



# **FD CONTROLLER INSTRUCTION MANUAL Socket Communication**

**1st edition**

- Before attempting to operate the robot, please read through this operating manual carefully, and comply with all the safety-related items and instructions in the text.
- The installation, operation and maintenance of this robot should be undertaken only by those individuals who have attended one of our robot course.
- When using this robot, observe the law related with industrial robot and with safety issues in each country.
- This operating manual must be given without fail to the individual who will be actually operating the robot.
- Please direct any queries about parts of this operating manual which may not be completely clear or any inquiries concerning the after-sale service of this robot to any of the service centers listed on the back cover.

**NACHI-FUJIKOSHI CORP.**



# Table of Contents

## Chapter 1 Outline

1.1 Outline .....	1-1
1.1.1 Socket interface function of robot language.....	1-1
1.1.2 Necessary device .....	1-1
1.2 Terminology .....	1-2
1.2.1 Socket.....	1-2
1.2.2 Port number .....	1-2
1.2.3 TCP/IP (Transport control protocol / Internet protocol) .....	1-2
1.2.4 TCP (Transport control protocol) .....	1-3
1.2.5 UDP (User Datagram Protocol).....	1-3
1.2.6 Endian.....	1-3
1.2.7 Byte.....	1-3
1.2.8 Server / Client.....	1-3

## Chapter 2 Creation of Communication Program

2.1 Creation of Communication Program .....	2-1
2.1.1 TCP/IP setting.....	2-1
2.1.2 Creation of robot language program .....	2-1
2.1.3 Transferring to this controller and compiling .....	2-1

## Chapter 3 Socket Functions

3.1 Socket Function .....	3-1
3.1.1 Outline .....	3-1
3.1.2 Socket program flowchart.....	3-2
3.1.3 Creating socket.....	3-3
3.1.4 Closing socket .....	3-3
3.1.5 Assigning port number to socket .....	3-4
3.1.6 Waiting for connection from client .....	3-5
3.1.7 Connecting to server .....	3-6
3.1.8 Transmitting data saved into buffer .....	3-7
3.1.9 Transmitting string data .....	3-8
3.1.10 Receiving data.....	3-9
3.1.11 Saving character string into buffer.....	3-10
3.1.12 Saving integer into buffer.....	3-11
3.1.13 Saving real into buffer.....	3-12
3.1.14 Saving one-byte data into buffer.....	3-13
3.1.15 Retrieving character string from buffer .....	3-14
3.1.16 Retrieving integer value from buffer .....	3-15
3.1.17 Retrieving real value from buffer .....	3-16
3.1.18 1 Retrieving one-byte data from buffer .....	3-17
3.1.19 Error variables .....	3-18

## Chapter 4 Example of Creating Program

4.1 Getting Value of Integer Variable (Server Program).....	4-1
4.2 Getting Error Number by Unit (Server Program).....	4-2

4.3 Getting Data from Vision System (Client Program).....	4-3
4.4 Data Receiving Procedure.....	4-4
4.4.1 Predetermining a data size.....	4-4
4.4.2 Transmitting data size with different data .....	4-4
4.4.3 Determining a character to be transmitted at the last .....	4-4

# Chapter 1 Outline

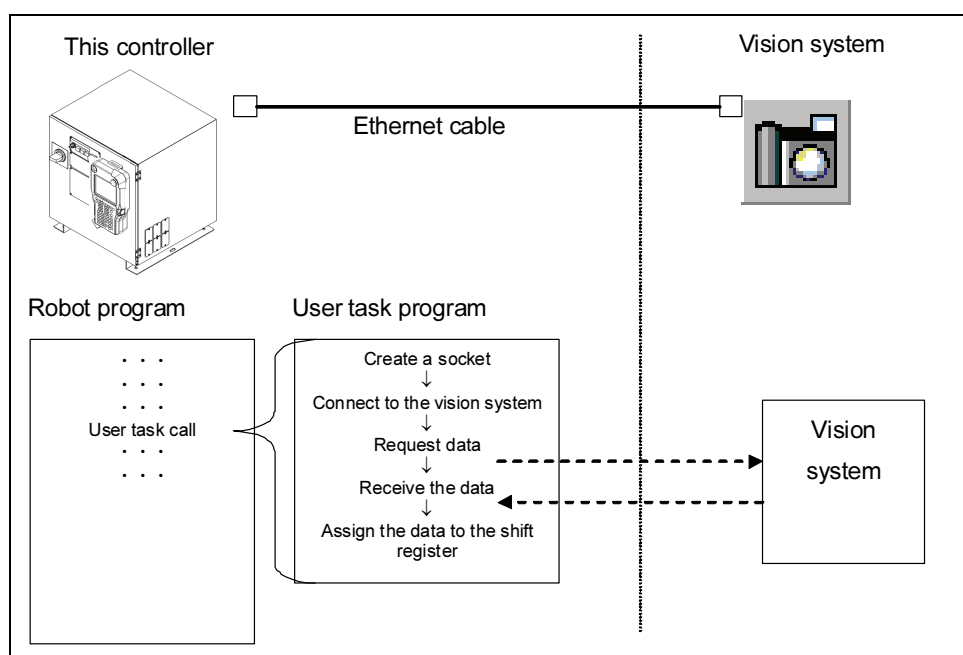
## 1.1 Outline

### 1.1.1 Socket interface function of robot language

This function performs Ethernet communication using various application commands from the user task program. Creating a user task program used to perform Ethernet communication makes it possible to reference and rewrite a wide variety of data from the outside of the controller via the Ethernet.

Example:

1. Monitor the integer variables of the controller from a personal computer.
2. Make changes to shift register values from the vision system or else.
3. Monitor the robot and controller status obtained by SYSTEM functions on a personal computer.



Example of connection to vision system,



The socket interface function can be only used from the user task program, not from the robot program.

To use the communication function from the robot program, use the application command "FN671 CALLMCR" to call the user task program from the robot program.



For the robot language, refer to the instruction manual "Robot Language".  
For the user task, refer to the instruction manual "User Task".



For networks using general sockets and terminology, refer to commercially-available specialized books in combination, as appropriate.

### 1.1.2 Necessary device

To use this function, Ethernet cables must be provided. For connections of the Ethernet cables, refer to the instruction manual "Setup" "Chapter 8 Connection to Ethernet".

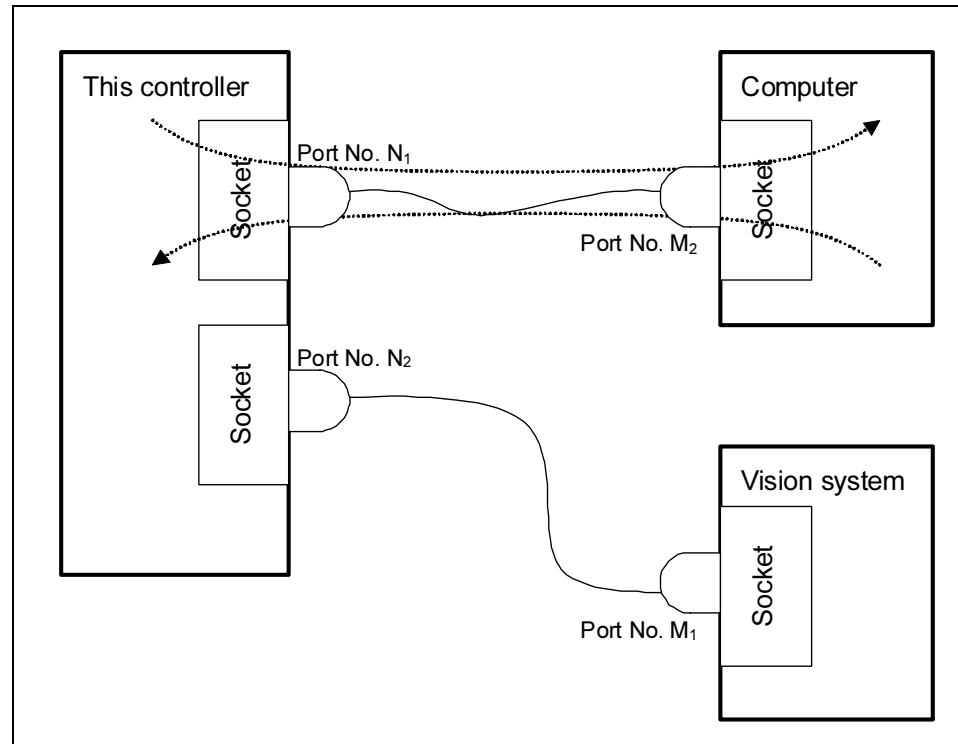
## 1.2 Terminology

This Section provide an explanation of communication-related terminology for the purpose of this Manual

