

**Kawasaki Robot Controller  
D Series**

**TCP/IP  
Communication  
Manual**

Robot

Kawasaki Heavy Industries, Ltd.

## PREFACE

This manual gives instructions for using the 10BASE-T/100BASE-TX Ethernet-connected TCP/IP communication function with the D Series Controller.

This manual provides as much detailed information as possible on the standard operating methods for using this function. However, not every possible operation, condition or situation that should be avoided can be described in full. Therefore, should any unexplained questions or problems arise during robot operation, please contact Kawasaki Machine Systems. Refer to the contact information listed on the rear cover of this manual for the nearest Kawasaki Machine Systems office.

Read the D Controller Basic Manuals (including safety manual), delivered with the robot, without fail together with this manual. Do not perform any procedure described herein until the contents of this manual are fully understood.

- 
1. This manual does not constitute a guarantee of the systems in which the robot is utilized. Accordingly, Kawasaki is not responsible for any accidents, damages, and/or problems relating to industrial property rights as a result of using the system.
  2. It is recommended that all personnel assigned for activation of operation, teaching, maintenance or inspection of the robot attend the necessary education/training course(s) prepared by Kawasaki, before assuming their responsibilities.
  3. Kawasaki reserves the right to change, revise, or update this manual without prior notice.
  4. This manual may not, in whole or in part, be reprinted or copied without the prior written consent of Kawasaki.
  5. Store this manual with care and keep it available for use at any time. If the robot is reinstalled or moved to a different site or sold off to a different user, attach this manual to the robot without fail. In the event the manual is lost or damaged severely, contact Kawasaki.
- 

All rights reserved. Copyright © 2008 Kawasaki Heavy Industries Ltd.

## SYMBOLS

The items that require special attention in this manual are designated with the following symbols.

Ensure proper and safe operation of the robot and prevent physical injury or property damage by complying with the safety matters given in the boxes with these symbols.



### **DANGER**

**Failure to comply with indicated matters can result in imminent injury or death.**



### **WARNING**

**Failure to comply with indicated matters may possibly lead to injury or death.**



### **CAUTION**

**Failure to comply with indicated matters may lead to physical injury and/or mechanical damage.**

### **[ NOTE ]**

Denotes precautions regarding robot specification, handling, teaching, operation and maintenance.



### **WARNING**

- 1. The accuracy and effectiveness of the diagrams, procedures, and detail explanations given in this manual cannot be confirmed with absolute certainty. Should any unexplained questions or problems arise, please contact Kawasaki Machine Systems.**
- 2. Safety related contents described in this manual apply to each individual work and not to all robot work. In order to perform every work in safety, read and fully understand the safety manual, all pertinent laws, regulations and related materials as well as all the safety explanation described in each chapter, and prepare safety measures suitable for actual work.**

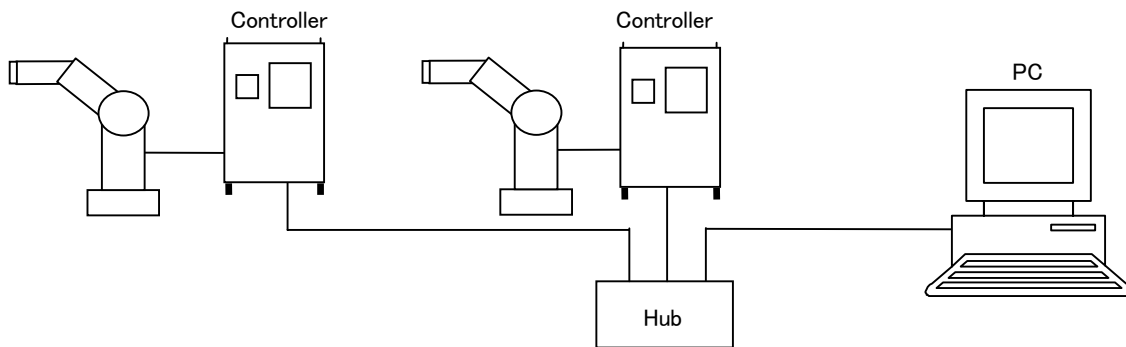
## CONTENT

Preface.....	1
Symbols.....	2
1.0 Overview of Ethernet-connected TCP/IP Communication .....	5
1.1 Remote Terminal Function.....	5
1.2 Socket Communication Function.....	5
1.2.1 TCP Communication Instructions.....	6
1.2.2 UDP Communication Instructions .....	7
1.3 Support Specifications for TCP/IP Communication.....	8
1.4 File Transfer Function .....	8
2.0 Ethernet Network Configuration and Connection with Robot Controller.....	9
2.1 Direct Connection without Hub.....	9
2.2 Network Connection with Hub .....	10
3.0 Setting IP Address and Confirming Connection.....	11
3.1 Setting IP Address and Subnet Mask .....	11
3.2 Checking Connection to the Network.....	13
4.0 Using KCwinTCP Remote Terminal Function .....	15
4.1 What is KCwinTCP? .....	15
4.2 System Requirements for KCwinTCP.....	15
4.3 Installing KCwinTCP and Registering Robot IP Address .....	16
4.4 Operating KCwinTCP.....	17
4.4.1 Daily Operations .....	17
4.4.2 Menu .....	19
4.4.2.1 File(F).....	19
4.4.2.2 Com(munication) (C).....	20
4.4.2.3 View (V).....	22
4.4.2.4 Help (H) .....	22
4.4.3 Vertical/ Horizontal Scroll Bar.....	22
4.4.4 Title Display .....	22
4.5 Notes and Restrictions in Using KCwinTCP.....	23
4.6 To Avoid Saving/Loading of Mistaken Robot Data.....	24

5.0	Data Communication Instructions in UDP .....	26
5.1	Communication Errors in UDP_SENDTO and UDP_RECVFROM .....	31
6.0	Data Communication Instructions in TCP .....	32
6.1	Communication Errors in Socket Communication .....	47
6.2	Sample Program Using Socket Communication Instructions .....	49
6.2.1	When Robot is the Server-Side .....	49
6.2.2	When Robot is the Client-Side .....	52
7.0	Save/Load Files using FTP Client Function .....	54
7.1	Software Settings .....	55
7.1.1	Settings on the robot controller .....	55
7.1.2	Settings on Remote Host .....	55
7.2	FTP auxiliary functions .....	57
7.2.1	Aux. 0201 Save .....	57
7.2.2	Aux. 0202 Load .....	59
7.2.3	Aux. 0203 File configuration .....	60
7.2.4	Aux. 0205 Delete file .....	61
7.2.5	Aux. 0206 Rename file .....	62
7.2.6	Aux. 0210 Autosave configuration .....	62
7.3	Communication Error codes which occur with the ftp client function .....	65
8.0	Error Message .....	66

## 1.0 OVERVIEW OF ETHERNET-CONNECTED TCP/IP COMMUNICATION

To use the functions below, create an Ethernet network as shown in the diagram below by connecting Ethernet port on main CPU board with external devices (e.g. PC) using 10BASE-T/100BASE-TX cable.



- Remote terminal function
- Socket communication function

### 1.1 REMOTE TERMINAL FUNCTION

By installing KCwinTCP, software provided by Kawasaki, in a Windows computer, operations such as entering AS monitor commands from PC are made possible via Ethernet-connected TCP/IP communication in addition to the RS232C communication (standard feature).

### 1.2 SOCKET COMMUNICATION FUNCTION

This communication function provides commands based on socket interface in TCP/IP communication enabling data communication between robot controller and other devices.

Two types of communication instructions are provided:

1. TCP communication instructions
2. UDP communication instructions

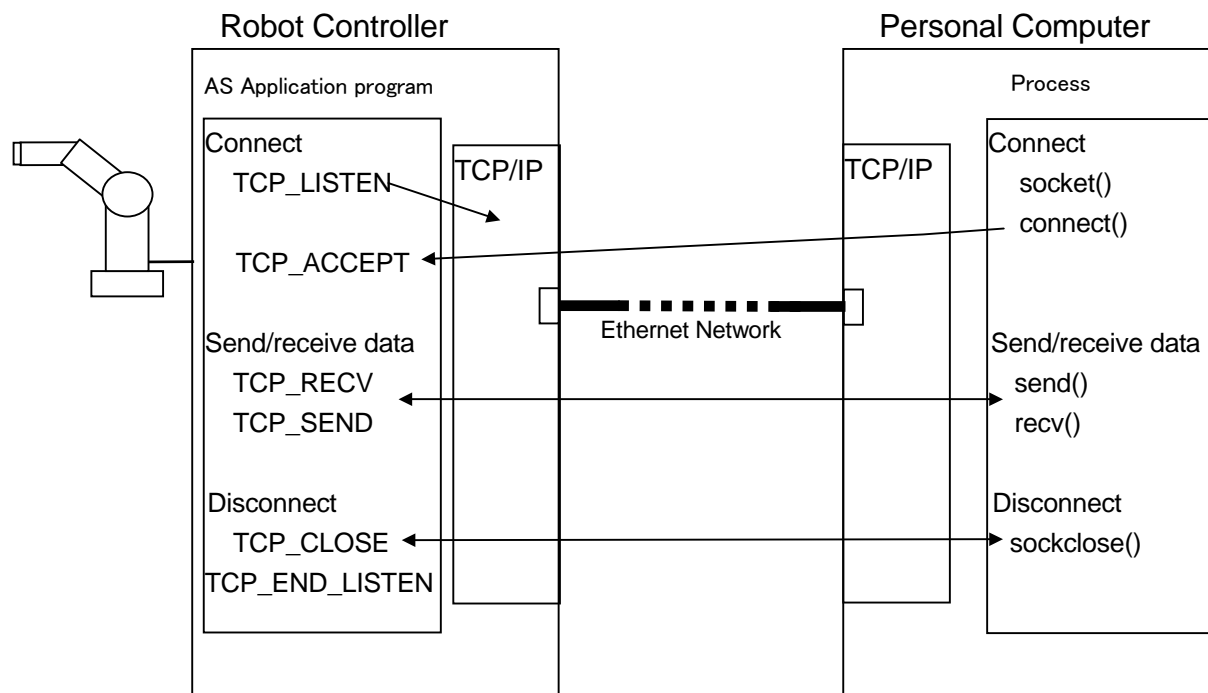
In TCP communication, data is automatically re-sent when communication errors occur. It is used in applications where certainty of the communication is important. In UDP communication, data is not re-sent after an error. It is used when speed is important.

## 1.2.1 TCP COMMUNICATION INSTRUCTIONS

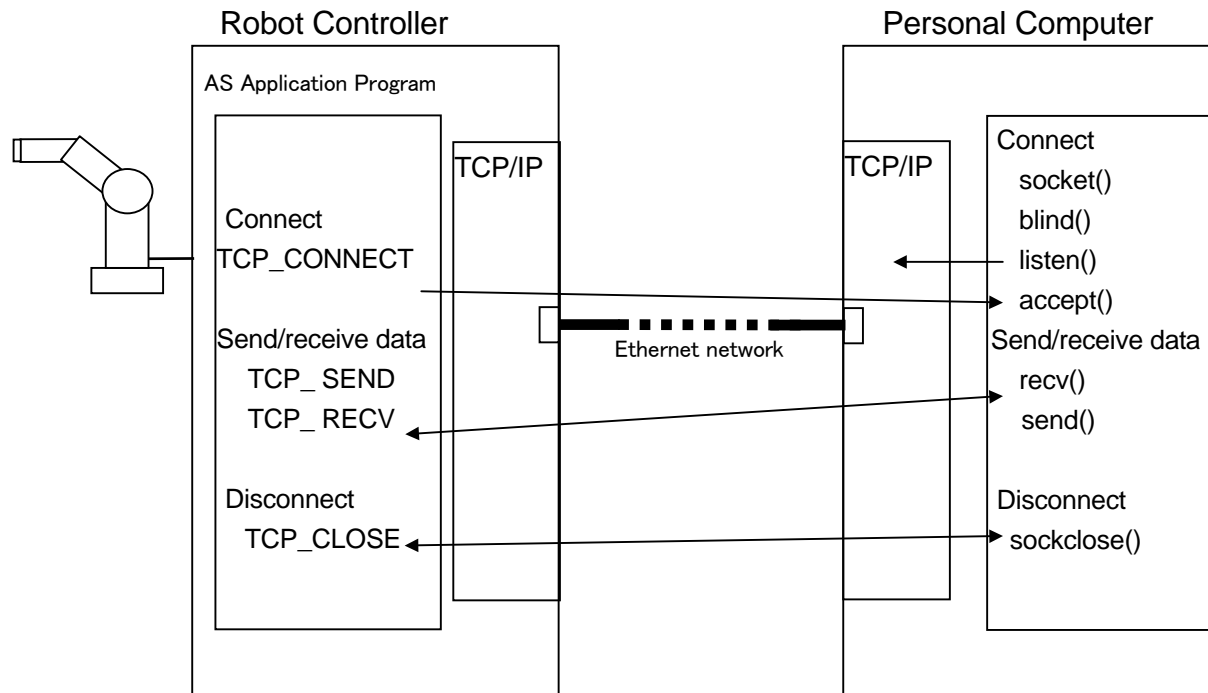
Available TCP communication instructions:

- |                 |  |
|-----------------|--|
| •TCP_LISTEN     | Creates socket and waits for connection requests |
| •TCP_ACCEPT     | Checks if connection request is received         |
| •TCP_CONNECT    | Creates a socket and sends connection request    |
| •TCP_SEND       | Sends data string                                |
| •TCP_RECV       | Receives data string                             |
| •TCP_CLOSE      | Aborts socket communication                      |
| •TCP_END_LISTEN | Ends waiting for connection request              |

Example: Robot as server

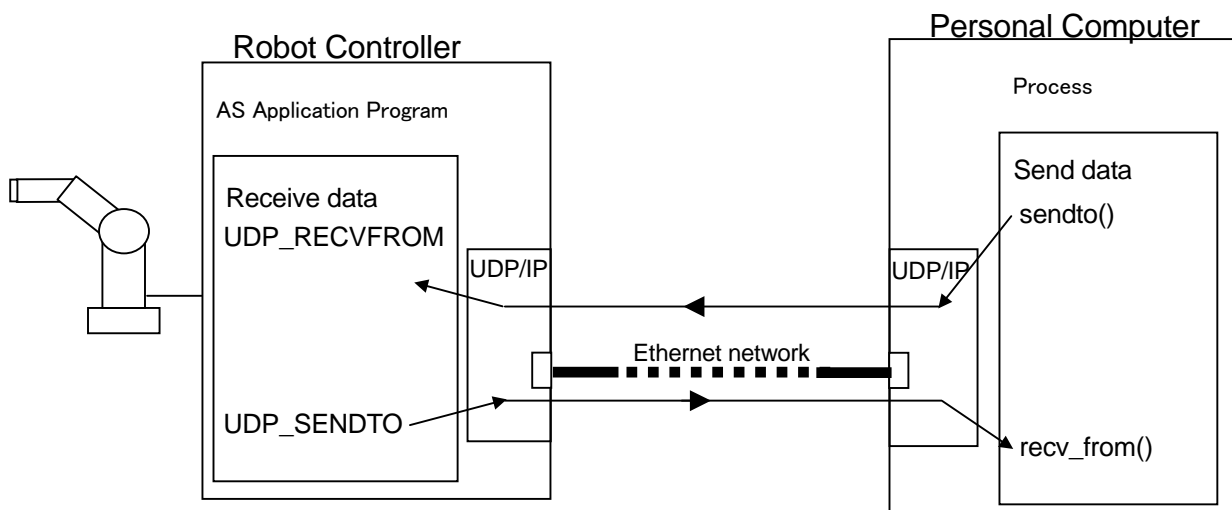


Example: Robot as client



### 1.2.2 UDP COMMUNICATION INSTRUCTIONS

There are two UDP communication instructions: UDP\_SENDTO (sends data) and UDP\_RECVFROM (receives data).



