓	✓	✓	✓
	Special relay	✓	✓	✓	✓
	Timer relay	✓	✓	✓	✓
	Counter relay	✓	✓	✓	✓
	Link relay	✓	✓	✓	✓
Word devices	Data register	×	✓	×	✓
	File register	×	✓	×	✓
	Shared register	×	✓	×	✓
	Index register	×	✓	×	✓
	Special register	×	✓	×	✓
	Link register	×	✓	×	✓
	Timer preset value	×	✓	×	×
	Timer current value	×	✓	×	✓
	Timer current value (for count-up timers)	×	✓	×	✓
	Counter preset value	×	✓	×	×
	Counter current value	×	✓	×	✓
	Counter current value (for count-up counters)	×	✓	×	✓

✓: Supported
×: Not supported

■ Specifying a Device in ASCII Format

Specify a device using a six- or seven-character name string as shown in the table below.

Table 4.4.6   Specifying a Device in ASCII Format

Device Type		How to Specify	Device Type		How to Specify
Input relay	X	"X□□□□□"	Data register	D	"D□□□□□"
Output relay	Y	"Y□□□□□"	File register	B	"B□□□□□□"
Internal relay	I	"I□□□□□"	Shared register	R	"R□□□□□"
Shared relay	E	"E□□□□□"	Index register	V	"V□□□□□"
Special relay	M	"M□□□□□"	Special register	Z	"Z□□□□□"
Timer relay	T	"TU□□□□"	Link register	W	"W□□□□□"
Counter relay	C	"CU□□□□"	Timer preset value	T	"TS□□□□"
Link relay	L	"L□□□□□"	Timer current value		"TP□□□□"
			Timer current value (for count-up timers)		"TI□□□□"
			Counter preset value	C	"CS□□□□"
			Counter current value		"CP□□□□"
			Counter current value (for count-up counters)		"CI□□□□"

□: Device number

Example: Specify data register 123 (D0123)

"D" $44	"0" $30	"0" $30	"1" $31	"2" $32	"3" $33

FA5511.VSD

Figure 4.4.14   Example for Specifying a Device (in ASCII format)

■ Specifying a Device in Binary Format

Specify a device by its device attribute and device number as follows:

FA5521.VSD
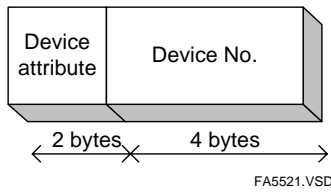
Figure 4.4.15   How to Specify a Device (in binary format)

● Device Attribute

The table below shows the mapping between device type and device attribute.

Table 4.4.7   Device Attributes in Binary Format

Device Type		Device Attribute	Device Type		Device Attribute
Input relay	X	$0018	Data register	D	$0004
Output relay	Y	$0019	File register	B	$0002
Internal relay	I	$0009	Shared register	R	$0012
Shared relay	E	$0005	Index register	V	$0016
Special relay	M	$000D	Special register	Z	$001A
Timer relay	T	$0014	Link register	W	$0017
Counter relay	C	$0003	Timer preset value	T	$0020
Link relay	L	$000C	Timer current value		$0021
			Timer current value (for count-up timers)		$0025
			Counter preset value	C	$0030
			Counter current value		$0031
			Counter current value (for count-up counters)		$0035

● Device Number

Specify the device number using 4 bytes.

Example: Specify data register 123 (D0123).

FA5522.VSD
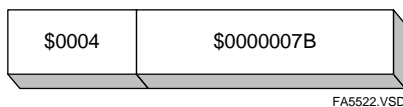
Figure 4.4.16   Example for Specifying a Device (in binary format)

4.4.6　Setup for Higher-level Link Service via Ethernet

This subsection describes how to configure higher-level link service via Ethernet before use.

■ Basic Setup

The table below shows the required setup for higher-level link service via Ethernet before use.

Table 4.4.8　Basic Setup for Higher-level Link Service via Ethernet

Name of Setup	Type of Setup	SEE ALSO [1]
Ethernet setup	CPU properties	A9.5.2, "Ethernet Setup"

[1]: For details on individual setup items, see "Sequence CPU – Functions (for F3SP66-4S, F3SP67-6S)" (IM34M6P14-01E).