### 1.2.1 Socket

The term “socket” means a group of functions used to create a TCP/IP-based network program and defines functions such as network connection, data transmission, and data reception. Normally, the Ethernet network is used via sockets.

This controller performs Ethernet communication by making the functions aforementioned available from the robot language.



Socket concept diagram

### 1.2.2 Port number

The term “port number” means numbers used in a communication system in order to identify services that perform communication. Since the port numbers are used to identify the services, no port number can be redundantly assigned to the services. Consequently, port numbers that are already used in this controller cannot be used.

Since numbers 1 to 1024 are used for commonly-used services (e.g. FTP and HTTP), these numbers are not normally used for port numbers.

Port numbers used in this controller in the initial state:  
233, 42353, and 51201

### 1.2.3 TCP/IP (Transport control protocol / Internet protocol)

No network communication can be performed unless communication systems know how they should perform communication each other. TCP/IP is one of the rules for communication and contains rules commonly used for Ethernet communication.

This function enables communication using TCP or UDP.

---

### 1.2.4 TCP (Transport control protocol)

TCP is the rules for performing highly reliable communications.

In order to ensure highly reliable communications, TCP checks whether data has reached the destination port and, if not, sends the same data once again, or determines network traffic congestion to control the data traffic.

---

### 1.2.5 UDP (User Datagram Protocol)

UDP is the rules for performing high-speed communications.

UDP enables communication processing at high speeds by omitting a process such as checking whether data has been transferred like TCP.

However, any case where data has not reached the destination port cannot be detected.

---

### 1.2.6 Endian

The term “endian” represents the order to handle numeric data.

In communications, data are normally transmitted by bytes. A byte is only available to represent values of 0 to 255. Consequently, to represent any value larger than 255 such as 10000, divide it from “ $1000 = 39 \times 256 + 16$ ” to  $39 \cdot 16$ , and then divide by bytes to transmit the data. The data transmission order from the most significant byte value (36) or the least significant byte value (16) is referred to as endian.

Transmission of data from the most significant byte value is referred to as a big-endian system, while that from the least significant byte value is referred to as a little-endian system. This function transmits data in the big-endian system.

---

### 1.2.7 Byte

A byte most often consists of eight bits (binary digits). Data size for the purpose of communication takes a byte as the minimum unit.

---

### 1.2.8 Server / Client

The term “server” means a program to provide services or functions upon receipt of a connection request from a destination port, while the term “client” means a program to make a connection request to the server.

NOTE



# Chapter 2 Creation of Communication Program

## 2.1 Creation of Communication Program

### 2.1.1 TCP/IP setting

To perform Ethernet communication, this controller IP address and sub net mask settings need to be made. To make these settings, select the screen [Constant Setting] → [8 Communication] → [Ethernet] → [TCP/IP] in the order described.

For TCP/IP setting, refer to the instruction manual "Setup" "Chapter 8 Connection to Ethernet".

### 2.1.2 Creation of robot language program

Create a program used to perform communication in the robot language. For the procedure and detail for using functions to perform communication, refer to information in Chapter 3.

For the robot language grammar, refer to the instruction manual "Robot Language".

Example of communication program

```
SOCKCREATE 1,0
SOCKCONNECT 1,1,23,5
SOCKSENDSTR 1,"USR",LEN("USR"),5,V1%,3
```

•  
•  
•

File name rule	USERTASK-A.*** (** represents a program number) *User task program numbers range from 000 to 999.
Editing method	Refer to the instruction manual of a personal computer or text editor in use.
File size	Use a maximum of 64 KB (65,536 bytes) for file size.
Precautions for sentenced, line, and characters	A maximum of 254 characters per line and 999 lines per program are describable. Alphabetical characters are not case-sensitive. Use one-byte characters for anything except for comments and string variables. It's no problem to describe the blank line.(It is ignored when compiled)
Other	Create two-byte characters in shifted JIS code and linefeed codes using CR + LF.

### 2.1.3 Transferring to this controller and compiling

To load the robot language program created by an external personal computer into this controller, provide a compact flash memory or connect the Ethernet.

To copy the robot language program from the USB memory, select the screen [Service]→ [4 File Manager] → [1 Copy] in the order described.

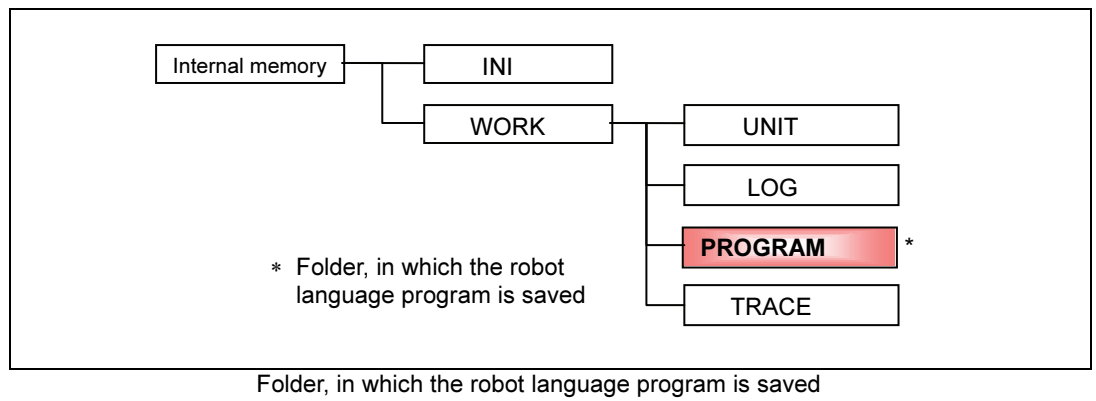
Specify the PROGRAM folder in which work programs are saved as a folder in which the robot language program is written. Any robot language programs other than those saved in the PROGRAM folder are not included in the scope of edition or compilation.

For the procedure for [File Manager], refer to information in the instruction manual "Basic Operation".

Furthermore, to transfer the robot language program via the Ethernet, select the screen [Service]→ [4 File Manager] → [8 File transfer (Ethernet FTP)] in the order described.

In this case, just as to copy the robot language program from the USB memory, specify the PROGRAM folder in which work programs are saved as a folder in which the robot language program is written.

For the procedure for transferring file with the use of FTP, refer to information in the instruction manual "Setup" "Chapter 8 Connection to Ethernet".