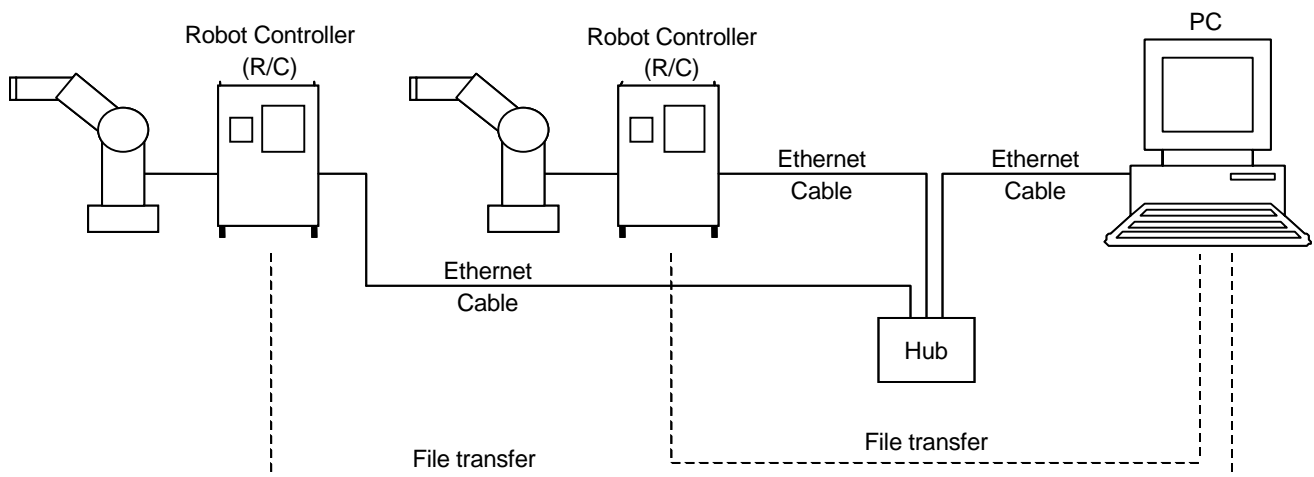


### 1.3 SUPPORT SPECIFICATIONS FOR TCP/IP COMMUNICATION

1. 10BASE-T/100BASE-TX Ethernet
2. Network connection transmission rate: 10Mbps/100Mbps
3. Connected using RJ045 Connector
4. Protocols: IP, ARP, RARP, ICMP, TCP, UDP, TELNET (Server)

### 1.4 FILE TRANSFER FUNCTION

The robot controller is equipped with FTP client function so that saving/loading can be done for all types of data: robot, program, auxiliary data, error logs and all the data by using (TCP/IP) communication in same way as a PC card. Operation is performed by auxiliary function on teach pendant.



i.e)

- AS taught data is saved from R/C to the server through LAN by instructions from R/C.
- AS taught data is loaded from the server through LAN to R/C by instructions from R/C.

## 2.0 ETHERNET NETWORK CONFIGURATION AND CONNECTION WITH ROBOT CONTROLLER

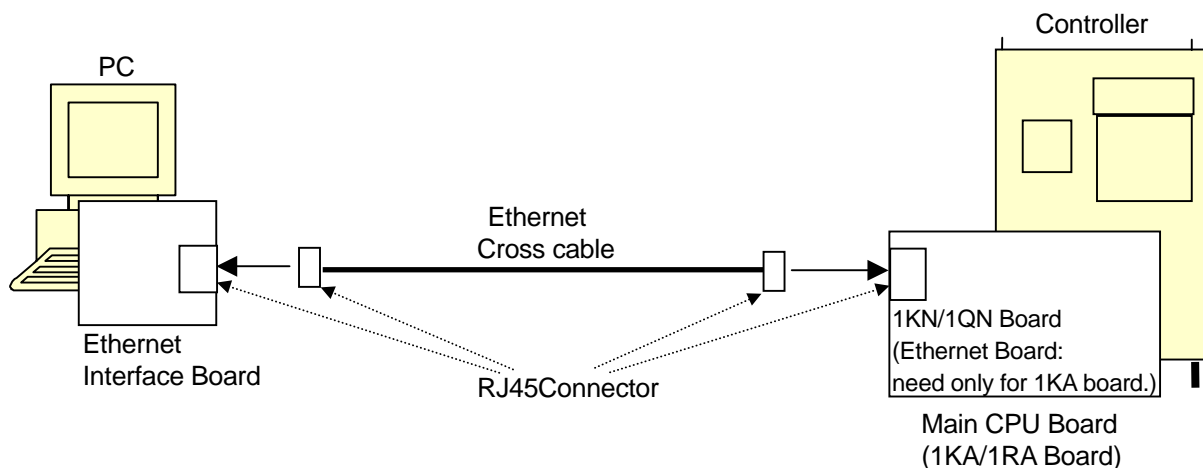
This section explains how an Ethernet network is created using an example of connecting a PC (external computer) with a robot controller.

The PC can be connected with the robot controller in two ways:

1. Connection of one computer to one controller without using a hub
2. Connection of two or more computer systems (including robot controllers) using Ethernet via a hub.

### 2.1 DIRECT CONNECTION WITHOUT HUB

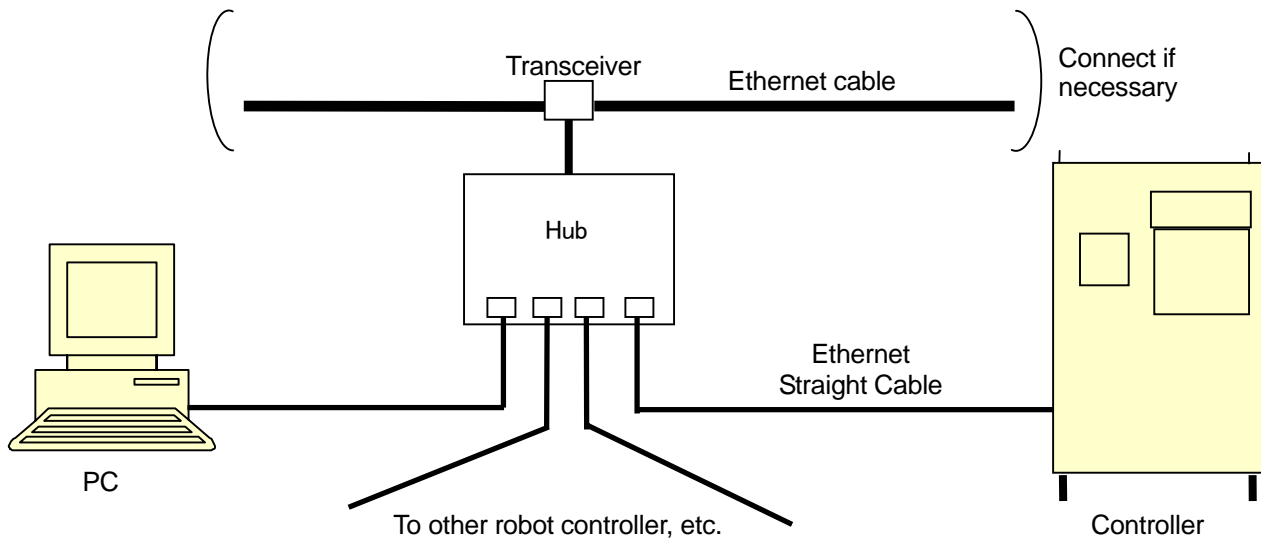
This diagram shows connection without hub.



In this case, use Ethernet cross cable for connecting.

## 2.2 NETWORK CONNECTION WITH HUB

This diagram shows the Ethernet network using a hub for 10BASE-T/100BASE-TX connection.



In this case, use Ethernet straight cable for connection.

### 3.0 SETTING IP ADDRESS AND CONFIRMING CONNECTION

To set the IP address and confirm connection, the robot controller and a Windows-based PC have to be connected to the network. After connecting the hardware, an IP address for the robot controller must be set on the network in order to use the TCP/IP communication function.

#### 3.1 SETTING IP ADDRESS AND SUBNET MASK

Register the IP address and the subnet mask for the controller before using the TCP/IP communication function.

To set the IP address and subnet mask, execute NETCONF monitor command via teach pendant or a KCwin/ KCwinTCP remote terminal screen on a RS232C-connected PC.

##### Example

```
>NETCONF 
IP address = 192.168.0.2
Change ? (If not, press RETURN only)
123.123.123.123 

IP address = 123.123.123.123
Change ? (If not, press RETURN only) 

Host name =
Change ? (If not, press RETURN only)
KAWASAKI 

Host name = KAWASAKI
Change ? (If not, press RETURN only) 

Subnet mask = 255.255.240.0
Change ? (If not, press RETURN only)
255.255.255.0 

Subnet mask= 255.255.255.0
Change ? (If not, press RETURN only) 
Gateway IP = 0.0.0.0
```

Change ? (If not, press RETURN only)

255.255.255.0

Gateway IP = 255.255.255.0

Change ? (If not, press RETURN only)

DNS server 1 IP = 0.0.0.0

Change ? (If not, press RETURN only)

123.123.123.1

DNS server 1 IP = 123.123.123.1

Change ? (If not, press RETURN only)

DNS server 2 IP = 0.0.0.0

Change ? (If not, press RETURN only)

123.123.123.2

DNS server 2 IP = 123.123.123.2

Change ? (If not, press RETURN only)

Domain name =

Change ? (If not, press RETURN only)

kawasaki\_robot

Domain name = kawasaki\_robot

Change ? (If not, press RETURN only)

MAC address = 00:09:0f:00:ff:01

(W1018)Turn control power ON/OFF to change network setting.

>

IP address can be set via teach pendant. See the Operation Manual for setting the IP address via TP (Auxiliary function 0812 Network Setting).

After registering the IP address, reboot the robot controller by turning control power OFF → ON.

### 3.2 CHECKING CONNECTION TO THE NETWORK

Once the robot controller Ethernet-connected interface is opened and properly started, connect a Windows PC to the same network and confirm the connection using the ping command in MS-DOS Prompt screen. Before checking with ping command, the Windows PC needs to be set for Ethernet connection (register IP address and subnet mask) and rebooted.

1. Open MS-DOS Prompt screen in a Windows PC.
2. In MS-DOS Prompt screen, enter:

```
>ping IP_address
```

IP\_address is the IP address of the robot controller to be connected. If the connection is functioning properly, the following message appears.

**Example** C:>ping 001.002.003.004  
Pinging 001.002.003.004 with 32 bytes of data:  
Reply from 001.002.003.004: bytes=32 time=53ms TTL=64  
Reply from 001.002.003.004: bytes=32 time=5ms TTL=64  
Reply from 001.002.003.004: bytes=32 time=24ms TTL=64  
Reply from 001.002.003.004: bytes=32 time=11ms TTL=64  
Ping statistics for 001.002.003.004:  
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),  
Approximate round trip times in milli-seconds:  
Minimum = 5ms, Maximum = 53ms, Average = 23ms

If the connection is not properly established, error messages such as timeout error will be displayed.

**Example** C:>ping 001.002.003.005  
Pinging 001.002.003.005 with 32 bytes of data:  
Request timed out.  
Request timed out.  
Request timed out.  
Request timed out.  
Ping statistics for 001.002.003.005:  
Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),  
Approximate round trip times in milli-seconds:  
Minimum = 0ms, Maximum = 0ms, Average = 0ms

This error can occur due to following causes. Check and correct the error and retry until connection is successful.

1. The power for hub is not ON.
2. The network cable connector is faulty or not connected correctly.
3. Incorrect cable is used.
4. IP address setting for the robot is not correct.

Generally, there are LEDs on the hub or Ethernet connection interface devices. The signal status can be checked by these LEDs. Use this information for troubleshooting when connection does not succeed.

