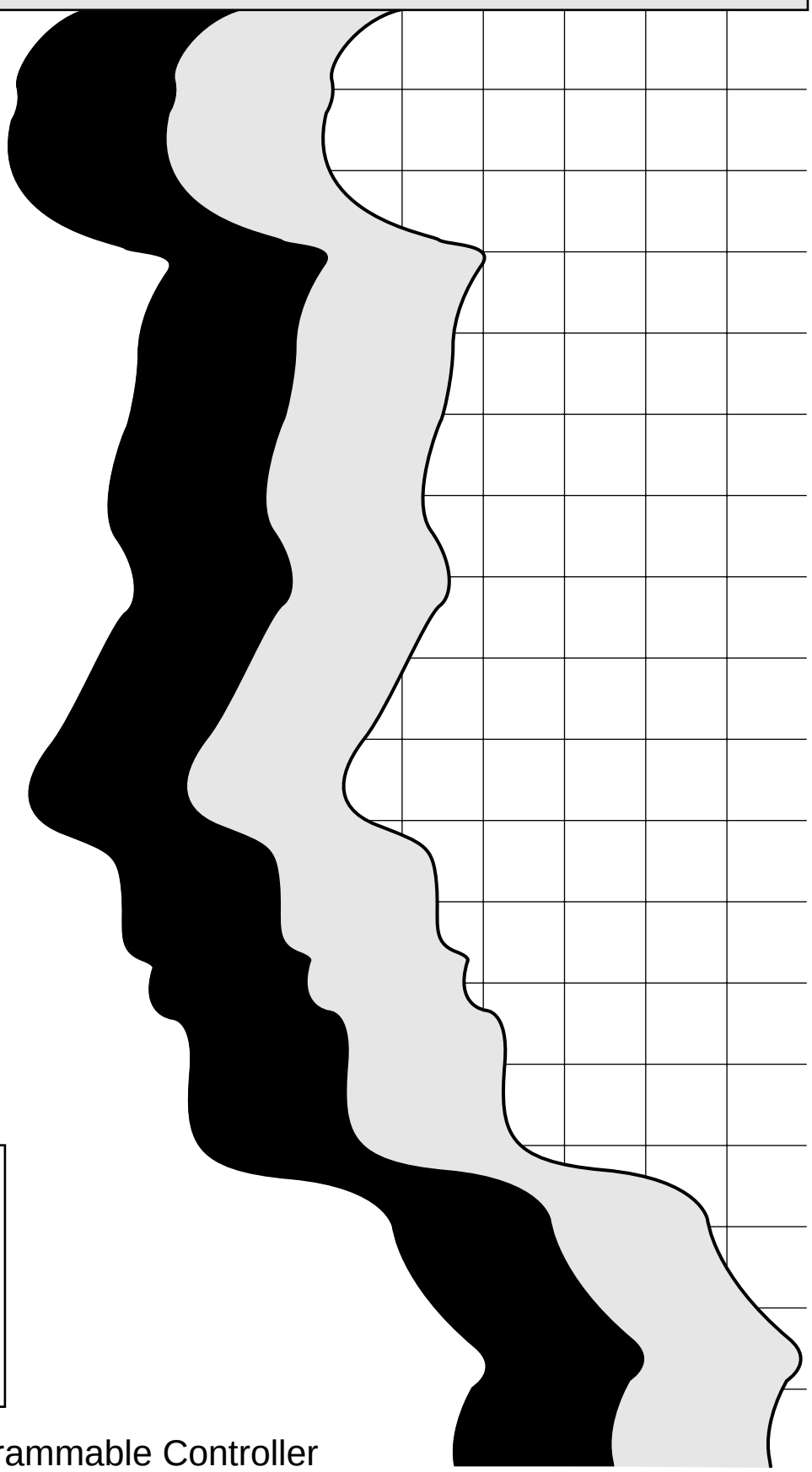
MITSUBISHI

Ethernet Interface Module type A1SJ71E71-B2/B5

User's Manual



MELSEC

Mitsubishi Programmable Controller

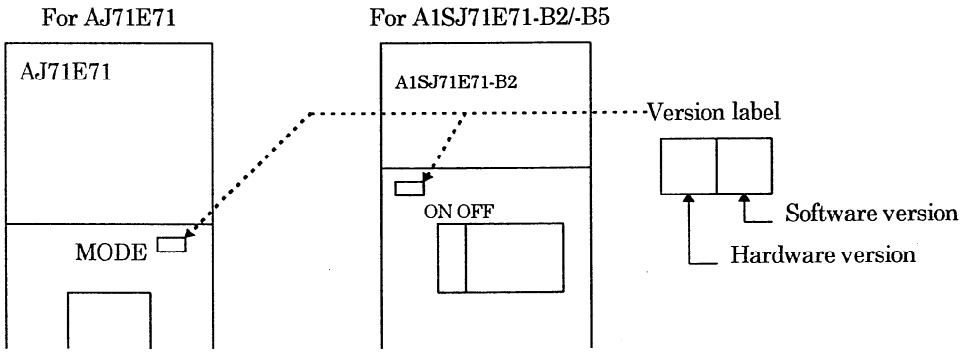
TECHNICAL BULLETIN

[Issue No.] T08-0002 **[Page]** 1/4
[Title] Notice on changes in MELSEC-A Series Ethernet interface module specifications, etc. **[Date of Issue]** Dec. '96
[Relevant Models] AJ71E71, A1SJ71E71-B2, A1SJ71E71-B5

Thank you for your continued patronage of the Mitsubishi general-purpose programmable logic controller (PLC) MELSEC-A Series.
The MELSEC-A Series Ethernet interface module specifications have been partially changed from the module software version "R". The details of the changes and the relevant manual revisions are as described below. (The above three models will be generically called AJ71E71 hereafter.)

Whereas

The module software version can be confirmed with the module version label on the front of the module.



The product with the changed specifications is compatible with the conventional product. When using the AJ71E71 with changed specifications, the program on the other node side used for exchanging data with the conventional AJ71E71 can be used as it is for exchanging data.
Note that if the "unit for each timer's setting value" is changed to "500 ms", the response timeout time with the other node being used for data exchange, etc., must be adjusted.

Details of specification changes

The details of the AJ71E71 specification changes are as described below.

(1) Addition of function to change each timer's setting value unit

(a) During the initialization of the AJ71E71, the unit of each timer value set in the buffer memory by the user can be set to a "500 ms unit" or a "2 s unit".
* The conventional product was fixed to a "2 s unit".

(b) By writing "500" in the buffer memory's "unit of each timer setting value" when initializing the AJ71E71, the unit for each timer value can be changed to a "500 ms unit".

Address	Buffer memory of AJ71E71
0	IP address of AJ71E71/A1SJ71E71
1	
2	Not used area (use not possible)
New → 3	Unit for each timer setting value
4	Not used area (use not possible)
9	
10	TCP ULP timeout value
11	TCP zero window timer value

← Set the unit for each timer's setting value

Setting value	Unit
500	500ms
Other than 500	2000ms (2s)

(Default value 2000: 2s unit)

* Each timer value will be the "timer setting value" x "each timer's setting value unit".
(Example) If the TCP ULP timeout value is set to 15 and each timer's setting value unit is set to 500, the TCP ULP timeout time will be 15 × 500 = 7500 ms.

(c) Designate the setting value for each timer from within the following range using buffer memory address 10 to 15 according to the value set for "each timer's setting value unit".

Each timer's setting value unit	Each timer's setting value setting range	Set time range
500	1 ~ 32767 (1 ~ 7FFFH)	500ms ~ 16383.5s
Other than 500	1 ~ 8191 (1 ~ 1FFFH)	2s ~ 16382s

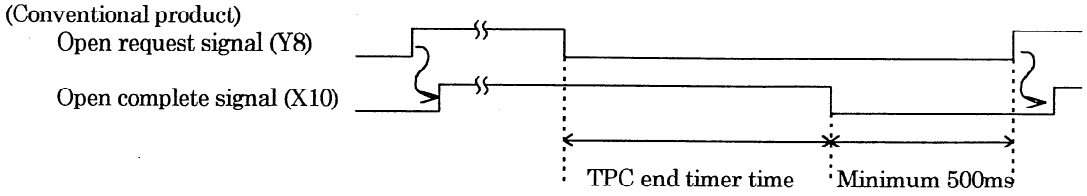
* If a value not within the above range is designated, the corresponding timer operation will not be guaranteed.

(2) Improvement of reopen time after connection is closed

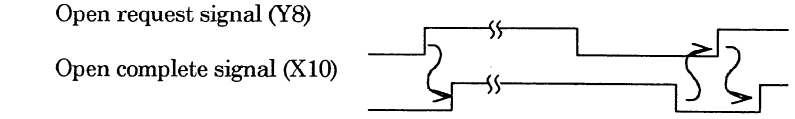
(a) When a random connection is closed and then reopened, the open request can be made immediately after the open complete signal (X10 to X17) turns OFF.
* With the conventional product, the reopen request could be made only when the following open request signal (Y8 to YF) had turned OFF, and the "TCP end timer time" and "minimum of 500 ms" had passed.

(Example) For connection No. 1

(Conventional product)



(Product with changed specifications)



[Issue No.] T08-0002 **[Page]** 4/4
[Title] Notice on changes in MELSEC-A Series Ethernet interface module specifications, etc. **[Date of Issue]** Dec. '96
[Relevant Models] AJ71E71, A1SJ71E71-B2, A1SJ71E71-B5

Details of User's Manual Revisions

The details of the revisions in the Ethernet interface module User's Manual are as follow.

- AJ71E71 type Ethernet interface module User's Manual .... (IB-68204-G and previous)
- A1SJ71E71-B2, A1SJ71E71-B5 type Ethernet interface module User's Manual (Details Section) .... (SH-3533-A)

The manual number and version can be confirmed on the lower left of the back cover.

IB(NA) 66310-B
IB(NA) 66547-A
Version
Manual number

* The following page numbers are the revised page number in each User's Manual.

(page 2-2~3/2-3)

- Revision of explanation for remarks in section 2.1.2 (1)/(2) (a)

Items that satisfy IEEE802.3 10BASE5 standards
↓
Items that satisfy Ethernet standards

In IEEE802.3 (described only in IB(NA)66547)
↓
In the transceiver electrical characteristics

(page 5-9)

- Addition to section 5.3.1 (1)

Bit 0 : Setting of fixed buffer usage application
Set "0" when carrying out communication with the random access buffer or read/write communication of data in PLC CPU.
Bit 14, 15 : Setting of opening method
Set "00: when UDP/IP open is to be used.

(page 9-4)

- Revision of buffer memory address in 4th line of explanation in section 9.1.4

Storage area (buffer memory address 168 to 178).
↓
Storage area (buffer memory address 169 to 179).

REVISIONS

*The manual number is given on the bottom left of the back cover.

Print Date	*Manual Number	Revision
Aug., 1995	IB (NA) 66547-A	First edition

INTRODUCTION

Thank you for choosing the Mitsubishi MELSEC-A Series of General Purpose Programmable Controllers. Please read this manual carefully so that the equipment is used to its optimum. A copy of this manual should be forwarded to the end User.

CONTENTS

1. INTRODUCTION

This manual gives the specifications, handling, and programming method of the A1SJ71E71 Ethernet interface module (hereafter called the A1SJ71E71) which is used to connect an A-series PC CPU to a computer using the Ethernet TCP/IP method.

The A1SJ71E71 functions as a node in 10BASE2 (Cheapernet) or 10BASE5 (Ethernet) network. Incorporating A1SJ71E71s into such networks allows data communications between an A-series PC CPU and a personal computer, or between A-series PCs.

In this manual, the term "Ethernet" is used to cover both 10BASE2 (Cheapernet) and 10BASE5 (Ethernet).

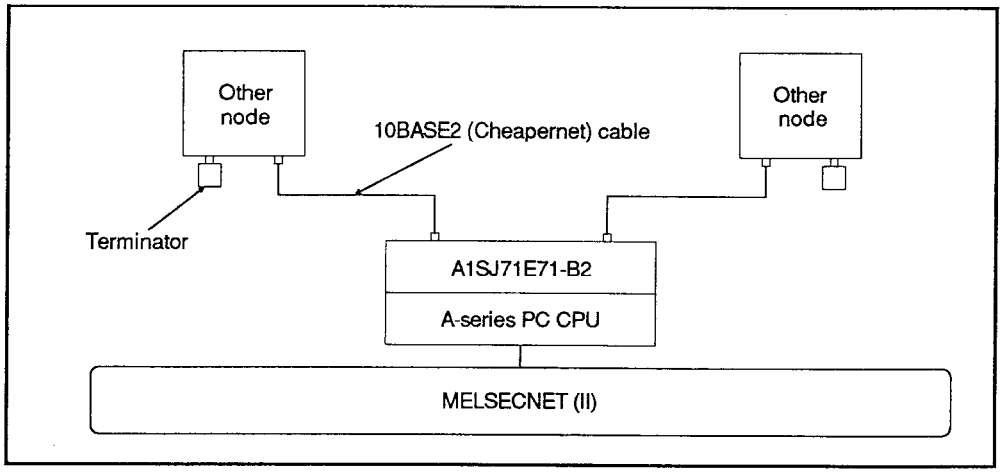


Fig. 1.1 10BASES2 (Cheapernet) Connection

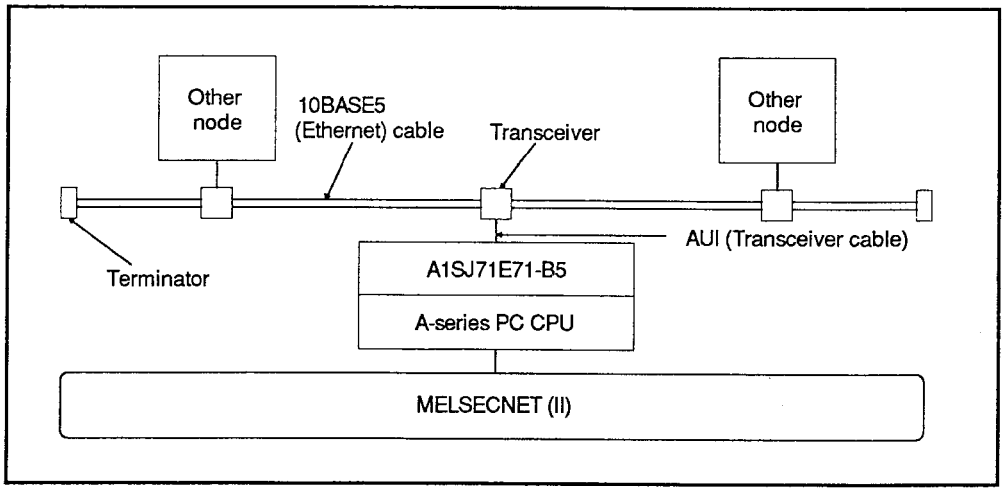


Fig. 1.2 10BASE5 (Ethernet) Connection

Confirm that the following product is contained in the A1SJ71E71E71-B2/B5 package.

See Section 2.1.2 for the parts and materials necessary in addition to this product.

Model	Product Name	Number
A1SJ71E71-B2	A1SJ71E71-B2 Cheapernet Interface Module	1
	BNC T-adapter UG-274/U	1
A1SJ71E71-B5	A1SJ71E71-B5 Ethernet Interface Module	1

1.1 Software Architecture

The A1SJ71E71 supports two kinds of protocols; TCP/IP and UDP/IP.

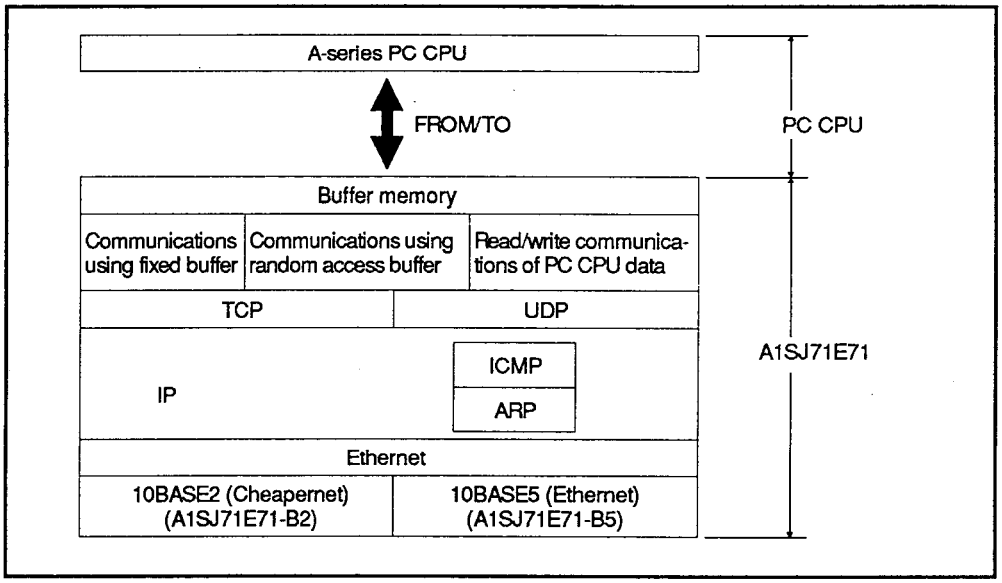


Fig. 1.3 Software Architecture

(1) TCP (Transmission Control Protocol)

This protocol guarantees the reliability or the certainty of data.

- By establishing a connection between nodes, this protocol makes a logical connection which is used exclusively for communications between the nodes.
- Up to 8 connections can be established simultaneously. Also, simultaneous communications using several buffers is possible.
- Reliability of data is assured by the sequential control using sequence numbers, the retransmission function of data, and the use of the check sum.
- Flow of communications data is controlled by the window operation.
- The MAX SEGMENT option is supported.

(2) UDP (User Datagram Protocol)

This protocol does not guarantee the reliability and the certainty of data.

Therefore, even if data fails to reach to a destination node, the data is not retransmitted.

- Because connections are not necessary, a high-speed communications is enabled.
- Check sum is added to improve reliability of communications data.
- However, if higher reliability is necessary, use a user application or TCP.

(3) IP (Internal Protocol)

- This protocol transmits/receives the communications data in the datagram form.
- This protocol can split and assemble communications data.
- This protocol does not support a routing option.

(4) ARP (Address Resolution Protocol)

- This calculates the Ethernet physical address from the IP address by the broadcast function.

(5) ICMP (Internal Control Message Protocol)

- This has the function that transmits the error messages of the IP.
- This does not support the ICMP options.

1.2 Features of A1SJ71E71

The following describes features of the A1SJ71E71.

(1) Data communications with a specific node by handshaking (Fixed buffer communications)

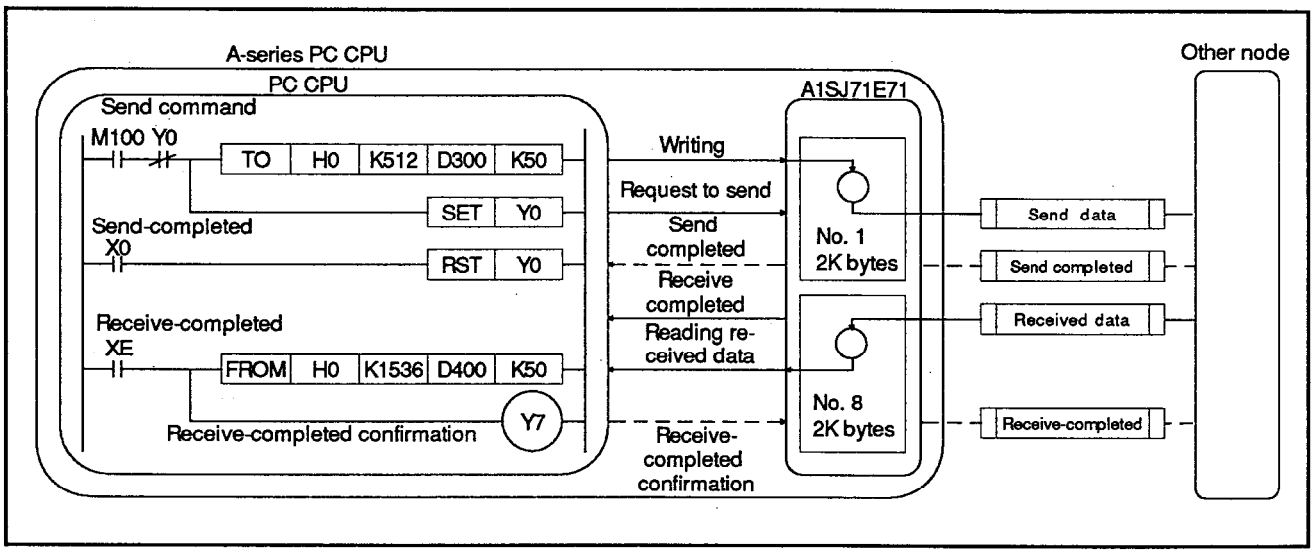


Fig. 1.4 Fixed Buffer Communications

(a) The PC CPU can write send data to the A1SJ71E71 and can read received data from the A1SJ71E71 while using handshaking with other nodes.

(b) When communications is done using fixed buffer, communications (data transmission/receive) with specific nodes can be done.

When data transmission and receive are done with a specific node, 2 fixed buffer areas are required. Communicating nodes and the type of communications (send or receive) for each fixed buffer area are set with communications parameters.

Communications between two A1SJ71E71s is also possible.

(c) There are 8 fixed buffer areas from No. 1 to No. 8 (1K words/area). (Section 3.3 gives details of allowable data capacity per communications.)

(2) Communications by read/write requests from several nodes (Random access buffer communications)

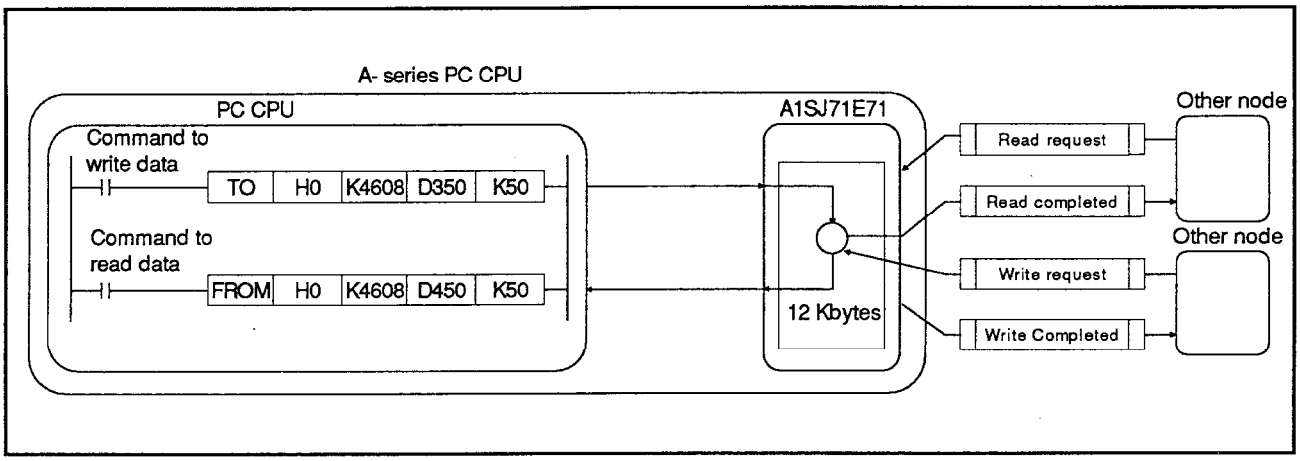


Fig. 1.5 Random Access Buffer Communications

(a) When random access buffer is used, data read/write with several nodes can be done with one same buffer address.

However, because the communications between a PC CPU and communicating nodes is asynchronous, the user has to add interlock processing.

(b) The random access buffer holds 6K words (3K for channel 0 and 3K for channel 1).

Buffer area is not set for each connection.

(c) The PC CPU reads and writes data to and from the random access buffer by switching channels in the 3K word unit.

However, communicating nodes use this buffer area as one continuous area of 6K words. (Section 3.3 gives details of allowable data capacity per communications.)