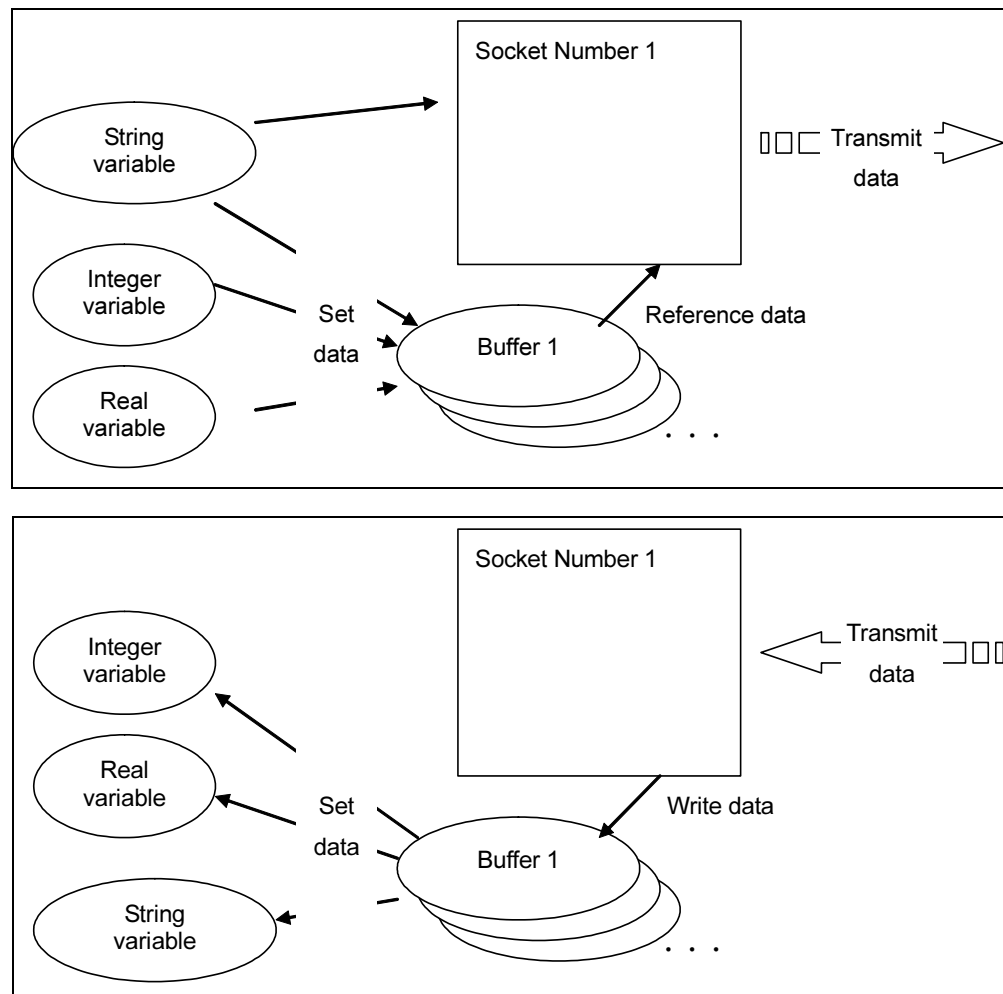
To compile programs, select the screen [Service]→ [9 Program conversion] → [8 Language conversion] in the order described.  
For compiling procedure, refer to the instruction manual “Robot Language”.

# Chapter 3 Socket Functions

## 3.1 Socket Function

### 3.1.1 Outline

The socket function basically exchanges data with the network through a buffer. Data are exchanged with the buffer using a dedicated application command.  
This controller is able to define 16 sockets (Socket Number 1 to 16) at one time, and 16 buffers (Buffer Number 1 to 16) of 1024 bytes in size can be used.



If a communication-related error occurs, the socket function will write the error in the error variable to complete the steps. However, the communication-related error will not stop the playback of the program. After the socket function completes its steps, monitor the error variable to check for the communication status.

**POINT**

Several buffers can be used in one socket. And one buffer can be also used in several sockets. In this case it is necessary to use the "I signal waiting" interlock when not access in the same time.

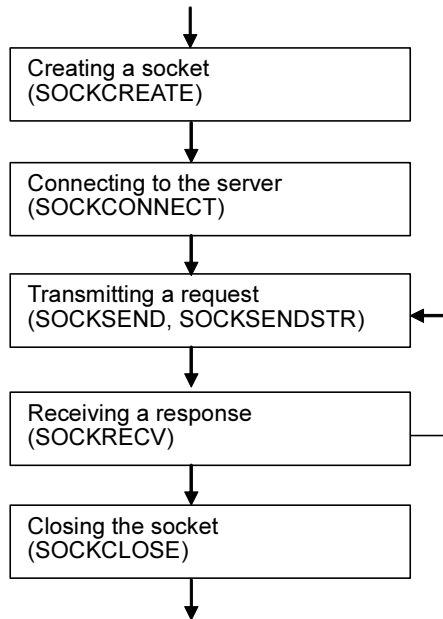
**POINT**

Normally same program can be executed by several user tasks simultaneously. But this is impossible for the socket communication because the same number of the socket can not be created.

### 3.1.2 Socket program flowchart

Here is the flowchart commonly used for socket programs. Even complicated programs basically follow the flowcharts shown below.

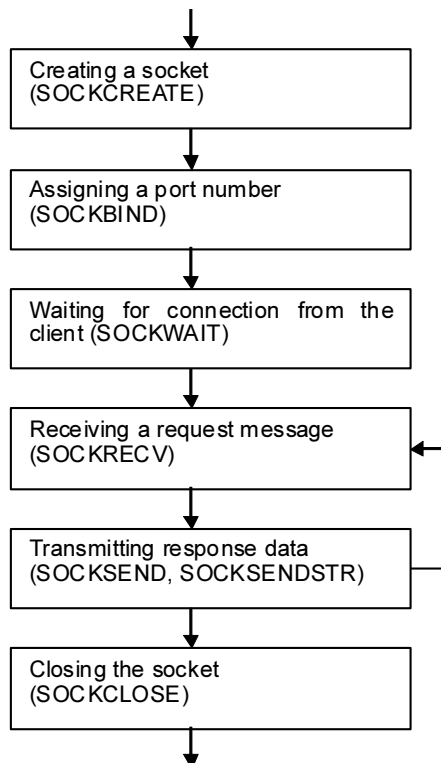
#### Client



For client socket programs, basically follow the flowchart shown on the left.

1. Creating a socket  
Create a socket.
2. Connecting to the server  
Connect the socket to a port specified by the server.
3. Transmitting a request  
Transmit a message requesting service (e.g. data return or login) to the server.
4. Receiving a response  
Receive the implementation status of service requested or data requested.
5. Closing the socket  
Open the socket number used.

#### Server



For server socket programs, basically follow the flowchart shown on the left.

1. Creating a socket  
Create a socket.
2. Assigning a port number  
Specify a port number to wait for a connection from the client.
3. Waiting for connection from the client  
Stop the program and wait until the server receives a request for connection from the client.
4. Receiving a request message  
Receive message from the client.
5. Transmitting response data  
Transmit a response message at the request of the client.
5. Closing the socket  
Open the socket number used.

### 3.1.3 Creating socket

Command name	SOCKCREATE
FN code	570
Title name	Creating socket
Outline	Used to create a socket.

#### Outline

This function is used to create a socket. To perform communication using a socket, execute this function before executing any other socket function.

#### Parameters

Parameter 1	Socket number	Used to specify a socket number to use. Specifying the number of socket that has been already created result in an error. (Specifying range: 1 to 16)
Parameter 2	TCP/UDP	Used to specify whether to use the set socket with TCP or UDP.  To use TCP, specify "0". To use UDP, specify "1". (Specifying range: 0 or 1)

#### Example of description in robot language

##### **SOCKCREATE 1, 0**

The SOCKCREATE command creates a socket for TCP in Socket Number 1.

### 3.1.4 Closing socket

Command name	SOCKCLOSE
FN code	571
Title name	Closing socket
Outline	Used to close a socket.