## 4.0 USING KCWINTCP REMOTE TERMINAL FUNCTION

While KCwin has only RS232C terminal function, this also has Ethernet-connected remote terminal function. When installed on Windows PC, this software allows remote operation of robots from PC. But robot controller to be used needs to be specified by IP address before the terminal operation. When the robot is successfully connected, terminal operations such as entering monitor commands are possible. After the necessary operations are finished, end the connection using the designated instructions.

Three PCs can be connected to one controller at a time. On the other hand, when a controller is connected to three PCs, remote terminal operations from other PC are not possible until that connection is terminated.

Instructions for operating the KCwinTCP remote terminal function are explained below.

### 4.1 WHAT IS KCWINTCP?

KCwinTCP is terminal software for the Kawasaki robot controller. When a PC is installed with this software and connected to controller by Ethernet connection, the following operations are possible by TCP/IP communication.

- AS monitor command input from PC
- Saving/loading controller memory contents to/from PC etc.

### 4.2 SYSTEM REQUIREMENTS FOR KCWINTCP

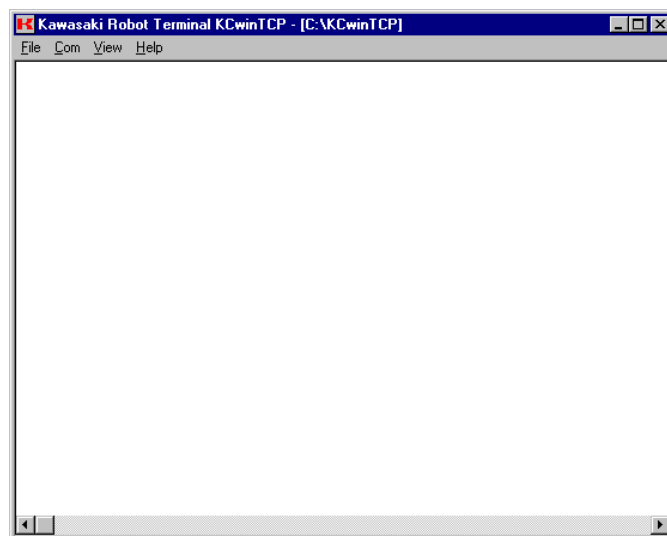
Hardware	PC with CPU of 80486 or more, and operating with Microsoft Windows
OS	Microsoft Windows 95/98/NT4.0/2000/XP
Confirmed operation models*	Toshiba PC Dynabook Satellite 2520 (Windows98) Compaq Armada 1500C (Windows98) IBM ThinkPad 365X (Windows95)

**NOTE\*** Kawasaki is not responsible for use of KCwinTCP on unconfirmed hardware/OS.

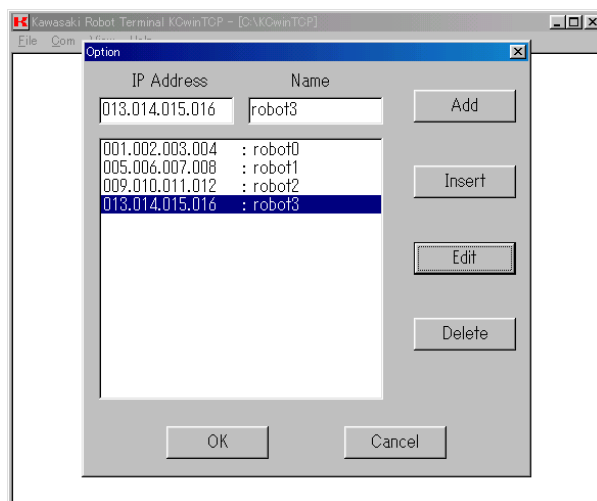


### 4.3 INSTALLING KCWINTCP AND REGISTERING ROBOT IP ADDRESS

Install (copy) KCwinTCP provided by Kawasaki into the desired directory of your Windows PC. After installing, double-click on the KCwinTCP icon. The window below appears.



Register the robot controllers to be connected. Select [Com]→[Options] from the menu bar.



In the window that appears, enter the IP address and the name for the robot controllers on the network and click <OK>.

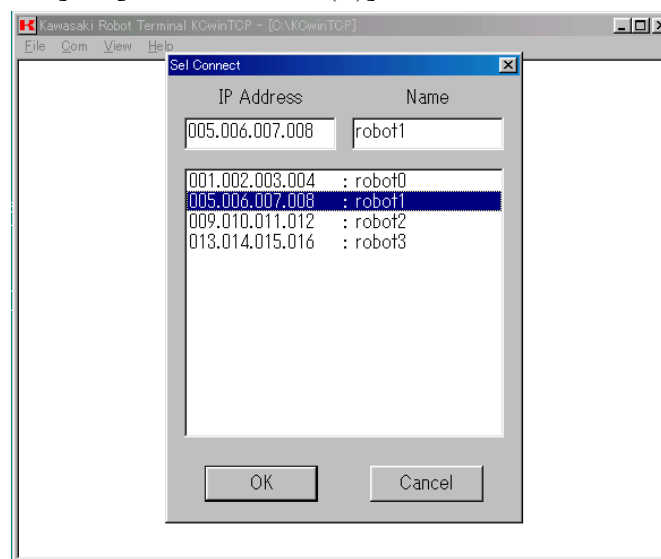
## 4.4 OPERATING KCWINTCP

### 4.4.1 DAILY OPERATIONS

1. Turn ON the controller. Do not initialize the controller. TELNET server function starts and the controller is now waiting for connection from client PC.

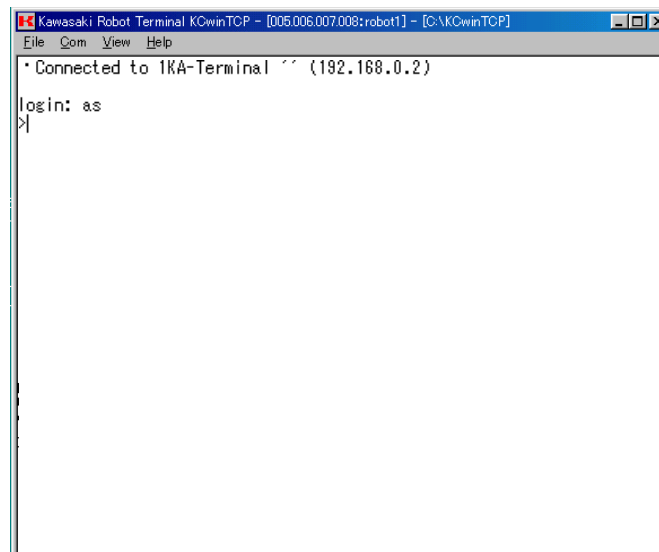
2. Start KCwinTCP

Double-click on the KCwinTCP icon. KCwinTCP starts and a window appears. In this window, select [Com] → [Sel Connection (E)] from the menu bar.



Select the IP address of the robot to be connected using the cursor keys. Click <OK>.

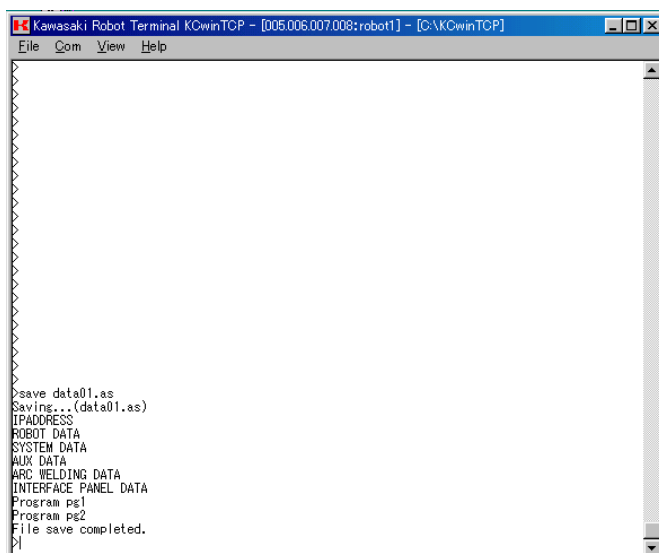
PC tries to connect with the specified IP address. When the connection is successfully established, login: appears following several other messages. Enter “as” after this message, and the robot will return a prompt.



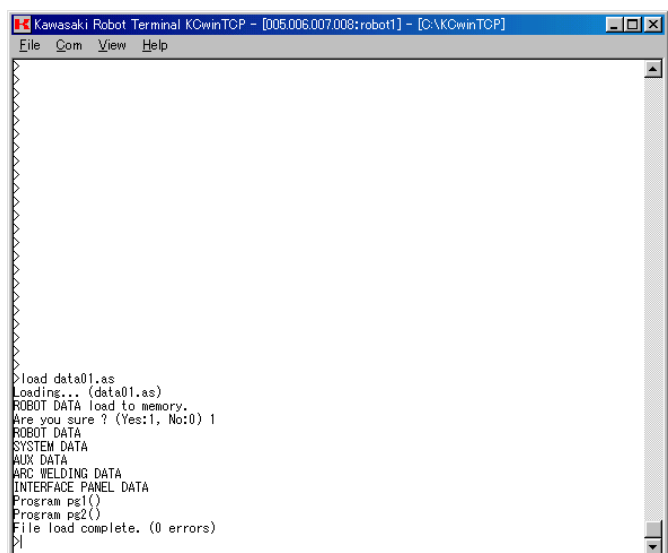
Now, AS monitor commands can be entered from the PC.

When uploading or downloading teach data between PC and controller, enter SAVE/LOAD monitor commands the same as in AS terminal.

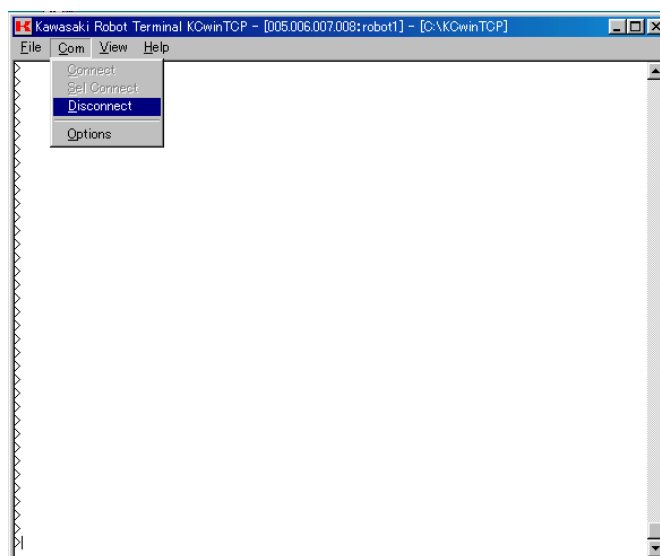
▪ When executing SAVE command



When executing LOAD command



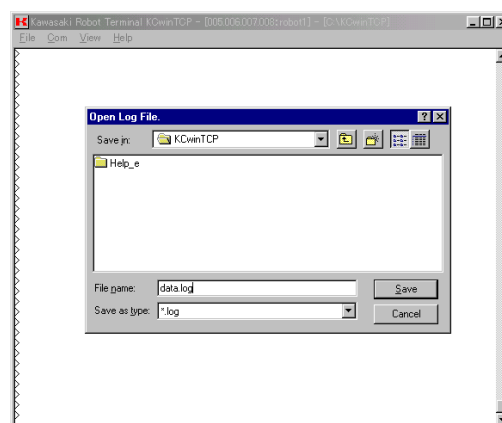
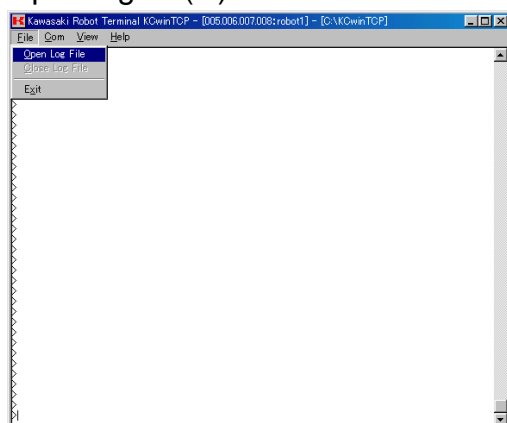
After all operations are finished, abort the connection. Select [Communication] → [Disconnect] from the menu bar.



## 4.4.2 MENU

### 4.4.2.1 FILE(F)

#### Open log file(O)



To save the characters displayed on screen as a log file, specify a file as above. (Default for extension log files is .log)

#### Close log file(C)

Ends logging.

#### Exit(X)

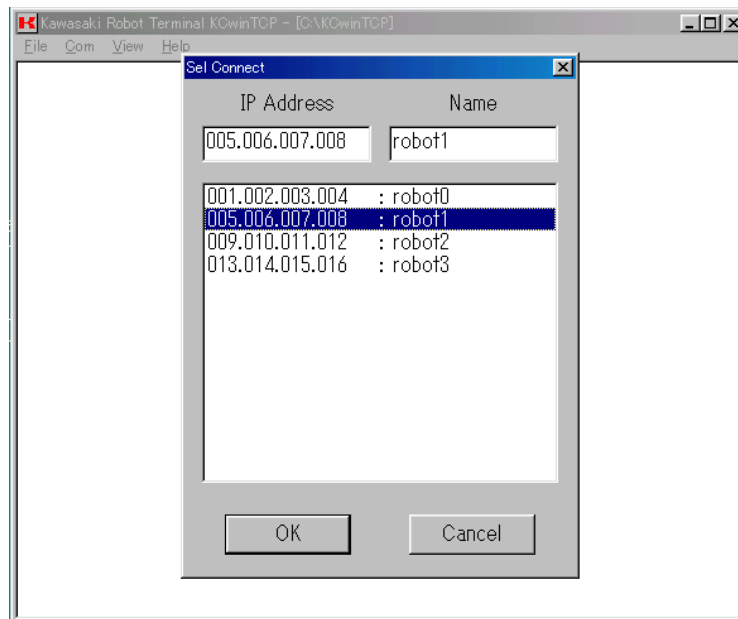
Terminates KCwinTCP.