(3) Read/write of data in the PC CPU by the request from other nodes

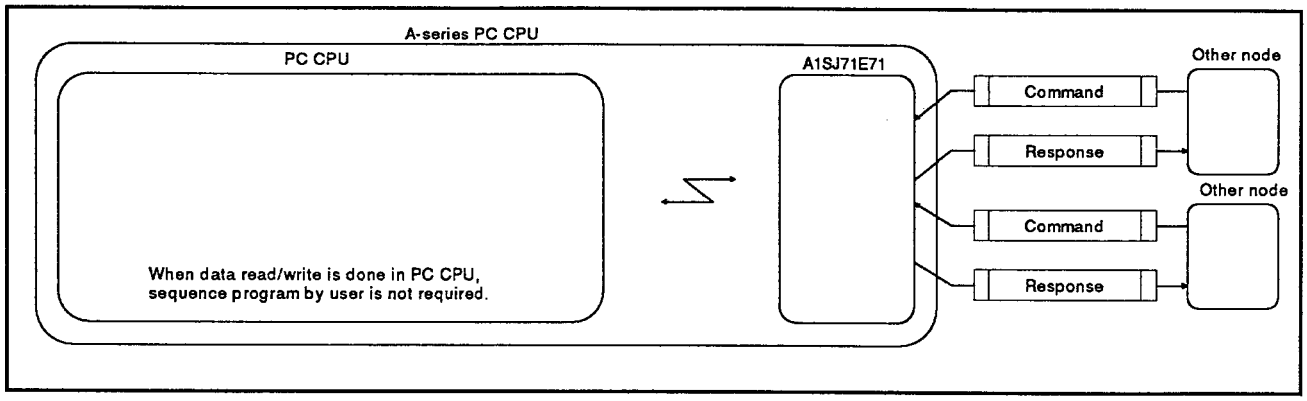


Fig. 1.6 Data Read/Write in the PC CPU

(a) Upon receiving a read/write request for the data in the PC CPU from communicating nodes, the A1SJ71E71 transmits or receives the data of devices, programs, comments and parameters.

(b) When a PC CPU which is loaded with an A1SJ71E71 is connected to MELSECNET, data communications can be done with the PC CPUs in the MELSECNET. (See Section 8.1.2.)

(c) Since data communications is done between the A1SJ71E71 and communicating nodes, any special sequence program to perform data communications is not needed.

(4) Selection (ASCII/binary) of the data code of communications data is enabled.

Communications data code used between the A1SJ71E71 and other nodes can be set to either ASCII or binary. (See Section 3.3.)

(5) Communications method (TCP/IP and UDP/IP) can be selected.

Each connection (communicating node) can select either TCP/IP or UDP/IP for communications with the A1SJ71E71. (See Section 5.3.)

2. SYSTEM CONFIGURATION

This section explains the system configuration that can be used with the A1SJ71E71-B2/B5.

2.1 Overall Configuration

2.1.1 Connection between an independent PC CPU and 10BASE2 (Cheapernet)/ 10BASE5 (Ethernet)

(1) When a PC CPU which is not connected to the MELSECNET data link is connected with Ethernet/Cheapernet, use the following system configuration.

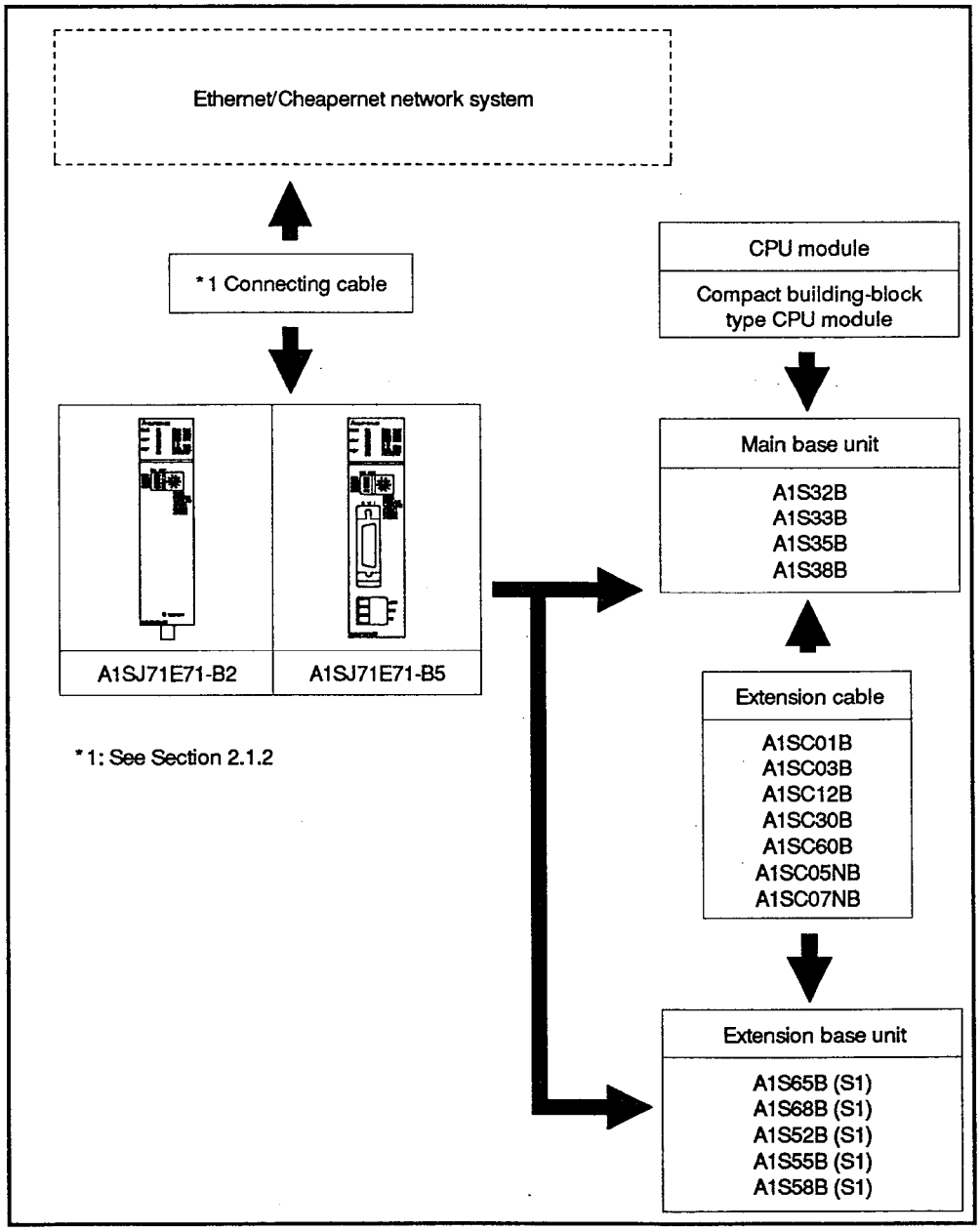


Fig. 2.1 System Configuration with a Single PC CPU

2.1.2 Equipment necessary to construct a network

(1) When using an A1SJ71E71-B2, the user must prepare the equipment shown in Figure 2.2.

(a) Coaxial cable for 10BASE2 (Cheapernet).

RG-58/U

(b) BNC plug (for connection to the BNC T-adapter)

UG-88/U (made by Hirose) or equivalent

(c) Terminator

Plug type terminator, BNC type (made by Fujikura Densen) or equivalent

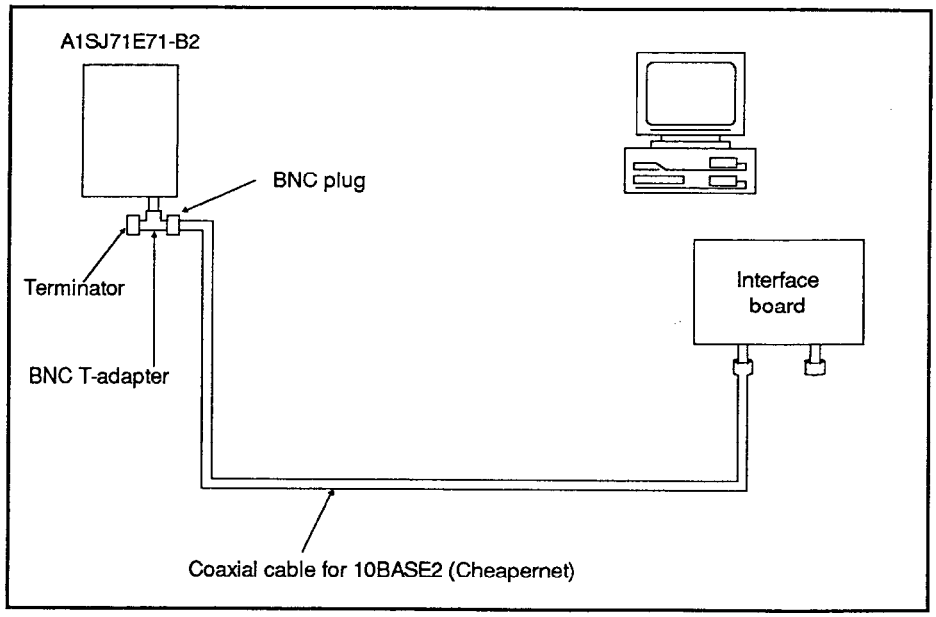


Fig. 2.2 Example Network System Configuration

(2) When using an A1SJ71E71-B5, the user must prepare the equipment shown in Figure 2.3.

(a) Use a coaxial cable for 10BASE5 (Ethernet), N-connector, N-terminator, transceiver, and transceiver cable that satisfy IEEE802.3 10BASE5 standards.
In general, use a transceiver that has a signal designated "SQETEST" or called the "heart beat signal" (this signal executes a transceiver function which checks whether the transceiver operates normally after data is sent).

(b) Use a 12 VDC power supply to the transceiver that will satisfy the transceiver and transceiver cable specifications, taking into account the voltage drop (max. 0.8 V) in the A1SJ71E71-B5.

REMARK

The IEEE802.3 standard includes the following stipulations:

- Transceiver input terminal voltage: 12 V $^{-6\%}$ to 15 V $^{+15\%}$
- Transceiver cable DC resistance: 40 Ω/km max., length: 50 m
- Transceiver max. current consumption: 500 mA or less

Accordingly when the voltage drop of 0.8 V in the A1SJ71E71-B5 is taken into account, the guide for the transceiver power supply is 13.08 V to 15.75 V.

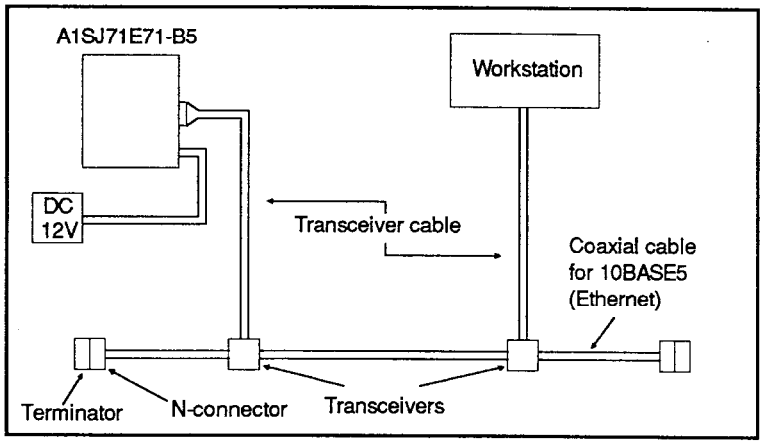


Fig. 2.3 Network System Configuration Example

POINT

Entrust 10BASE2 (Cheapernet) and 10BASE5 (Ethernet) installation work to a specialist contractor since adequate safety measures are required. For the installation environment, refer to JIS X 5252.

2.2 Applicable CPU Modules

The A1SJ71E71-B2/B5 can be used with the following CPU modules.

(1) Applicable CPU modules and the maximum number of A1SJ71E71-B2/B5s

CPU Module	Maximum Number of Modules	Note
A1SCPU-C24	1	When the following modules are used with the A1SJ71E71-B2/B5, they must be included in the maximum number of modules. • A1SJ71C24-R2/R4/PRF, A1SD51S • AD51(S3), AD51H(S3) AJ71C21(S1) {when using the BASIC program mode} AJ71C22 (S1)/C23/C24 (S3/S6/S8) AJ71UC24 AJ71P41 AJ71E71
A1SCPU(S1) A1SJCPU A2SCPU(S1)	2	
A2ASCPU(S1)	6	

(2) Applicable base unit

The A1SJ71E71 can be loaded in any slot of the main base unit or extension base unit.

3. SPECIFICATIONS

This chapter describes the general specifications performance specifications, and I/O conversion characteristics, and I/O conversion characteristics of the A1SJ71E71.

3.1 General Specifications

Table 3.1 shows the general specifications of the A1SJ71E71.

Table 3.1 General Specifications

<table>
<tr><th>Item</th><th colspan="5">Specifications</th></tr>
<tr><td>Operating ambient temperature</td><td colspan="5">0 to 55 °C</td></tr>
<tr><td>Storage ambient temperature</td><td colspan="5">–20 to 75 °C</td></tr>
<tr><td>Operating ambient humidity</td><td colspan="5">10 to 90% RH, non-condensing</td></tr>
<tr><td>Storage ambient humidity</td><td colspan="5">10 to 90% RH, non-condensing</td></tr>
<tr><td rowspan="3">Vibration resistance</td><td rowspan="3">Conforms to *JIS C 0911</td><td>Frequency</td><td>Acceleration</td><td>Amplitude</td><td>Sweep Count</td></tr>
<tr><td>10 to 57 Hz</td><td>—</td><td>0.075 mm (0.003 in)</td><td rowspan="2">10 times *(1 octave/minute)</td></tr>
<tr><td>57 to 150 Hz</td><td>9.8 m/s^2 (1 G)</td><td>—</td></tr>
<tr><td>Shock resistance</td><td colspan="5">Conforms to JIS C 0912 (147 m/s^2 {15 G} x 3 times in 3 directions)</td></tr>
<tr><td>Noise resistance</td><td colspan="5">By noise simulator of 1500 Vpp noise voltage, 1 μs noise width, and 25 to 60 Hz noise frequency.</td></tr>
<tr><td>Withstanding voltage</td><td colspan="5">500 VAC for 1 minute across DC external terminals and ground</td></tr>
<tr><td>Insulation resistance</td><td colspan="5">5 MΩ or larger, measured with 500 VDC insulation resistance tester across AC terminals and ground.</td></tr>
<tr><td>Grounding</td><td colspan="5">Class 3 grounding if possible. If not possible, ground to panel.</td></tr>
<tr><td>Operating environment</td><td colspan="5">Free of corrosive gases. Dust should be minimal.</td></tr>
<tr><td>Cooling method</td><td colspan="5">Self-cooling</td></tr>
</table>

REMARK

One octave, marked *, indicates a change from the initial frequency to double or half frequency. For example, any of these changes - from 10 Hz to 20 Hz, from 20 Hz to 40 Hz, from 40 Hz to 20 Hz, and from 20 Hz to 10 Hz - is referred to as one octave.

* JIS: Japanese Industrial Standard

3.2 Performance Specifications

The performance specifications of the A1SJ71E71-B2/B5 are tabled below.

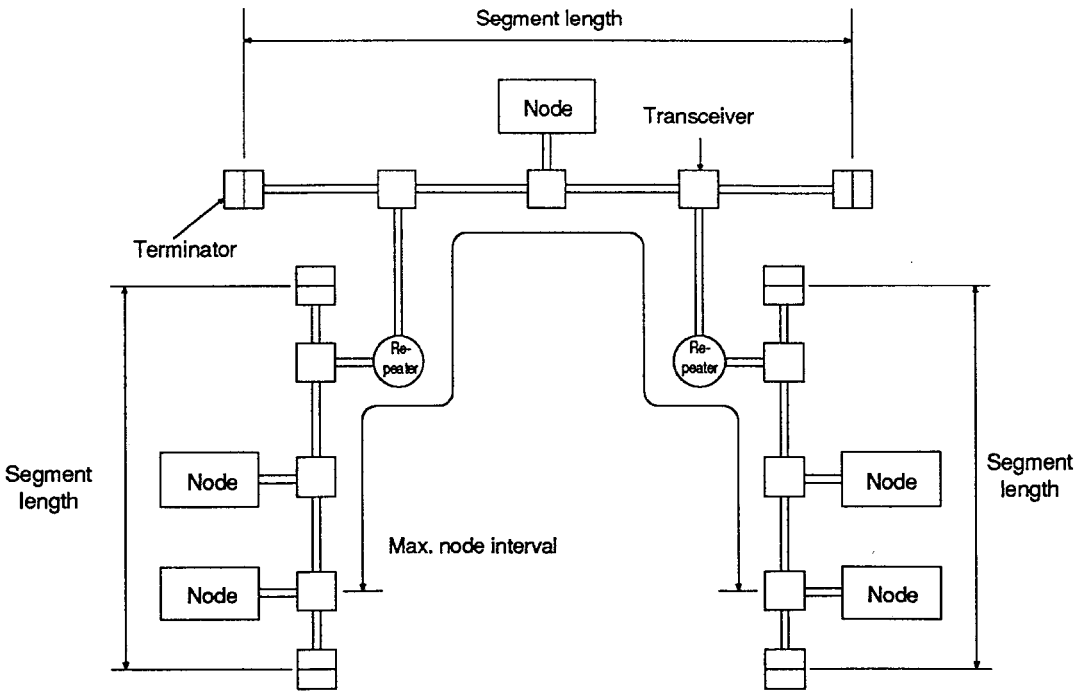
Table 3.2 Performance Specifications

Item		Specifications	
		A1SJ71E71-B2 10BASE2 (Cheapernet)	A1SJ71E71-B5 10BASE5 (Ethernet)
Transmission specifications	Data transmission speed	10 Mbps	
	Transmission method	Base band	
	Max. node interval (m) (ft)	925 (3034.93)	2500 (8202.50)
	Max. segment length (m) (ft)	185 (606.99)	500 (1640.50)
	Max. number of nodes	30/segment	100/segment
	Min. node interval (m) (ft)	0.5 (1.64)	2.5 (8.20)
Communications data storage memory	Fixed buffer	2k bytes x 8	
	Random access buffer	12k bytes x 1	
Number of inputs and outputs		32 points	
5 VDC internal current consumption [A]		0.52	0.35
12 VDC external power supply capacity			Must satisfy the transceiver and transceiver cable specifications, taking the voltage drop in the module (max. 0.8 V) into account.
Outside dimensions (mm) (in)		130 (H) x 34.5 (W) x 93.6 (D) (5.12 (H) x 1.36 (W) x 3.7 (D))	
Weight (kg) (lb)		0.30 (0.66) *1	0.27 (0.59)

*1: This includes the weight of the T-adapter (20 g) and the terminal resistor (10 g).

REMARK

The maximum node interval and segment length are illustrated below.



3.3 Data Code Used for Communications

The following explains the data code used for communications between an A1SJ71E71 and communicating nodes or between an A1SJ71E71 and a PC CPU.

(1) Data code used for communications

Between A1SJ71E71 and other nodes ..... Switching of binary/ ASCII code is enabled.

Between A1SJ71E71 and a PC CPU ..... Binary code

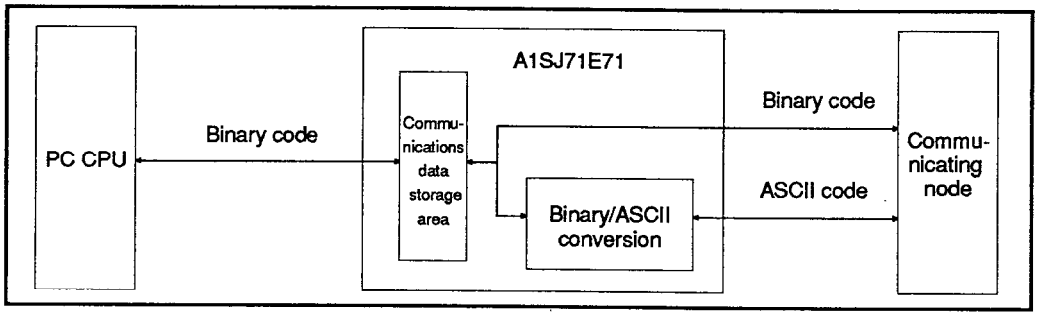


Fig. 3.1 Code system of communication data

(2) Switch from binary to ASCII by using a DIP switch on the front of A1SJ71E71 (As for the details, refer to Section 4.3.3.).

(3) When communications is done in the ASCII code, 1-byte binary code data is converted automatically to 2-byte ASCII code.

Example:

Binary code data		ASCII code data
15_H	⟶	$31_H, 35_H$ "1" "5"
(1 byte)		(2 bytes)
1234_H	⟶	$31_H, 32_H, 33_H, 34_H$ "1" "2" "3" "4"
(2 bytes)		(4 bytes)

(4) The data capacity that can be communicated at one time between an A1SJ71E71 and other node varies as follows according to the function and data code (binary/ASCII).

Function \ Data Code	Binary Code	ASCII Code
Communications using fixed buffer	Maximum 1017 words	Maximum 508 words
Communications using random access buffer		
Read/write of data in the PC CPU	Up to the maximum number of words that can be set with each command when either data code is used.	

3.4 Function of A1SJ71E71-B2/B5

Table 3.3 Functions of A1SJ71E71

Function	Description	Communicating Nodes		
		Other node ↓ A1SJ71E71	A1SJ71E71 ↓ Other node	A1SJ71E71 ↓ A1SJ71E71
Communications using fixed buffer	(1) Using the handshaking signal in the one-to-one connection ratio, a PC CPU performs communications with other node. (Communications between two AJ71E71s are also possible.) (2) There are 8 areas (1K words per 1 area) to do communications with other nodes. (Section 3.3 gives the data capacity per one time of communication.) (3) Set a communicating node and the type (send/receive) at fixed buffer by using communications parameters. Two fixed buffer areas are needed to do data send and receive with one node. (4) Communications with up to 8 nodes for which connection has been opened are possible.	o	o	o
Communications using randam access buffer memory	(1) Read/write communications is possible between several nodes and the random access buffer memory of the A1SJ71E71. (Communications between A1SJ71E71s is impossible.) (2) The area used for communications with other nodes has 6K words (channel 1: 3K words and channel 0: 3K words). The PC CPU can read/write data using all areas by switching the channels. Communicating nodes can read/write data using the area as one continuous area. (Section 3.3 gives the data capacity per one time of communication.) (3) The random access area is not provided with allocated connections.This buffer can be used as common buffer memory in the network. (4) Communications with up to 8 nodes for which connection has been opened is possible.	o		x
Read/write communi-cations of data in the PC CPU	(1) Upon receiving a request from a communicating nodes, the A1SJ71E71 can read/write data of devices, programs, comments and parameters in the PC CPU which is loaded with the A1SJ71E71. (2) When a PC CPU which is loaded with an A1SJ71E71 is connected to MELSECNET, data communications can be done between any node and any PC CPU in the MELSECNET. (3) Communications with up to 8 nodes for which connection has been opened is possible.	o		x
Self-loopback test	(1) The A1SJ71E71 hardware containing the data transmission and receive circuits is checked.			

REMARK

Communication using the fixed buffer is also possible between an A1SJ71E71-B2/B5 and AJ71E71.

3.5 I/O Signals Used for the PC CPU

The following list shows the I/O signals used for communications between the A1SJ71E71 and the PC CPU.

The X/Y number allocation in the following table is used when the A1SJ71C24 is loaded in slot 0 of the main base unit.

Devices X indicate the input from an A1SJ71E71 to a PC CPU, and devices Y indicate the output from a PC CPU to an A1SJ71E71.

Table 3.4 I/O Signals for PC CPU

Signal direction A1SJ71E71 → PC CPU			Signal direction PC CPU → A1SJ71E71		
Device No.	Signal		Device No.	Signal	
X0	Send-completed or receive-completed	Connection No. 1 fixed buffer communications	Y0	Connection No. 1	Send request or receive-completed confirmation
X1	Send error-detected		Y1	Connection No. 2	
X2	Send-completed or receive-completed	Connection No. 2 fixed buffer communications	Y2	Connection No. 3	
X3	Send error-detected		Y3	Connection No. 4	
X4	Send-completed or receive-completed	Connection No. 3 fixed buffer communications	Y4	Connection No. 5	
X5	Send error-detected		Y5	Connection No. 6	
X6	Send-completed or receive-completed	Connection No. 4 fixed buffer communications	Y6	Connection No. 7	
X7	Send error-detected		Y7	Connection No. 8	
X8	Send-completed or receive-completed	Connection No. 5 fixed buffer communications	Y8	Connection No. 1	Open request
X9	Send error-detected		Y9	Connection No. 2	
XA	Send-completed or receive-completed	Connection No. 6 fixed buffer communications	YA	Connection No. 3	
XB	Send error-detected		YB	Connection No. 4	
XC	Send-completed or receive-completed	Connection No. 7 fixed buffer communications	YC	Connection No. 5	
XD	Send error-detected		YD	Connection No. 6	
XE	Send-completed or receive-completed	Connection No. 8 fixed buffer communications	YE	Connection No. 7	
XF	Send error-detected		YF	Connection No. 8	
X10	Connection No. 1	Open completed	Y10	Unused	
X11	Connection No. 2		Y11		
X12	Connection No. 3		Y12		
X13	Connection No. 4		Y13		
X14	Connection No. 5		Y14		
X15	Connection No. 6		Y15		
X16	Connection No. 7		Y16		
X17	Connection No. 8		Y17	COMM, ERR LED OFF	
X18	Open error-detected		Y18	Unused	
X19	Initial-completed		Y19	Initial request	
X1A	Initial error-detected		Y1A	Unused	
X1B	Unused		Y1B		
X1C			Y1C	Buffer memory channel switching	
X1D			Y1D	Unused	
X1E			Y1E		
X1F	Watchdog timer error-detected		Y1F		

IMPORTANT

Since the device numbers indicated as "unused" in Table 3.4 are used by the system, the user must not use them.