#### Outline

This function is used to clear the socket created executing the SOCKCREATE command to boot the system again. If the user task program to use the socket is complete without executing the SOCKCLOSE command, the socket will be automatically closed.

#### Parameters

Parameter 1	Socket number	Used to specify a socket number to use. Specifying the number of socket that has not been created will result in an error. (Specifying range: 1 to 16)
-------------	---------------	---

#### Example of description in robot language

##### **SOCKCLOSE 1**

The SOCKCLOSE command closes the socket created in Socket Number 1.

### 3.1.5 Assigning port number to socket

Command name	SOCKBIND
FN code	572
Title name	Binding socket
Outline	Used to specify a standby port.

#### Outline

In order to use this controller as the server, this function is used to assign a standby port to wait for connection from the client to the socket number.

#### Parameters

Parameter 1	Socket number	Used to specify a socket number to use. Specifying the number of socket that has not been created will result in an error. (Specifying range: 1 to 16)
Parameter 2	Port number	Used to specify a standby port to operate the socket as server. Specifying a port number that has been in use will result in an error. (Specifying range: 1 to 65535)

#### Example of description in robot language

##### **SOCKBIND 1, 48952**

The SOCKBIND command assigns the port number 48952 to the socket created in Socket Number 1.

### 3.1.6 Waiting for connection from client

Command name	SOCKWAIT
FN code	573
Title name	Waiting for reception
Outline	Used to wait until an external connection is made to the port number assigned.

#### Outline

This function is used to wait until an external connection is made to the port number assigned. During the connection time-out the program stops playing back. When the time-out period is complete, the program will restart playing back.

The SOCKWAIT command is only available for TCP.

#### Parameters

Parameter 1	Connection time-out socket number	Used to specify a socket number to use. Specifying the number of socket that has not been created will result in an error. (Specifying range: 1 to 16)
Parameter 2	Communication socket number	Used to specify a socket number used to connect to a client that has been accessing to the system. When the client is accessing, a socket will be created in the set socket number. Consequently, this function needs to be closed using the SOCKCLOSE function after the completion of communication. (Specifying range: 1 to 16)
Parameter 3	Time-out period	Used to specify connection time-out by seconds. Specifying "0" will not complete the time-out period to keep waiting for connection (Specifying range: 0 to 20)

#### Example of screen display

##### SOCKWAIT 1,2,5

The SOCKWAIT command waits for connection at Socket Number 1. When the connection is made from the client, connect to Socket Number 2. If no connection is made from the client within a period of five seconds, the time-out period will be complete.

### 3.1.7 Connecting to server

Command name	SOCKCONNECT
FN code	574
Title name	Connecting to server
Outline	Used to connect to the server.

#### Outline

This function is used to connect to the server if TCP mode is selected. If UDP mode is selected, connection to the server is not made, but IP address setting needs to be made with the SOCKCONNECT command.

#### Parameters

Parameter 1	Socket number	Used to specify a socket number to use. Specifying the number of socket that has not been created will result in an error. (Specifying range: 1 to 16)
Parameter 2	IP address	Used to specify the least significant byte of the server IP address. For the three high bytes, use the same bytes as those of this controller TCP/IP setting. (Specifying range: 1 to 254)
Parameter 3	Port number	Used to specify a standby port of the server. (Specifying range: 1 to 65535)
Parameter 4	Time-out period	Used to specify connection time-out by seconds. (Specifying range: 0 to 20)

#### Example of description in robot language

##### **SOCKCONNECT 1,101,80,10**

The SOCKCONNECT command uses Socket Number 1 to make a connection request to the server. The server IP address comes to a value found by taking "101" as the last byte of the IP address, and the port number comes to "80". If no response is made from the server within a period of 10 seconds, the time-out period will be complete.



### 3.1.8 Transmitting data saved into buffer

Command name	SOCKSEND
FN code	575
Title name	Transmitting data
Outline	Used to transmit data saved in the specified buffer.

#### Outline

This function is used to transmit data saved in the specified buffer. If the client closes the socket when no connection is made to the server, an error will result. However, since the SOCKSEND command cannot recognize that the client disconnects the socket if the client does so without closing it, an error may not result at the time when the command is executed.

To transmit data of length exceeding the buffer size, transmit it in two batches.

#### Parameters

Parameter 1	Socket number	Used to specify a socket number to use. Specifying the number of socket that has not been created will result in an error. (Specifying range: 1 to 16)
Parameter 2	Buffer number	Used to specify a buffer number. (Specifying range: 1 to 16)
Parameter 3	Transmitting data length	Used to specify how many bytes of data saved in the specified buffer to be sent. (Specifying range: 1 to 1024)
Parameter 4	Time-out period	Used to specify connection time-out by seconds. Specifying "0" will not complete the time-out period to keep waiting for the completion of transmitting data. (Specifying range: 0 to 20)
Parameter 5	Integer variable	Used to specify a variable used to write the data size sent (by bytes). The data size actually sent may be smaller than data size specified by the parameter 3 (Transmitting data length).

#### Example of description in robot language

##### **SOCKSEND 1,1,10,5,V1%**

The SOCKSEND command transmits a maximum of 10 bytes of data saved into Buffer Number 1 to the communication equipment connected to Socket Number 1. The SOCKSEND command transmits data, and then writes the data size actually sent into V1% integer variable, thus completing the step. If the command cannot transmit the data within a period of five seconds, the time-out period will be complete.

### 3.1.9 Transmitting string data

Command name	SOCKSENDSTR
FN code	576
Title name	Transmitting character string
Outline	Used to transmit a character string specified.

#### Outline

This function is used to transmit a character string specified. At this moment, termination characters indicating the end of the string can be added.

If the client closes the socket when no connection is made to the server, an error will result. However, since the SOCKSENDSTR command cannot recognize that the client disconnects the socket if the client does so without closing it, an error may not result at the time when the command is executed.

#### Parameters

Parameter 1	Socket number	Used to specify a socket number to use. Specifying the number of socket that has not been created will result in an error. (Specifying range: 1 to 16)
Parameter 2	Character string	Used to specify string data to transfer. The string data can be specified by string variables and string constants.
Parameter 3	Transmitting data length	Used to specify transmitting data length by bytes. (Specifying range: 0 to 199)
Parameter 4	Time-out period	Used to specify connection time-out by seconds. Specifying "0" will not complete the time-out period to keep waiting for connection (Specifying range: 0 to 20)
Parameter 5	Integer variable	Used to specify a variable used to write the data size sent (by bytes). The data size actually sent may be smaller than data size specified by the parameter 3 (Transmitting data length).
Parameter 6	Termination characters	Used to specify termination characters added to a character string to be sent. If any termination characters are specified, the transmitting data length becomes longer by the number of termination characters. If the specification of termination characters is omitted, the robot will make the same motion as that when "0" of this parameter is specified. 0: No termination characters added 1: Data sent with "¥r" added to the end of character string 2: Data sent with "¥n" added to the end of character string 3: Data sent with "¥r¥n" added to the end of character string (Specifying range: 0 to 3)

#### Example of description in robot language

##### SOCKSENDSTR 1, "ABC", LEN("ABC"), 5, V1%

The SOCKSENDSTR command transmits a character string "ABC" to the communication equipment connected to Socket Number 1. The SOCKSENDSTR command transmits data, and then writes the data size actually sent into V1% integer variable, thus completing the step. If the command cannot transmit the data within a period of five seconds, the time-out period will be complete, which means the parameter 6 is omitted. In this case, the description means the same as that when "0" of the parameter 6 is specified.

### 3.1.10 Receiving data

Command name	SOCKRECV
FN code	577
Title name	Receiving data
Outline	Used to receive data.

#### Outline

This function is used to receive data and conduct different operation depending on using TCP or UDP.