#### 4.4.2.2 COM(MUNICATION) (C)

##### Connect(C)

Establishes connection with the robot whose IP address and port number were last used.

##### Sel(ect) Connect(ion)(E)



The IP addresses and names registered in [Options (O)] are displayed in a list. Select the IP address and name of the robot to be connected and click <OK> to establish connection with that robot.

##### IP address

Displays the selected IP address.

##### Name

Displays the selected name (nickname).

##### <OK>

Connects with the selected IP address.

##### <Cancel>

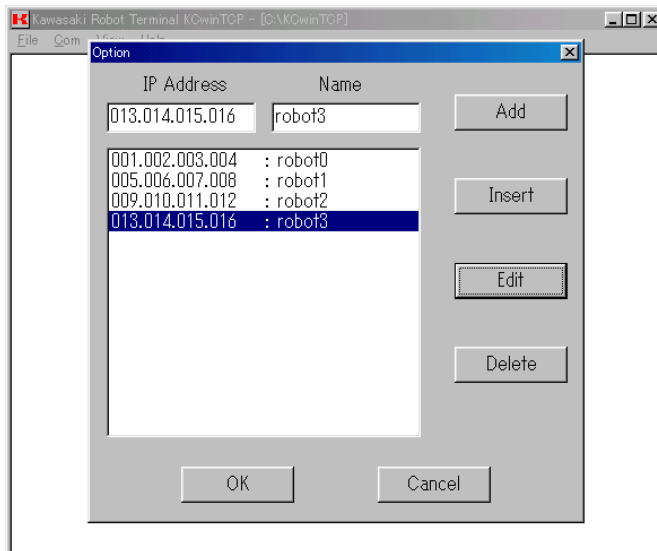
Terminates without making any connections.

##### Disconnect (D)

Terminates connection.

## Options (O)

Registers the IP address and names of the robots on the network. Maximum of 256 robots can be registered.



### IP address

Enter the IP address of the robot to be connected.

### Name

Enter the name (nickname) of the robot to be connected.

### <Add>

Adds the content input in the IP address and name.

### <Insert>

Inserts the content input in IP address and name where the cursor is.

### <Edit>

Overwrites the selected IP address and name with new contents.

### <Delete>

Deletes the selected IP address and name from the list.

### <OK>

Overwrites contents and puts in effects all changes made to KCWINTCP.INI and terminates the Option screen.

<Cancel>

Terminates the screen without saving the changed contents. (KCWINTCP.INI is not overwritten.)

#### **4.4.2.3 VIEW (V)**

Font (F)

Selects the font of the characters used on the screen.

SCREEN (W)

Sets the number of scroll buffers, colors of the screen and text.

#### **4.4.2.4 HELP (H)**

Contents (C)

Opens the table of contents for KCwinTCP operation instructions.

About KCwin (A)

Displays KCwinTCP version information.

### **4.4.3 VERTICAL/ HORIZONTAL SCROLL BAR**

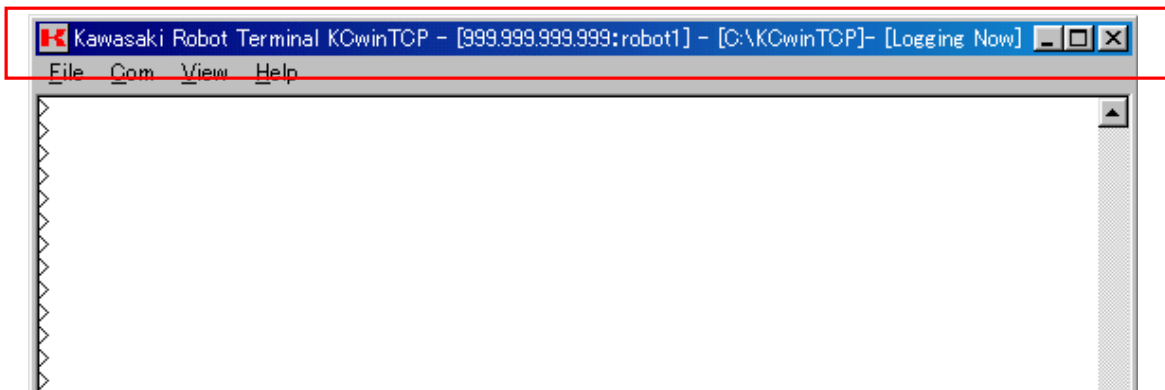
The vertical scroll bar appears when the data exceeds the number of rows in the screen.

Maximum of 200 rows can be displayed using the scroll bar.

#### **4.4.4 TITLE DISPLAY**

The following information is shown on the title bar.

### Example



[192.168.29.128: econ]

The current client is shown in the format [client IP address: name]. When no connection is established, nothing is displayed here.

[C:¥KCWINTCP]

If only the file name is specified in AS language LOAD/SAVE command, the data will be saved/ loaded to/ from this directory.

[Logging]

This is displayed while the screen data is being saved to the log file.

## 4.5 NOTES AND RESTRICTIONS IN USING KCWINTCP

When connection is aborted during an execution of a monitor command, that command (e.g. EDIT) should first be terminated in the robot controller using ABORT.

Do not disconnect from the KCwinTCP side until the command has been executed. If you have disconnected by mistake, immediately re-connect and check if that command was executed properly.

When the connection is aborted from the robot side (i.e. the OS displays a disconnection message), KCwinTCP automatically cuts off connection. If it does not cut off automatically, manually disconnect KCwinTCP.

AS commands FDIRECTORY, FDELETE, FORMAT are not supported in KCwinTCP (Use Windows Explorer to perform such operations.)



## 4.6 TO AVOID SAVING/LOADING OF MISTAKEN ROBOT DATA

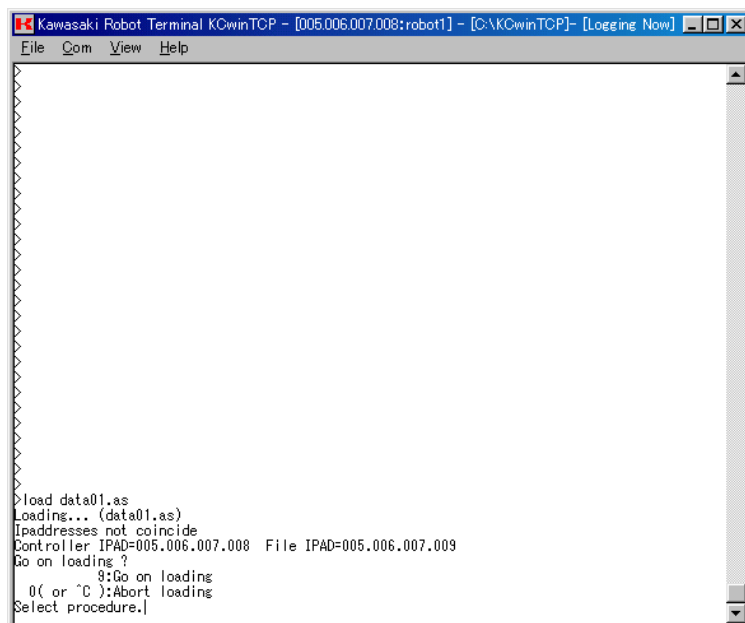
Because more than one robot is connected to the network, there is the chance that a wrong robot data may be loaded to the robot. Also, data from a wrong robot may overwrite a saved file.

To avoid this, KCwinTCP checks if the data is correct before loading.

At the beginning of the data, the IP address data is stored when saving the file. (See line starting with .NETCONF below).

```
*****
.*==== AS GROUP ===           : AS_01000107 2003/01/23 12:05
.*USER IF AS                  : UAS01000107 2003/01/23 12:01
.*USER IF TP                  : UTP01000107 2003/01/23 12:02
.*ARM CONTROL AS              : AAS01000107 2003/01/22 17:47
.*USER IF AS MESSAGE FILE     : MAS000107EN 2003/01/22 16:41
.*USER IF TP MESSAGE FILE     : MTP000107EN 2003/01/22 16:42
.*ARM DATA FILE              : ARM01000107 2003/01/22 16:16
.*USER IF IPL                 : UIP01190000 2002/12/25
.*ARM CONTROL IPL             : AIP01170000 2003/01/09
.*==== SERVO GROUP ===       : SV_02000005 2003/02/05 17:24
.*ARM CONTROL SERVO           : ASV010000-1 2003/02/04 12:03
.*SRV DATA FILE              : ASP010000-1 2003/01/29 09:10
.*ARM CONTROL SERVO CONT.    : ARMSH Ver1.2.3 Dec-05-2001 ZX165U ; (shi
.*   [Shipment setting data]
.*There is no Shipment setting data.
*****
.NETCONF
123.123.123.123,"KAWASAKI",255.255.255.0,255.255.255.0,123.123.123.1,123.123.123.2,"kawasaki
_robot"
.ROBOTDATA1
ZROBOT.TYPE    10  11   6   1      -421   FS010N-A001 ( 2003-02-06 14:36 )
ZSYSTEM        1   5   1      -106
ZLINEAR        0   0   0   0   0   0   -1  -1  -1  -1  -1  -1  -1  -1  -1  -1  -1
179
CONT_CODE      701      -10516
ZZERO          268435456 268435456 268435456 268435456 268435456 268435456 268435456
268435456 268435456 268435456 268435456 268435456 268435456 268435456 268435456
268435456 268435456 268435456 536870911
```

KCwinTCP checks if the IP address in the line beginning with .NETCONF is the same as the IP address of the controller. If they are not the same, KCwinTCP asks whether loading should be continued or not. Check if the data is correct and enter 9 to continue, and 0 to terminate.



## 5.0 DATA COMMUNICATION INSTRUCTIONS IN UDP

The following instructions enable data communication between robot controller and computer system via Ethernet UDP/IP.

UDP\_SENDTO . . . . . Program Instruction to send data

Sends the specified character string data based on UDP/IP. This instruction creates socket, sends data and closes the socket in one sequence.

UDP\_RECVFROM . . . . . Program Instruction to receive data

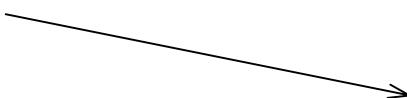
Receives and stores data in the specified character string array based on UDP/IP. This instruction creates socket, receives data and closes the socket in one sequence.

### Example

Keyword



Parameter




---

**UDP\_SENDTO   variable for returned value, IP address array variable, ...**

---

Parameters marked with   can be omitted.

Always enter a space between the keyword and the first parameter.

 represents the Enter (Return) key in the examples.

---

Program Instruction

---

**UDP\_SENDTO** variable for returned value, IP address array variable,  
port number, character string variable array for sending data,  
number of elements, **timeout**

---

**Function**

Sends data string based on UDP protocol. The data to be sent is specified in a character string variable array. This instruction creates socket, sends data and closes socket in one sequence. If communication error occurs, the error code is stored in the returned value storage variable. However, program execution does not stop.

**Parameter**

Variable for returned value

- Specifies variable that stores the execution results.
- 0 is stored when execution completes normally.
- Error code is stored when communication error occurs. However, program execution does not halt at communication error.
- Error code is not stored when non-communication errors occur (e.g. parameter error). In this case, the program execution results in error and stops.
- See 5.1 for details on communication error codes.

IP address array variable

- Specifies the first element of the array variable storing the IP address of the destination computer system. The IP address is stored in 8 bits increments to each element of the array variable.
- Data beyond 0 - 255 is regarded as error. The first data must be between 0 - 247, if not an error occurs.

Port number

- Specifies the port number that indicates the channel of the other end of the connection (process). The socket is bound to this port number.
- Acceptable range is 8192 - 65535, otherwise error occurs.

Character string variable array for sending data

- Specifies the first element of the character string array that stores the data to be sent.
- Array elements are sent in order from first to last.
- Change the data to be sent into character string format. Numeric data can be sent once transformed into character string format. Ensure the encoding format is consistent with that in the recipient (big endian/little endian, etc.).

- The maximum amount of data that can be sent in execution of one instruction is 1472 bytes.

#### Number of elements

- Specifies the number of elements to send in the character string variable array.
- These array elements are sent to the recipient in order of index number.

#### Timeout

- Specifies the time (seconds) to wait before outputting timeout error when connection is not established. Default is 1 second.
- When a timeout error occurs, the error code is stored in the returned value storage variable.
- Timeout error does not stop AS program execution.

#### Example

Increasing the count whenever robot operation is performed.

```
.PROGRAM pg1
:
Motion instructions
:
count = count + 1
$cnt[0]=$ENCODE(/D,count)
CALL send(.$cnt[])
.END
```

Sending the number of motions robot performed.

```
.PROGRAM send(.$cnt[])
UDP_SENDTO ret,ip[0],49152,.$cnt[0],1,2
IF ret<>0 THEN
TYPE "ERROR!"
HALT
END
.END
```

---

Program Instruction

---

**UDP\_RECVFROM** variable for returned value, port number,  
character string variable array for received data, number of elements,  
timeout, IP address array variable, Max. number of bytes

---

**Function**

Receives and stores data in the character string variable array based on UDP protocol. This instruction creates socket, receives data, and closes socket in one sequence. If a communication error occurs, the error code is stored in the returned value storage variable, and program execution does not stop.

**Parameter**

Variable for returned value

- Specifies variable that stores the execution results.
- 0 is stored when execution completes normally.
- Error code is stored when communication error occurs. However, program execution does not halt at communication error.
- Error code is not stored when non-communication errors occur (e.g. parameter error). In this case, the program execution results in error and stops.
- See 5.1 for details on communication error codes.

Port number

- Specifies the port number that indicates the channel of the other end of the connection (process). The socket is bound to this port number.
- Acceptable range is 8192 – 65535, otherwise error occurs.