If used by the user, normal operation cannot be guaranteed.

3.5.1 Details of I/O signals

The following explains the ON/OFF timing and conditions of the I/O signals shown in Table 3.4.

SA( ) shows the device number that corresponds to Table 3.4.

(1) Send-completed or receive-completed signal (X0, X2, X4, X6, X8, XA, XC and XE)

This signal is used for communications using fixed buffer.

This signal is not used for communications using random access buffer and read/write of data in the PC CPU.

When fixed buffer is used for sending data, this signal is used as a send-completed signal.

When fixed buffer is used for receiving data, this signal is used as a receive-completed signal.

(a) When using this signal as a send-completed signal

1) When s send request signal (Y0 to Y7) turns ON, data is transmitted.

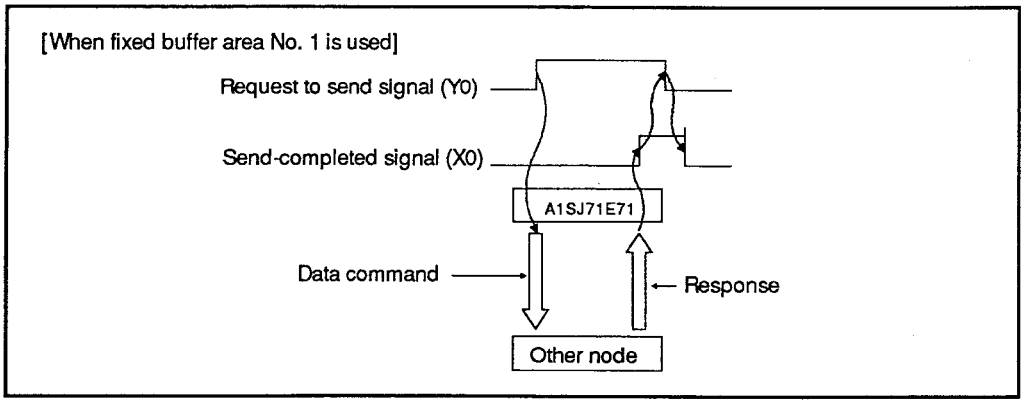
2) A node which received data sends a response to the A1SJ71E71.

3) When a response is sent from the node, the send-completed signal turns ON.

4) When a request to send signal (Y0 to Y7) turns OFF, a send-completed signal is turned OFF, too.

5) When the completion code of a response from other node is other than H, the send-completed signal does not turn ON.

Send error-detected signal (X1, X3, X5, X7, X9, XB, XD, and XF) turns ON.



(b) When using this signal as a receive-completed signal

1) When the A1SJ71E71 receives data from a communicating node, this signal turns ON.

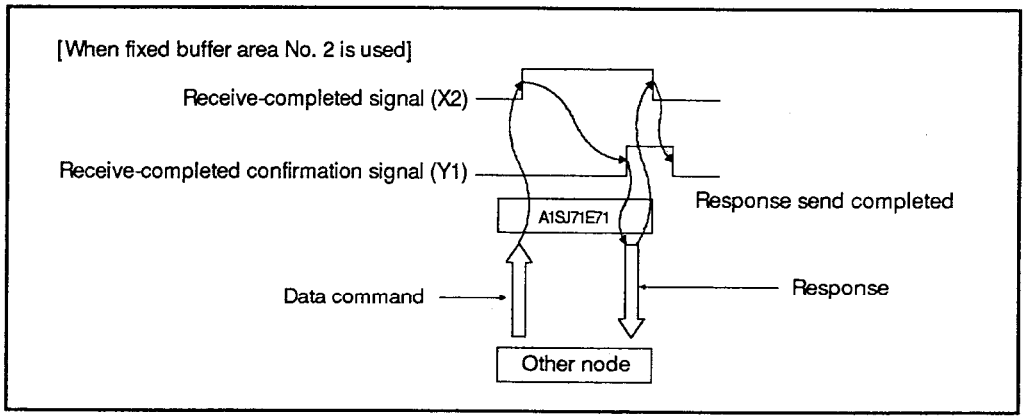
2) When reading received data to the PC CPU using the FROM instruction, this signal can be used for handshaking.

3) After reading received data using the FROM instruction, a receive-completed confirmation signal (Y0 to Y7) turns ON.

A response is sent to a node which transmitted data.

4) The receive-completed signal turns OFF automatically after sending a response to a node.

5) When error data is transmitted from a node, the receive-completed signal does not turn ON.



(2) Send error-detected signal (X1, X3, X5, X7, X9, XB, XD, and XF)

This signal is used for communications using fixed buffer.

This signal is not used for read/write of data in the PC CPU and for the communications using random access buffer.

(a) When a response is not sent from a communicating node after data transmission using fixed buffer within the response watchdog timer value (Section 5.2.1), a send error-detected signal turns ON.

(b) When ACK is not sent after data transmission with fixed buffer using the TCP connection, specified retry processing (Section 5.2.1) is executed, and then, send error-detected signal turns ON.

(c) When the completion code of the response from a communicating node after data transmission using fixed buffer is other than 00H, a send error-detected signal turns ON.

(d) When a request to send signal (Y0 to Y7) of fixed buffer turns OFF, a send error-detected signal is turned OFF, too.

(3) Connection open-completed signal (X10 to X17)

(a) When an open request signal (Y8 to YF) of each connection is turned ON with a sequence program, communications parameters are checked, and the open processing is executed.

And then, when the open processing is executed normally, an open-completed signal (X10 to X17) turns ON.

(b) When an open request signal is turned ON and the open processing is not executed normally, an open error-detected signal (X18) turns ON.

The connection open-completed signal does not turn ON in this case.

(c) Data communications (fixed buffer communications, random access buffer communications, and read/write of data in the PC CPU) is enabled only with the node(s) for which the connection open-completed signal (X10 to Y17) is turned ON.

(d) The ON/OFF states of the connection open-completed signal (X10 to X17) can be confirmed by the LEDs (BUF1 to BUF8) on the front of the A1SJ71E71.

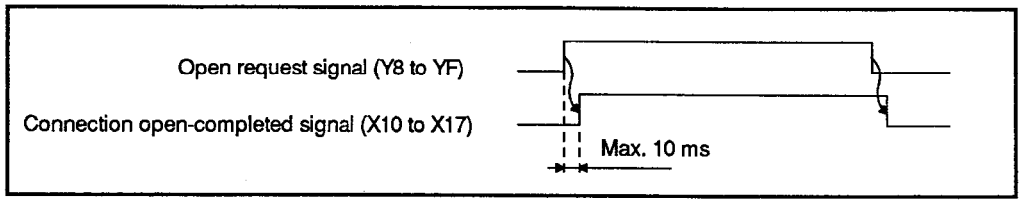
(e) When an open request signal is turned OFF with a sequence program, the connection open-completed signal (X10 to X17) turns OFF.

The connection open-completed signal turns OFF also in the following cases.

1) When the TCP time out error occurs. (Refer to Section 5.3.2.)

2) When the CLOSE or ABORT instruction is received from a communicating node. (Refer to Section 5.3.2.)

3) When a response watchdog timer error occurs. (Refer to Section 5.3.2.)



(4) Open error-detected signal (X18)

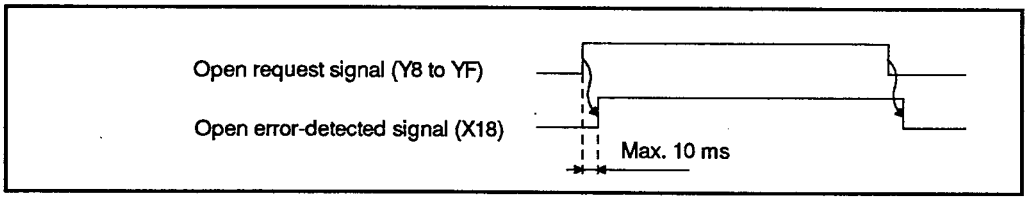
(a) When an open request signal (Y8 to YF) of each connection is turned ON by the sequence program, communications parameters are checked. The error-detected signal turns ON.

(b) When an open request signal (Y8 to YF) is turned ON, and the open processing is not executed normally, an open error-detected signal turns ON.

(c) When an open error-detected signal turns ON, read an open error code storage area (buffer addresses 93, 103 and 113...163) in the communications state storage area. This enables the connection No. and the error description of the current error to be monitored.

(d) The open error-detected signal (X18) turns OFF by turning OFF the open request signal (Y8 to YF) of the connection in which an open error is occurring.

(e) When several open errors occur, all corresponding open request signals must be turned OFF to turn OFF the open error-detected signal (X18).



(5) Initial-completed signal (X19)

(a) When an initial request signal (Y19) is turned ON with a sequence program, the initial parameters are checked, and the initial processing is executed.

And then, when the initial processing is executed normally, the initial-completed signal (X19) turns ON.

(b) When the initial processing is not executed normally, the initial error-detected signal (X1A) turns ON.

The initial-completed signal (X19) does not turn ON in this case.

(6) Initial error-detected signal (X1A)

(a) When the initial request signal (Y19) is turned ON, and when the initial processing is not executed normally, the initial error-detected signal (X1A) turns ON.

(b) When initial error-detected signal (X1A) turns ON, read an initial error code storage area (buffer memory 80) in the communications state storage area (See Section 5.2.3.), and then, the error description can be monitored.

(c) The initial error-detected signal (X1A) turns OFF by turning OFF the initial request signal.

(7) WDT error-detected signal (X1F)

When a WDT error is detected by the self-diagnostic function of the A1SJ71E71, a WDT error-detected signal (X1F) turns ON.

(8) Request to send or receive-completed confirmation signal (Y0 to Y7)

This signal is used for communications using fixed buffer.

This signal is not used for communications using random access buffer and read/write of data in the PC CPU.

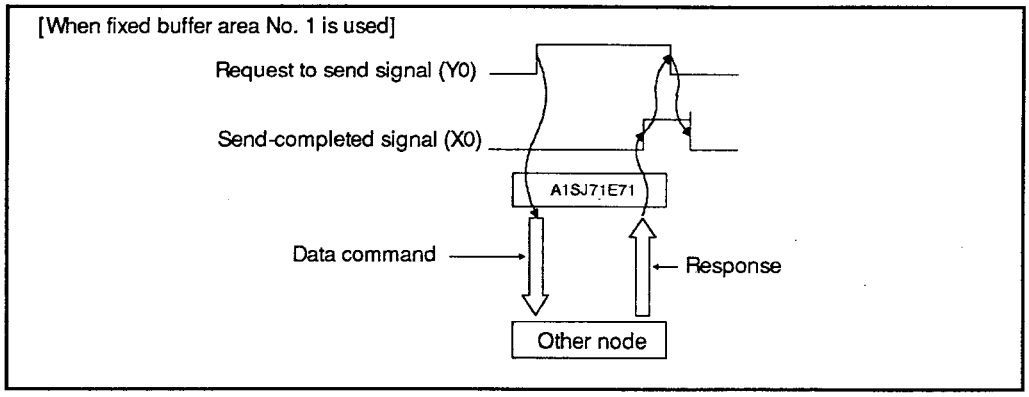
When fixed buffer is used for sending, this signal can be used as a request to send signal.

When fixed buffer is used for receiving, this signal can be used as a receive-completed confirmation signal.

(a) When this signal is used as a request to send signal

1) The A1SJ71E71 transmits data to a node specified by the communications parameter by turning ON the request to send signal (Y0 to Y7) with a sequence program.

2) A response is sent back from the node after data transmission, and a send-completed signal (X0 when fixed buffer No.1 is used) turns ON. And the send is completed.

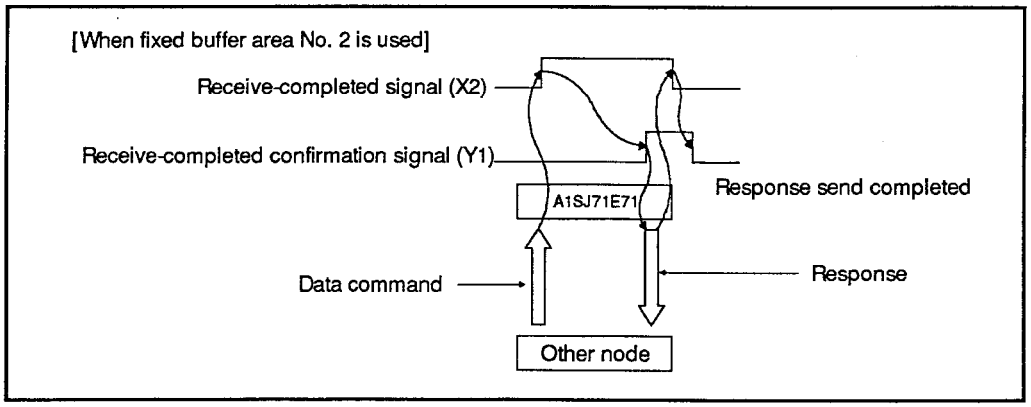


(b) When using this signal as a receive-completed signal

1) After the A1SJ71E71 receives data from a node, a receive-completed signal (X2 when fixed buffer No. 2 is used) turns ON.

2) The sequence program confirms the receive-completed signal (X2 when fixed buffer No. 2 is used) ON state.

Then, when the receive-completed confirmation signal (Y0 to Y7) turns ON, the A1SJ71E71 sends a response to the communicating node.



(9) Open request signal (Y8 to YF)

(a) This signal is turned ON so that the A1SJ71E71 can execute data communications (fixed buffer communications, random access buffer communications and data read/write in the PC CPU) with other nodes.

(b) The communications parameters are checked when an open request signal (Y8 to YF) of each connection is turned ON by a sequence program.

If the check result is normal, the open processing is executed.

When an error is detected, the open error-detected signal (X18) turns ON.

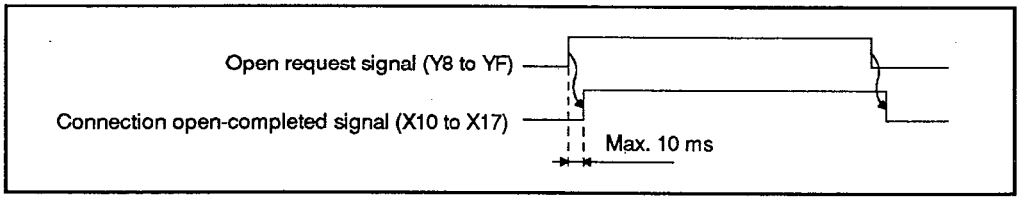
(c) When the open processing is executed normally when an open request signal is turned ON, the connection open-completed signal (X10 to X17) turns ON.

When an error is detected, the open error-detected signal (X18) turns ON.

(d) The open error-detected signal (X18) turns OFF, when the open request signal (Y8 to YF) is turned OFF. When several connections are causing errors, turn OFF all the open request signals for those connections.

When the open error-detected signal (X18) is turned OFF, the open error code storage area in the communications state storage area is also cleared.

Therefore, be sure to read the open error code storage area (buffer addresses 93, 103 and ...163) before turning OFF the open request signal when an error occurs.



(10) The "COM.ERR" LED OFF signal (Y17)

This signal is used to turn off the "COM.ERR" LED that turns on when a communications error occurs.

The "COM.ERR" LED turns off by turning ON the turn OFF signal (Y17) with a sequence program.

While the turn OFF signal (Y17) is ON, the turn OFF processing is executed.

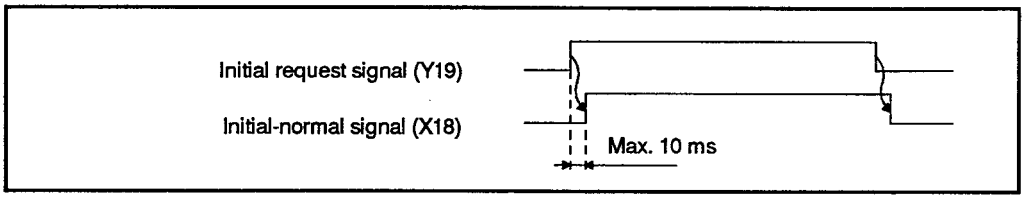
(11) Initial request signal (Y19)

(a) This signal is used to initialize the A1SJ71E71 before starting data communications.

(b) The initial parameters are checked by turning ON the initial request signal (Y19) with a sequence program.

If the check result is normal, the initial processing is executed.

When an error is detected, the initial error-detected signal (X1A) turns ON.



(c) When the initial processing is executed normally when the initial request signal (Y19) is turned ON, the initial-completed signal (X19) turns ON.

And then, when an error is detected, the initial error-detected signal (X1A) turns ON.

(12) Buffer memory channel switching signal (Y1C)

This signal is used to specify a channel to be used with buffer memory.

Before doing read/write with the A1SJ71E71 buffer memory by the FROM/TO instruction of a sequence program, this signal is turned ON or OFF with a sequence program.

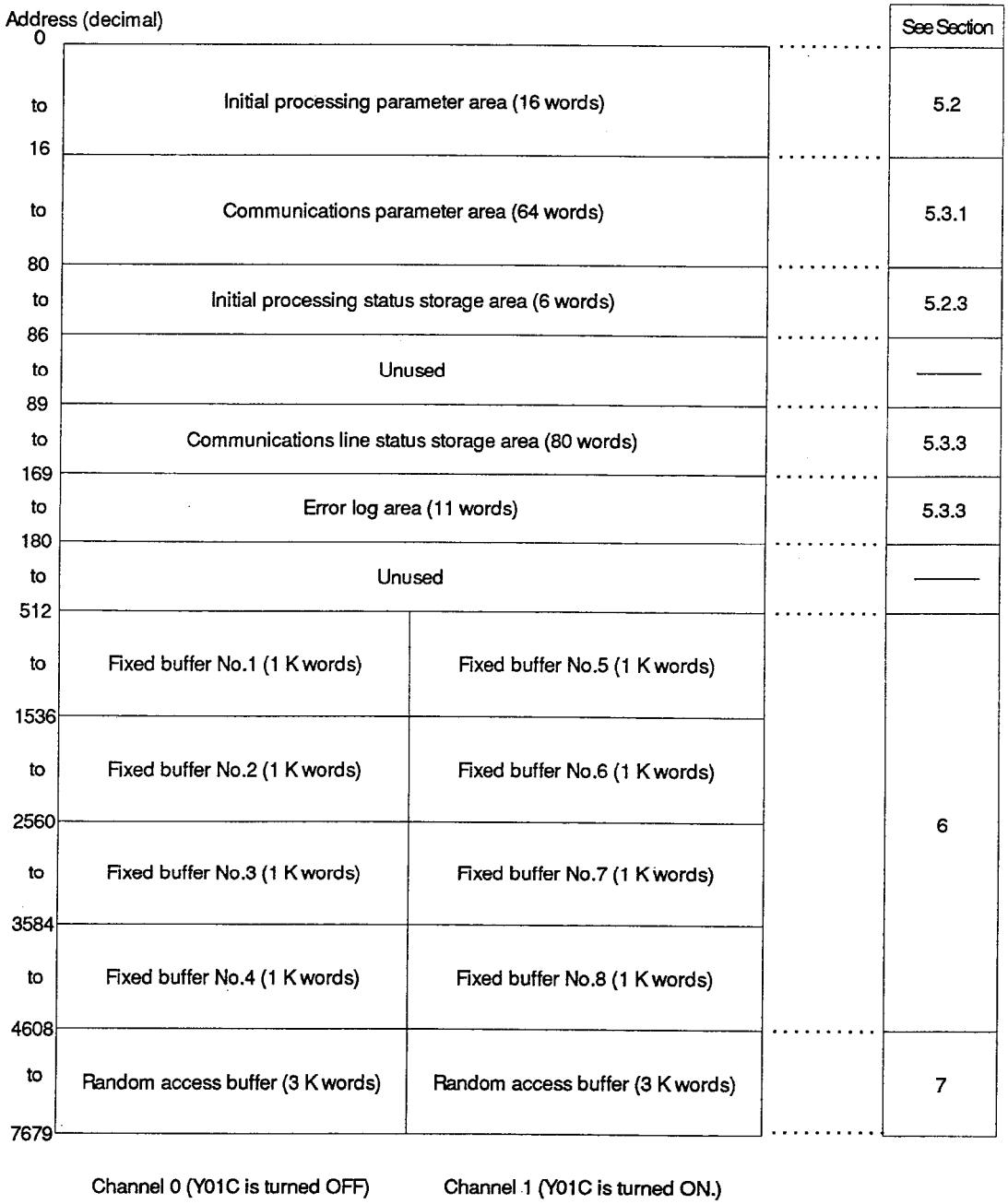
OFF : Channel 0 becomes valid.

ON : Channel 1 becomes valid.

3.6 Buffer Memory Map

The following explains the A1SJ71E71 buffer memory used for data communications with a PC CPU.

As shown in the figure below, the memory map of A1SJ71E71 is composed of an initial processing parameter area, a communications parameter area, a communications state storage area, an Ethernet address setting area, fixed buffer areas and random access buffer areas.



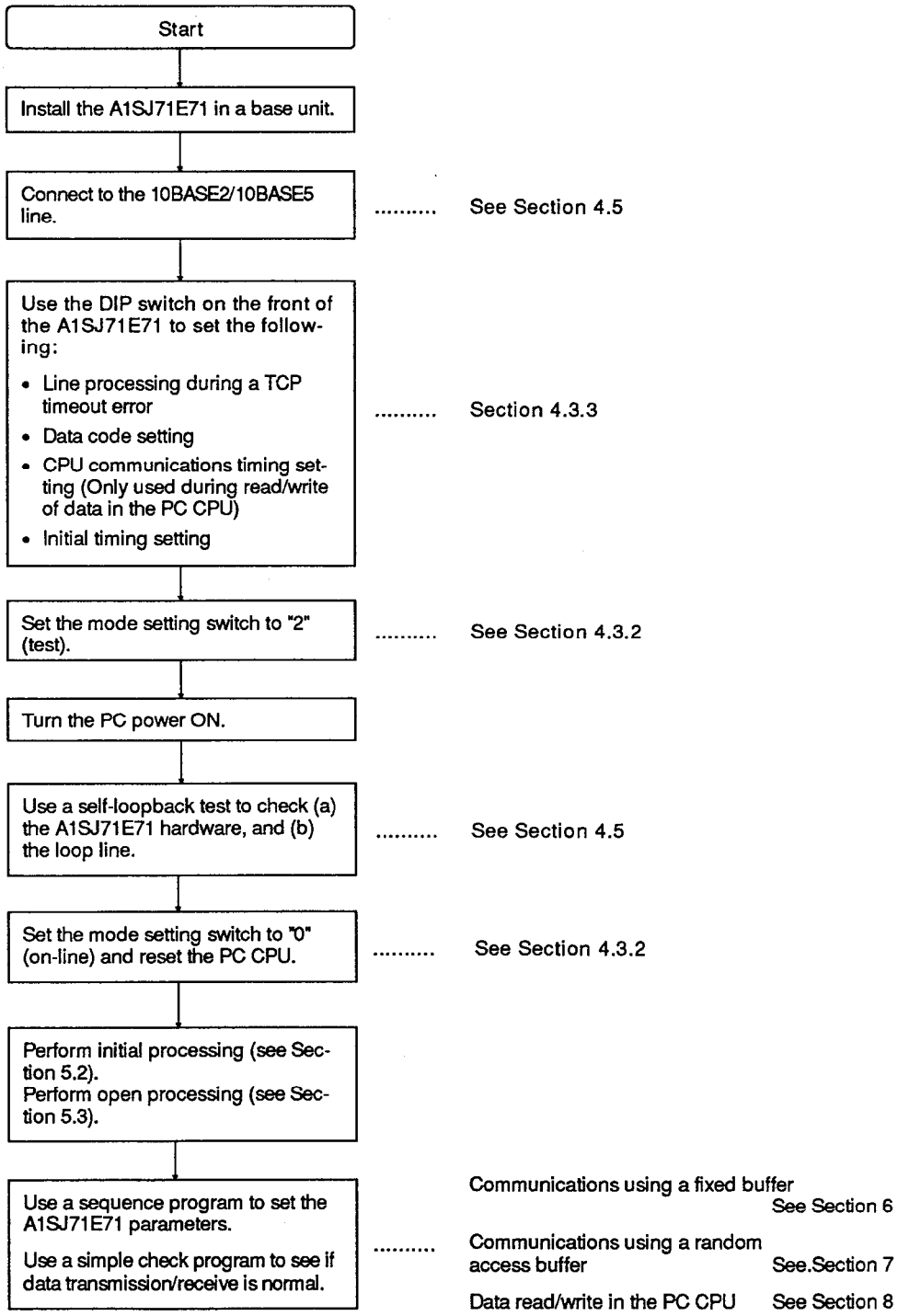
POINT

Execute buffer memory read/write operations only when necessary, by pulse generation (FROMP, TOP, etc.).
If read/write operations are executed continually, the data communication time may be made longer.