When TCP is used, the SOCKRECV command will complete its function step in any of the following cases.

- The size of data actually received exceeds the length of data received.
- The client closed the socket.
- Time-out period is complete.

If the command receives data in length shorter than the length of data received, it will wait for receiving data once again. It completes its function when it repeats receiving data and cumulative data length exceeds the length of data received.

When UDP is used, the SOCKRECV command will complete its function step in any of the following cases.

- Data is received.
- Time-out period is complete.

While in UDP mode, when data is received, the SOCKRECV command will save the data received in a buffer to complete its function. Consequently, the length of data actually received may be shorter than the specified receiving data length. Furthermore, if the length of data actually received is longer than the specified receiving data length, the command will cut the excess portion to return an error.

In addition, by executing the function after receiving data, the SOCKSEND can reply to the transmitter of the data. To transmit data to communication equipment different from the transmitter of the data, make resetting of destination using the SOCKCONNECT command.

#### Parameters

Parameter 1	Socket number	Used to specify a socket number to use. Specifying the number of socket that has not been created will result in an error. (Specifying range: 1 to 16)
Parameter 2	Buffer number	Used to specify a buffer number in which data received is saved. (Specifying range: 1 to 16)
Parameter 3	Receiving data length	Used to specify receiving data size by bytes. (Specifying: 0 to 1024)
Parameter 4	Time-out period	Used to specify connection time-out by seconds. Specifying "0" will not complete the time-out period to keep waiting for connection (Specifying: 0 to 20)
Parameter 5	Integer variable	Used to specify a variable used to write the data size received and written in the transmitting/receiving buffer (by bytes). The data size actually received may be smaller than data size specified by the parameter 3 (Receiving data length).

#### Example of description in robot language

##### SOCKRECV 1, 1, 10, 5, V1%

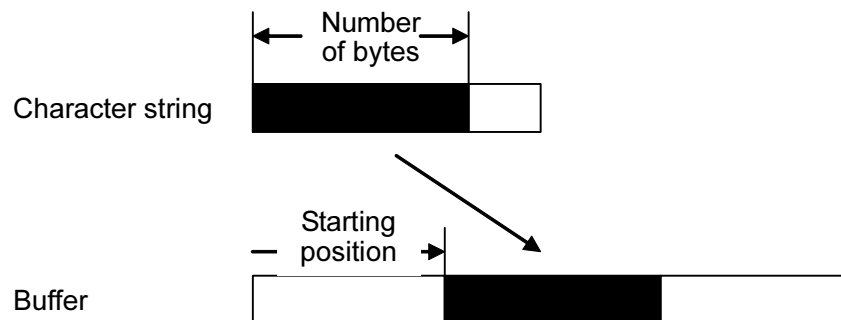
The SOCKRECV command receives data transmitted from the communication equipment connected to Socket Number 1 and saves data for a maximum of 10 bytes into Buffer Number 1 to the communication equipment connected to Socket Number 1. The size of data actually received is written into V1% integer variable. If the command cannot receive the data within a period of five seconds, the time-out period will be complete.

### 3.1.11 Saving character string into buffer

Command name	SETSTR
FN code	580
Title name	Saving into buffer (character string)
Outline	Used to write string data in any position of the buffer.

#### Outline

This function is used to write string data in any position of the buffer. If the buffer is flooded with character strings to be written (i.e., if “starting position + number of bytes” exceeds the buffer size), the error “E2250 Step data has an error” will result.



#### Parameters

Parameter 1	Buffer number	Used to specify a buffer number into which data is written. (Specifying range: 1 to 16)
Parameter 2	Character string	Used to specify a character string to be written into the buffer. The character string can be specified with string variables and string constants.
Parameter 3	Starting position	Used to specify a position in which the buffer writes data. (Specifying range: 0 to 1023)
Parameter 4	Number of bytes	Used to specify data size to be written. (Specifying range: 1 to 199)

#### Example of description in robot language

##### SETSTR 1, V1\$, 0, 10

The SETSTR command saves data for the leading 10 bytes of character string saved in the V1\$ variable from the head of Buffer Number 1.

### 3.1.12 Saving integer into buffer

Command name	SETINT
FN code	581
Title name	Saving into buffer (integer)
Outline	Used to save integer into the specified position of the buffer.

#### Outline

This function is used to save an integer into the specified position of the buffer. For the integer value, write signed integer data for four bytes in the big-endian system.

#### Parameters

Parameter 1	Buffer number	Used to specify a buffer number into which data is written. (Specifying range: 1 to 16)
Parameter 2	Integer	Used to specify an integer value to be written into the buffer. The integer value can be specified with integer variables and integer constants. (Specifying range: -2147483647 to 2147483647)
Parameter 3	Starting position	Used to specify a position in which the buffer writes data and writes data for four bytes from the specified position. (Specifying range: 0 to 1020)

#### Example of description in robot language

##### **SETINT 1,25,0**

The SETINT command saves the constant of 25 into the region of four bytes from the head of Buffer Number 1.

### 3.1.13 Saving real into buffer

Command name	SETREAL
FN code	582
Title name	Saving into buffer (real)
Outline	Used to save real into the specified position of the buffer.

#### Outline

This function is used to save a real into the specified position of the buffer. For the real value, write data represented in the format of single-precision floating-point number defined under the IEEE754 for four bytes in the big-endian system.

#### Parameters

Parameter 1	Buffer number	Used to specify a buffer number into which data is written. (Specifying range: 1 to 16)
Parameter 2	Real	Used to specify a real value to be written into the buffer. The real value can be specified with real variables and real constants. (Specifying range: $-10^{38}$ to $10^{38}$ )
Parameter 3	Starting position	Used to specify a position in which the buffer writes data and writes data for four bytes from the specified position. (Specifying range: 0 to 1020)

#### Example of description in robot language

##### SETREAL 1, V2!, 0

The SETREAL command saves a value saved into the real variable V2! into four bytes from the head of Buffer Number.

#### POINT

The IEEE754 is the standard that defines formats for representing floating-point numbers (real).

### 3.1.14 Saving one-byte data into buffer

Command name	SETBYTE
FN code	583
Title name	Saving into buffer
Outline	Used to write one-byte data into any position of the buffer.

#### Outline

This function is used to write one-byte data into any position of the buffer. Consequently, use this function to save data that the SETSTR, SETINT, and SETREAL commands cannot support into the buffer.

#### Parameters

Parameter 1	Buffer number	Used to specify a buffer number into which data is written. (Specifying range: 1 to 16)
Parameter 2	Integer	Used to specify a value to be written into the buffer. The value can be specified with variables and constants. (Specifying range: 0 to 255)
Parameter 3	Starting position	Used to specify a position in which the buffer writes data and writes data for one byte from the specified position. (Specifying range: 0 to 1023)

#### Example of screen display

##### **SETBYTE 1, 12, 3**

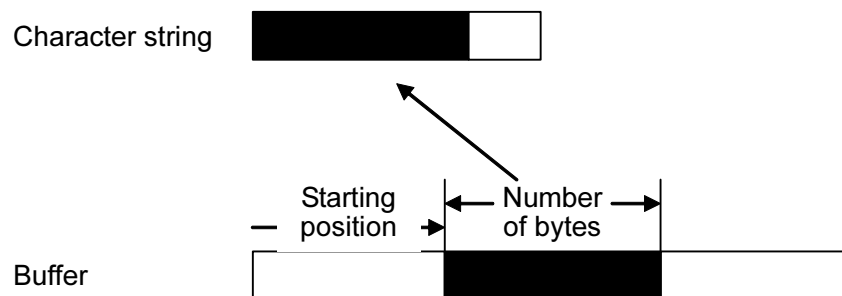
The SETBYTE command saves the constant of 12 into the third byte of Buffer Number 1.