Character string variable array for received data

- Specifies the first element of the character string variable array that stores the received data.
- Data is received as continuous character data of 1 byte each. When receiving numeric data (transformed into character string data), ensure the encoding format is consistent with the sender (big endian/ little endian, data segmentation etc).
- The maximum amount of data that can be received in one instruction is 1472 bytes.

Number of elements

- Sets the variable to substitute the number of elements in the received character string variable array.
- Should be specified as a variable.

#### Timeout

- Specifies the time (seconds) to wait before outputting timeout error when connection is not established. The default setting is 1 second.
- When a timeout error occurs, the error code is stored in the returned value storage variable.
- Timeout error does not stop AS program execution.

#### IP address array variable

- Specifies the first element of the array variable to store the IP address of the node that sent the data. The 32 bits in the IP address is divided into 4 elements with 8 bits each, and each element is stored in the array elements in the specified array.

#### Maximum number of bytes

- Specifies the maximum number of bytes that can be stored in an element of the character string variable array.
- Acceptable range is 1 – 255, otherwise error occurs. Default setting is 255 bytes.

#### Example

```
Robot activated by value .start_flg.
.PROGRAM pg1
CALL rcv(.start_flg)
IF .start_flg == 1 THEN
:
Motion instructions
:
END
.END

Operating the robot based on operations received from another robot
.PROGRAM rcv(.start_flg)
UDP_RECVFROM ret,49152,$cnt[0],p,1,ip[0],255
IF ret<>0 THEN
TYPE "ERROR!"
HALT
END
num = VAL($cnt[0])
IF num%10 == 0 THEN
.start_flg = 1
ELSE
.start_flg = 0
END
.END
```

## 5.1 COMMUNICATION ERRORS IN UDP\_SENDTO AND UDP\_RECVFROM

Code	Error message	Description	Countermeasure
E4019	TCPIP) Socket open failure. CODE=XX Dst.IP=XX.XX.XX.XX	Failed to open socket when executing UDP_SENDTO/UDP_RECVFROM. Too many sockets may have been opened. Or, parameter error e.g. wrong IP address.	Do not execute too many of this instruction. Fix incorrect parameter. CODE shows the SUB error code of TCP/IP package.
E4020	TCPIP) Socket close failure. CODE=XX Dst.IP=XX.XX.XX.XX	Failed to close socket when executing UDP_SENDTO/UDP_RECVFROM. Possible problem along Ethernet or at other end of connection.	Check for problems in node at the other end of connection or in the connection line. Fix accordingly. CODE shows the SUB error code of TCP/IP package.
E4021	TCPIP)Communication Error. CODE=XX Dst.IP=XX.XX.XX.XX	Communication error during execution of UDP_SENDTO/UDP_RECVFROM. Possible problem along Ethernet or at other end of connection.	Check for problems in node at the other end of connection or in the connection line. Fix accordingly. CODE shows the SUB error code of TCP/IP package.
E4022	TCPIP)Too Long Message.	Tried to send data longer than the limit while executing UDP_SENDTO/UDP_RECVFROM.	Fix data to be within limit size and re-try the program.
E4023	TCPIP)Cannot reach the Host	Could not connect with the host when executing UDP_SENDTO/UDP_RECVFROM. There is no host with this IP address on the network. Port number or IP address may be wrong. Or, possible problem along Ethernet or at other end of connection.	Check if port number or IP address is correct and correct it if it is wrong. Check for problems in node at the other end of connection or in the connection line. Fix accordingly.
E4024	TCPIP)Communication Time Out. Dst.IP=XX.XX.XX.XX	Timeout occurred during execution of UDP_SENDTO/UDP_RECVFROM. Possible problem along Ethernet or at other end of connection.	Check for problems in node at the other end of connection or in the connection line. Fix accordingly.
E4025	TCPIP)Connection aborted.	Connection was aborted while executing UDP_SENDTO/UDP_RECVFROM. Possible problem along Ethernet or at other end of connection.	Check for problems in node at the other end of connection or in the connection line. Fix accordingly.
E4026	TCPIP)No Buffer Space.	Communication error occurred during execution of UDP_SENDTO/UDP_RECVFROM. TCP/IP system resource (queue, buffer) may be lacking. Or, possible problem along Ethernet or at other end of connection.	Stop some TCP/IP applications working at the same time. Reboot the controller.
E4027	TCPIP)Bad Socket.	Could not connect when executing UDP_SENDTO/UDP_RECVFROM. Port number or IP address may be wrong. Or, possible problem along Ethernet or at other end of connection.	Check port number or IP address and correct if necessary. Check for problems in node at the other end of connection or in the connection line. Fix accordingly.



## 6.0 DATA COMMUNICATION INSTRUCTIONS IN TCP

The following program instructions are provided to allow data transmission between robot controller and other computer systems via Ethernet TCP.

**TCP\_LISTEN** .....Program Instruction to start waiting for connection request (used by server-side to start communication service)

Creates socket, binds it to the specified port number and waits for connection request to the socket. This is used when robot controller is used as the server.

Check if connection request is received using the below **TCP\_ACCEPT** instruction.

**TCP\_ACCEPT** .....Program Instruction to check connection request (used by server-side to start communication service)

Checks if the specified port has received connection request for socket communication. This instruction is completed when the specified port receives connection request.

Use **TCP\_ACCEPT** after creating a socket using **TCP\_LISTEN**. After connection is confirmed by this command, use the **TCP\_SEND** and **TCP\_RECV** instructions below to send/ receive data.

**TCP\_CONNECT** .....Program Instruction to request connection (Used by client-side to start communication service)

Creates socket and binds it to the specified port number. Then, connection request for socket communication is send to the specified node, and connection is established. Use **TCP\_CONNECT** when robot is the client.

After connecting via **TCP\_CONNECT**, use **TCP\_SEND** and **TCP\_RECV** instructions to send/ receive data to/ from the server.

**TCP\_SEND** ..... Program Instruction to send data

Sends the specified character string data based on TCP protocol.

Use **TCP\_SEND** after connecting via **TCP\_ACCEPT** or **TCP\_CONNECT** instructions.

**TCP\_RECV** ..... Program Instruction to receive data

Receives data and stores them in the specified character string variable based on TCP protocol.

Use **TCP\_RECV** after connecting via **TCP\_ACCEPT** or **TCP\_CONNECT** instructions.

**TCP\_CLOSE** ..... Program Instruction to abort connection (Used to terminate communication service)

Closes the current socket connection. To close sockets created by TCP\_LISTEN instruction, use TCP\_END\_LISTEN instruction below.

**TCP\_END\_LISTEN** ..... Program Instruction to abort connection

Closes socket currently waiting for connection request. To close sockets that are already connected, use the above TCP\_CLOSE instruction.

#### Example

Keyword




Parameter



**TCP\_ACCEPT** variable for returned value, port number, timeout, ...

Parameters marked with    can be omitted.

Always enter a space between the keyword and first parameter.

 represents the Enter (Return) key in the examples.

---

Program Instruction

---

**TCP\_LISTEN   variable for returned value, port number**

---

**Function**

Program Instruction to start waiting for connection request (used by server-side to start communication service)

Creates socket, binds it to the specified port number, and waits for connection request to that socket. If communication error occurs during execution, the error code is stored in the returned value storage variable and program execution does not stop.

**Parameter**

Variable for returned value

- Specifies variable that stores the execution results.
- 0 is stored when execution completes normally.
- Error code is stored when communication error occurs. However, program execution does not halt at communication error.
- Error code is not stored when non-communication errors occur (e.g. parameter error). In this case, the program execution results in error and stops.
- See 6.1 for details on communication error codes.

Port number

- Specifies the port number that indicates the channel of the other end of the connection. The socket is bound to this port number.
- Acceptable range is 8192 – 65535, otherwise error occurs.

---

Program Instruction

---

**TCP\_ACCEPT** variable for returned value, port number, timeout,  
client IP address array variable

---

**Function**

Program Instruction to check connection request (used by server-side to start communication service)

Checks if the connection request for socket communication has been received by the specified port, and if it has, establishes connection. Connection is completed when this instruction terminates normally. If communication error occurs during execution, the error code is stored in the specified returned value storage variable and program execution does not stop.

**Parameter**

Variable for returned value

- Sets the variable that stores the execution results.
- 0 is stored when execution completes normally.
- Error code is stored when communication error occurs. However, program execution does not halt at communication error.
- Error code is not stored when non-communication errors occur (e.g. parameter error). In this case, the program execution results in error and stops.
- See 6.1 for details on communication error codes.

Port number

- Specifies the port number that indicates the channel of the other end of the connection (process). The socket is bound to this port number.
- Acceptable range is 8192 – 65535, otherwise error occurs.

Timeout

- Specifies the time (seconds) to wait before outputting timeout error when connection is not established. Default is 1 second. Acceptable range is 0 - 60, otherwise error occurs. If not specified, 1 is assumed.
- When a timeout error occurs, the error code is stored in the returned value storage variable.
- Timeout error does not stop AS program execution.

Client IP address array variable

- Specifies array variable that stores the IP address of the client (32 bits) when successfully connected.

- The IP address is stored in 8 bit increments to each element in the array variable in order starting from the start of the IP address.

**Example**

IP address is stored in the array variable elements ipa[0] through ipa[3], when the connection is established with computer “192.168.0.3”.

ipa[0] : 192

ipa[1] : 168

ipa[2] : 0

ipa[3] : 3

---

Program Instruction

---

**TCP\_CONNECT** variable for returned value, port number,  
server IP address array variable, timeout

---

**Function**

Program Instruction to request connection (Used by client-side to start communication service)

Creates socket and binds to the specified port number. Then, connection request is sent to the specified node, and connection is established. The node is determined by the IP address of the server. Waiting time to establish the connection is 60 seconds. (fixed)

If communication error occurs during execution, the error code is stored in the returned value storage variable and program execution does not stop.

**Parameter**

Variable for returned value

- Specifies variable that stores the execution results.
- 0 is stored when execution completes normally.
- Error code is stored when communication error occurs. However, program execution does not halt at communication error.
- Error code is not stored when non-communication error occurs (e.g. parameter error). In this case, the program execution results in error and stops.
- See 6.1 for details on communication error codes.

Port number

- Specifies the port number that indicates the channel of the other end of the connection (process). The socket is bound to this port number.
- Acceptable range is 8192 – 65535, otherwise error occurs.

Server IP address array variable

- Specifies array variable that stores the IP address of the server (32 bits) when successfully connected.
- The IP address is stored in 8 bit increments to each element in the array variable in order from the start of the IP address.

Timeout

- Specifies the time (seconds) to wait before outputting timeout error when connection is not established. Default is 1 second. Acceptable range is 0 - 60, otherwise error occurs. If not specified, 1 is assumed.

- When a timeout error occurs, the error code is stored in the returned value storage variable.
- Timeout error does not stop AS program execution.

### **Example**

IP address is stored in the array variable elements ipa[0] through ipa [3], when the connection is established with computer “192.168.0.3”.

ipa[0] : 192

ipa[1] : 168

ipa[2] : 0

ipa[3] : 3

---

Program Instruction

---

**TCP\_SEND** variable for returned value, socket ID number,  
character string variable array for sending data, number of elements,  
**timeout**

---

**Function**

Program Instruction to send data

Sends data based on TCP protocol. The data to be sent is specified as character string variable array.

If communication error occurs, the error code is stored in the returned value storage variable and program execution does not stop.

**Parameter**

Variable for returned value

- Sets variable that stores the execution results.
- 0 is stored when execution completes normally.
- Error code is stored when communication error occurs. However, program execution does not halt at communication error.
- Error code is not stored when non-communication errors occur (e.g. parameter error). In this case, the program execution results in error and stops.
- See 6.1 for details on communication error codes.

Socket ID number

- Specifies the socket ID number obtained by TCP\_ACCEPT or TCP\_CONNECT instructions.

Character string variable array for sending data

- Specifies the first element of the character string array that stores the data to be sent.
- Array elements are sent in order from first to last.
- Change the data to be sent into character string variable. Numeric data can be sent once transformed into character string format. Ensure that the encoding format is consistent with the recipient (big endian/little endian, etc.).
- The maximum amount of data that can be sent in execution of one instruction is 4096 bytes.

Number of elements

- Specifies the number of elements to send in the character string variable array.
- The specified number of elements are sent to the recipient in order of index number.



#### Timeout

- Specifies the time (seconds) to wait before outputting timeout error when connection is not established. Default is 1 second. Acceptable range is 0 - 60, otherwise error occurs. If not specified, 1 is assumed.
- When a timeout error occurs, the error code is stored in the returned value storage variable.
- Timeout error does not stop AS program execution.

---

Program Instruction

---

**TCP\_RECV** variable for returned value, socket ID number,  
character string variable array for received data, number of elements,  
timeout, max. number of characters

---

**Function**

Program instruction to receive data

Receives data based on TCP protocol, and stores it in the specified character string variable array.

If communication error occurs, the error code is stored in the returned value storage variable and program execution does not stop.

**Parameter**

Variable for returned value

- Specifies variable that stores the execution results.
- 0 is stored when execution completes normally.
- Error code is stored when communication error occurs. However, program execution does not halt at communication error.
- Error code is not stored when non-communication errors occur (e.g. parameter error). In this case, the program execution results in error and stops.
- See 6.1 for details on communication error codes.

Socket ID number

- Specifies the socket ID number obtained by TCP\_ACCEPT or TCP\_CONNECT instruction.

Received data storage character string variable array

- Specifies the first element of the character string variable array that stores the received data.
- Data is received as continuous character data of 1 byte each. When receiving numeric data (transformed into character string data), ensure that the encoding format is consistent with the sender (big endian/ little endian, data segmentation etc).
- The maximum amount of data that can be received in one instruction is 4096 bytes.

Number of elements

- Sets the variable to substitute the number of elements in the received character string variable array.
- Should be specified as a variable.