4. PRE-OPERATION SETTINGS

This section explains the pre-operation settings of the A1SJ71E71.

4.1 Pre-Operation Settings



4.2 Handling Instructions

(1) Protect the A1SJ71E71 and its terminal block from impact.

(2) Do not touch or remove the printed circuit board from the case.

(3) Do not allow metal particles or wire offcuts to enter the A1SJ71E71.

(4) Tighten the module mounting and terminal screws as specified below.

Screw	Tightening Torque kg.cm
Power supply cable connection terminal screw (M4)	10 to 14
Module mounting screws (optional) (M4)	8 to 12

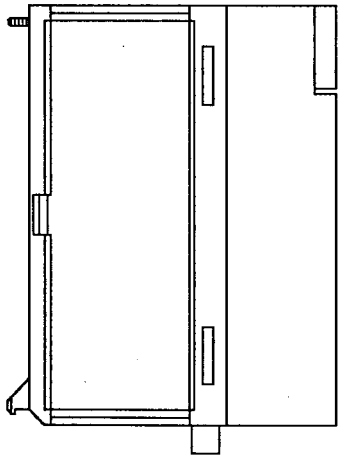
(5) Never install the system in the following environments:

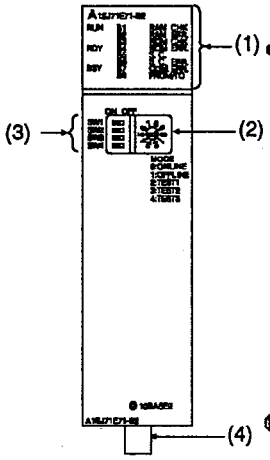
- Locations where ambient temperature is outside the range 0 to 55 °C (32 to 131 °F).
- Locations where ambient humidity is outside the range of 10 to 90% RH.
- Locations where dew condensation takes place due to sudden temperature changes.
- Locations where there are corrosive gasses and combustible gasses.
- Locations where there is a high level of conductive powder, such as dust and iron filings, oil mist, salt, and organic solvent.
- Locations exposed to the direct rays of the sun.
- Locations where strong power and magnetic fields are generated.
- Locations where vibration and shock are directly transmitted to the main unit.

4.3 Nomenclature

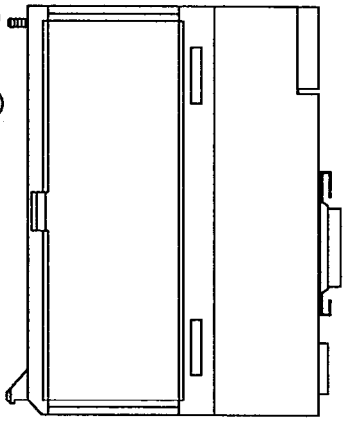
A1SJ71E71-B2/B5 nomenclature and settings are explained below.

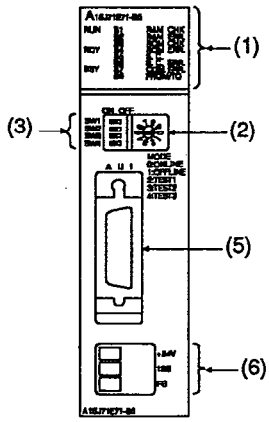
A1SJ71E71-B2





A1SJ71E71-B5





Number	Name	Description	See Section
(1)	Display LED	Indicates operating states, data communications displays, and error descriptions Each LED reflects various aspects of the A1SJ71E71 operation, so ON/OFF states cannot be directly referred to in the same meaning. The ON/OFF state has different meanings for different LEDs.	4.3.1
(2)	Mode setting switch	This selects the operating mode for online, offline, and self-diagnostic tests. It is factory-set at "0" (online).	4.3.2
(3)	Communications state setting switch	This sets the conditions for starting, communications processing, communications code classifications, and TCP timeout error processing. SW1 to SW4 are factory-set at OFF.	4.3.3
(4)	10BASE2 (Cheapernet) connector	This connects an A1SJ71E71-B2 to a 10BASE2 (Cheapernet).	2.1.2
(5)	AUI (transceiver cable) connector	This connects an A1SJ71E71-B5 to an AUI (transceiver cable).	2.1.2
(6)	External power supply terminal	When using an A1SJ71E71-B5, the power to the transceiver is supplied here. Length of bared wire: 13 mm Applicable wire size: 0.5 to 2 mm^2	2.1.2

4.3.1 LED signal names and indicator descriptions

The following table gives the signal names and indicator descriptions of the display LEDs on the upper front side of an A1SJ71E71-B2/B5.

Table 4.1 LED Indicator Description List

LED Location Chart	LED Name	LED Indicator Description	LED ON	LED OFF
RUN B1 RAM CHK B2 RAM ERR B3 ROM CHK RDY B4 ROM ERR B5 S.C. B6 S.C. ERR BSY B7 COM. ERR B8 FROM/TO	RUN	Normal operation	Normal	Error
	RDY	Communications preparation completed	This goes ON at the beginning of an online operation.	
	BSY	Communications processing is being executed	This goes ON during communications with a node.	
	B1	Connection status of a No. 1 connection	Open-completed	Closed
	B2	Connection status of a No. 2 connection		
	B3	Connection status of a No. 3 connection		
	B4	Connection status of a No. 4 connection		
	B5	Connection status of a No. 5 connection		
	B6	Connection status of a No. 6 connection		
	B7	Connection status of a No. 7 connection		
	B8	Connection status of a No. 8 connection		
	RAM. CHK	RAM test is being executed	This goes ON during a RAM test.	
	RAM. ERR	RAM error detection	RAM error	Normal
	ROM. CHK	ROM test is being executed	This goes ON during a ROM test	
	ROM. ERR	ROM error detection	ROM error	Normal
	S.C.	Self-loopback test is being executed	ON during a self-loopback test	
	S.C. ERR	Self test error detection	Self-loopback error	Normal
	COM. ERR	Communications error detection	Communications error	Normal
	FROM/TO	Data is being read/written	ON while a FROM/TO instruction is being executed.	

(1) If a WDT error causes the RUN LED to go OFF after power is turned ON, the WDT error detection signal (X1F) goes ON.

(2) The RDY LED goes ON just after the beginning of an online mode operation. (The mode setting switch is set at 0.) When the initial timing setting (SW8) of a communications condition setting switch is in the normal mode, this LED goes ON about 20 seconds after an online mode operation is started.

(3) As indicated above, BSY "communications processing is being executed" can be defined in the following ways:

(a) Time until timeout or until receiving a response after transmitting a command

(b) Time until timeout or until transmitting a response after receiving a command

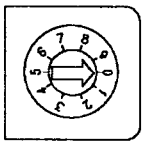
(4) The connection status of the B1 to B8 LEDs means the connection status of a line with a node set by a communications parameter.

The ON/OFF connection states of the open-completed signals (X10 to X17) can be confirmed by these LEDs.

After which, only open-completed connections can perform data communications.

4.3.2 Operating mode settings

Table 4.2 Description List of Operating Mode Settings



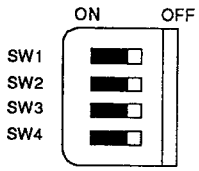
Setting Number	Setting Name	Setting Description
0	Online	Communications with a node is done in the usual operating mode.
1	Offline	The A1SJ71E71-B2/B5 is disconnected from the network.
2	Test 1	A self-diagnostic test is done using a self-loopback test.
3	Test 2	A RAM test is done.
4	Test 3	A ROM test is done.
5 to 9	Unused	

POINT

If the operating mode is changed, switch the mode setting switch.
Then, reset the PC CPU.
When it is reset, an operating mode selection is started.

4.3.3 Communications condition settings

Table 4.3 Description List of Communications Condition Settings



Switch	Setting Item	Setting Description	
SW1	Line processing selection during a TCP timeout error	If a TCP timeout error occurs, line processing is selected:	
		OFF	The line is closed by the TCP timeout error.
		ON	Even a the TCP timeout error occurs, the line is not closed.
SW2	Data code setting	A code classification for data communications with a node is selected:	
		OFF	Communications is done in binary code.
		ON	Communications is done in ASCII code.
SW3	CPU communication s timing setting	When a PC CPU is in the RUN state, data write enable/disable from a node is selected:	
		OFF	When the PC CPU is in the RUN state, a write operation from the node is disabled.
		ON	When the PC CPU is in the RUN state, a write operation from the node is enabled.
SW4	Initial timing setting	Timing to start initial processing is selected.	
		OFF	Quick start (starting without a delay time.) When communications is made by a single network, this is set.
		ON	Normal start (starting 20 seconds after a delay time.) When composed of several networks, this is set.

(1) Line processing selection by a TCP timeout error

If the TCP protocol is used (even if specified retry processing is done) and an ACK is not sent, a TCP timeout error occurs.
At this time, the connection processing is selected.

(2) Data code setting

When data communications is done with a node, a data code classification (ASCII or binary) is selected.

(3) Initial timing setting

When the TCP/IP protocol is used, the closed connection is frozen for about 20 seconds.
When a system starts, the time needed for startup is automatically set. The time between the initial processing request signal (Y19) coming ON and the initial-completed signal (X19) coming ON changes in accordance with this setting.

POINT

Make sure the power supply to the A1SJ71E71-B2/B5 is OFF when the communications condition setting switch is set.

4.4 Connecting to the Network

The method for connecting an A1SJ71E71-B2/B5 to a10BASE2 (Cheapernet) or 10BASE5 (Ethernet) is given below.

4.4.1 Connecting to a 10BASE2 (Cheapernet)

The following explains how to connect an A1SJ71E71 to a 10BASE2 (Cheapernet) network.

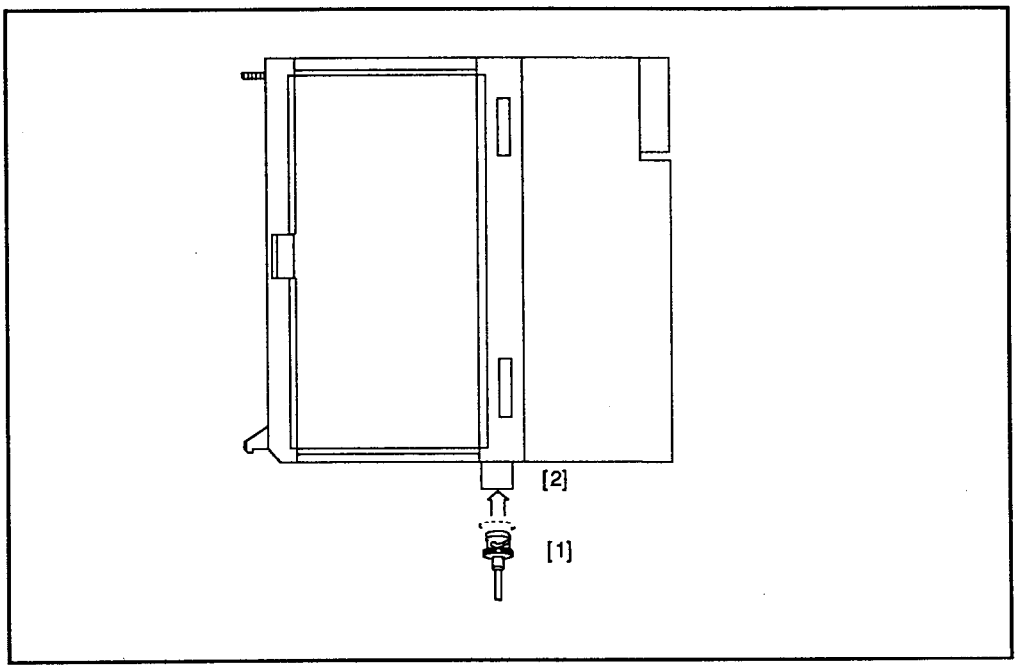


Fig. 4.1 Connecting a 10BASE2 (Cheapernet) Coaxial Cable

How to connect a Cheapernet coaxial cable

(1) Make sure the connector [1] and the slot [2] line up. Then, push the connector in.

(2) While pushing the connector, rotate it 1/4 turn to the right.

(3) Keep rotating the connector until it locks in place.

(4) Make sure the connector is securely fixed.

POINT

The coaxial cable has a constant allowable bend radius.
When connecting a 10BASE2(Cheapernet) coaxial cable, a space that is larger than the allowable bend radius of a coaxial cable is needed with an A1SJ71E71.
Find out the allowable bend radius of the coaxial cable from the manufacturer.

IMPORTANT

When connecting transceiver cables, make sure that the cables are 50 mm or more from both the power line and the large current main ladder; otherwise there will be a malfunction.

4.4.2 Connection to a 10BASE5 (Ethernet)

The following explains how to connect an A1SJ71E71 to a 10BASE5 (Ethernet) network.

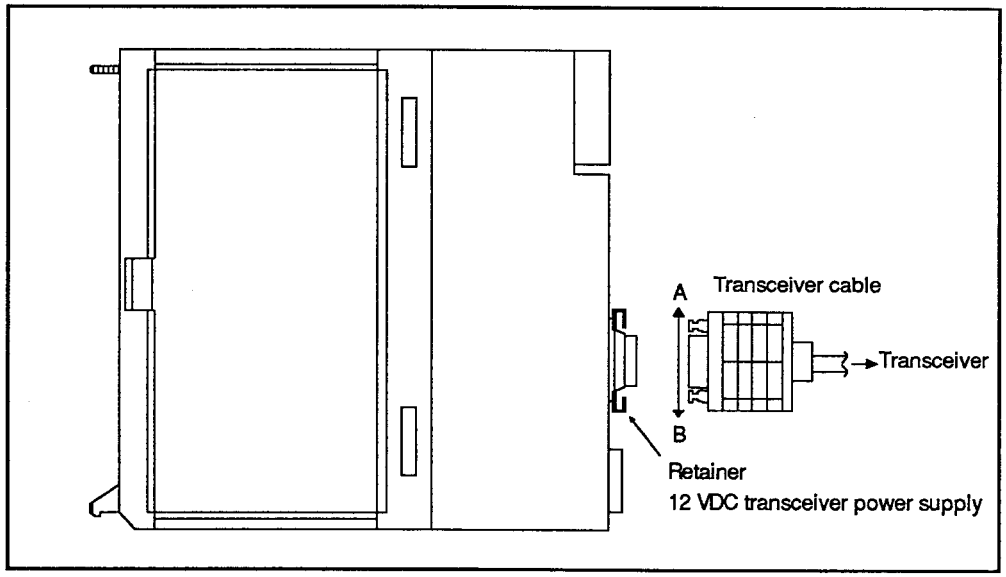


Fig. 4.2 Connecting a Transceiver Cable

How to connect a transceiver cable

(1) Slide the retainer towards "A" in the figure.

(2) Insert the connector of a transceiver cable connector so that the cable is fully secured by the retainer.

(3) Slide the retainer towards "B" in the figure.

(4) Make sure the transceiver cable is securely locked in place.

(5) Input power to the transceiver.
(Do not connect the cable while the power supply is ON.)

IMPORTANT

When connecting transceiver cables, make sure that the cables are 50 mm or more from both the power line and the large current main ladder; otherwise there will be a malfunction.

REMARK

Consult a specialist about terminal processing of an Ethernet cable and connecting a trunk line cable.

REMARK

Connection of a coaxial cable connector

The following explains how to connect a BNC connector (coaxial cable connector plug) to a cable.

(1) Configuration of a BNC connector and a coaxial cable

Figure 4.3 shows the configurations of a BNC connector and a coaxial cable.

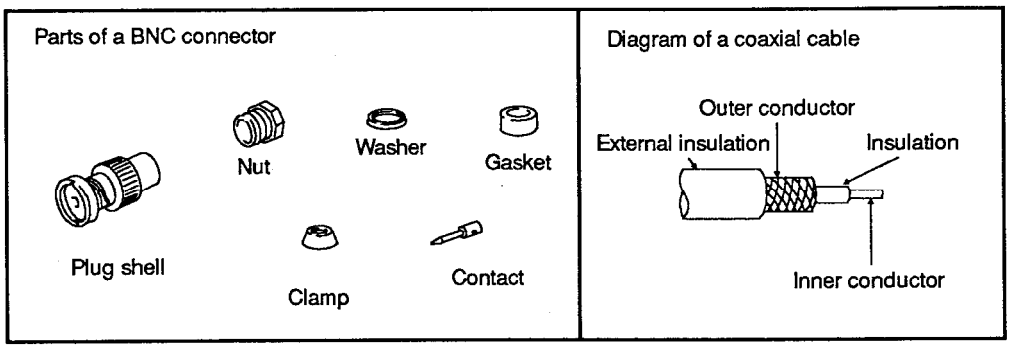
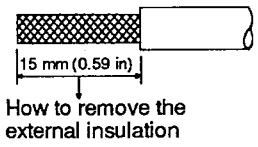


Fig. 4.3 Configurations of a BNC Connector and a Coaxial Cable

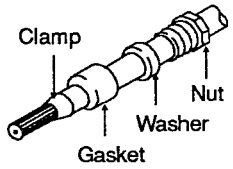
(2) How to connect a BNC connector to a coaxial cable

The following describes how to connect a BNC connector and a coaxial cable.

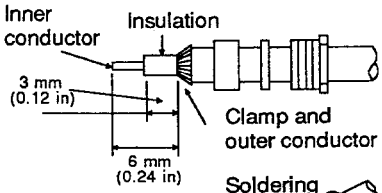
(a) Remove the external insulation of the coaxial cable as shown on the right. Take care not to damage the outer conductor.



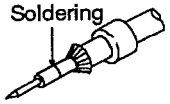
(b) As shown to the right, put the nut, washer, gasket, and clamp on the coaxial cable. Then, loosen the outer conductor.



(c) Cut the outer conductor, the insulator, and the inner conductor as shown on the right. However, cut the outer conductor just enough to cover the tapering part of the clamp. Then, cover the clamp with it.

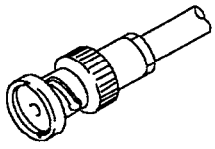


(d) Solder the contact to the inner conductor.



(e) Insert the contact assembly of (d) into the plug shell.

Finally, put the nut on the plug shell.



POINT

When soldering the inner conductor to the contact, pay close attention to the following:

(1) When soldering, make sure the solder does not swell.

(2) There should be no space between the contact and the cable insulation. Also, make sure the contact does not cut into the insulation.

(3) Do the soldering quickly so that an insulation is not altered in any way.

4.5 Self-Diagnostic Tests

4.5.1 Self-loopback test

The following explains the self-loopback test for checking the hardware containing the communications ladder of an A1SJ71E71.

The A1SJ71E71 transmits a test message to itself in the self-loopback test and receives this test message through the network.

The A1SJ71E71 then examines whether or not the received test message is the same as a transmitted test message.

The self-loopback test (which takes about five seconds) is explained below.

How to do a self-loopback test

1) Connect the A1SJ71E71 to 10BASEB2 or 10BASEB5 line.
2) Set the operating mode setting rotary switch on the front of the A1SJ71E71 at "2".
3) Set the RUN/STOP keyswitch of the PC CPU at STOP.
4) Reset the PC CPU. Then, start the self-loopback test.
   Make sure the S.C. LED goes ON.

Test results

1) When the S.C. LED goes OFF, the self-loopback test is completed.
2) Confirm the test result with the S.C.ERR LED.
   Normal.....The S.C.ERR LED is OFF.
   Faulty.....The S.C.ERR LED is ON.
3) The fault cause is one of the following:
   - Faulty A1SJ71E71 hardware
   - Faulty 10BASE2 or 10BASE5 line
   - Faulty 12 VDC external power supply (when testing a 10BASE5)

Post-test operation

Switch the operating mode setting rotary switch on the front of the A1SJ71E71 to the online mode or another test mode. Then, reset the PC CPU.

POINT

Even if a self-loopback test is done, if a node is online, the problem is not in the hardware.
Also, if a packet is interfered with in a line (because it collides with other packets), this test will not be completed within five-second time span.
After stopping data communications between nodes, do a self-loopback test.

4.5.2 RAM test

The following explains the RAM test for checking the RAM memory of an A1SJ71E71.

How to do a RAM test

1) Set the operating mode setting rotary switch on the front of the A1SJ71E71 at "3".
2) Set the RUN/STOP keyswitch of the PC CPU at STOP.
3) Reset the PC CPU. Then, start the RAM test.
   Make sure the RAM CHK LED goes ON.

Test results

1) When the RAM CHK LED goes OFF, the RAM test is completed.
2) Confirm the test result with the RAM ERR LED.
   Normal.....The RAM ERR LED is OFF.
   Error.....The RAM ERR LED is ON.

Post-test operation

Switch the operating mode setting rotary switch on the front of the A1SJ71E71 to the online mode or another test mode. Then, reset the PC CPU.

4.5.3 ROM test

The following explains the ROM test for checking the ROM memory of an A1SJ71E71.

How to do a ROM test

1) Set the operating mode setting rotary switch of the front of the A1SJ71E71 at "4".
2) Set the RUN/STOP keyswitch of the PC CPU at STOP.
3) Reset the PC CPU. Then, start the ROM test.
   Make sure the ROM CHK LED goes ON.

Test results

1) When the ROM CHK LED goes OFF, the RAM test is completed.
2) Confirm the test result with the ROM ERR LED.
   Normal.....The ROM ERR LED goes OFF.
   Erro.....The ROM ERR LED goes ON.

Post-test operation

Switch the operating mode setting rotary switch on the front of the A1SJ71E71 to the online mode or another test mode. Then, reset the PC CPU.

5. COMMUNICATING WITH OTHER NODES

5.1 Communicating with Other Nodes

To start communications between an A1SJ71E71 and any other node, initial processing and open processing need to be executed to establish a valid connection between the two. Communications are possible only between these nodes which have an established connection between them.

To end communications, close processing and end processing are executed. The connection between nodes is made invalid and all communication processings end.