### 3.1.15 Retrieving character string from buffer

Command name	GETSTR
FN code	584
Title name	Retrieving from buffer (character string)
Outline	Used to get data of the specified size from the specified position of the buffer and save the data into any string variable.

#### Outline

This function is used to get data of the specified size from the specified position of the buffer and save the data into any string variable. If the scope of loading data exceeds the buffer region (i.e., if "starting position + number of bytes" exceeds the buffer size), the error "E2250 Step data has an error" will result.



#### Parameters

Parameter 1	Buffer number	Used to specify a buffer number from which data is retrieved. (Specifying range: 1 to 16)
Parameter 2	String variable	Used to specify a variable into which string data loaded is saved.
Parameter 3	Starting position	Used to specify a head position which the buffer retrieves data. (Specifying range: 0 to 1023)
Parameter 4	Number of bytes	Used to specify the size of character string to be retrieved. (Specifying range: 1 to 199)

#### Example of description in robot language

##### GETSTR 1, V1\$, 0, 3

The GETSTR command copies data for three bytes from the head of data saved into Buffer Number 1 to the string variable V1\$.



### 3.1.16 Retrieving integer value from buffer

Command name	GETINT
FN code	585
Title name	Retrieving from buffer (integer)
Outline	Used to load data for four bytes from any position of the buffer and save the data into any integer variable.

#### Outline

This function is used to load data for four bytes from any position of the buffer and save the data into any integer variable. To use the GETINT command, ensure that data saved into the buffer is created in the big-endian system and a signed four-byte integer value. For any other integer values, use the GETBYTE function.

#### Parameter

Parameter 1	Buffer number	Used to specify a buffer number from which data is loaded. (Specifying range: 1 to 16)
Parameter 2	Integer variable	Used to specify an integer variable into which a value retrieved is saved.
Parameter 3	Starting position	Used to specify a head position in which the buffer loads data for four bytes from the set position. (Specifying range: 0 to 1020)

#### Example of description in robot language

##### **GETINT 1, V1%, 10**

The GETINT command retrieves data for four bytes from the 10th byte of data saved into Buffer Number 1, and then saves the data into the integer variable V1%.

### 3.1.17 Retrieving real value from buffer

Command name	GETREAL
FN code	586
Title name	Retrieving from buffer (real)
Outline	Used to load data for four bytes from any position of the buffer and save the data into any real variable.

#### Outline

This function is used to load data for four bytes from any position of the buffer and save the data into any real variable. To use the GETREAL command, ensure that data saved into the buffer is created in the big-endian system and a four-byte integer value provided under the IEEE754. For any other real values, use the GETBYTE function.

#### Parameters

Parameter 1	Buffer number	Used to specify a buffer number from which data is loaded. (Specifying range: 1 to 16)
Parameter 2	Real variable	Used to specify a real variable into which a value retrieved is saved.
Parameter 3	Starting position	Used to specify a head position in which the buffer loads data for four bytes from the set position. (Specifying range: 0 to 1020)

#### Example of description in robot language

##### **GETREAL 1, V1!, 3**

The GETREAL command retrieves data for four bytes from the 3rd byte of data saved into Buffer Number 1, and then saves the data into the real variable V1!.

### 3.1.18 1 Retrieving one-byte data from buffer

Command name	GETBYTE
FN code	587
Title name	Retrieving from buffer
Outline	Used to load data for one byte from any position of the buffer and save the data into any integer variable.

#### Outline

This function is used to load data for one byte from any position of the buffer and save a value of 0 to 255 into any integer variable.

#### Parameter

Parameter 1	Buffer number	Used to specify a buffer number from which data is loaded. (Specifying range: 1 to 16)
Parameter 2	Integer variable	Used to specify an integer variable into which a value retrieved is saved.
Parameter 3	Starting position	Used to specify a head position in which the buffer loads data for one byte from the set position. (Specifying range: 0 to 1023)

#### Example of description in robot language

##### **GETBYTE 1, V1%, 5**

The GETBYTE command retrieves data of the 5th byte of data saved into Buffer Number 1, and then saves the data into the integer variable V1%.

### 3.1.19 Error variables

As described in “3.1.1 Outline”, if the socket function detects any communication error, the function will write the error in the error variable to exit its processing. Since the error function is overwritten when executing the subsequent socket function, create a program so that the error will be confirmed immediately after the function is executed.

The error variable can be got in the form of E1% or E2% from the robot language program.

Error codes output by the error variable in the form of E1%

Number	Description
0	Processing has been normally completed.
-1	Communication service has not completed the startup sequence. When this controller power supply turns ON, the initialization of the communication-related function is in progress. If this error is detected, wait a few moments, and then execute the initialization.
-2	Parameter is not in the normal state. Detected if the SOCKWAIT command causes a problem with parameter such as the same socket number waiting for connection. If the parameter falls outside the detection range, the error “E2250 Step data has an error” will result to stop the program running.
-3	Socket has been already created. Detected if the SOCKCREATE command is executed for the already-created socket number.
-4	No socket has been created. Detected if communication is attempted to perform using a socket for which the SOCKCREATE has not been executed.
-5	Processing with the use of a socket is in execution. Detected if processing with the use of the specified socket number is in execution under a different user task program.
-7	Time-out period has been completed. In this case, do not directly use the socket, but close it once.
-8	No IP address setting has been made. Detected if any IP address is specified when transmitting data with the use of UDP. In this case, make IP address setting with the use of the SOCKCONNECT command.
-999	Detected if any error other than errors listed above. For detail, refer to information on E2%.

Error codes output by the error variable in the form of E2%

Number	Description
10022	Detected if the SOCKWAIT command is used before the SOCKBIND command specifies a port number.
10040	Detected if data larger than the buffer size specified by UDP communication and consequently the data is cut.
10045	Detected if the SOCKWAIT command is executed for the socket created as UDP using the SOCKCREATE command. The SOCKWAIT command is only available for TCP communication.
10048	Detected if the port number used in the SOCKBIND command is specified.
10054	Detected if data transmission or reception is attempted in TCP communication between devices that have blocked communication due to a trouble after connection. However, since it takes time to detect blocking of communication, it cannot be detected immediately after that.
10057	Detected if data transmission is attempted without making connection in TCP communication using the SOCKCONNECT or SOCAKWAIT command.

\* The error variable in the form of E2% returns error codes defined by the WinSock specification. If any error code not listed in the tables above is displayed, refer to information in commercially available technical book on network program.

Error variable can not be confirmed by the abnormal history or variable monitor. And the error of the user task monitor can not be showed either. You can confirm the error variable by following the next program.

■ EXP: the way to confirm error variable when create the socket.

- ① If integer variable substituted into error variable, the integer variable can be confirmed by integer variable.

POINT

Program EXP①	
' Create the socket V101% = 0 V102% = 0 SOCKCREATE 1, 0 IF E1% < 0 THEN *ERROR ... EXIT *ERROR V101% = E1% V102% = E2% EXIT	<u>Disposal the error</u> Error variable will be finished if Integer variable 101.102 substituted into error variable

- ② To confirm the error is showed in user screen.

Program EXP②	
' Create the socket SOCKCREATE 1, 0 IF E1% < 0 THEN *ERROR ... EXIT *ERROR WINDOW 0, 0, 200, 100 PRINT #0, STR\$(E1%) PRINT #0, STR\$(E2%) SOCKCLOSE 1 PAUSE 3000 EXIT	<u>Disposal the error</u> The program will be finished 3seconds later after the error code showed on the TP when the error occurred.