#### Timeout

- Specifies the time (seconds) to wait before outputting timeout error when connection is not established. Default is 1 second. Acceptable range is 0 - 60, otherwise error occurs. If not specified, 1 is assumed.
- When a timeout error occurs, the error code is stored in the returned value storage variable.
- Timeout error does not stop AS program execution.

#### Maximum number of characters

- Specifies the maximum number of bytes stored in an element of the character string variable array.
- Acceptable range is 1 – 255, otherwise error occurs.
- Default setting is 255 bytes.

#### Example

When character string “abcdefghijklmnopqrstuvwxyz” is received, this data is stored in array elements \$recv\_dt[0] through \$recv\_dt[3] of the received data storage character string variable array as shown below. The number of elements is specified as “rcv\_cnt”, and the maximum number of characters in the element is specified as 10.

```
$recv_dt[0]: "abcdefghij"  
$recv_dt[1]: "klmnopqrst"  
$recv_dt[2]: "vwxyz"  
rcv_cnt      : 3
```

---

Program Instruction

---

**TCP\_CLOSE    variable for returned value, socket ID number**

---

**Function**

Program Instruction to abort connection (Used to terminate communication service)

Cuts off connection for socket communication and closes socket. If communication error occurs, the error code is stored in the returned value storage variable, and program execution does not stop.

When TCP\_CLOSE execution is stopped by communication error etc., the connection may not be completely disconnected. In this case, execute TCP\_CLOSE instruction again. This will internally shutdown (force CLOSE) the connection.

**Parameter**

Variable for returned value

- Sets variable that stores the execution results.
- 0 is stored when execution completes normally.
- Error code is stored when communication error occurs. However, program execution does not halt at communication error.
- Error code is not stored when non-communication errors occur (e.g. parameter error). In this case, the program execution results in error and stops.
- See 6.1 for details on communication error codes.

Socket ID number

- Specifies the socket ID number obtained by TCP\_ACCEPT or TCP\_CONNECT instruction.

---

Program Instruction

---

**TCP\_END\_LISTEN    variable for returned value, port number**

---

**Function**

Program Instruction to abort connection

Ends waiting for connection request on the socket specified by TCP\_LISTEN and closes that socket. If communication error occurs, the error code is stored in the returned value storage variable, and program execution does not stop.

**Parameter**

Variable for returned value

- Specifies variable that stores the execution results.
- 0 is stored when execution completes normally.
- Error code is stored when communication error occurs. However, program execution does not halt at communication error.
- Error code is not stored when non-communication errors occur (e.g. parameter error). In this case, the program execution results in error and stops.
- See 6.1 for details on communication error codes.

Port number

- Specifies the socket currently waiting for connection request (socket specified in TCP\_LISTEN instruction).

---

Program Instruction

---

<b>TCP_STATUS</b>	<b>Returned value variable, Port number array variable, Socket ID array variable, Error code array variable, Sub error code array variable, IP address character array variable</b>
-------------------	---

---

**Function**

Stores to the specified variables, the status of the sockets used in the TCP communication instructions. Data are stored in order of sockets ID No., as controlled within the robot controller. The data with the same array number in each parameter are data from the same socket. If there is an error in the parameter, the program execution stops.

**Parameter**

Returned value variable

- Stores the number of sockets used.
- The data is not stored when there is an error in the parameter.

Port number array variable

- Stores the port number of the sockets used.
- Only stores data specified by TCP\_LISTEN, TCP\_CONNECT instructions. The data for the socket specified by TCP\_ACCEPT instruction is stored as 0.

Socket ID array variable

- Stores the socket ID of the socket in use.

Error code array variable

- Stores the error codes for the communication errors that occurred recently.
- These may include internal TCP/IP package error codes not explained in 6.10.

Sub error code array variable

- Stores the description codes for the error codes.
- SUB error codes for TCP/IP package

IP address character array variable

- Stores the IP address of the connected device in character string.  
Example: 192.168.0.1
- Only stores data specified by TCP\_ACCEPT, TCP\_CONNECT instructions.

The data for the socket specified by TCP\_LISTEN instruction is stored as “0.0.0.0”.

Example of stored data

•Status of the socket

Port number	Socket ID	Error code	Description code	IP Address
8192	1	0	0	0;Socket specified by TCP_LISTEN
0	2	0	0	192.168.0.2; Socket specified by TCP_ACCEPT
49152	3	0	0	192.168.0.5; Socket specified by TCP_CONNECT
49153	4	-21670	90	192.168.0.4; Example of a socket in error status

•TCP\_STATUS instruction

TCP\_STATUS sock\_cnt, port\_no[0], sock\_id[0], err\_cd[0], sub\_cd[0], \$ip\_adrs[0]

•Contents of stored data

```

sock_cnt      : 4
port_no[0]    : 8192
port_no[1]    : 0
port_no[2]    : 49152
port_no[3]    : 49153
sock_id[0]    : 1
sock_id[1]    : 2
sock_id[2]    : 3
sock_id[3]    : 4
err_cd[0]     : 0
err_cd[1]     : 0
err_cd[2]     : 0
err_cd[3]     : -21670
sub_cd[0]     : 0
sub_cd[1]     : 0
sub_cd[2]     : 0
sub_cd[3]     : 90
$ip_adrs[0]   : 0.0.0.0
$ip_adrs[1]   : 192.168.0.2
$ip_adrs[2]   : 192.168.0.5
$ip_adrs[3]   : 192.168.0.4

```

## 6.1 COMMUNICATION ERRORS IN SOCKET COMMUNICATION

The following errors may be detected during execution of socket communication instructions, but will not stop the execution of the program\*. All errors are recorded in the error log.

**NOTE\*** Program execution halts if error occurs when the controller is in the middle of processing TCP\_ACCEPT.

Code	Error message	Description	Countermeasure
E4019	TCPIP) Socket open failure. CODE=XX Dst.IP=XX.XX.XX.X X	Failed to open socket when executing TCP_LISTEN/TCP_CONNECT. Too many sockets may have been opened. Or, parameter error e.g. wrong IP address.	Do not execute too many of this instruction. Fix incorrect parameter. CODE shows the SUB error code of TCP/IP package.
E4020	TCPIP) Socket close failure. CODE=XX Dst.IP=XX.XX.XX.X X	Failed to close socket when executing TCP_CLOSE/TCP_END_LISTEN. Possible problem along Ethernet or at the other end of connection.	Check for problems in the node at the other end of connection or in the connection line. Fix accordingly. CODE shows the SUB error code of TCP/IP package.
E4021	TCPIP)Communication Error. CODE=XX Dst.IP=XX.XX.XX.X X	Communication error during execution of TCP_SEND/TCP_RECV. Possible problem along Ethernet or at the other end of connection.	Check for problems in the node at the other end of connection or in the connection line. Fix accordingly. CODE shows the SUB error code of TCP/IP package.
E4022	TCPIP)Too Long Message.	Tried to send data longer than the limit while executing TCP_SEND.	Fix data so the size is within limit size and re-try the program.
E4023	TCPIP)Cannot reach the Host	Could not connect with the host when executing TCP_CONNECT. There is no host with this IP address on the network. Port number or IP address may be wrong. Possible problem along Ethernet or at the other end of connection.	Check if port number or IP address is correct and correct if necessary. Check for problems in the node at the other end of connection or in the connection line. Fix accordingly.
E4024	TCPIP)Communication Time Out. Dst.IP=XX.XX.XX.X X	Timeout occurred during execution of instructions based on TCP. Possible problem along Ethernet or at the other end of connection.	Check for problems in the node at the other end of connection or in the connection line. Fix accordingly.
E4025	TCPIP)Connection aborted.	Connection was aborted while executing TCP_CONNECT/TCP_SEND/TCP_RECV/TCP_ACCEPT. Possible problem along Ethernet or at the other end of connection.	Check for problems in the node at the other end of connection or in the connection line. Fix accordingly.
E4026	TCPIP)No Buffer Space.	Communication error occurred during execution of TCP_SEND/TCP_RECV. TCP/IP system source (queue, buffer) may be lacking. Or, possible problem along Ethernet or at the other end of connection.	Abort some TCP/IP applications working at the same time. Reboot the controller.
E4027	TCPIP)Bad Socket.	Could not connect when executing TCP_CONNECT. Port number or IP address may be wrong. Possible problem along Ethernet or at the other end of connection.	Check port number or IP address and correct if necessary. Check for problems in the node at the other end of connection or in the connection line. Fix accordingly.
E4056	TCP)This port is not in LISTEN (SOCK).	Failed to open socket when executing TCP_ACCEPT. Or failed to close socket when executing TCP_END_LISTEN. May have specified socket not created by TCP_LISTEN.	Check if port number and correct if necessary.



Code	Error message	Description	Countermeasure
E4057	TCP)Illegal Socket ID	Communication error occurred during execution of TCP_SEND/TCP_RECV. Or, failed to close socket when executing TCP_CLOSE/TCP_END_LISTEN. May have specified socket ID number or port number of socket not created. Or, the order of program execution is wrong.	Check socket ID number or port number and correct if necessary. Check order of program execution, and correct if necessary.
E4055	TCP)Cannot create a socket.	Failed to open socket when executing TCP_LISTEN/TCP_CONNECT. Too many sockets are opened. Or, the same port is used too many times.	Avoid executing too many instructions at a time. Correct parameter. Close port previously used via TCP_CLOSE/TCP_END_LISTEN before using it again.

## 6.2 SAMPLE PROGRAM USING SOCKET COMMUNICATION INSTRUCTIONS

### 6.2.1 WHEN ROBOT IS THE SERVER-SIDE

```
.PROGRAM main()           ;Communication main program
port = 49152
max_length = 255
tout_open = 60
tout_rec = 60
CALL open_socket           ;Connecting communication
IF sock_id < 0 THEN
    GOTO exit_end
END
text_id = 0
tout = 60
eret = 0
rret = 0
$sddata[1] = "001"
CALL send(eret,$sddata[1]) ;Instructing processing 1
IF eret < 0 THEN
    PRINT "CODE 001 ERROR END code=",eret
    GOTO exit
END
CALL recv                   ;Receiving the result of processing 1
IF rret < 0 THEN
    PRINT "CODE 001 RECV ERROR END code=",rret
    GOTO exit
END
eret = 0
$sddata[1] = "002"
CALL send(eret,$sddata[1]) ;Instructing processing 2
IF eret < 0 THEN
    PRINT "CODE 002 ERROR END code=",eret
    GOTO exit
END
CALL recv                   ;Receiving the result of processing 2
IF rret < 0 THEN
    PRINT "CODE 002 RECV ERROR END code=",rret
    GOTO exit
END
exit:
CALL close_socket          ;Closing communication
exit_end:
.END
```

```
.PROGRAM open_socket() ;Starting communication
    er_count = 0
listen:
    TCP_LISTEN retl,port
    IF retl<0 THEN
        IF er_count >= 5 THEN
            PRINT "Connection with PC is failed (LISTEN). Program is stopped."
            sock_id = -1
            goto exit
        ELSE
            er_count = er_count+1
            PRINT "TCP_LISTEN error=",retl," error count=",er_count
            GOTO listen
        END
    ELSE
        PRINT "TCP_LISTEN OK ",retl
    END
    er_count = 0
accept:
    TCP_ACCEPT sock_id,port,tout_open,ip[1]
    IF sock_id<0 THEN
        IF er_count >= 5 THEN
            PRINT "Connection with PC is failed (ACCEPT). Program is stopped."
            TCP_END_LISTEN ret,port
            sock_id = -1
        ELSE
            er_count = er_count+1
            PRINT "TCP_ACCEPT error id=",sock_id," error count=",er_count
            GOTO accept
        END
    ELSE
        PRINT "TCP_ACCEPT OK id=",sock_id
    END
exit:
.END
.PROGRAM send(.ret,.$data) ;Communication Sending data
    $send_buf[1] = .$data
    buf_n = 1
    .ret = 1
    TCP_SEND sret,sock_id,$send_buf[1],buf_n,tout
    IF sret < 0 THEN
        .ret = -1
        PRINT "TCP_SEND error in SEND",sret
    ELSE
        PRINT "TCP_SEND OK in SEND",sret
    END
.END
.PROGRAM recv() ;Communication Receiving data
    .num=0
    TCP_RECV rret,sock_id,$recv_buf[1],.num,tout_rec,max_length
    IF rret < 0 THEN
        PRINT "TCP_RECV error in RECV",rret
        $recv_buf[1]="000"
    ELSE
        IF .num > 0 THEN
            PRINT "TCP_RECV OK in RECV",rret
        ELSE
            $recv_buf[1]="000"
        END
    END
    END
.END
```

```
.PROGRAM close_socket() ;Closing communication
TCP_CLOSE ret,sock_id      ;Normal socket closure
IF ret < 0 THEN
    PRINT "TCP_CLOSE error ERROE=(",ret," ) ",$ERROR(ret)
    TCP_CLOSE ret1,sock_id    ;Forced closure of socket (shutdown)
    IF ret1 < 0 THEN
        PRINT "TCP_CLOSE error id=",sock_id
    END
ELSE
    PRINT "TCP_CLOSE OK id=",sock_id
END
TCP_END_LISTEN ret,port
IF ret < 0 THEN
    PRINT "TCP_CLOSE error id=",sock_id
ELSE
    PRINT "TCP_CLOSE OK id=",sock_id
END
.END
```