● Ethernet setup

Ethernet setup configures the CPU module for joining an Ethernet network. Minimally, you must specify the IP address and subnet mask. If you set the subnet mask to "0.0.0.0", the default mask for the class of the IP address is used.

■ Optional Setup

Higher-level link service via Ethernet may be configured as required before use.

Table 4.4.9　Basic Setup for Higher-level Link Service via Ethernet

Name of Setup	Type of Setup	SEE ALSO [1]
Higher-level link service setup	CPU properties	A9.5.4, "Higher-level Link Service Setup"
Network filter setup	CPU properties	A9.5.8, "Network Filter Setup"
Function removal	Configuration	A9.2.12, "Function Removal"

[1]: For details on individual setup items, see "Sequence CPU – Functions (for F3SP66-4S, F3SP67-6S)" (IM34M6P14-01E).

● Higher-level link service setup

To modify the default values for the communications protocol to be used for higher-level link service (TCP/IP or UDP/IP) or the personal computer link command data format (ASCII or binary), perform higher-level link service setup of CPU properties.

To prohibit write access to devices via higher-level link service, enable the write protection.

Table 4.4.10　Default Values of Higher-level Link Service Setup

Setup Item	Default Value
Higher-level link service port A protocol (port no. 12289)	TCP/IP
Higher-level link service port A command data format	ASCII
Higher-level link service port B protocol (port no. 12291)	TCP/IP
Higher-level link service port B command data format	Binary
Write protection	Disabled

TIP

The Ethernet interface module (F3LE□□-□T) has no protocol selection setting as it automatically detects whether the connected network is TCP/IP or UDP/IP. If a monitor or application originally communicating through an Ethernet interface module fails to run after the Ethernet interface module is replaced by the sequence CPU module, check whether the protocol selection of the CPU module is consistent with the connected equipment.

● **Network filter setup**

You may perform network filter setup to restrict the IP addresses for connection to the module. By default, connections from all IP addresses are allowed. This setup affects all functions (e.g. remote programming service, FTP server, etc.) supporting the higher-level link service via Ethernet function.

● **Function removal**

You may disable the higher-level link service using function removal of configuration. Once deleted, the higher-level link service stops running and generating responses. The module has several interfaces for higher-level link service and personal computer link function and not all these interfaces can be removed by function removal.

Table 4.4.11   Higher-level Link Service Interfaces Removed by Function Removal

Interface	State of Higher-level Link Service after it is Removed using Function Removal
Via the 10BASE-T/100BASE-TX connector located on the front panel of the local CPU module.	Disabled
Via the SIO port located on the front panel of the local CPU module	Disabled
Via the 10BASE-T/100BASE-TX connector located on the front panel of another CPU module in a multi-CPU configuration.	Enabled
Via the SIO port located on the front panel of another CPU module in a multi-CPU configuration.	Enabled
Via the programming port located on the front panel of another CPU module in a multi-CPU configuration.	Enabled
Via an advanced function module (Ethernet interface module)	Enabled
Via an advanced function module (personal computer link module)	Enabled

4.4.7　Using Higher-level Link Service via Ethernet

This subsection describes how to use the Higher-level Link Service via Ethernet.

■ Connecting to a Monitor

To connect to a monitor, perform the following steps:

1. Create screen data on a PC.
2. Transfer screen data to a monitor.
3. Match the communication setup of the monitor and the module.
4. Connect the monitor and the module. Begin monitoring.

SEE ALSO

For details on how to connect to the monitor, see the documentation of the monitor.

■ Connecting to a PC

● Connecting to a program created on a PC

To connect to a program on a PC, perform the following steps:

1. Create a communication program on a PC (using Visual Basic, etc.)
2. Match the communication setup of the communication program and the module.
3. Connect the communication program and the module. Begin monitoring.

● Connecting to a SCADA application

To connect to a SCADA application on a PC, perform the following steps:

1. Create screen data on a PC.
2. Match the communication setup of SCADA and the module.
3. Connect SCADA and the module. Begin monitoring.