Note

# Chapter 4 Example of Creating Program

## 4.1 Getting Value of Integer Variable (Server Program)

The following section describes a program to return the value of an integer variable in response to external request. When the program starts running, the system will create a socket for TCP to wait for connection at Port Number 48952. After an external connection is made, the system will receive one-byte numeric data (ranging from 1 to 100) as a request message and, in response, return the value of integer variable 1 to return in the four-byte big endian.

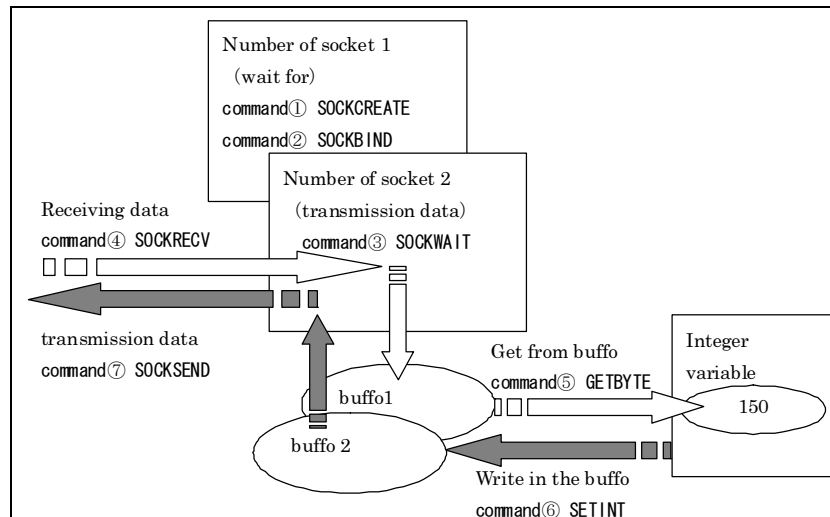
After transmitting data, the system closes the socket to exit the program.

Used variable

variable	No	Used content
Integer variable	100	Receiving data size
Integer variable	120	Transmission data size
Integer variable	150	Receiving data (number of the demanded integer variable)

### Example of creating program

'RETURN INTEGER VARIABLE SERVER		
'CREATE SOCKET		
Command ①	SOCKCREATE 1,0	
IF E1%<0 THEN *ERROR		
'ASSIGN PORT NUMBER		
Command ②	SOCKBIND 1,48952	
IF E1%<0 THEN *ERROR		
'WAIT FOR CONNECTION		
Command ③	SOCKWAIT 1,2,0	<u>Waiting for connection</u>
IF E1%<0 THEN *ERROR		The system waits for connection from the client, and perform communication with the client connected using Socket Number 2.
'RECEIVE DATA		
Command ④	SOCKRECV 2,1,1,5,V150%	<u>Receiving request message</u>
IF E1%<0 THEN *ERROR		The system receives the request message from the client, and subsequently checks for data. If the data falls outside of the range, it will consider the data as an error.
Command ⑤	GETBYTE 1,V150%,0	
IF V150%<1 THEN *ERROR		
IF V150%>100 THEN *ERROR		
'TRANSMIT DATA		
Command ⑥	SETINT 1,V%[V150%],0	<u>Transmitting data</u>
Command ⑦	SOCKSEND 2,1,4,5,V150%	The system transmits data to the client in response to its request.
IF E1%<0 THEN *ERROR		
'CLOSE SOCKET		
SOCKCLOSE 1		
SOCKCLOSE 2		<u>Closing sockets</u>
EXIT		The system closes sockets. Not only the socket used for standby but that used to transmit and receive data need to be closed.
*ERROR		
WINDOW 0,0,200,100		
PRINT #0,STR\$(E1%)		<u>Processing error</u>
PRINT #0,STR\$(E2%)		If any error occurs, the system will display the relevant error code on the teach pendant and, after a lapse of three seconds, exit the program.
SOCKCLOSE 1		To facilitate understanding, this sample shows a batch processing of errors.
SOCKCLOSE 2		
PAUSE 3000		
EXIT		



## 4.2 Getting Error Number by Unit (Server Program)

As an example of UDP program, the following section describes a program to return the code of error occurring in this controller. When the program starts running, the system will create a socket for UDP to wait for connection at Port Number 48952. When the client transmits a unit number as its request message, the system will get an error code from the SYSTEM functions and return data on integer value.

After transferring data, socket is closed and program will end.

Variable definition file 「SAMPLE.INC」 should be used in this program.

		Used variable
variable	NO	Used content
Integer variable	100	Receiving data size
Integer variable	120	Transmission data size
Integer variable	150	Receiving data size(number of unit)
Integer variable	200	Error code

### Example of creating program

```

INCLUDE "SAMPLE"
'CREATE SOCKET
SOCKCREATE SOCK_NO,UDP
IF E1%<0 THEN *ERROR

'ASSIGN PORT NUMBER
SOCKBIND SOCK_NO,PORT_NO
IF E1%<0 THEN *ERROR

'RECEIVE DATA
SOCKRCV SOCK_NO,BUF_NO,1,0,V100%
IF E1%<0 THEN *ERROR
GETBYTE BUF_NO,UNIT_NO,0
IF UNIT_NO<1 THEN *ERROR
IF UNIT_NO>9 THEN *ERROR

'TRANSMIT DATA
V3% = SYSTEM%(140+UNIT_NO)
SETINT BUF_NO,E_CODE,0
SOCKSEND SOCK_NO,BUF_NO,4,5,V3%
IF E1%<0 THEN *ERROR

'CLOSE SOCKET
SOCKCLOSE 1
'END

*ERROR
WINDOW 0,0,200,100
PRINT #0,STR$(E1%)
PRINT #0,STR$(E2%)
SOCKCLOSE 1
SOCKCLOSE 2
PAUSE 3000
EXIT

```

#### Receiving request message

Since UDP requires no connection processing, the system immediately waits for receiving data without waiting for connection (executing the SOCKWAIT command).

#### Receiving request message

The system gets and returns the code of error that is currently occurring on the specified unit with the use of the SYSTEM function.

#### Processing error

If any error occurs, the system will display the relevant error code on the teach pendant and, after a lapse of three seconds, exit the program.

To facilitate understanding, this sample shows a batch processing of errors.

### SAMPLE.INC

```

SOCK_NO,1
BUF_NO,1
PORT_NO,48952
TCP,0
UDP,1
UNIT_NO,V1%
E_CODE,V2%

```



About system function, please refer to the instruction manual "Robot language".  
About variable identification file, please refer to the instruction manual "User Task".



## 4.3 Getting Data from Vision System (Client Program)

The following section describes a program to get a shift amount from the vision system and assign it to the shift register. Communications with the vision system are performed using character strings and taking character strings that represent linefeed (two-byte data of 13, 10) as data delimiters. When the vision system receives the data request "GVA002 [Linefeed]", it will make an error response first, and then transmit the shift amount.