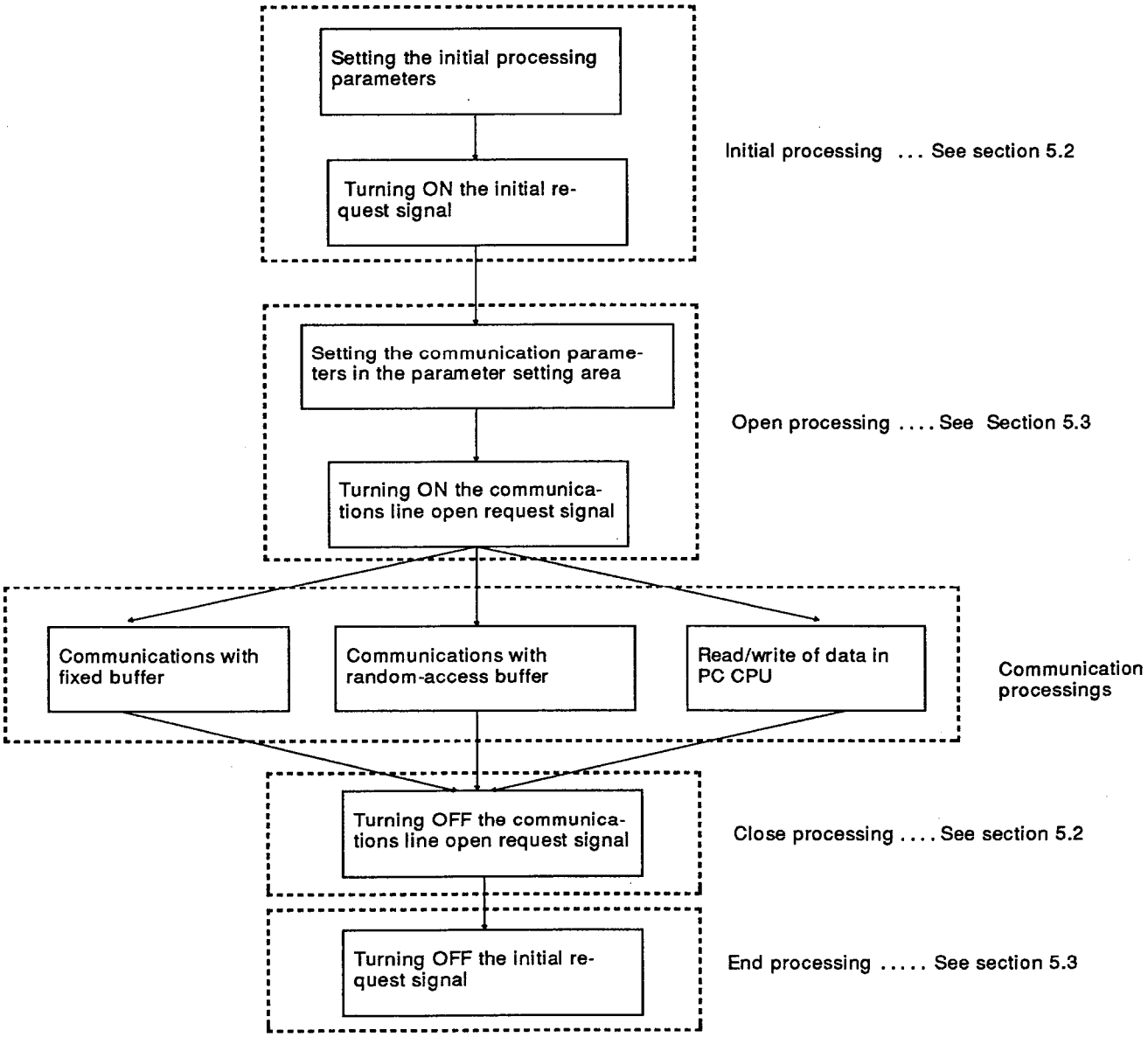


Fig. 5.1 Communicating with other nodes

POINTS

(1) Open processing is required to open a valid communications line to a communicating node to perform communications with fixed buffer or random-access buffer or to perform read/write of data in the PC CPU. These three kinds of communications can be performed between nodes which are connected with an open communications line.

(2) Open processing can be performed for up to eight nodes.
However, when fixed buffer is used for both send and receive communications with one node, two buffer areas are needed. The number of communicating nodes accordingly decreases.
Initial processing must be completed before starting open processing.

(3) When the ACPU is set to the STOP status, the A1SJ71E71 open request signal (Y8 to F) and initial request signal (Y19) go OFF, and lines to other nodes are closed.
Arrange for initial processing and open processing to be executed again when the ACPU is switched from STOP to RUN.

5.2 Initial Processing

This section explains the initial processing of an A1SJ71E71.

5.2.1 Data for initial setting

The parameter setting area (buffer addresses 0 to 15) for doing initial processing is shown below.

Set a value determined by the network manager (network planner, the IP address manager, etc.) here.

Buffer Memory Address	Setting Description
0	IP address of the A1SJ71E71
1	
2	Unused
3	
4	
5	
6	
7	
8	
9	
10	TCP and ULP timeout value
11	TCP zero window timer value
12	TCP retransmission timer value
13	TCP completed timer value
14	IP assembly timer value
15	Response watching timer value

(1) IP address of the A1SJ71E71 (2 words)

The IP address of A1SJ71E71 is set.

(2) TCP and ULP timeout value (default = 15; setting time = set value x 2 seconds)

This timeout value sets the packet lifetime of the TCP.
This is the timer that is transferred in the parameter form during the opening of a TCP and during data transmission.

(3) The TCP zero window timer value (default = 5; setting time = set value × 2 seconds)

When the send window size of a TCP becomes 0, a send window confirmation packet is retransmitted. This timer value sets the time.

(4) The TCP retransmission timer value (default = 5; setting time = set value × 2 seconds)

When ACK is not sent back, even if open data of TCP is transmitted, data is retransmitted. This timer value sets the time.

(5) TCP completed timer value (default = 10; setting time = set value × 2 seconds)

When the self closes the connection of TCP, a counterpart station is closed. This value sets the time needed for the close processing of a counterpart station.
In the case of software package "H" and later versions, if it has not been possible to close the connection within the time set by the TCP completed timer value, RST processing is performed at the communicating node to forcibly execute close processing.

(6) IP assembly timer value (default = 16; setting time = set value × 2 seconds)

When data split by the IP is received and the A1SJ71E71 waits for the next split data. This timer value sets this wait time.

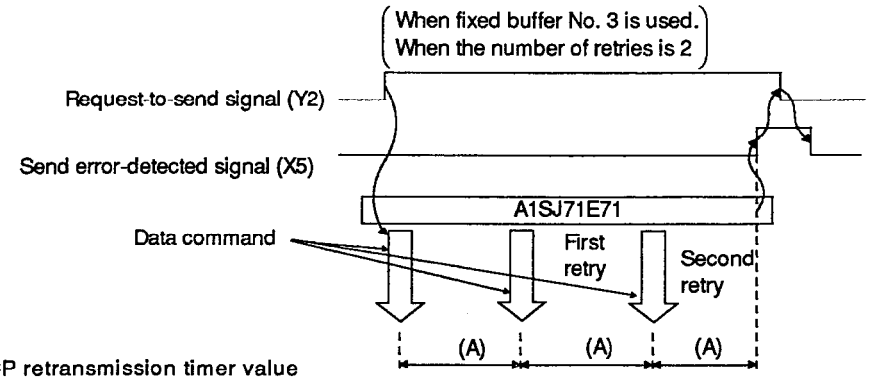
(7) Response watchdog timer value (default = 15; setting time = set value × 2 seconds)

When a command is transmitted, a response is sent back. This timer value sets the wait time.

REMARKS

(1) If communication errors occur due to noise, etc., change the settings so that a greater number of retries are executed.
The number of retries is determined using the following formula.

$$\text{Number of retries} = \frac{\text{TCP ULP timeout value}}{\text{TCP retransmission timer value}}$$



(A): TCP retransmission timer value
After data is transmitted, if an ACK is not sent back, the data will be retransmitted. The retransmission timer value sets this retransmission time.

(2) The only data that needs to be set in initial setting, provided there is no problem, is the IP address: the other data can be left as the default values.

5.2.2 Initial processing procedures

This section explains the initial processing procedures used with an A1SJ71E71.

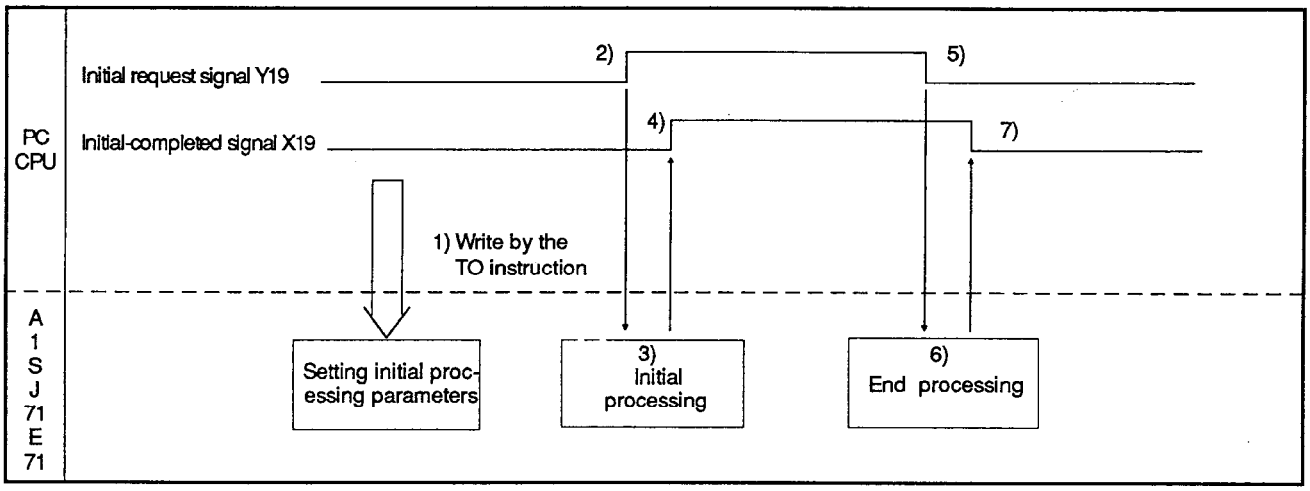


Fig. 5.2 Initial Processing

1) The initial processing parameters are written by the T0 instruction of a sequence program.

2) The initial processing request signal (Y19) is turned ON.

3) Initial processing for the A1SJ71E71 is performed.

4) When initial processing is completed, the initial-completed signal (X19) turns ON.

   When an error is detected during initial processing, the initial error-detected signal (X1A) turns ON.
   Check the initial error code (buffer address 80) and retry initial processing.

5) The initial processing request signal (Y19) is turned OFF.

   This signal is turned OFF when the initial error-detected signal (X1A) is turned ON or when initial processing is discontinued.

6) Initial processing for the A1SJ71E71 is completed.

7) When initial processing is completed, the initial-completed signal (X19) or, when an error is detected during initial processing, the initial error-detected signal (X1A) is turned OFF.
   All open lines are closed. (Refer to 5.3.2)

5.2.3 Initial processing state storage area

This section deals with the area (buffer addresses 80 to 88) where the initial processing state of the A1SJ71E71 is stored.

(1) Initial error code

(a) An occurring error code is stored when initial processing is executed.

(b) Section 9.1.1 gives details about the initial processing error code.

(c) The error code is stored as a binary value.

(d) The error code will be cleared in the following cases:

1) When the PC PCU is reset, or PC power is turned OFF

2) When writing "0" in an initial error code (address 80 of a buffer memory) using a sequence program

Word Address	Communications Status
80	Initial error code
81	A1SJ71E71 address setting monitoring (L) to (H)
82	
83	A1SJ71E71 Ethernet address setting monitoring (L) to (H)
84	
85	

(2) A1SJ71E71 IP address storage

(a) The A1SJ71E71 address set when initial processing is executed is stored.

(b) The A1SJ71E71 address is stored as a binary value.

Example: The following shows the data storage state; the IP address is A20009C0H (162.0.9.192).

Address	Buffer memory
81	09C0H
82	A200H

(3) A1SJ71E71 Ethernet address storage

(a) The physical address of an A1SJ71E71 is read from ROM and is stored in this area.

Because the physical address of the Ethernet is written in ROM, it cannot be changed.

(b) The A1SJ71E71 Ethernet address is stored as a binary value.

5.2.4 Sample initial processing program

This section shows the sequence program for the initial processing of an A1SJ71E71.

Example: The A1SJ71E71 is installed in the "0" slot of a main base.

The initial processing parameters are as follows:

(a) The IP address of the A1SJ71E71 is "A20009C0$_H$ (162.0.9.192)".

(b) The TCP ULP timeout value is a default "15" (15 × 2= 30 seconds).

(c) The TCP zero window value is a default "5" (5 × 2= 10 seconds).

(d) The TCP retransmission timer value is a default "5" (5 × 2= 10 seconds).

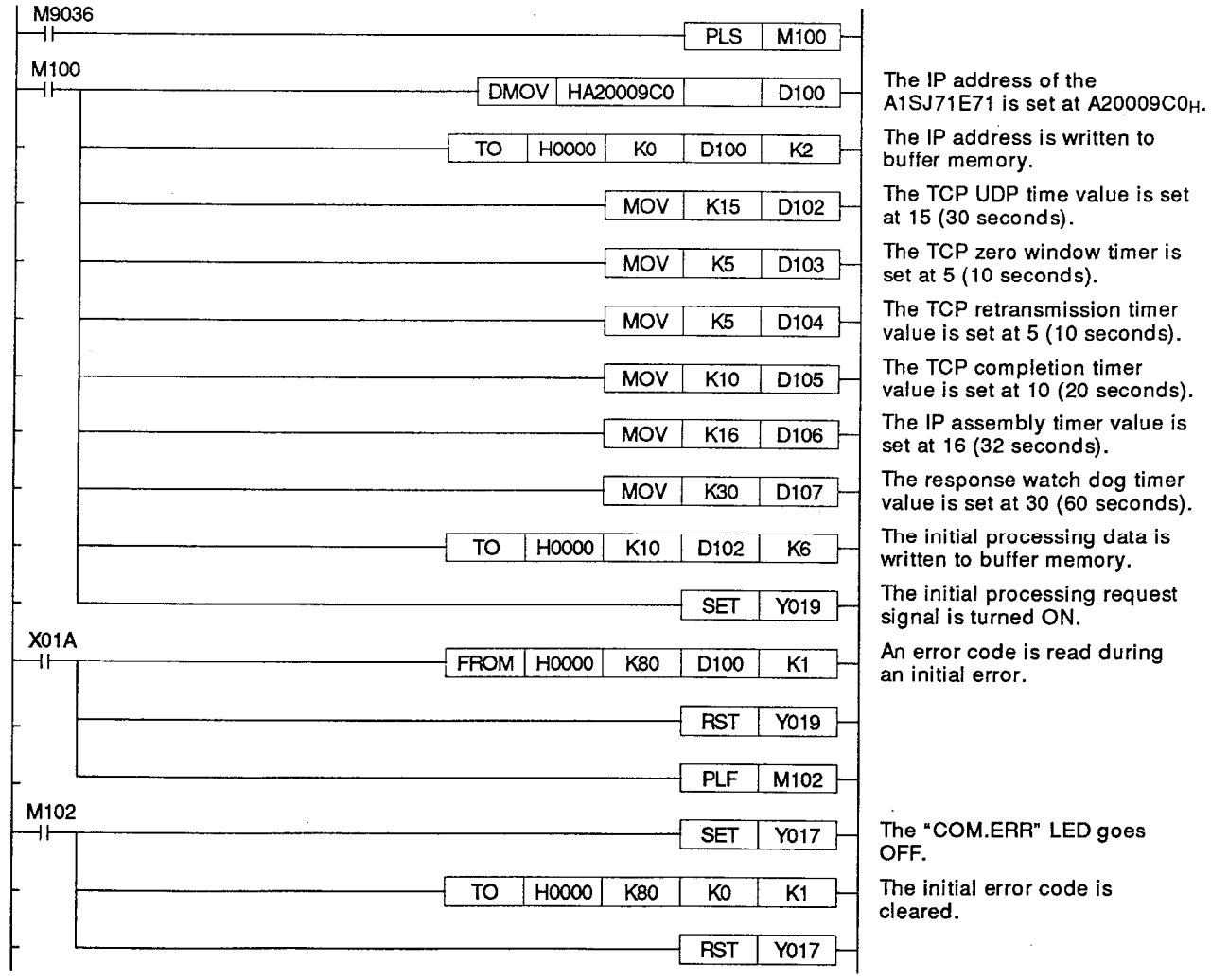
As a result, the retry count is 15/5 = 3 times.

(e) The TCP completion timer value is a default "10" (10 × 2= 20 seconds).

(f) The IP assembly timer value is a default "16" (16 × 2= 32 seconds).

(g) The response watch dog timer value is a default "30" (30× 2= 60 seconds).

Initial processing program



5.3 Open/Close of a Communications Line

Communications between an A1SJ71E71 and a maximum of eight nodes are enabled.

The A1SJ71E71 can allow fixed buffer communications, random access buffer communications, and data read/write communications in the PC CPU with a node to which a communications line has been opened.

Therefore, even if only data read/write in the random access buffer communications and the PC CPU is performed, open processing must be done.

5.3.1 Data for opening a communications line

This section shows the communications parameter setting area (buffer addresses 16 to 79) to use for open processing of a communications line.

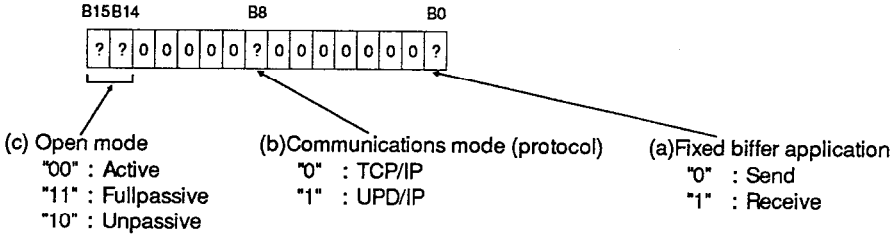
Buffer Memory Address	Setting Description	
16	Connection No. 1	Application setting area
17	Connection No. 2	
18	Connection No. 3	
19	Connection No. 4	
20	Connection No. 5	
21	Connection No. 6	
22	Connection No. 7	
23	Connection No. 8	
24	A1SJ71E71 port number	Connection No.1 communications address setting area
25	Node IP address	
26		
27	Node port number	
28	*Node Ethernet address (L)	
29	to	
30	(H)	
31	A1SJ71E71 port number	Connection No.2 communications address setting area
32	Node IP address	
33		
34	Node port number	
35	*Node Ethernet address (L)	
36	to	
37	(H)	
38	A1SJ71E71 port number	Connection No.3 communications address setting area
39	Node IP address	
40		
41	Node port number	
42	*Node Ethernet address (L)	
43	to	
44	(H)	
45	A1SJ71E71 port number	Connection No.4 communications address setting area
46	Node IP address	
47		
48	Node port number	
49	*Node Ethernet address (L)	
50	to	
51	(H)	

Buffer Memory Address	Setting Description	
52	A1SJ71E71 port number	Connection No.5 communications address setting area
53	Node IP address	
54		
55	Node port number	
56	*Node Ethernet address (L)	
57	to	
58	(H)	
59	A1SJ71E71 port number	Connection No.6 communications address setting area
60	Node IP address	
61		
62	Node port number	
63	*Node Ethernet address (L)	
64	to	
65	(H)	
66	A1SJ71E71 port number	Connection No.7 communications address setting area
67	Node IP address	
68		
69	Node port number	
70	*Node Ethernet address (L)	
71	to	
72	(H)	
73	A1SJ71E71 port number	Connection No.8 communications address setting area
74	Node IP address	
75		
76	Node port number	
77	*Node Ethernet address (L)	
78	to	
79	(H)	

* When the node to be connected has an ARP function (broadcast), set a default (value = FFFFFFFFFFFF$_H$).

(1) This section shows the application setting area (buffer addresses 16 to 23) for a communications parameter.

Set a condition of the communications of each connection from No. 1 to No. 8 at one-word data as bit information. One-word data is described below.



(a) Setting a fixed buffer application for each connection

Set either send or receive at a fixed buffer application for each connection.
Two fixed buffers (for send and receive) are required to transmit and receive to/from a specific node. Therefore, two connections must be set.
Even if the application of a fixed buffer is set as send or receive, it will be possible to read from and write to the random access buffer, and to read/write data in the PC CPU, from other nodes.

(b) Setting the protocol for each connection

Set the communications protocol for each connection at TCP/IP or UDP/IP.

(c) Open mode setting of each connection

This setting is valid only when the protocol is TCP/IP. (It is unnecessary with UDP/IP/.) Therefore, set it at "0".
When open processing with a UDP/IP, after completing open processing with a node to be opened by a fullpassive/unpassive open, do open processing with the node to be opened by an active open.

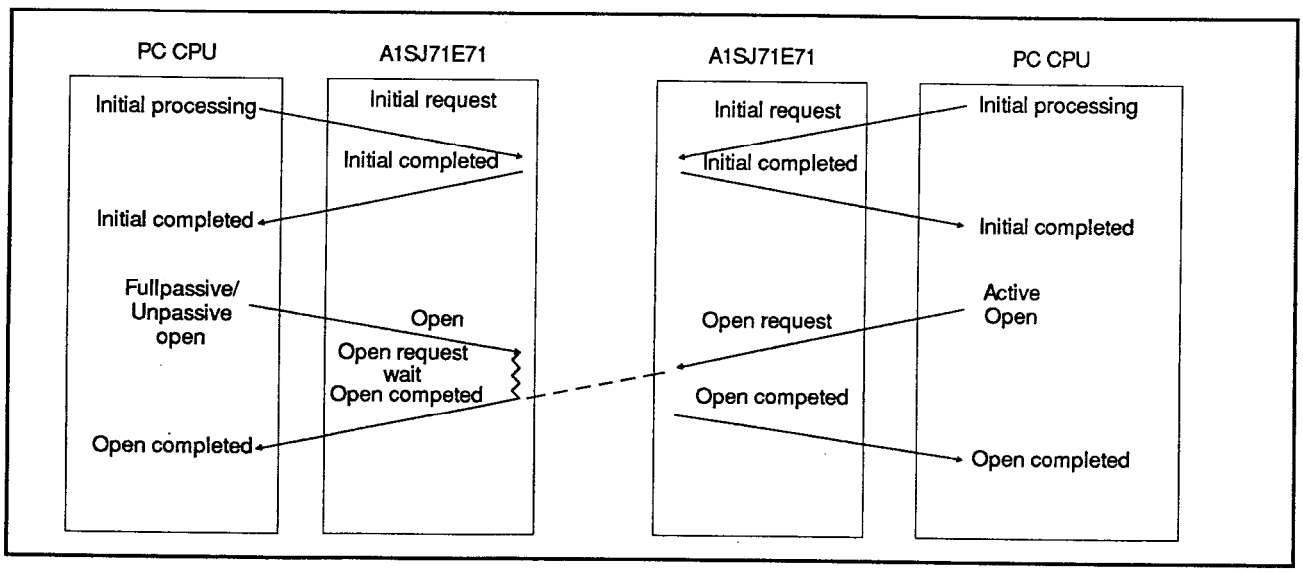


Fig. 5.3 TCP Open Processing Operations

1) Active open mode

Active open processing is done for the node in the open passive state.

2) Fullpassive open mode

Fullpassive open processing is done only for the specific node set at the communications address setting area.
Then, the A1SJ71E71 waits for an active open request from the node set at the communications address setting area.

3) Unpassive open mode

Unpassive open processing is done for all nodes connected to the network.
Then, the A1SJ71E71 waits for an active open request from all nodes in the network.

(d) Sample data setting of an application setting area

Table 5.1 Applications Setting Data

Protocol \ Application			Send	Receive
TCP	Active		0000H	0001H
	Passive	Fullpassive	C000H	C001H
		Unpassive	8000H	8001H
UDP			0100H	0101H

(2) This section shows the communications address setting area (buffer addresses 24 to 79) for a communications parameter.

Set the address and the port number of the node to be linked by each connection.
Also, set the value determined by the network manager here.

(a) A1SJ71E71 port number setting (Setting range is from 0100_H to $FFFF_H$.)

Set the port number of the A1SJ71E71 to be connected to a node.

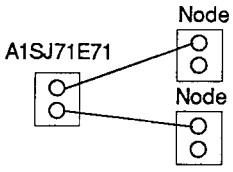
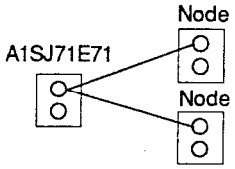
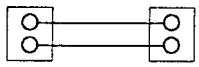
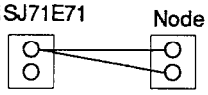
Connections	Connection Description	Communications Protocol	
		TCP	UDP
	In this protocol, when connected to more than one node, several A1SJ71E71 ports are set.	o	o
	In this protocol, when connected to more than one node, a single A1SJ71E71 port is set.	o	x
	In this protocol, when connected to more than one port in one node, several A1SJ71E71 ports are set.	o	o
	In this protocol, when connected to more than one port in one node, a single A1SJ71E71 port is set.	o	x
	Multiple connections cannot be done using a single node port and a single A1SJ71E71 port.	x	x

Fig. 5.4 Various Connections

(b) IP node address of a node

Set the IP address of a communicating node.

(c) Port number of a node

Set the port address of a communicating node.

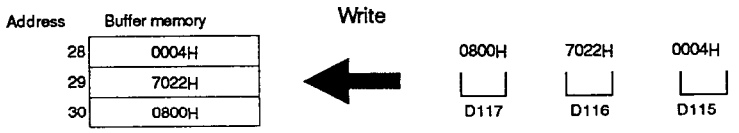
(d) Ethernet address of a node (Default = $FFFFFFFFFFFF_H$)

If a communicating node does not have an ARP (broadcast) function, set the Ethernet address to that node.