SEE ALSO

For details on the application (Visual BASIC or SCADA) on the PC end, see the documentation for the application.

4.5   List of Personal Computer Link Commands

A list of personal computer link commands supported by the module is given below.

SEE ALSO

For details on the command specifications of personal computer link commands, see "Personal Computer Link Command" (IM34M6P41-01E).

■ Device Bit Access Commands

Table 4.5.1   List of Personal Computer Link Commands (Device Bit Access Commands)

Command		Function	Number of points processed in one transmission
ASCII	Binary		
"BRD"	$01	Reads bits.	1 to 256 bits
"BWR"	$02	Writes bits.	1 to 256 bits
"BFL"	$03	Writes bits of the same data.	1 to 256 bits
"BRR"	$04	Reads bits randomly.	1 to 32 bits
"BRW"	$05	Writes bits randomly.	1 to 32 bits
"BRS"	$06	Specifies the devices to be monitored on a bit basis.	1 to 32 bits
"BRM"	$07	Monitors bits.	1 to 32 bits

■ Device Word Access Commands

Table 4.5.2   List of Personal Computer Link Commands (Device Word Access Commands)

Command		Function	Number of points processed in one transmission
ASCII	Binary		
"WRD"	$11	Reads words.	1 to 64 words
"WWR"	$12	Writes words.	1 to 64 words
"WFL"	$13	Writes words of the same data.	1 to 256 words
"WRR"	$14	Reads words randomly.	1 to 32 words
"WRW"	$15	Writes words randomly.	1 to 32 words
"WRS"	$16	Specifies the devices to be monitored on a word basis.	1 to 32 words
"WRM"	$17	Monitors words.	1 to 32 words

■ Special Module Access Commands

Table 4.5.3   List of Personal Computer Link Commands (Special Module Access Commands)

Command		Function	Number of points processed in one transmission
ASCII	Binary		
"SWR"	$31	Reads words.	1 to 64 channels
"SWW"	$32	Writes words.	1 to 64 channels
"SLR"	$33	Reads long words	1 to 32 channels
"SLW"	$34	Writes long words	1 to 32 channels

■ Program Access Commands

Table 4.5.4   List of Personal Computer Link Commands (Program Access Commands)

Command		Function
ASCII	Binary	
"PRI"	$41	Reads program information.
"PLC"	$42	Cancels program loading or saving.
"PLD"	$43	Loads a program.
"PSV"	$44	Saves a program.
"STA"	$45	Starts a program.
"STP"	$46	Stops a program.

■ Test Commands

Table 4.5.5   List of Personal Computer Link Commands (Test Commands)

Command		Function
ASCII	Binary	
"TST"	$51	Performs a (loopback) test.

■ Miscellaneous Commands

Table 4.5.6   List of Personal Computer Link Commands (Miscellaneous Commands)

Command		Function
ASCII	Binary	
"MDR"	$61	Resets the module.
"INF"	$62	Reads information.
"DTR"	$63	Reads date and time.
"DTW"	$64	Writes date and time.
"ERH"	$65	Reads error history.
"ULR"	$66	Reads user log.

5.   Remote Programming Service

Remote programming service enables a PC running WideField2 to connect with the module.
The connection can be made via USB, Ethernet or modem. The table below lists the available network configurations for connection to WideField2.

Table 5.1.1   List of Network Configurations for Connection to WideField2

Source	Via	Destination
PC	USB	FA-M3
	Ethernet	
	Modem	

⚠ **CAUTION**

Remote programming service is intended for temporary connection between the module and WideFIeld2 for FA-M3 maintenance purposes. As such, its correct operation is not guaranteed for applications (e.g. recorders) that require reliable connections over extended periods.

5.1   Remote Programming Service Specifications

This section describes the specifications of the remote programming service.

■ For USB Connection

The table below shows the remote programming service specifications for USB connection.

Table 5.1.2   Remote Programming Service Specifications for USB Connection

Item	Specifications
Communications protocol	USB1.1
Number of connections	1 max.
Compatible programming tool	WideField2 R4 or later version

■ For Ethernet Connection

The table below shows the remote programming service specifications for Ethernet connection.