## 6.2.2 WHEN ROBOT IS THE CLIENT-SIDE

```
.PROGRAM main()           ;Communication main program
port = 49152
ip[1] = 192
ip[2] = 168
ip[3] = 0
ip[4] = 2
max_length = 255
tout_open = 60
tout_rec = 60
eret = 0
ret = 0
CALL open_socket           ;Connecting communication
IF sock_id < 0 THEN
    GOTO exit_end
END
text_id = 0
tout = 60
WHILE (1) DO
    ret = 0
    CALL recv               ;Receiving instruction data from PC
    IF ret < 0 THEN
        PRINT "Communication error code=",ret
        GOTO exit
    ELSE
        CASE ret OF
            VALUE 0:
                text_id = val($mid($recv_buf[1],1,3)) ;Taking in instruction number
            ANY :
                END
        END
    END
    IF text_id > 0 THEN
        CALL com_test       ;Processing of each instruction
        IF eret < 0 THEN
            goto exit
        END
    END
END
END
exit:
CALL close_socket          ;Disconnecting communication
exit_end:
.END
.PROGRAM com_test()        ;Processing per separate instruction data
CASE text_id OF
    VALUE 0: ;No instructions
        GOTO break_exit
    VALUE 1: ;Instructing processing 1
        ;Separate processing
    VALUE 2: ;Instructing processing 2
        ;Separate processing
    VALUE 3: ;Instructing processing 3
        ;Separate processing
    any :
        GOTO break_exit
END
$send_buf[1] = $recv_buf[1] + " OK"
CALL send(eret,$send_buf[1]) ;Sending end processing
break_exit:
.END
```

```
.PROGRAM open_socket() ;Starting communication
.er_count =0
connect:
TCP_CONNECT sock_id,port,ip[1],tout_open
IF sock_id<0 THEN
    IF .er_count >= 5 THEN
        PRINT "Connection with PC failed. Program is stopped."
    ELSE
        .er_count = .er_count+1
        PRINT "TCP_CONNECT error id=",sock_id," error count=",.er_count
        GOTO connect
    END
ELSE
    PRINT "TCP_CONNECT OK id=",sock_id
END
.END

.PROGRAM recv() ;Communication Receiving data
.num=0
TCP_RECV ret,sock_id,$recv_buf[1],.num,tout_rec,max_length
IF ret < 0 THEN
    PRINT "TCP_RECV error in RECV",ret
    $recv_buf[1]="000"
ELSE
    IF .num > 0 THEN
        PRINT "TCP_RECV OK in RECV",ret
    ELSE
        $recv_buf[1]="000"
    END
END
.END

.PROGRAM send(.ret,$data) ;Communication Sending data
$send_buf[1] = .data
buf_n = 1
.ret = 1
send_rt:
TCP_SEND sret,sock_id,$send_buf[1],buf_n,tout
IF sret < 0 THEN
    .ret = -1
    PRINT "TCP_SEND error in SEND",sret
ELSE
    PRINT "TCP_SEND OK in SEND",sret
END
.END

.PROGRAM close_socket() ;Closing communication
TCP_CLOSE ret,sock_id ;Normal socket closure
IF ret < 0 THEN
    PRINT "TCP_CLOSE error ERROE=(",ret," ) “,$ERROR(ret)
    TCP_CLOSE ret1,sock_id ;Forced closure of socket (shutdown)
    IF ret1 < 0 THEN
        PRINT "TCP_CLOSE error id=",sock_id
    END
ELSE
    PRINT "TCP_CLOSE OK id=",sock_id
END
.END
```

## 7.0 SAVE/LOAD FILES USING FTP CLIENT FUNCTION

The functions below are provided to enable saving/loading of robot data using the FTP Client Function via Ethernet connection between the robot controller and an external computer system.

(1) Aux.0201: Save

Saves internal robot data to an external computer system.

(2) Aux.0202: Load

Loads internal robot data from an external computer system.

(3) Aux.0203: File list

Displays file information on the external computer system.

(4) Aux.0205: Delete file

Deletes files on the external computer system.

(5) Aux.0206: Rename file

Changes the name of the file in the external computer.

(6) Aux.0210: Autosave configuration

Sets/Displays the information of auto-save to the external computer.

(7) Aux.0812: Network setting

Sets/Displays the information of the local controller.

(8) Aux.0815: FTP server setting

Sets/Displays the information of connection to the external computer.

“2: FTP server” from a device list can not be selected in the following auxiliary functions.

Aux.0204 : Initialize

Aux.0207 : Copy file

Aux.0208 : Set verifying function

Aux.0209 : Set default device

### [ NOTE ]

< Selecting FTP server >

Enable server function in Aux.0815: FTP sever setting; otherwise,  
device list 2: FTP server can not be selected.

## 7.1 SOFTWARE SETTINGS

### 7.1.1 SETTINGS ON THE ROBOT CONTROLLER

Sets the items below for the robot controller by “Aux. 0812 Network setting”. The data below is saved or loaded as robot data.

Aux. 0812. Network Setting

IP Address: 192.168.0.2

Host Name:

Subnet Mask: 255.255.255.0

Gateway: 0.0.0.0

Primary DNS Server: 0.0.0.0

Secondary DNS Server: 0.0.0.0

Domain Name:

MAC Address: 00:09:0F:03:01:12

Undo

Sets IP Address

Input range : [0 - 255]

IP address	Input XXX in decimal. Initial setting: 192.168.0.2
Subnet mask	Input XXX in decimal. Initial setting: 255.255.255.0
Gateway IP address	Input XXX in decimal. Initial setting: 0.0.0.0

### 7.1.2 SETTINGS ON REMOTE HOST

Sets the items below for the FTP server by “Aux. 0815 FTP server setting”. The data below is saved or loaded as auxiliary data.

Aux. 0815. FTP Server Setting

Server Function: ☐ Enable ☒ Disable

IP Address: 192.168.0.1

User Name:

Password:

Path:

Account:

Transfer Mode: ☒ ASCII ☐ BINARY

Character Code: ☒ SJIS ☐ EUC

Command to get file name: ☒ LIST ☐ NLST

Undo



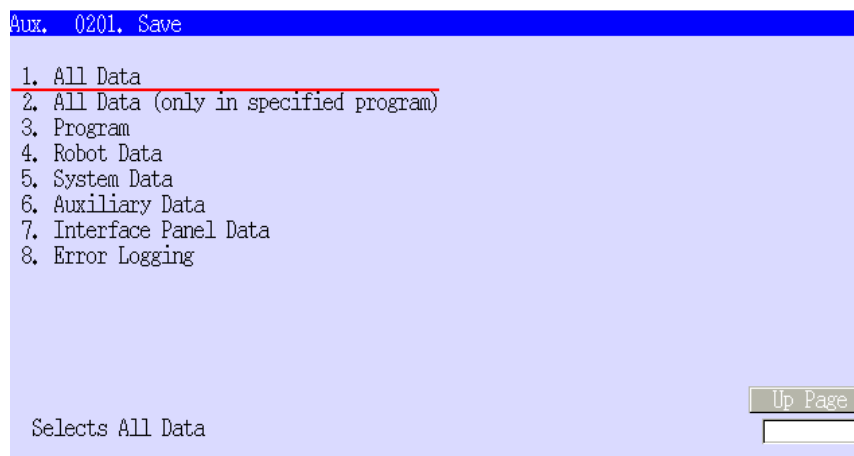
Server Function	Select Enable/Disable server function. Initial setting: Disable
IP address	Input XXX in decimal. Initial setting: 192.168.0.1
User name	Input characters starting with alphabet. (max. 24 characters.) User name can not be omitted.
Password	Input characters starting with alphabet. (max. 24 characters.) The data is displayed with “*” Password can not be omitted.
Path	Input on 3 lines (max. 24 characters/line) (max.72 characters in total) Backspace(Delete) space at the end on each line Path can not be omitted. i.e) When the host is Windows, C:\ftp\ or C:/ftp/ When the host is UNIX, /home/ftp/
Account	Input characters starting with alphabet. (max. 24 characters.) The data is displayed with “*”. Account can not be omitted.
Transfer mode	Select ASCII or BINARY. Default: ASCII
Character code	Select SJIS(shift JIS) code or EUC code. Default: SJIS
Command to get file name	Select LIST or NLST.

## 7.2 FTP AUXILIARY FUNCTIONS

### 7.2.1 AUX. 0201 SAVE

Through the FTP server, data and programs in memory are stored into a file in the directory on the path set via “Aux. 0815 FTP server setting”.

In the screen below, select the type of data to be saved by cursor or the function number.



Data to be saved (File extensions are displayed in [ ] when file are saved.)

1. All data [.as]
2. All data (only in specified program) [.as]
3. Program [.pg]
4. Robot data [.rb]
5. System data [.sy]
6. Auxiliary data [.au]
7. Interface panel data [.if]
8. Error logging [.el]

Next, enter the file name in the screen below.

Aux. 0201. Save 1. All Data

File Name

Device  Device List  
0: PC Card  
2: FTP Server

File Input

Input range : 12 characters

File name	Enter characters starting with alphabet. (max.: 8) File extensions are automatically added to the data to be saved.
Device	Selects device. 0: PC card or 2: FTP server Initial setting: 0: PC card

[ NOTE ]

< Limit on the number of characters that can be used for file names in the FTP server >  
The maximum number of characters is 16 when saving a file name. However, if file names displayed in the Windows FTP server directory exceed 8 characters, the remaining characters may be truncated and appear as (~) . Name files with 8 characters or less when using this kind of server.

## 7.2.2 AUX. 0202 LOAD

Loads data stored in file units in the directory on the path via “Aux. 0815 FTP server setting” to memory.

Select loading method, all data or specified data only, in the screen below by function number or cursor.

Aux. 0202. Load

1. All Data  
2. Specified Data

Selects All Data

Up Page

Next, file list in the directory according to the path set by “Aux. 0815 FTP server setting” is displayed below. Select the file to be read by cursor and press ENTER key.

Select file.						1/2
Input char		Next Page				
-rw-r--r--	1 dcon	jcon	4096	Jan 20	2002	FILE1.AS
-rw-r--r--	1 dcon	jcon	4096	Jan 20	2002	FILE2.AS
-rw-r--r--	1 dcon	jcon	4096	Jan 20	2002	FILE3.AS
-rw-r--r--	1 dcon	jcon	4096	Jan 20	2002	FILE4.AS
-rw-r--r--	1 dcon	jcon	4096	Jan 20	2002	FILE5.AS
-rw-r--r--	1 dcon	jcon	4096	Jan 20	2002	FILE6.AS
-rw-r--r--	1 dcon	jcon	4096	Jan 20	2002	FILE7.AS
-rw-r--r--	1 dcon	jcon	4096	Jan 20	2002	FILE8.AS
-rw-r--r--	1 dcon	jcon	4096	Jan 20	2002	FILE9.AS
-rw-r--r--	1 dcon	jcon	4096	Jan 20	2002	FILE10.AS

Similar screen appears when pressing: “2. Specified data”. (Only changing the display of the first line.)

### 7.2.3 AUX. 0203 FILE CONFIGURATION

Displays a list of the file names stored in the directory, path set by “Aux. 0815 FTP server setting”.

Aux. 0203, File List

Device

2

Device List

0: PC Card

2: FTP Server

Undo

Input range : [0 - 2]

Aux. 0203, File List

-rw-r--r--	1 dcon	jcon	4096 Jan 20	2002 FILE1.AS
-rw-r--r--	1 dcon	jcon	4096 Jan 20	2002 FILE2.AS
-rw-r--r--	1 dcon	jcon	4096 Jan 20	2002 FILE3.AS
-rw-r--r--	1 dcon	jcon	4096 Jan 20	2002 FILE4.AS
-rw-r--r--	1 dcon	jcon	4096 Jan 20	2002 FILE5.AS
-rw-r--r--	1 dcon	jcon	4096 Jan 20	2002 FILE6.AS
-rw-r--r--	1 dcon	jcon	4096 Jan 20	2002 FILE7.AS
-rw-r--r--	1 dcon	jcon	4096 Jan 20	2002 FILE8.AS
-rw-r--r--	1 dcon	jcon	4096 Jan 20	2002 FILE9.AS
-rw-r--r--	1 dcon	jcon	4096 Jan 20	2002 FILE10.AS
-rw-r--r--	1 dcon	jcon	4096 Jan 20	2002 FILE11.AS
-rw-r--r--	1 dcon	jcon	4096 Jan 20	2002 FILE12.AS

Next Page

## 7.2.4 AUX. 0205 DELETE FILE

Deletes the file selected by cursor from the directory, path set by “Aux. 0815 FTP server setting”.

Aux. 0205, Delete file

File Name

Device  Device List  
0: PC Card  
2: FTP Server

Input range : 12 characters

Select file.					1/2
Input char	Next Page				
-rw-r--r--	1 dcon	jcon	4096 Jan 20	2002 FILE1.AS	
-rw-r--r--	1 dcon	jcon	4096 Jan 20	2002 FILE2.AS	
-rw-r--r--	1 dcon	jcon	4096 Jan 20	2002 FILE3.AS	
-rw-r--r--	1 dcon	jcon	4096 Jan 20	2002 FILE4.AS	
-rw-r--r--	1 dcon	jcon	4096 Jan 20	2002 FILE5.AS	
-rw-r--r--	1 dcon	jcon	4096 Jan 20	2002 FILE6.AS	
-rw-r--r--	1 dcon	jcon	4096 Jan 20	2002 FILE7.AS	
-rw-r--r--	1 dcon	jcon	4096 Jan 20	2002 FILE8.AS	
-rw-r--r--	1 dcon	jcon	4096 Jan 20	2002 FILE9.AS	
-rw-r--r--	1 dcon	jcon	4096 Jan 20	2002 FILE10.AS	

### 7.2.5 AUX. 0206 RENAME FILE

Changes the file name defined by Source file name to the file name defined by Dest. File name from among the files stored in the directory (path set by “Aux. 0815 FTP server setting”).

Source file name	Enter the file name including extension. Input characters starting with alphabet. (max.: 8)
Dest. file name	Enter the file name including extension. Input characters starting with alphabet. (max.: 8)

### 7.2.6 AUX. 0210 AUTOSAVE CONFIGURATION

Auto-saving is a function that automatically saves data from the robot controller to the remote host daily at a specified time and by a specified procedure.

Autosaving is executed according to specified procedure. Data below is saved/ loaded as auxiliary data.