Used variable		
variable	NO	Used content
Integer variable	50	Receiving data size
Integer variable	100	Receiving data size
Integer variable	150	Receiving data (error reply)
Integer variable	151	Receiving data (offset)
Transmission string variable	10	Receiving data convert to transmission string data
Integer variable	160	Transmission string convert to retrieving data

### Example of creating program

```
'CONNECT VISION
```

```
'CREATE SOCKET
SOCKCREATE V150%,0
IF E1%<0 THEN *ERROR
```

```
'CONNECT TO VISION SYSTEM
SOCKCONNECT V150%,1,23,5
IF E1%<0 THEN *ERROR
```

```
'TRANSMIT DATA REQUEST MESSAGE
SOCKSENDSTR 1,"GVA002",LEN("GVA002"),0,V151%,3
IF E1%<0 THEN *ERROR
```

#### Transmitting data request message

The system transmits a data request message to vision system. Adding "3" to the last parameter of the SOCKSENDSTR command makes it possible to add two-byte data of 13, 10 representing linefeed.

```
'RECEIVE RESPOND MESSAGE
SOCKRECV V150%,1,3,5,V151%
IF E1%<0 THEN *ERROR
GETBYTE 1,V151%,0
IF V151%<>49 THEN *ERROR
```

```
'RECEIVE DATA
*GET_DATA
SOCKRECV V150%,1,1,5,V151%
IF E1%<0 THEN *ERROR
GETBYTE 1,V151%,0
IF 13=V151% THEN *GET_DATA_END
V10$ = V10$ + CHR$(V151%)
GOTO *GET_DATA
*GET_DATA_END
```

#### Transmitting data

The system gets data character by character and receives data until a data delimiter is received.

```
SOCKRECV V150%,1,1,5,V151%
```

```
'ASSIGN TO SHIFT REGISTER
V1% = VAL(V10$)
R1 = (0,V1%,0,0,0,0)
```

#### Assigning to shift register

The system uses the general function VAL to convert a character string into a numeric value and assign the value as a shift amount in the Y direction to the shift register.

```
SOCKCLOSE V150%
EXIT
```

```
'PROCESS ERROR
*ERROR
WINDOW 1,1,100,100
PRINT #0,STR$(E1%)
SOCKCLOSE V150%
PAUSE 5000
EXIT
```

#### Processing error

If any error occurs, the system will display the relevant error code on the teach pendant and, after a lapse of three seconds, exit the program. To facilitate understanding, this sample shows a batch processing of errors.

## 4.4 Data Receiving Procedure

To receive data with TCP, any of the following procedures is available to know the end of data.

1. Predetermining a data size
2. Transmitting data size with different data
3. Determining a character to be transmitted at the last

To perform communication using a socket, it is needed to know the end of data by any of the procedures. Procedure 1 is used for “4.1 Getting Value of Integer Variable (Server Program)”, while Procedure 3 is used for “4.3 Getting Data from Vision System (Client Program)”.

Also to transmit data, it is needed to make sure of what procedure is used on the receiving side and transmit the data.

### 4.4.1 Predetermining a data size

In this procedure, since the data size is already known, the system directly specifies data size to the third parameter of the SOCKRECV command to get the data. For example, if the data size is already known to be four bytes, specify “4” as shown below.

```
SOCKRECV 1, 1, 4, 5, V1%
```

### 4.4.2 Transmitting data size with different data

In this procedure, the system transmits a data size with data of predetermined size first, and then variable-length data. To transmit such data, describe as shown below. (In this case, the data size is to be transmitted in the four-byte big-endian system.)

```
SOCKRECV 1, 1, 4, 5, V1%
GETINT 1,%2%,0
SOCKRECV 1, 1, V2%, 5, V1%
```

Reception of data size

Reception of variable-length data

### 4.4.3 Determining a character to be transmitted at the last


In this procedure, the system transmits a data delimiter at the end of data. In this case, it receives data byte by byte and completes receiving data at the time when the data delimiter is received. In this case, the data ends as shown below. (“0” is to be transmitted as the data delimiter.)

```
*GET_DATA
SOCKRECV 1, 1, 1, 5, V1%
GETBYTE 1, V2%, 0
IF 0=V2% THEN *GET_DATA_END
V1$ = V1$ + CHR$(V2%)
GOTO *GET_DATA
*GET_DATA_END
```

**POINT**

In the procedure that determines a character to be transmitted at the last, data reception is processed byte by byte. Consequently, the number of steps tends to increase and processing time tends to be longer compared to other procedures. Particularly in case of low-priority user task, the processing time becomes longer.



 <b>NACHI</b> NACHI-FUJIKOSHI CORP.		<a href="http://www.nachi-fujikoshi.co.jp/">http://www.nachi-fujikoshi.co.jp/</a>	
<b>JAPAN MAIN OFFICE</b>	Phone: +81-3-5568-5245 Fax: +81-3-5568-5236	Shiodome Sumitomo Bldg. 17F, 1-9-2 Higashi-Shinbashi Minato-ku, TOKYO, 105-0021 JAPAN	
<b>NACHI NORTH AMERICA</b>		<a href="http://www.nachirobotics.com/">http://www.nachirobotics.com/</a>	
<b>North America Headquarters</b>	Phone: 248-305-6545	Fax: 248-305-6542	22285 Roethel Drive, Novi, Michigan 48375 U.S.A.
<b>Greenville Service Office</b>	Use 248-305-6545	Use 248-305-6542	South Carolina, U.S.A.
<b>San Antonio Service Office</b>	Use 248-305-6545	Use 248-305-6542	Texas, U.S.A.
<b>Kentucky Branch Office</b>	Phone: 502-695-4816	Fax: 502-695-4818	116 Collision Center Drive, Suite A, Frankfort, KY 40601 U.S.A
<b>Training Office</b>	Phone: 248-334-8250	Fax: 248-334-8270	22213 Roethel Drive, Novi, Michigan 48375 U.S.A.
<b>Toronto Branch Office</b>	Phone: 905-760-9542	Fax: 905-760-9477	89 Courtland Avenue, Unit 2, Vaughan, Ontario L4K3T4 CANADA
<b>Mexico Branch Office</b>	Phone : +52-555312-6556	Fax: +52-55-5312-7248	Urbina # 54, Parque Industrial Naucalpan, Naucalpan de Juarez, 53370, Estado de México, MEXICO
<b>Saltillo Service Office</b>	Phone : +52-844416-8053	Fax: +52-844416-8053	Canada 544 Privada Luxemburgo C. P. 25230, Saltillo, Coahuila, MEXICO
<b>NACHI ROBOTIC EUROPE</b>		<a href="http://www.nachi.de/">http://www.nachi.de/</a>	
<b>Germany</b>			
<b>Nachi Europe GmbH</b>	Phone: +49-(0)2151-65046-0	Fax: +49-(0)2151-65046-90	Bischofstrasse 99, 47809, Krefeld, GERMANY
<b>United Kingdom</b>		<a href="http://www.nachi.co.uk/">http://www.nachi.co.uk/</a>	
<b>Nachi U.K. LTD.</b>	Phone: +44-(0)121-250-1895	Fax: +44-(0)121-250-1899	Unit 7, Junction Six Industrial Estate, Electric Avenue, Birmingham B6 7JJ, U.K.
<b>Czech Republic</b>			
<b>Nachi Europe</b>	Phone: + 420-255-734-000	Fax: +420-255-734-001	Prague 9, VGP Park, Czech republic
<b>NACHI ROBOTIC ASIA</b>		<a href="http://www.nachi-korea.co.kr/">http://www.nachi-korea.co.kr/</a>	
<b>Korea</b>			
<b>Korea</b>	Phone: +82-(0)2-469-2254	Fax: +82-(0)2-469-2264	2F Dongsan Bldg. 276-4, Sungsu 2GA-3DONG, Sungdong-ku, Seoul 133-123, KOREA

Copyright NACHI-FUJIKOSHI CORP.

**Robot Division**

1-1-1, FUJIKOSHIHONMACHI, TOYAMA CITY, JAPAN 930-8511  
 Phone +81-76-423-5137  
 Fax +81-76-493-5252

NACHI-FUJIKOSHI CORP. holds all rights of this document. No part of this manual may be photocopied or reproduced in any form without prior written consent from NACHI-FUJIKOSHI CORP. Contents of this document may be modified without notice. Any missing page or erratic pagination in this document will be replaced.

In case that an end user uses this product for military purpose or production of weapon, this product may be liable for the subject of export restriction stipulated in the Foreign Exchange and Foreign Trade Control Law. Please go through careful investigation and necessary formalities for export.

Original manual is written in Japanese.

**NACHI-FUJIKOSHI CORP. ©**