Table 5.1.3   Remote Programming Service Specifications for Ethernet Connection

Item	Specifications
Communications protocol	TCP/IP
Number of connections	2
Port number used	12290($3002)
Compatible programming tool	WideField2 R4 or later version

5.2 Network Configurations

This section describes the possible network configurations for remote programming service.

SEE ALSO

For details on individual network configurations and connection methods, see "FA-M3 Programming Tool WideField2" (IM34M6Q15-01E).

5.2.1 USB Connection

This subsection describes the network configuration for remote programming service with the PC running the WideFIeld2 software connected to the USB port of the module.

⚠ CAUTION

- Depending on the chipset used in the PC running the WideField2 software, connection may sometimes be unreliable. The USB connection function of the module is not guaranteed to work with all PCs (chipsets).
- A USB connection may become unreliable or even disconnected due to noise. If this happens, remove and re-attach the USB cable to the PC.

■ Connecting PC and FA-M3 using USB

The figure below shows the configuration for remote programming service by connecting a PC to FA-M3 using a USB cable.
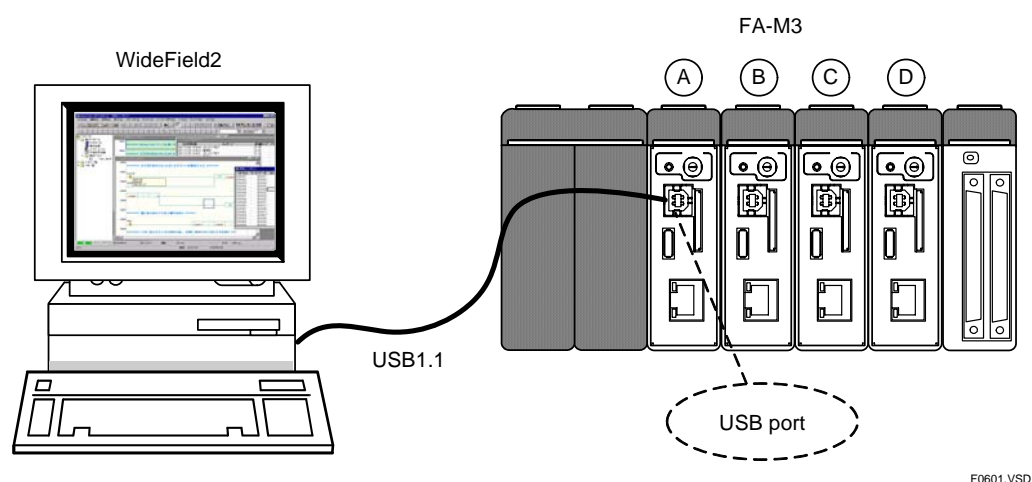
In a multi-CPU configuration as shown in the figure below, by connecting the cable to CPU module A, you can also access CPU modules B, C and D.
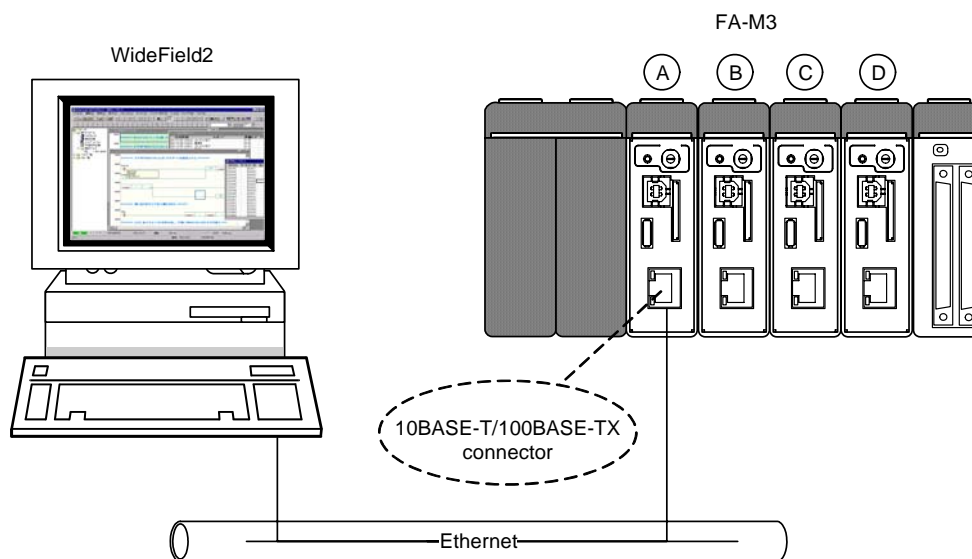