When a value is set, the node does not have any ARP function except a default value. Therefore, the A1SJ71E71 is accessed by the set Ethernet address.

Be sure to set a default (value = FFFFFFFFFFFF$_H$) when a node has the ARP function.

Example: If the Ethernet address is 080070220004$_H$, the data settings are as follows:



(e) Table 5.2 shows the relationship between the open mode and communications parameters.

Table 5.2 Relationship Between Open Mode and Communications Parameter Data Settings

			A1SJ71E71 Port Number	Node IP Address	Node Port Number	Ethernet Address
TCP	Active	Node with an ARP function	Setting needed	Setting needed	Setting needed	Default value (0)
		Node without an ARP function				Setting needed
	Passive	Unpassive	Setting needed	Setting not needed	Setting not needed	Setting not needed
		Fullpassive	Setting needed	Setting needed	Setting needed	Setting not needed
UDP		Node with an ARP function	Setting needed	Setting needed	Setting needed	Default value (0)
		Node with an ARP function				Setting needed

5.3.2 Open processing of communications line

(1) This section shows the open processing of an A1SJ71E71.

Initial processing must be completed before open processing can be done.

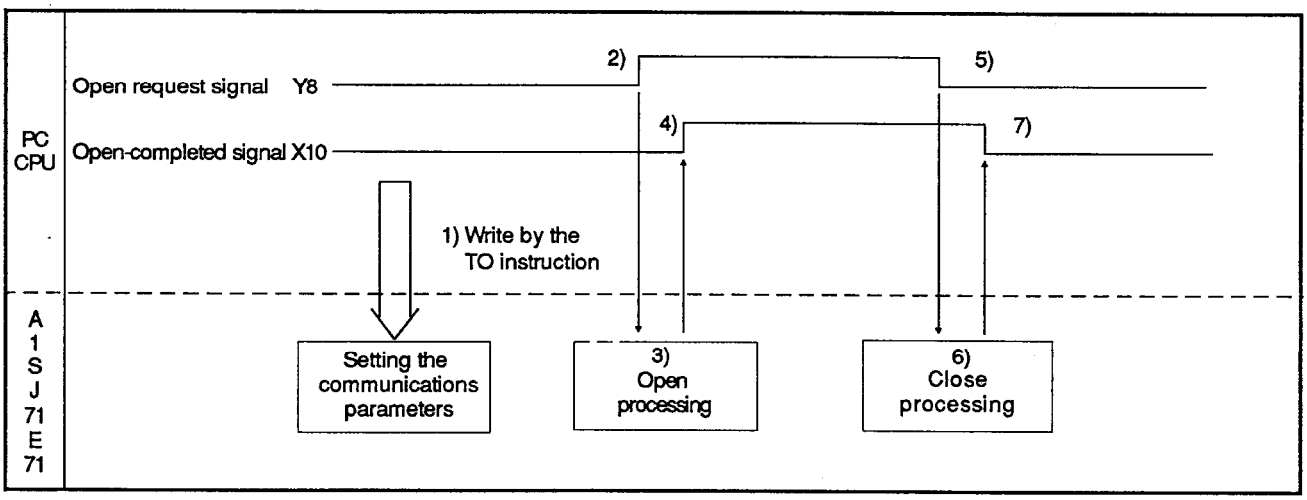


Fig. 5.5 Open Processing

1) Communications parameters are written to the parameter setting area by the TO instruction of a sequence program.
2) The open request signal (Y8 to YF) is turned ON by the sequence program.
3) The A1SJ71E71 executes open processing.
4) When open processing is completed, the A1SJ71E71 turns ON the open-completed signal (X10 to X17).
5) The open request signal (Y8 to YF) is turned OFF by the sequence program.
6) The A1SJ71E71 executes close processing.
7) When close processing is completed, the A1SJ71E71 turns OFF the open-completed signal (X10 to X17).

POINT

If it has not been possible to close the connection within the time set by the TCP completed timer value, RST processing is performed at the communicating node to forcibly execute close processing.

Except for close processing using a sequence program (open request signal OFF), the connection is automatically closed (open completed signal is turned OFF) in the following cases:
To reopen the connection, start open processing after turning the open request signal (Y8 to YF) OFF.

Table 5.3 Closed Communications Lines

Closed Processing	Cause of Closed Processing	Open Processing		
		Error Signal (X 18)	Completed Signal (X10 to X17)	Error Code (Address of Buffer Memory; 93, 103, 113, 123, 133, 143, 153, 163)
TCP ULP timeout error	When the TCP protocol is used, even if retry processing is done, ACK is not transmitted back (see Section 9.1.2).	ON	Changed by the ON/OFF DIP switch (see Section 4.3.3)	9059H
Response monitoring timeout error	The data set value in application data set by the user is larger than an actual data quantity (see Section 9.2).	ON	OFF	71H
Close request from a node	"CLOSE" or "ABORT" instruction sent from a node.	OFF	OFF	–

(2) The method for reopening a connection after closing it is explained here.

(a) To reopen a connection after close processing (switching the open request signal OFF) has been executed by the sequence program, switch the open request signal ON a minimum of 500 ms after the open completed signal has gone OFF (after close processing has been completed).
The open request signal and open completed signal come ON and go OFF in accordance with the timing as shown in figure 5.6.

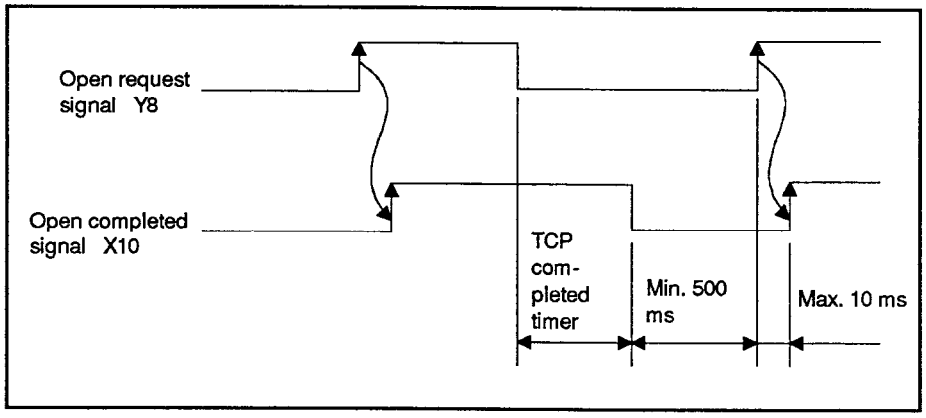


Fig. 5.6 Reopen Processing When Connection Closed by Sequence Program

(b) To reopen a connection that has been closed from the communicating node (see Table 5.3), first switch OFF the open request signal. Then, at least 500 ms after the open request signal has gone OFF, switch the open request signal ON.
The open request signal and open completed signal come ON and go OFF in accordance with the timing as shown in figure 5.7.

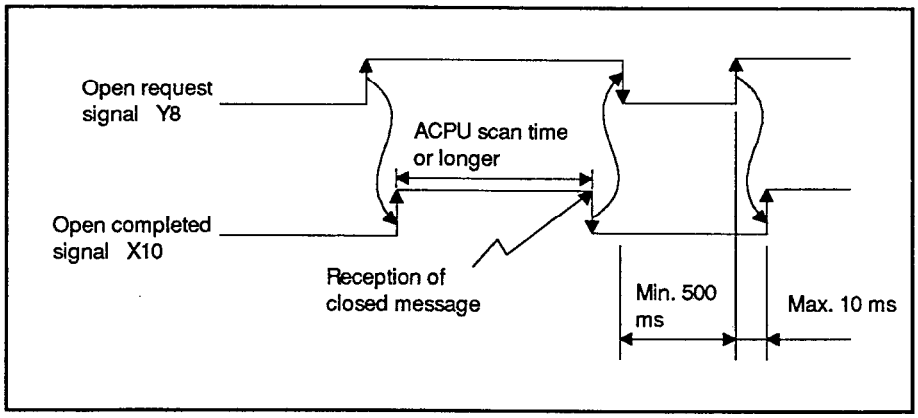


Fig. 5.7 Reopen Processing When Connection Closed From Communicating Node

POINT

In order to detect open completion with the sequence program, the ON time of the open completed signal (X10) must be at least as long as the ACPU scan time.
If the close message is received for a shorter time than the ACPU scan time after completion of open processing, it may not be possible to detect the open completed status with the sequence program.

5.3.3 Communications line status storage area

This section shows the A1SJ71E71 communications state storage area (buffer addresses 80 to 178).

This area is used for storing the communications status of an A1SJ71E71 port, the IP address of a communicating node, various error codes, the fixed buffer communications time, etc., of each communications state.

The communications status of different connections can be confirmed by reading this area.

Buffer Memory Address	Setting Description		
89	A1SJ71E71 port number		Connection No.1
90	Node IP address		
91			
92	Node port number		
93	Open error code		
94	Fixed buffer send error code		
95	Fixed buffer send completed code		
96	Fixed buffer communications time storage	Maximum	
97		Minimum	
98		Present	
99	A1SJ71E71 port number		Connection No.2
100	Node IP address		
101			
102	Node port number		
103	Open error code		
104	Fixed buffer send completed code		
105	Fixed buffer send error code		
106	Fixed buffer communications time storage	Maximum	
107		Minimum	
108		Present	
109	A1SJ71E71 port number		Connection No.3
110	Node IP address		
111			
112	Node port number		
113	Open error code		
114	Fixed buffer send error code		
115	Fixed buffer send completed code		
116	Fixed buffer communications time storage	Maximum	
117		Minimum	
118		Present	
119	A1SJ71E71 port number		Connection No.4
120	Node IP address		
121			
122	Node port number		
123	Open error code		
124	Fixed buffer send error code		
125	Fixed buffer send completed code		
126	Fixed buffer communications time storage	Maximum	
127		Minimum	
128		Present	

Buffer Memory Address	Setting Description		
129	A1SJ71E71 port number		Connection No.5
130	Node IP address		
131			
132	Node port number		
133	Open error code		
134	Fixed buffer send error code		
135	Fixed buffer send completed code		
136	Fixed buffer communications time storage	Maximum	
137		Minimum	
138		Present	
139	A1SJ71E71 port number		Connection No.6
140	Node IP address		
141			
142	Node port number		
143	Open error code		
144	Fixed buffer send error code		
145	Fixed buffer send completed code		
146	Fixed buffer communications time storage	Maximum	
147		Minimum	
148		Present	
149	A1SJ71E71 port number		Connection No.7
150	Node IP address		
151			
152	Node port number		
153	Open error code		
154	Fixed buffer send error code		
155	Fixed buffer send completed code		
156	Fixed buffer communications time storage	Maximum	
157		Minimum	
158		Present	
159	A1SJ71E71 port number		Connection No.8
160	Node IP address		
161			
162	Node port number		
163	Open error code		
164	Fixed buffer send error code		
165	Fixed buffer send completed code		
166	Fixed buffer communications time storage	Maximum	
167		Minimum	
168		Present	
169			
170			
171			
169			
170			
171	Error log area		
172	This area is for all areas not resulting from a		
173	fixed buffer send (ring buffer).		
174	For details on error codes, see Section 9.1.4.		
175			
176			
177			
178			
179			

(1) A1SJ71E71 port number (buffer addresses 89, 99, 109 through 159)

These store the port number of a set A1SJ71E71 during the open processing of each connection.

(2) Node IP address (buffer addresses 90, 100, 110 through 160)

These store the set node IP address during the open processing of each connection.

If the IP address is "A20009C0H", the storage data is shown below.

Address	Buffer memory storage data
90	09C0H
91	A200H

(3) Node port number (buffer addresses 92, 102, 112 through 162)

These store the set node port number during the open processing of each connection.

(4) Open error codes (buffer memory 93, 103, 113 through 163)

(a) These store the error codes that occur in the open processing of each connection (binary value).

(b) Section 9.1.2 gives details about open error codes.

(c) The error code is cleared in the following cases:

1) The connection where the open error occurred could be opened again, and it could be opened normally.

2) The PC CPU is reset, or PC power is turned OFF.

(5) Error codes during fixed buffer send (buffer addresses 94, 104, 114 through 164)

(a) These store the error codes that occur during the fixed buffer send of each connection (binary value).

(b) Section 9.1.3 gives details about fixed buffer send error codes.

(c) The error code is cleared in the following cases:

1) Turn OFF the fixed buffer request-to-send signal of the connection where the send error occurred.

2) The PC CPU is reset, or PC power is turned OFF.

(6) Response-completed codes during fixed buffer send (buffer addresses 95, 105, 115 through 165)

(a) These store the response-completed code to be sent back during the fixed buffer send of each connection (binary value)

(b) Section 9.1.3 gives details about response-completed codes.

(c) Response-completed code is cleared in the following cases: The PC CPU is reset, PC power is turned OFF.

(7) Storage of the communicating times of fixed buffer communications (buffer addresses 96, 106, 116 through 166)

(a) The maximum value, the minimum value and the present value of the processing time of fixed buffer communications are stored.

(b) The processing time is stored in 10 msec units (binary value).

(c) During processing time, the following values are stored:

1) Send processing time with a fixed buffer

The time after a request-to-send signal goes ON until the A1SJ71E71 completes a send.

2) Receive processing time with a fixed buffer

The time after a receive-completed signal goes ON until the A1SJ71E71 completes the reply processing.

(8) Error log area (buffer addresses 169 to 179)

(a) This is the area where A1SJ71E71 stores the errors (IP level error, receive data check sum error, etc.) that did not result from a fixed buffer send.

Any error that occurred while transmitting a fixed buffer is stored in the fixed buffer send error code area (buffer addresses 94, 103, and 112 through 164).

(b) This error area can store 11 words. This area also includes a ring buffer that can store information on up to 10 error cases.

This data area is initially set to 0000_H.

Therefore, it is possible to see if a relevant data is old or not.

(c) Usually, This area does not need to be read. However, it is necessary to read it during maintenance.

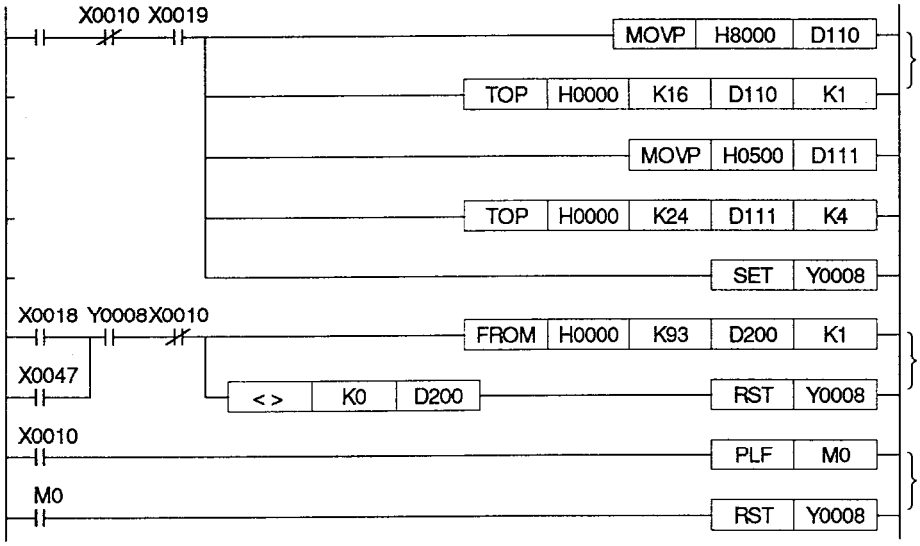
5.3.4 Open processing program example

This section shows the sequence program to open connections between an A1SJ71E71 and a node.

Example: When an A1SJ71E71 performs data communications with a node, the communications parameters are as follows:

Parameter Name		Connection No. 1	Connection No. 2
Fixed buffer	Application	Send	Receive
	Protocol	TCP	TCP
	Open mode	Unpassive	Unpassive
A1SJ71E71 port number		500H	501H
Node	IP address	—	A20009C1H
	Port number	—	501H
Node Ethernet address		Default with ARP	Default with ARP

Open processing of connection No. 1 (Unpassive)



The setting data "Send, TCP, Unpassive" is written in the applications area of connection No.1.

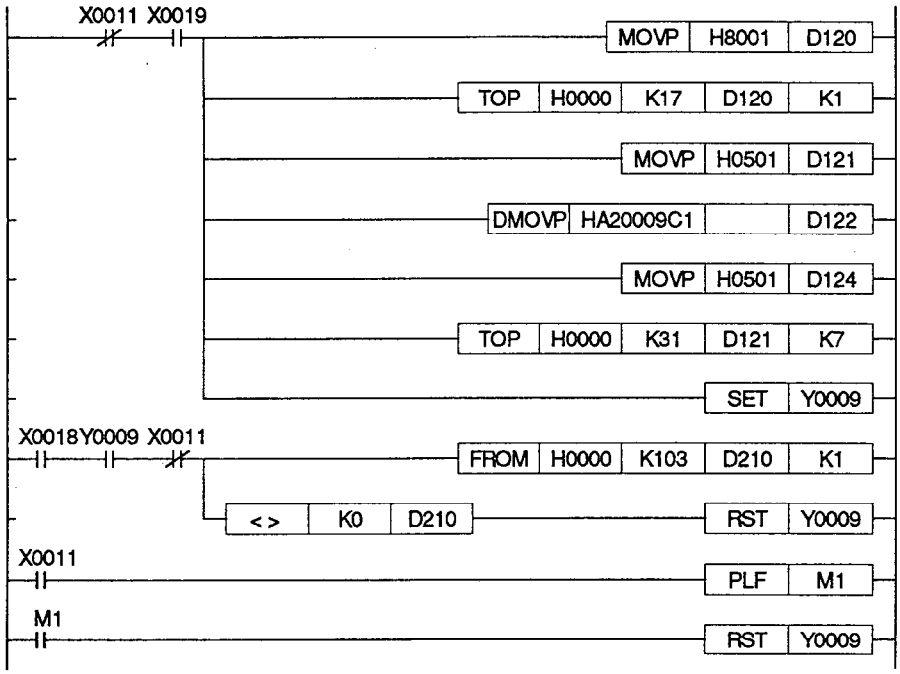
The port number of the A1SJ71E71 is set at 500H.
Writing to the communications address setting area of connection No.1.

The open request signal of connection No.1 is turned ON.

Connection No.1 open error processing.

When the line is closed due to a cause indicated in Table 5.3, the open request signal is turned OFF.

Open processing of connection No. 2 (Fullpassive)



The setting data "Receive, TCP, Unpassive" is written in the applications area of connection No.2.

The port number of the A1SJ71E71 is set at 501_H.

The IP address of the other node is set as $A20009C1_H$.

The port number of the other node is set as 501_H.

Writing to the communications address setting area of connection No.2.

The open request signal of connection No.2 is turned ON.

Connection No.2 open error processing.

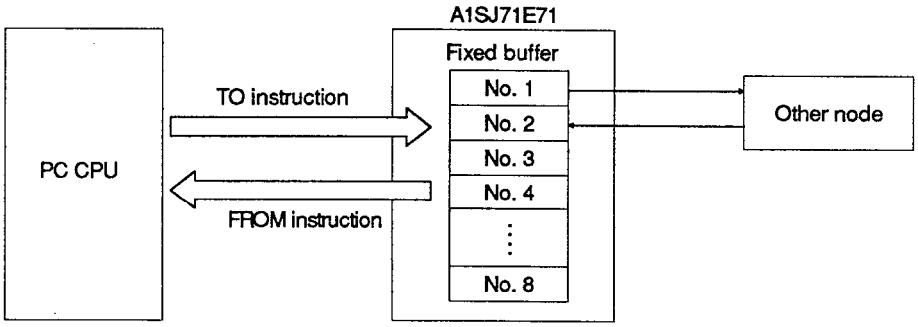
When the line is closed due to a cause indicated in Table 5.3, the open request signal is turned OFF.

6. COMMUNICATIONS PROCESSING USING FIXED BUFFER MEMORY

6.1 Control Methods

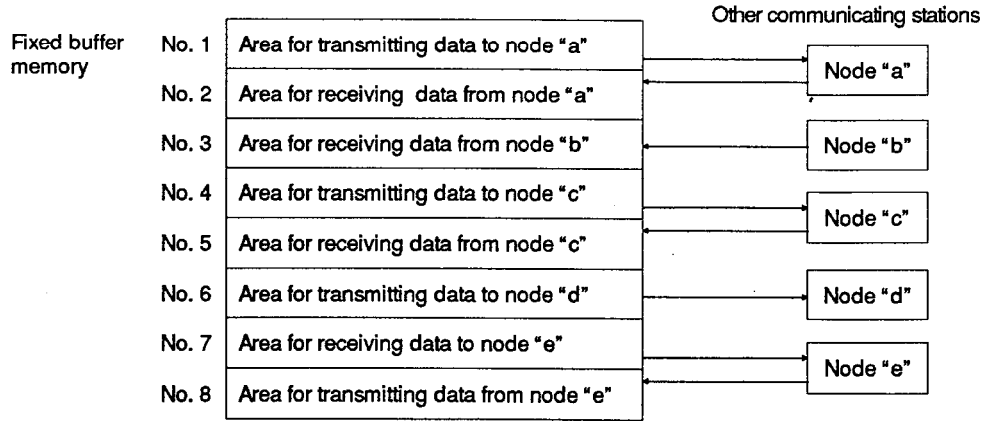
During communications processing with another node using fixed buffer memory, read/write of data from/to a PC CPU are executed by using hand-shake signals.

(1) The data flow in data communications processing to and from fixed buffer memory areas is shown below.



(2) During data communications to and from fixed buffer memory areas, the particular communicating node and the use (send or receive) are set for each of the fixed buffer memory areas (Nos. 1 to 8) when the communications line of an A1SJ71E71 is opened (see Section 5.3). This fixes the buffer memory areas allocated for communicating with other nodes.

Example:

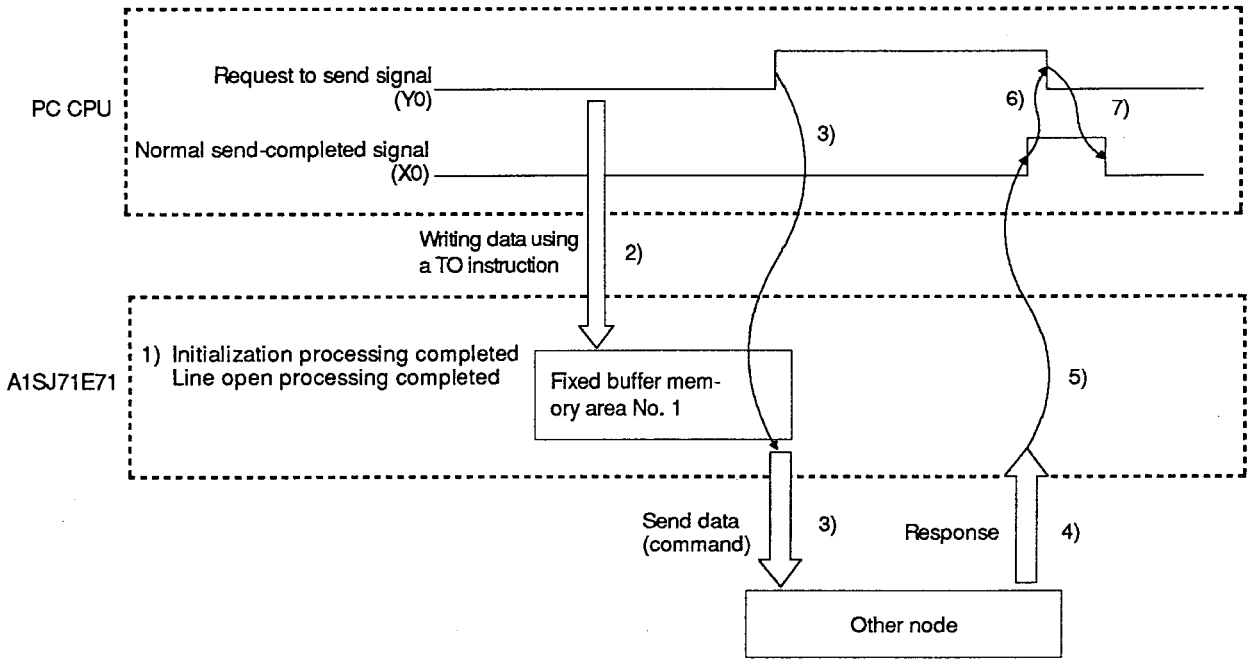


(3) The parameter setting of the fixed buffer memory areas for each communicating station becomes valid at the leading edge of the A1SJ71E71 communications line open-completed signal.