Aux. 0210. Autosave Configuration 1. Save Data1

Autosave	<input type="radio"/> Enable	<input checked="" type="radio"/> Disable
Day Of Week	<input type="radio"/> Mon	<input type="radio"/> Tue <input type="radio"/> Wed <input type="radio"/> Thu <input type="radio"/> Fri <input type="radio"/> Sat <input type="radio"/> Sun
	<input type="radio"/> Everyday	<input checked="" type="radio"/> Only Dedicated Signals
Time	00:00	
Device	0	Device List
Save File Name	FILE0.as	0: PC Card
Save Data	All Data	
Specified Program		2: FTP Server

Undo

Autosave	Set execution of Auto saving function: ENABLE/DISABLE. Initial setting: DISABLE.
Day of week	Select the day to execute autosaving.
Time	Input the time to start autosaving. Hour: 0 to 23 Min.: 0 to 59
Device	Select device. 0: PC card 2: FTP server Initial setting: 0: PC card
Save file name	Input characters starting with alphabet. (max.: 8) Initial setting: FILE0.as
Save data	Select data to be saved. 1. All data 2. All data (only in specified program) 3. Program 4. Robot data 5. System data 6. Auxiliary data 7. Interface panel data 8. Error logging
Specified program	Specify program to be saved. This function is available when selecting the following in save data screen. 2. All data (only in specified program) 3. Program

Data is automatically saved to the directory specified by the path in “Aux. 0815 FTP server setting” under the conditions set in “Aux. 0210 Autosave configuration”. Processing conditions are indicated on next page.



(1) Start autosaving everyday at specified time.

Start autosaving process everyday at the specified time.

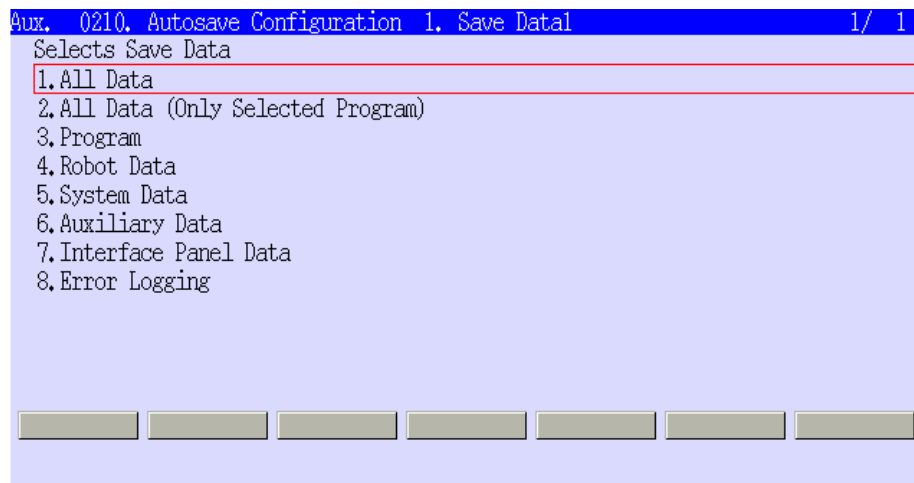
(2) Start autosaving everyday on the specified day.

Start autosaving process on only specified day at set time.

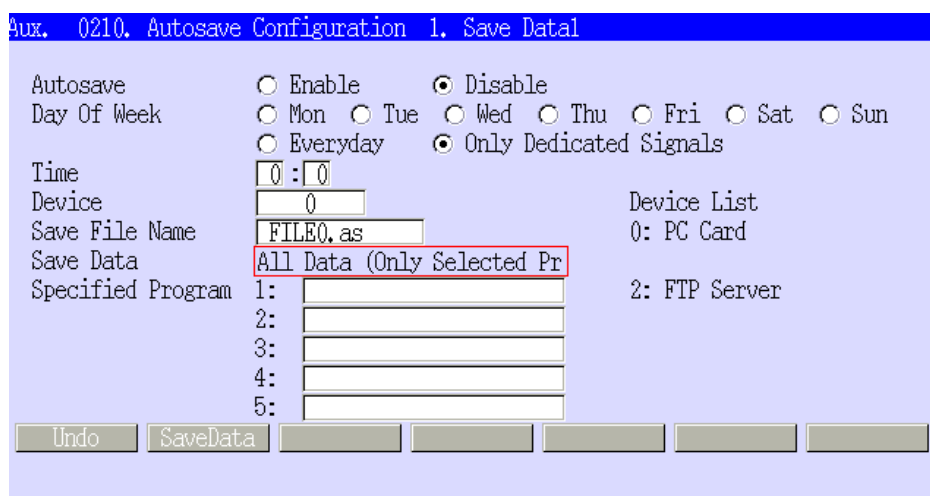
(3) Start when dedicated signal is input

Starts autosaving process at the specified time, when dedicated signal set in condition 1 to 3 in “Aux. 0602 Dedicated input signal” is input.

[Selects Save Data] screen appears by selecting items in save data.



2. All data (only in specified program) or 3. Program is selected, boxes to choose programs are displayed on the right side of the specified program.



## 7.3 COMMUNICATION ERROR CODES WHICH OCCUR WITH THE FTP CLIENT FUNCTION

Even if errors relating to the save/load function by the FTP function are detected, only a warning is displayed without them being an error. However, the communication errors are recorded in the error log. Errors are detected in the following cases.:

- Information for the LAN setting is left unset during start-up.
- There is a malfunction in the automatic backup.
- There is an error on manually saving/loading.

Error Code	Error Message	Explanation	Counter-measures
P2008	Device is not ready.	Designated directory does not exist. (Occurs only when auxiliary functions relating to FTP are attempted.)	After verifying the directory path to be referenced, check if Aux.0815 FTP server setting is set correctly, and reset as necessary.
P2076	SAVE/LOAD in progress.	Because save/load is already being executed by another operation, save/load could not be done.	After saving/loading has been completed by the other server, try to re-execute.
E4009	Communication time out error	Execution is terminated because there is no reply from the FTP server. (Occurs only when auxiliary functions relating to FTP are attempted.)	Verify that the communication cable is connected properly and re-execute.
E4028	FTP)Data receive error(code=%d)	Failure to receive the teach data via LAN→Memory (Manually LOAD) or failure on receiving LAN file directory list information. A misconnection of the communication cable is the probable cause.	Verify that the communication cable is connected properly and re-execute.
E4029	FTP)Data send error(code=%d)	Failure to transmit the teach data via Memory→LAN (Manually SAVE). A misconnection of the communication cable is the probable cause.	Verify that the communication cable is connected properly and re-execute.
E4031	FTP)Failed to disconnect with FTP server(code=%d)	Failure to terminate when completing communication with the FTP server. The following are considered as causes: 1. Communication cable is not connected correctly. 2. The remote host FTP server is OFF or is in an abnormal condition.	Re-execute after carrying out these measures: 1. Verify that the communication cables connected properly. 2. Reboot the remote host FTP server.
E4033	FTP)Failed to connect with FTP Server (code=%d)	The connection failed when communication with the FTP server was started. 1. code=26 : the designated address does not exist on the network 2. code=64 : communication cable is not connected properly 3. code=530 : failed to login to FTP server	Re-execute after carrying out these measures: 1. Verify the IP address of the remote host. Correct the address setting via Aux.0815 FTP server setting. 2. Verify that the communication cable is connected properly. 3. Confirm the user name and password on the remote host. Correct the user name and password for the LAN remote host save/load settings via Aux.0815.
E4037	FTP)Failed AUTO-SAVING	Some kind of error occurred during auto save. The error that occurred is displayed immediately after this message and is recorded in the error log.	Carry out counter measures on the error displayed immediately after the message: FTP)AUTO-SAVING failed.

## 8.0 ERROR MESSAGE

Code	Error Message	Description	Countermeasure
E4013	TELNET) SEND error. CODE=XX	Error occurred when sending data by TCP/IP. Possible problem along communication line, or connection was aborted or closed.	Check for problems in node at end of connection or in the connection line. Fix accordingly.
E4014	TELNET) RECV error. CODE=XX	Error occurred when receiving data by TCP/IP. Possible problem along communication line, or connection was aborted or closed.	Check for problems in node at end of connection or in the connection line. Fix accordingly.
E4015	TELNET) IAC receive error. CODE=XX	Error occurred when receiving data by TCP/IP. Possible problem along communication line, or connection was aborted or closed.	Check for problems in node at end of connection or in the connection line. Fix accordingly. Check the KCwinTCP in the PC at other end of connection.
E4016	TELNET) Close failure. CODE=XX	Error occurred in LISTEN/ACCEPT function in TCP/IP, and attempt to close the socket failed. TCP/IP system resources (queue, buffer) may be lacking. Or, possible problem along communication line, or in node at other end of connection.	Stop some TCP/IP applications working at the same time. Reboot the controller. CODE shows the SUB error code of TCP/IP package.
E4017	TELNET) Main socket close failure. CODE=XX	Error occurred in LISTEN/ACCEPT function in TCP/IP, and tried to close the main socket but could not. TCP/IP system source (queue, buffer) may be lacking. Or, possible problem along communication line, or in node at other end of connection.	Stop some TCP/IP applications working at the same time. Reboot the controller. CODE shows the SUB error code of TCP/IP package.
E4018	TELNET) System error. CODE=XX	The parameter in IPADDRESS command is wrong.	Set the parameter in IPADDRESS correctly. CODE shows the SUB error code of TCP/IP package.
E4019	TCPIP) Socket open failure. CODE=XX Dst.IP=XX.XX.XX.XX	Failed to open socket when executing UDP_SENDTO/UDP_RECVFROM/TCP_LISTEN/TCP_CONNECT. Too many sockets may have been opened. Or, parameter error e.g. wrong IP address.	Do not execute too many of this instruction. Fix incorrect parameter. CODE shows the SUB error code of TCP/IP package.
E4020	TCPIP) Socket close failure. CODE=XX Dst.IP=XX.XX.XX.XX	Failed to close socket when executing UDP_SENDTO/UDP_RECVFROM/TCP_CLOSE/TCP_END_LISTEN. There may be problem with the Ethernet or at the other end of the connection.	Check for problems in node at end of connection or in the connection line. Fix accordingly. CODE shows the SUB error code of TCP/IP package.
E4021	TCPIP)Communication Error. CODE=XX Dst.IP=XX.XX.XX.XX	Communication error during execution of UDP_SENDTO/UDP_RECVFROM/TCP_SEND/TCP_RECV. Possible problem along communication line, or connection was aborted or closed.	Check for problems in node at end of connection or in the connection line. Fix accordingly. CODE shows the SUB error code of TCP/IP package.
E4022	TCPIP)Too Long Message.	Tried to send data longer than the limit while executing UDP_SENDTO/UDP_RECVFROM/TCP_SEND.	Fix data so the size is within limit size and re-try the program.

Code	Error Message	Description	Countermeasure
E4023	TCPIP)Cannot reach the Host.	Could not connect with the host when executing UDP_SENDTO/UDP_RECVFROM/TCP_CONNECT. There is no host with this IP address on the network. Port number or IP address may be wrong. Possible problem along communication line, or connection was aborted or closed.	Check port number or IP address and correct if necessary. Check for problems in node at end of connection or in the connection line. Fix accordingly.
E4024	TCPIP)Communication Time Out. Dst.IP=XX.XX.XX.XX	Timeout occurred during execution of instructions based on TCP. Possible problem along communication line, or connection was aborted or closed.	Check for problems in node at end of connection or in the connection line. Fix accordingly.
E4025	TCPIP)Connection aborted.	Connection was aborted while executing UDP_SENDTO/UDP_RECVFROM/TCP_CONNECT/TCP_SEND/TCP_RECV/TCP_ACCEPT. Possible problem along communication line, or connection was aborted or closed.	Check for problems in node at end of connection or in the connection line. Fix accordingly.
E4026	TCPIP)No Buffer Space.	Communication error occurred during execution of UDP_SENDTO/UDP_RECVFROM/TCP_SEND/TCP_RECV. TCP/IP system source (queue, buffer) may be lacking. Or, possible problem along communication line, or connection was aborted or closed.	Stop some TCP/IP applications working at the same time. Reboot the controller.
E4027	TCPIP)Bad Socket.	Could not connect when executing UDP_SENDTO/UDP_RECVFROM/TCP_CONNECT. Port number or IP address may be wrong. Possible problem along communication line, or connection was aborted or closed.	Check port number or IP address correct if necessary. Check for problems in node at end of connection or in the connection line. Fix accordingly.
E4056	TCP)This port is not in LISTEN (SOCK).	Failed to open socket when executing TCP_ACCEPT. Or failed to close socket when executing TCP_END_LISTEN. May have specified socket not created by TCP_LISTEN.	Check port number and correct if necessary.
E4057	TCP)Illegal Socket ID	Communication error occurred during execution of TCP_SEND/TCP_RECV. Or, failed to close socket when executing TCP_CLOSE/TCP_END_LISTEN. May have specified socket ID number or port number of socket not created. Or, the order of program execution is wrong.	Check socket ID number or port number and correct if necessary. Check order of program execution, and correct if necessary.
E4055	TCP)Cannot create a socket.	Failed to open socket when executing TCP_LISTEN/TCP_CONNECT. Too many sockets are opened. Or, the same port is used too many times.	Avoid executing too many instructions at a time. Correct parameter. Close port previously used using TCP_CLOSE/TCP_END_LISTEN before using it again.



MEMO

---

---

Kawasaki Robot Controller D Series  
TCP/IP Communication MANUAL

---

March 2003 : 1st Edition  
January 2007 : 2nd Edition  
December 2008 : 3rd Edition

Publication : KAWASAKI HEAVY INDUSTRIES, LTD.

90210-1190DEC

---

---

All rights reserved. Copyright © 2008 KAWASAKI HEAVY INDUSTRIES, LTD.