Figure 5.2.1 Connecting PC and FA-M3 using USB

5.2.2 Ethernet Connection

This subsection describes the network configurations for remote programming service with the PC running the WideFIeld2 software connected to the 10BASE-T/100BASE-TX connector of the module.

■ Connecting PC and FA-M3 using Ethernet

The figure below shows the configuration for remote programming service by connecting a PC to FA-M3 using Ethernet.

In a multi-CPU configuration as shown in the figure below, by connecting the cable to CPU module A, you can also access CPU modules B, C and D.

Figure 5.2.2   Connecting PC and FA-M3 using Ethernet

5.2.3 Modem Connection

In this configuration for remote programming service, connection to the module is made via a telephone line using a modem. This enables maintenance of the FA-M3 from a remote PC using WideField2.

SEE ALSO

For details on how to establish a modem connection, see "Personal Computer Link Modules" (IM34M6H41-02E)

5.3 Remote Programming Service Setup

This section describes how to configure the remote programming service before use.

5.3.1 For USB Connection

This section describes how to configure the remote programming service before use when using USB connection.

■ Basic Setup

Remote programming service requires no basic setup before use when using USB connection.

■ Optional Setup

Remote programming service may be configured as required when using USB connection.

Table 5.3.1   Optional Setup for Remote Programming Service using USB Connection

Name of Setup	Type of Setup	SEE ALSO [1]
Function removal	Configuration	A9.2.12, "Function Removal"

*1: For details on individual setup items, see "Sequence CPU – Functions (for F3SP66-4S, F3SP67-6S)" (IM34M6P14-01E).

● Function removal

To disable remote programming service, remove remote programming service using function removal of configuration.

5.3.2   For Ethernet Connection

This section describes how to configure the remote programming service before use when using Ethernet connection.

■ Basic Setup

The table below shows required setup for remote programming service before use when using Ethernet connection.

Table 5.3.2 Required Setup for Remote Programming Service using Ethernet Connection

Name of Setup	Type of Setup	SEE ALSO [1]
Ethernet setup	CPU properties	A9.5.2, "Ethernet Setup"

[1]: For details on individual setup items, see "Sequence CPU – Functions (for F3SP66-4S, F3SP67-6S)" (IM34M6P14-01E).

● Ethernet setup

The Ethernet setup configures the CPU module for joining an Ethernet network. Minimally, you must specify the IP address and subnet mask. If you set the subnet mask to "0.0.0.0", the default mask for the class of the IP address is used.

■ Optional Setup

Remote programming service may be configured as required when using Ethernet connection.

Table 5.3.3 Optional Setup for Remote Programming Service using Ethernet Connection

Name of Setup	Type of Setup	SEE ALSO [1]
Function removal	Configuration	A9.2.12, "Function Removal"
Network filter setup	CPU properties	A9.5.8, "Network Filter Setup"

[1]: For details on individual setup items, see "Sequence CPU – Functions (for F3SP66-4S, F3SP67-6S)" (IM34M6P14-01E).

● Function removal

To disable remote programming service, remove remote programming service using function removal of configuration.

● Network filter setup

You may perform network filter setup to restrict the IP addresses for connection to the module. By default, connections to all IP addresses are allowed. This setting affects all functions using Ethernet connection (e.g. socket communications function and FTP server function).

FA-M3

Sequence CPU – Network Functions
(for F3SP66-4S, F3SP67-6S)

INDEX

i

Revision Information

Document Name : Sequence CPU – Network Functions (for F3SP66-4S, F3SP67-6S)
Document No.　　: IM 34M6P14-02E

Edition	Date	Revised Item
1st	Jun. 2007	New publication

Blank Page