The communicating station cannot be changed while the communications line open-completed signal is ON.

6.1.1 Send control methods

Data send control methods when transmitting data from an A1SJ71E71 to another node are explained below (taking data transmitted from fixed buffer memory area No. 1 to another node as an example).



1) The A1SJ71E71 is initialized (see Section 5.2).

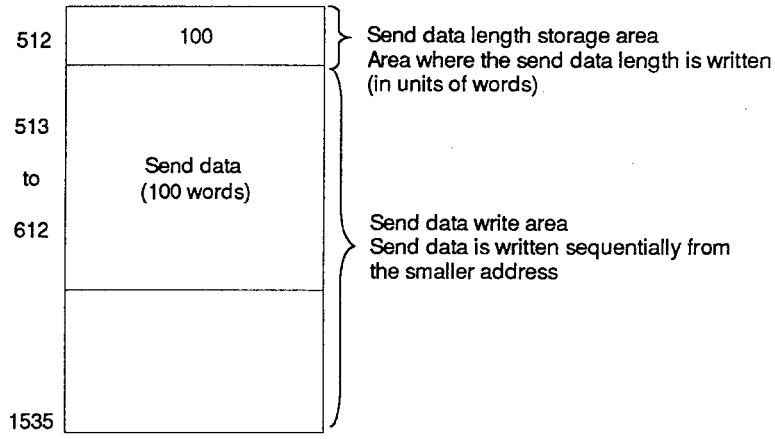
   The communications line open processing is executed (see Section 5.3).

2) The sequence program TO instruction writes the send data length and the send data to the fixed buffer memory areas of the A1SJ71E71.

   The send data length is written to the head addresses (512, 1536, 2560, 3584) of fixed buffer memory areas Nos. 1 to 8.

   The data to be transmitted is written to the area that follows the head address.

   The following example shows the procedure for transmitting 100 words of data using fixed buffer memory area No. 1.



3) By turning ON the request to send signal (Y0 for fixed buffer memory area No. 1) with a sequence program, data in the designated fixed buffer memory address areas is transmitted to the parameter-set designated node.

4) The designated node, in response to data receives from an A1SJ71E71, returns a "response" to the A1SJ71E71.

5) The A1SJ71E71 turns the normal send-completed signal (X0 for fixed buffer memory area No. 1) ON when it receives the "response" from the designated node.

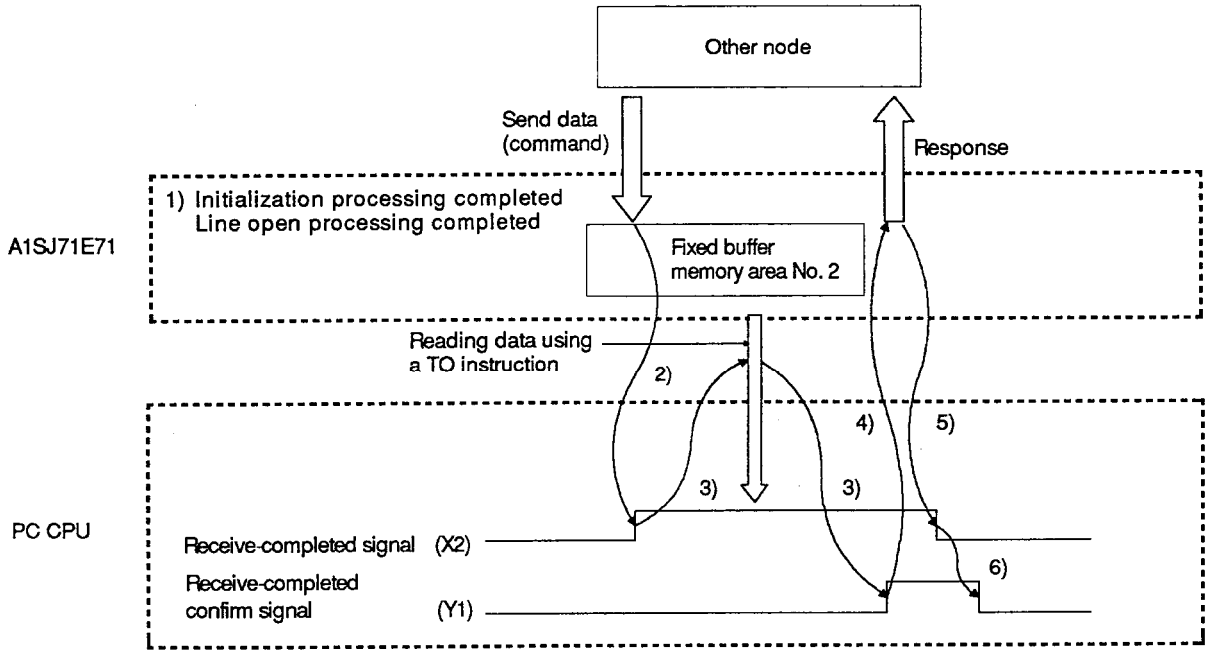
6) The sequence program turns OFF the request to send signal (Y0 for fixed buffer memory area No. 1) when the normal send-completed signal goes ON.

7) The normal send-completed signal goes OFF when the request to send signal is turned OFF.

   If data send is not correctly completed (no response from another node or if the responding end code is not "00_H"), the send error detection signal (X1) goes ON. In this case, the normal send-completed signal (X0) does not go ON.
   If the X1 signal is ON, turn the request to send signal ON and retry data send processing.

6.1.2 Receive control methods

The data receive control method when an A1SJ71E71 receives data from another node is explained below (taking data received by fixed buffer memory area No. 2 from another node as an example).

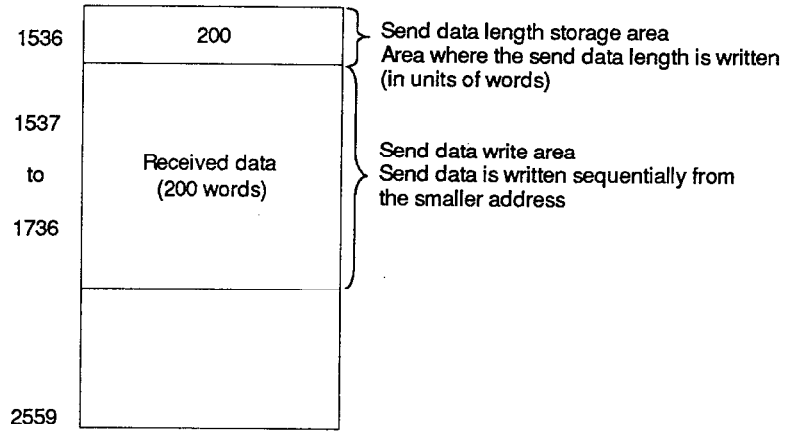


1) The A1SJ71E71 is initialized (see Section 5.2).

   The communications line open processing is executed (see Section 5.3).
   As a condition for executing communications with the fixed buffer, initial processing and open processing must be completed.

2) The A1SJ71E71 turns ON the receive-completed signal (X2 for fixed buffer memory area No. 2) when data from another node is received in the set fixed buffer memory areas.

   The received data length and received data are stored in the fixed buffer memory areas.
   The received data length is written to the head addresses (512, 1536, 2560, 3584) of fixed buffer memory areas Nos. 1 to 8.
   The received data is written to the area that follows the head address.
   The following chart shows what happens when 200 words of data are received in fixed buffer memory area No. 2.



3) When the receive-completed signal is turned ON, the FROM instruction in a sequence program reads the received data length and received data stored in the fixed buffer memory areas.

   At the same time, turn ON the receive-completed confirmation signal (Y1 for fixed buffer memory area No. 2) with a sequence program.

4) The A1SJ71E71, in response to the turning ON of the receive-completed confirmation signal, returns a "response" to the parameter-set other node.

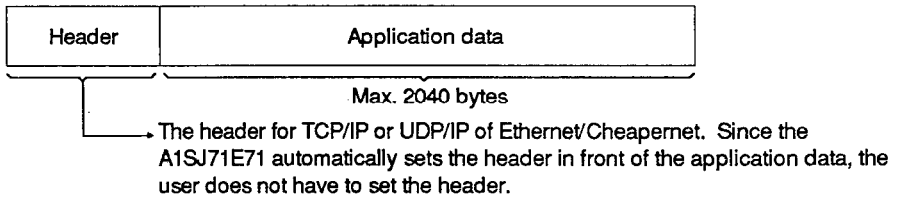
5) After returning the "response", the A1SJ71E71 automatically turns OFF the receive-completed signal.

6) The sequence program turns OFF the receive-completed confirmation signal when the receive-completed signal goes OFF.

   If data receive is abnormal, (a) the receive-completed signal (X2) is not turned ON and, (b) the received data is not stored in fixed buffer memory area No. 2.

6.2 Data Format

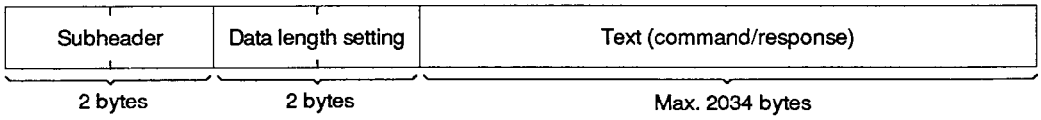
Communications data consists of "header" and "application data" as explained below.



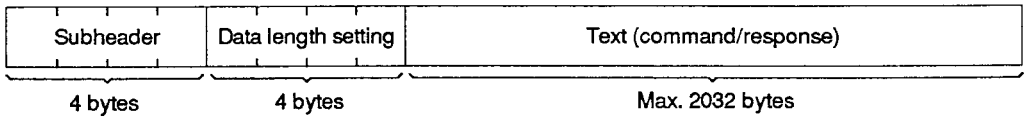
6.2.1 Application data format

The application data format varies depending on which code (binary or ASCII) will be used.
Whether the binary or ASCII code is used is set with the dip switches on the front panel of the A1SJ71E71. Section 4.3.3 gives setting details.

Communications in binary code:

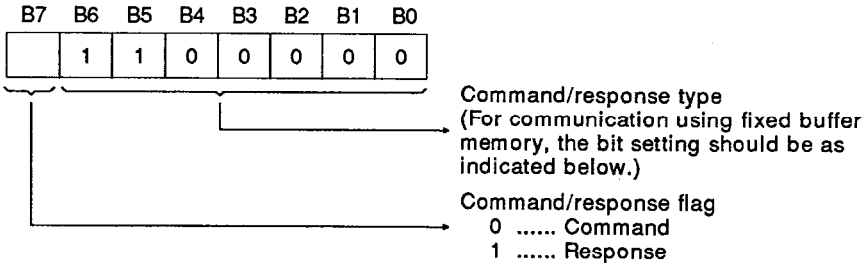


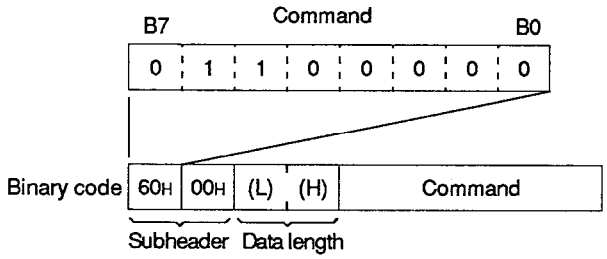
Communications in ASCII code:

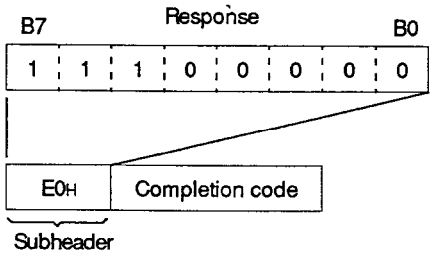


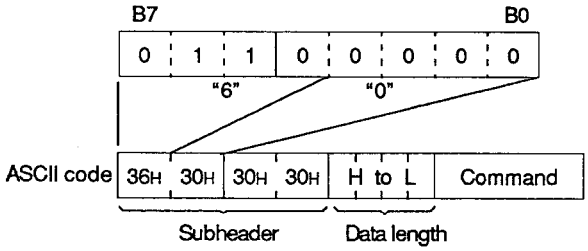
6.2.2 Subheader

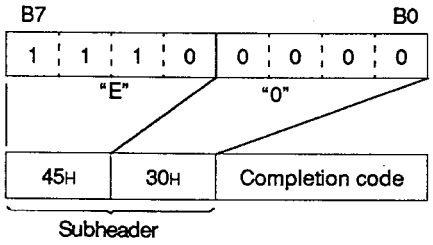
Since the subheader is automatically set by the A1SJ71E71, the user does not have to set the subheader.







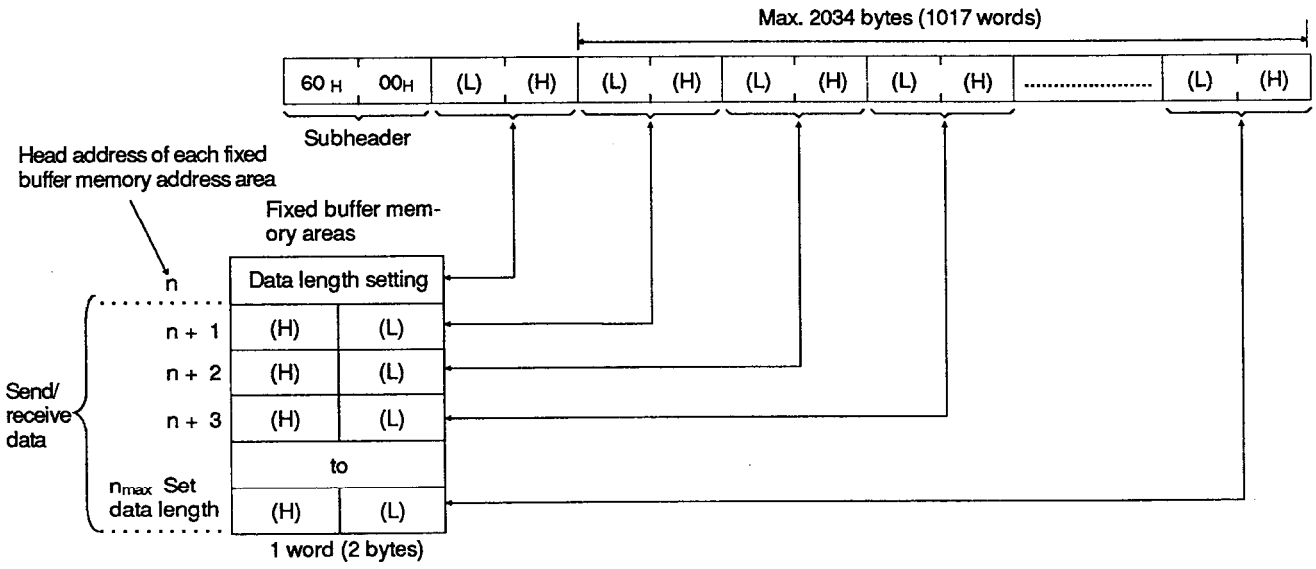




6.2.3 Command/response format

Binary Code Designation:

(1) Command format

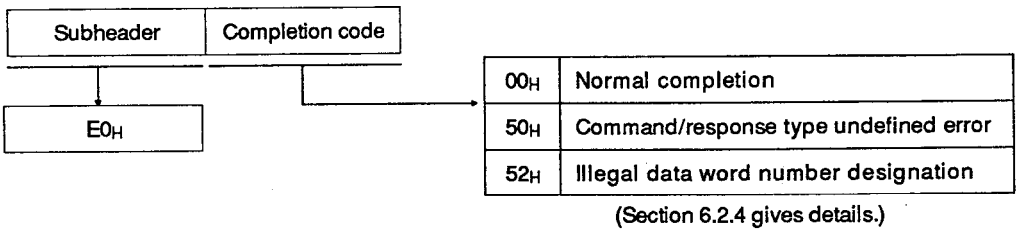


POINTS

(1) When binary code is used, the maximum communications data length is 1017 words.

(2) The data length setting range is 1 to 1017 (in units of words).

(2) Response format



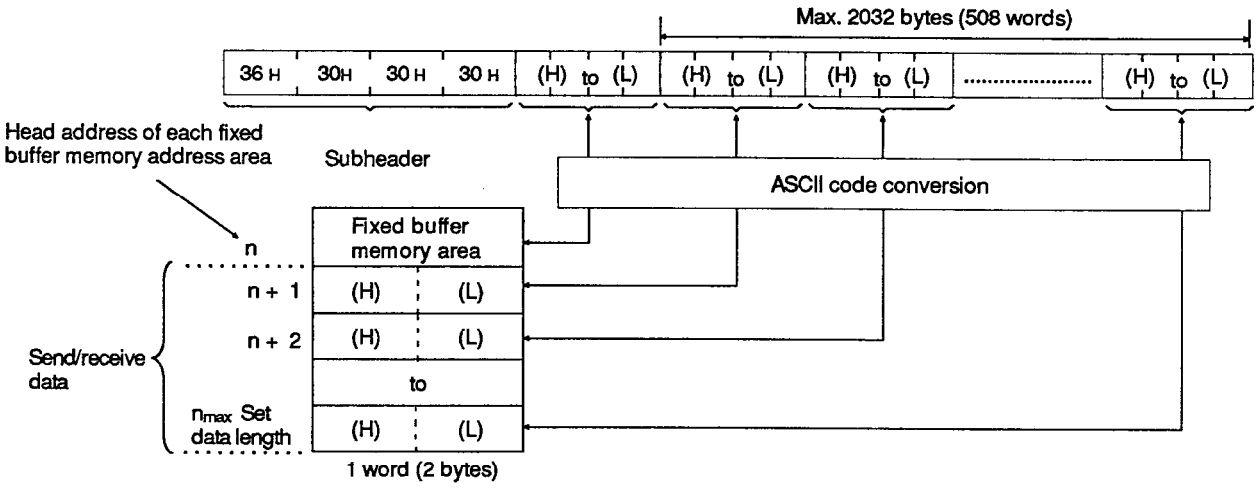
(Section 6.2.4 gives details.)

REMARK

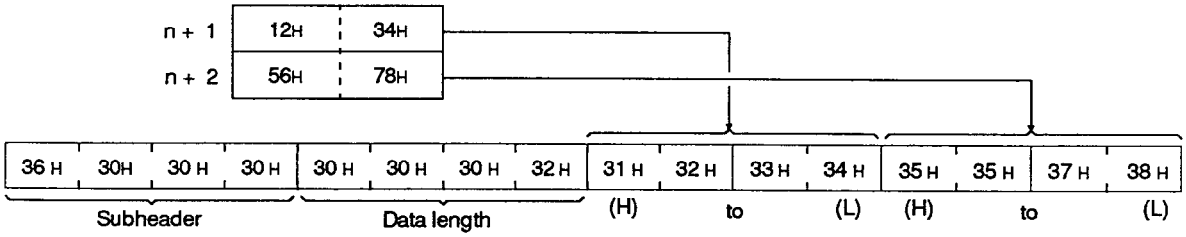
Since the subheader is automatically set by the A1SJ71E71, the user does not have to set the subheader.

ASCII Code Designation

(1) Command format



Example:

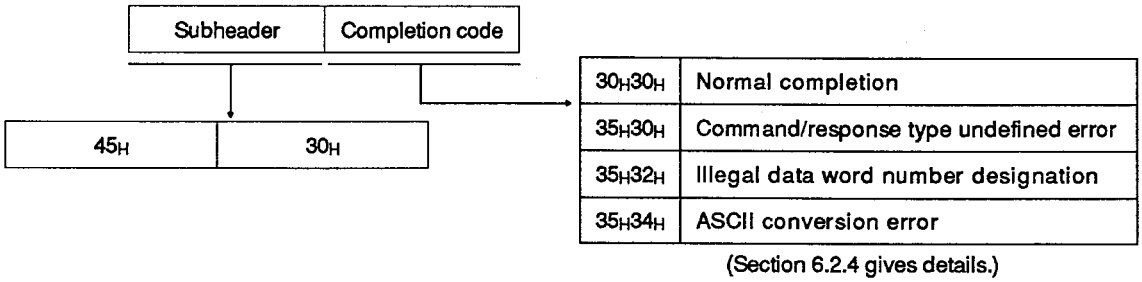


POINTS

(1) When ASCII code is used, the maximum communications data length is 508 words.
This is approximately half the maximum data length when binary code is designated.

(2) The data length setting range is 1 to 508 (in units of words).

(2) Response format



(Section 6.2.4 gives details.)

REMARK

Since the subheader is automatically set by the A1SJ71E71, the user does not have to set the subheader.

6.2.4 Completion code list

A completion code is stored in the communications status storage areas in buffer memory (Section 5.3.3 gives details).

Table 6.1 Completion Code List

<table>
<tr><th>Completion Code</th><th>Description</th><th>Corrective Action</th></tr>
<tr><td>00H</td><td>Normal completion</td><td>–</td></tr>
<tr><td>50H</td><td>The command/response type in the subheader is not allowable code.
<table><tr><th>Communications Processing</th><th>Command/ Response Type</th></tr><tr><td>Communications using fixed buffer memory</td><td>60H</td></tr><tr><td>Communications using random access buffer memory</td><td>61H, 62H</td></tr><tr><td>Read/write of data in PC CPU</td><td>00H to 3CH</td></tr></table>If the set data length is smaller than the actual length of data to be transmitted/ received, any data exceeding the set length is regared as the second data.
This might cause a command/response type of undefined error (see Section 9.2).</td><td>• Check the command/response type set by the communicating node and make corrections as necessary.
(Since the command/response type is automatically set by the A1SJ71E71, the user does not have to set the command/ response type.)
• Check and correct the data length.
• See the REMARK in Section 9.1.4.</td></tr>
<tr><td>52H</td><td>Data of the designated number of words cannot be transmitted in one frame.
The following lengths are excessive:
1017 words for binary code
508 words for ASCII code</td><td>Check and correct the number of data words of the communicating station.</td></tr>
<tr><td>54H</td><td>If the A1SJ71E71 code setting is ASCII,an ASCII code that cannot be converted into the binary code is transmitted from the communicating station.</td><td>Check and correct the send data of the communicating station.</td></tr>
</table>

6.3 Programming

6.3.1 Precautions when programming

(1) Communications using fixed buffer memory areas are possible only when the communications line open-completed signal (X10 to X17) is ON.

Initialization processing and communications line open processing must be completed (see Section 5).

(2) The contents of parameter settings are received by the A1SJ71E71 at the leading edge of the communications line open request signal (Y8 to YF).

Therefore, control contents cannot be changed even if parameter settings are rewritten while the communications line open-completed signal (X10 to X17) is ON.

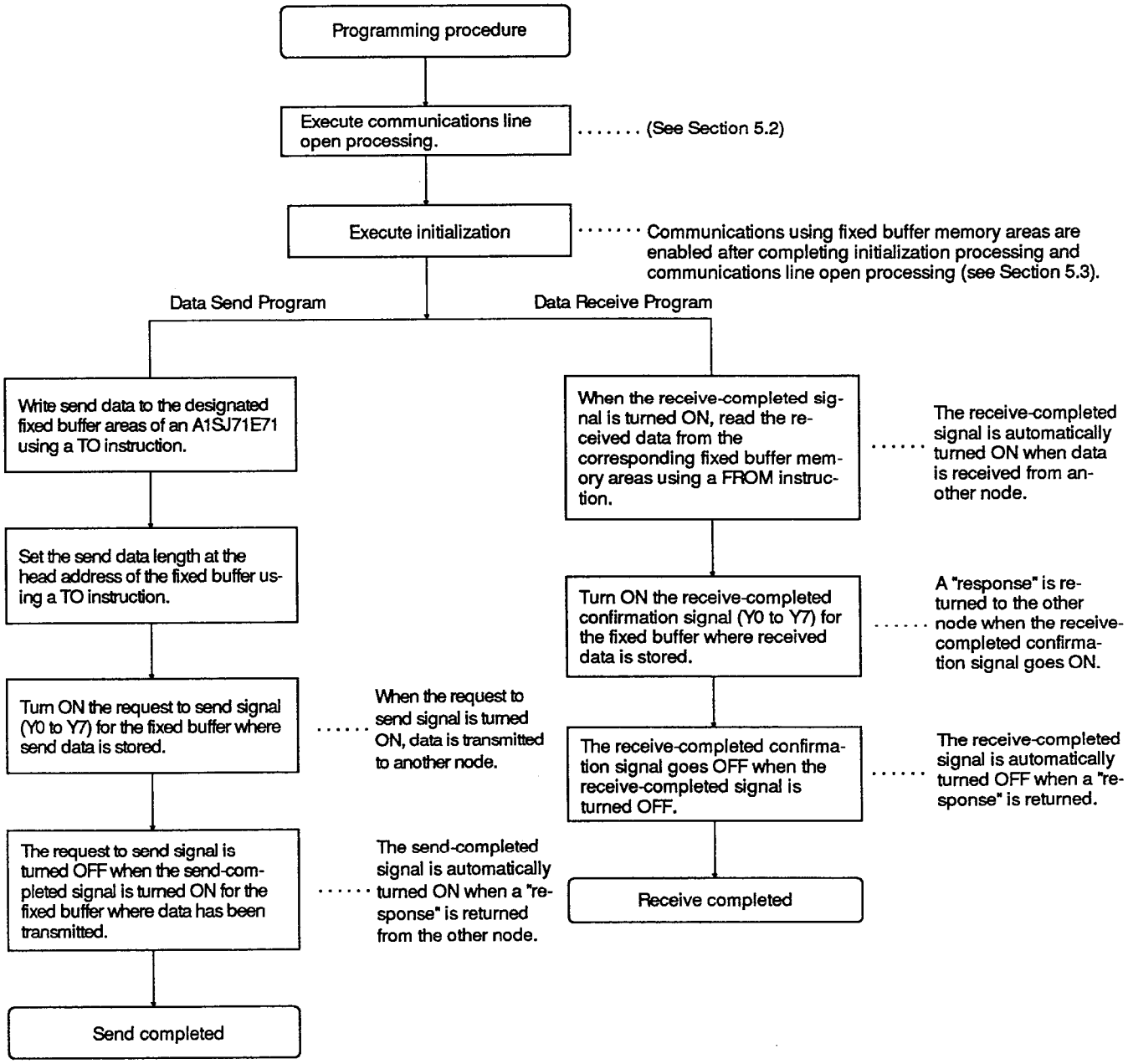
(3) When transmitting data using the fixed buffer memory areas, set the length of data to be transmitted in the designated fixed buffer memory areas.

Using a set length greater than the allowable range causes a communications error and communications cannot be done.

(4) When receiving data using the fixed buffer memory areas, turn ON the receive-completed confirmation signal (Y0 to Y7) when data receive is completed (receive-completed signal ON).

A "response" is returned to the other node when the receive-completed confirmation signal is turned ON.
If the receive-completed comfirmation signal is not turned ON, a communications error occurs at the communicating node because a "response" has not been returned to that node.

6.3.2 Programming procedure



6.3.3 Sample communications program

This example assumes the following conditions:

(a) The communications parameters for each connection are set as in Section 5.3.4.

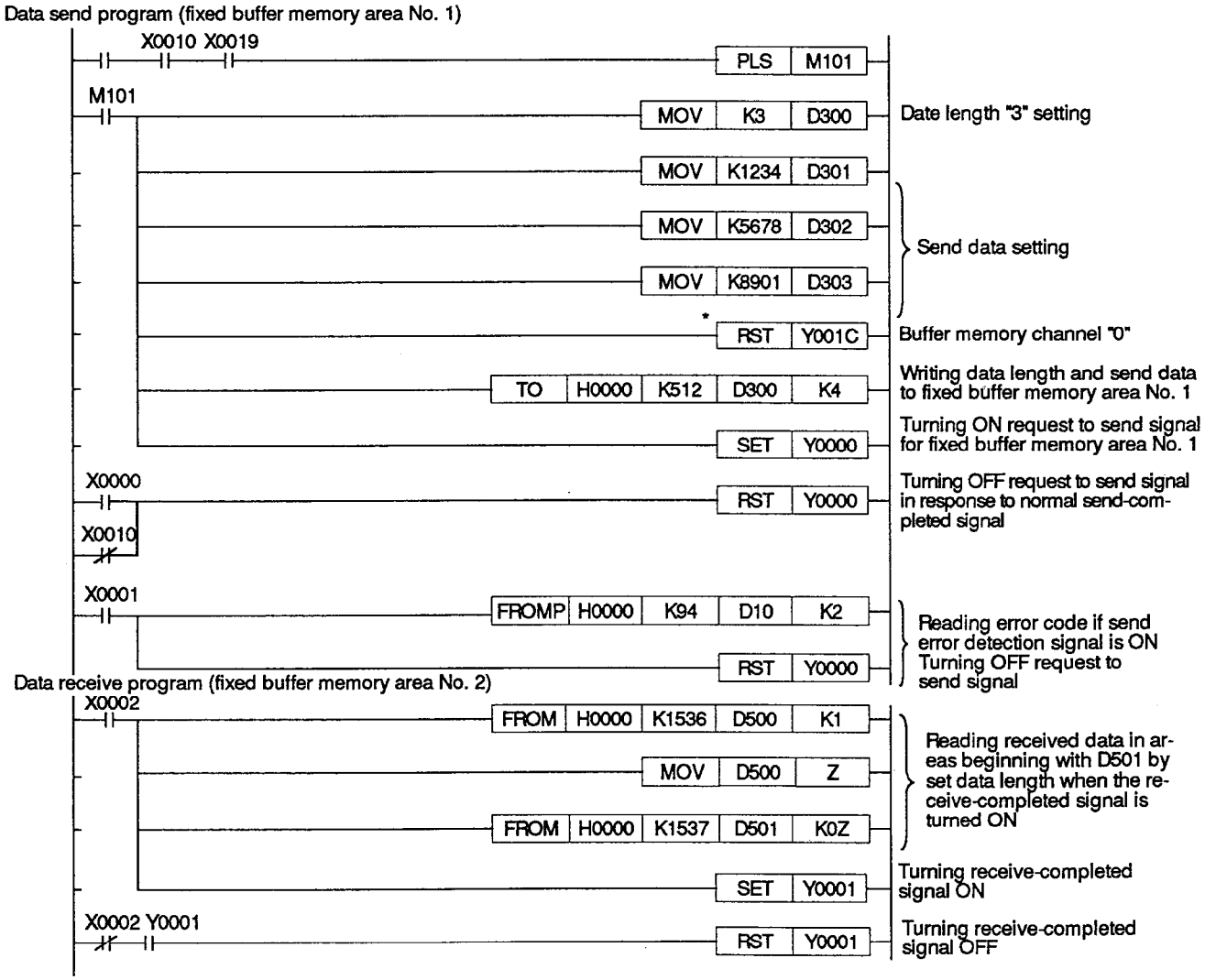
(b) Send data is set in D300 to D399.

(c) Received data is stored in D500 to D599.

(d) Error code and completion code are stored at:

D10: Send error code

D11: Communications end code



POINT

If the I/O control method used for the ACPU is the refresh method, Y1C (marked by the asterisk) will be output directly to the A1SJ71E71 and it is therefore necessary to add a partial refresh (SEG) command.
Adding this command ensures that Y1C is output to the A1SJ71E71 before the TO command in the next step is executed.

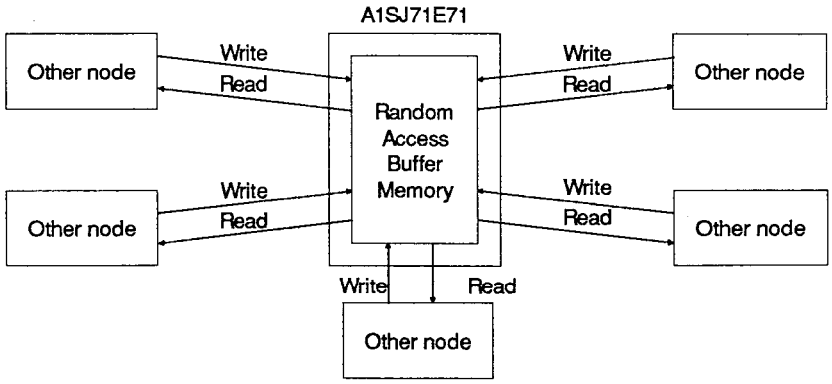
7. COMMUNICATION PROCESSING USING RANDOM ACCESS BUFFER MEMORY AREA

7.1 Control Method

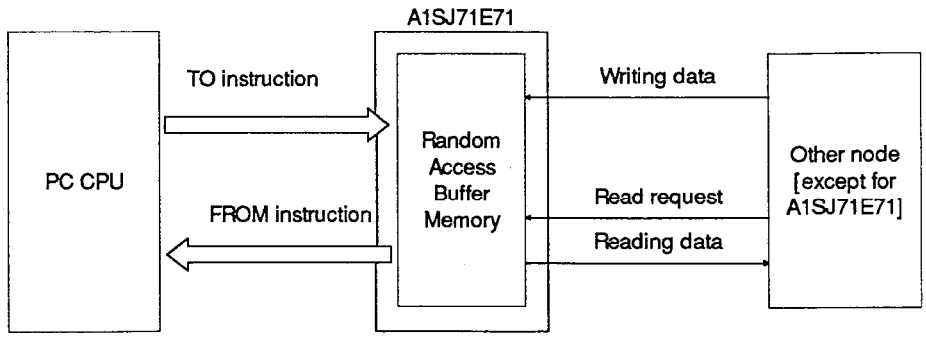
In communication processing using random access memory area, writing data to random access buffer memory area and reading data from random access buffer memory area are initiated by a command (request) from other node.
This data read/write using the random access buffer memory area is executed independently of the execution of PC CPU's sequence program.

(1) The random access buffer memory area is accessible from any node (except for A1SJ71E71) without designating the mating communication node.

Therefore, this area can be used as the buffer memory area common to all nodes connected to the 10BASE2/10BASEB5.



(2) Data flow in data communication processing using the random buffer memory area is shown below.

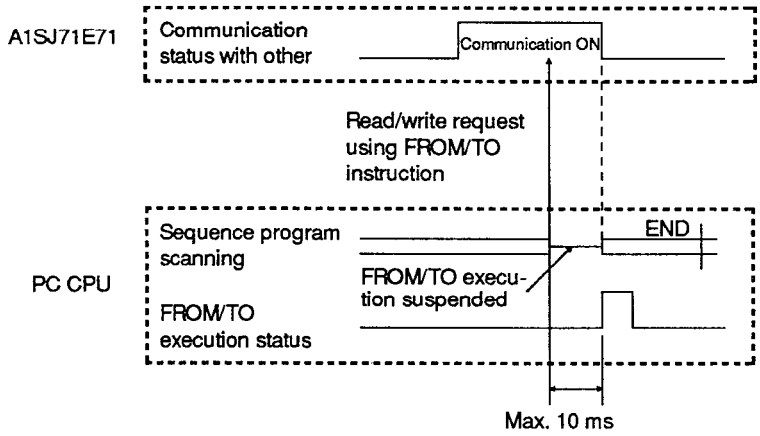


(3) Communications between two A1SJ71E71 modules is not possible using random access buffer memory area.

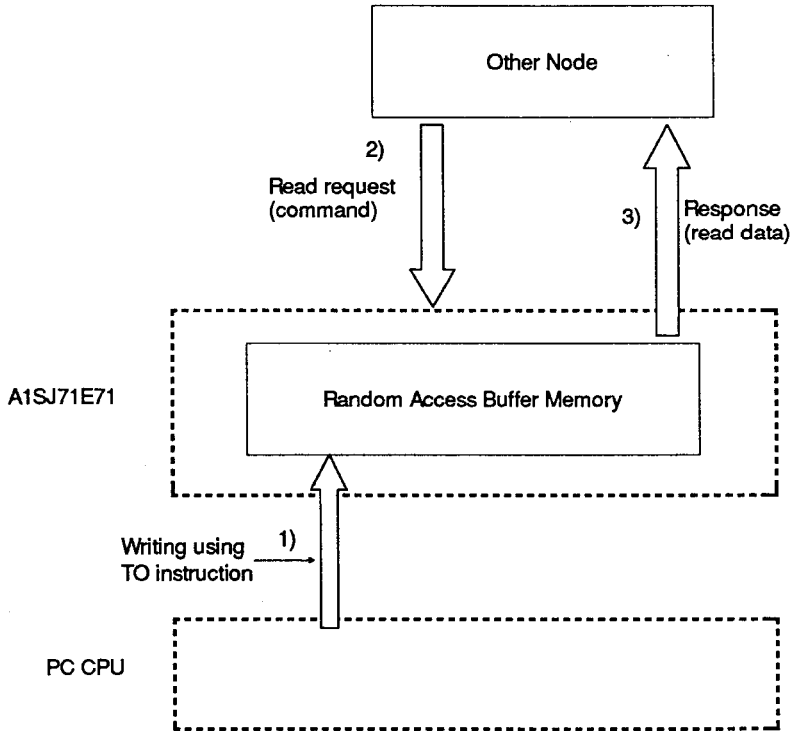
(4) Execution timing of FROM/TO instruction used for read/write operation between an A1SJ71E71 and a PC CPU is explained below.

Execution of FROM/TO instruction is suspended during communications with another node and is executed after the completion of communications.

Execution timing:



7.1.1 Communication control in response to read request from other node



1) Write data to A1SJ71E71 random access buffer memory area using a TO instruction in a sequence program.

   Write data to A1SJ71E71 random access buffer memory area from other node.

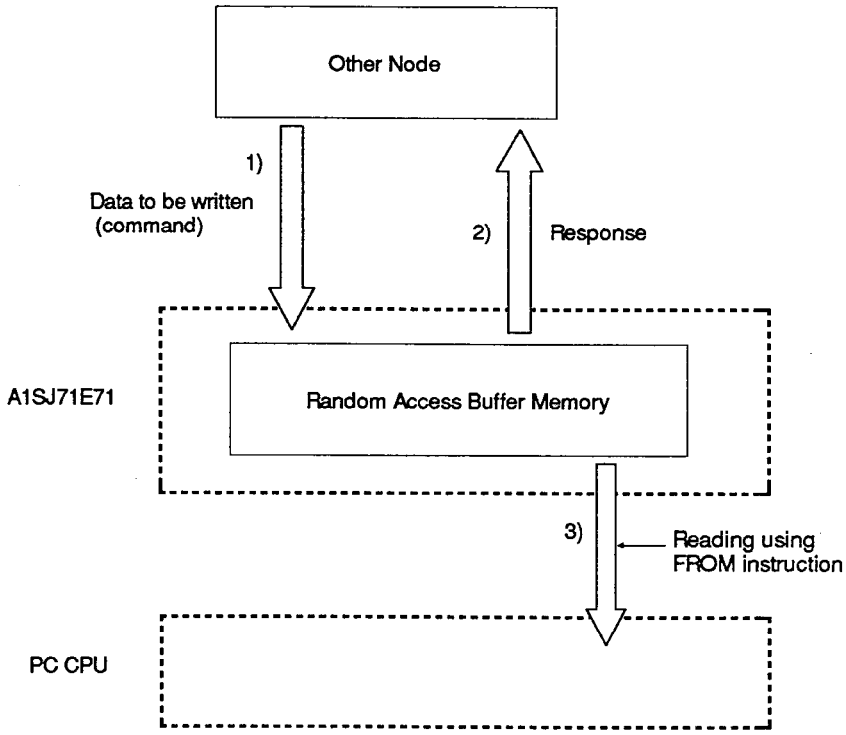
2) Transmit the read request signal from the node that is to read the contents in the random access buffer memory area to the A1SJ71E71.

3) Receiving the read request from the node, the A1SJ71E71 transmits the data stored in the random access buffer memory area as the response.

POINT

In the communications using random access buffer memory area, communications is possible only with the node for which the communication line open complete signal (X10 to X17) is ON.

7.1.2 Communication control in response to write request from other node



1) Write data to A1SJ71E71 random access buffer memory area from other node.

2) Upon receiving data from other node, the A1SJ71E71 returns "response" to the node that has transmitted data.

3) Read data received to the random access buffer memory area using a FROM instruction in a sequence program.

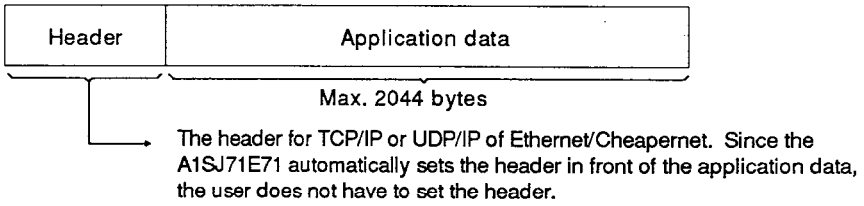
   It is also possible to read received data from the random access buffer memory area by other node.

POINT

In the communications using random access buffer memory area, communications is possible only with the node for which the communication line open complete signal (X10 to X17) is ON.

7.2 Data Format

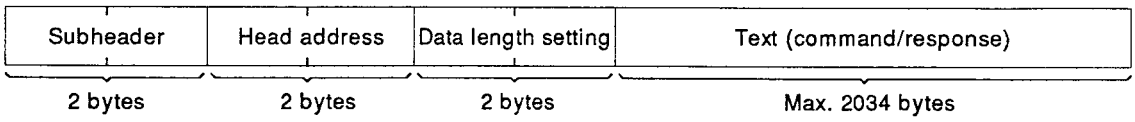
Communications data consists of "header" and "application data" as explained below.



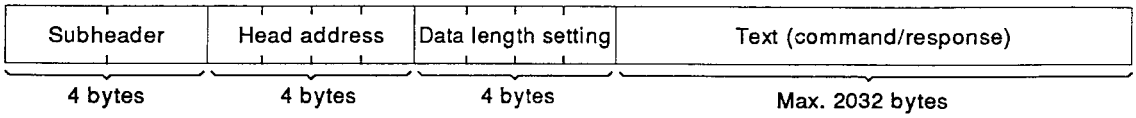
7.2.1 Application data format

The application data format varies depending on which code (binary or ASCII) will be used.
Whether the binary or ASCII code is used is set with the dip switches on the front panel of the A1SJ71E71. Section 4.3.3 gives setting details.

Communications in binary code:

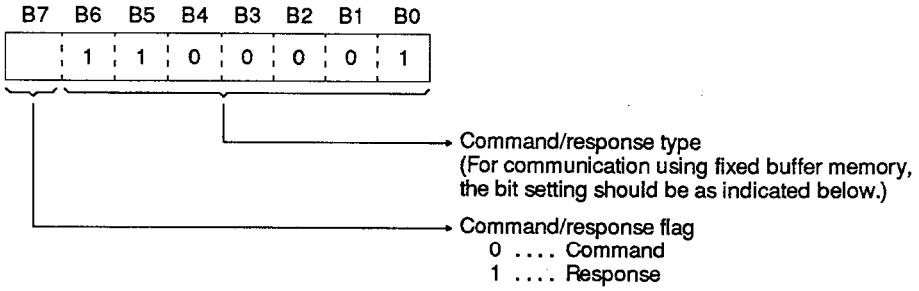


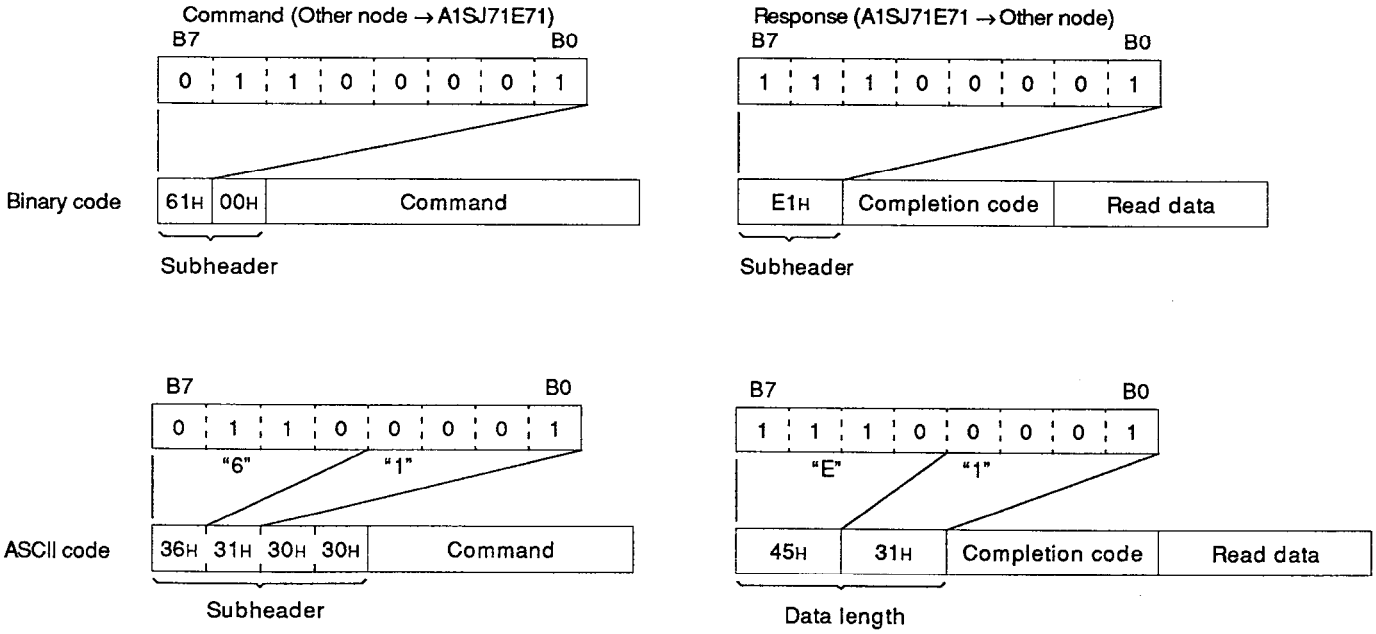
Communications in ASCII code:



7.2.2 Subheader

Since the subheader is automatically set by the A1SJ71E71, the user does not have to set the subheader.



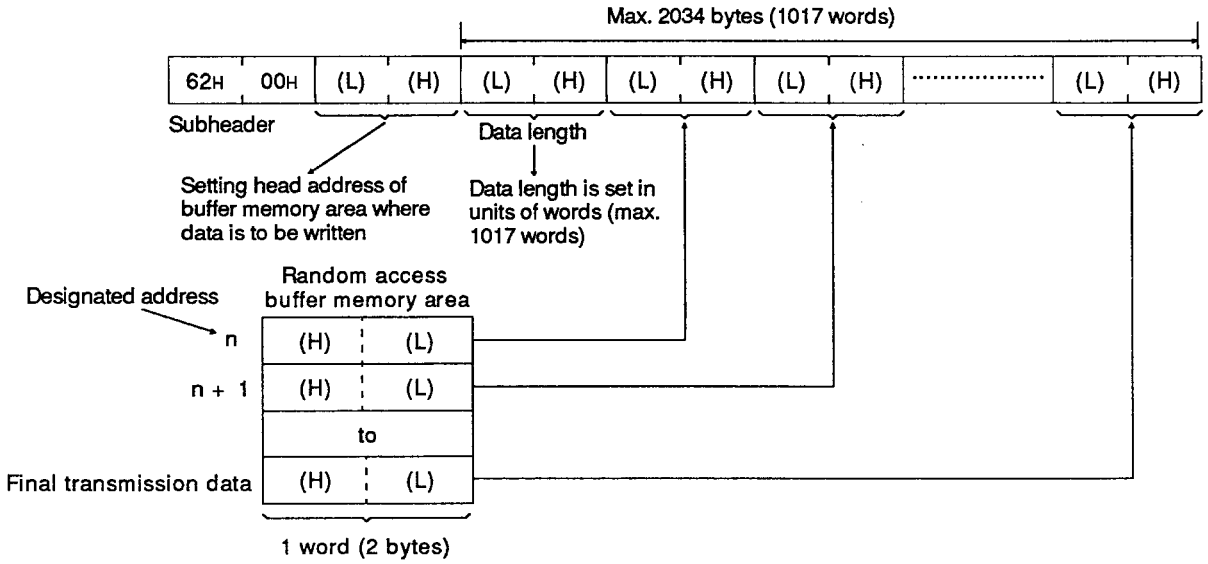


7.2.3 Command/response format

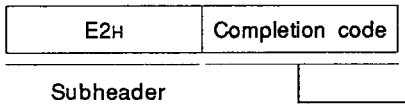
(1) Writing data to buffer memory area by write request from other node

Binary Code Designation

(a) Command format (other node → A1SJ71E71)



(b) Response format (A1SJ71E71 → other node)



00H	Normal complete
50H	Command/response type undefined error
51H	Illegal head address
52H	Illegal data word number

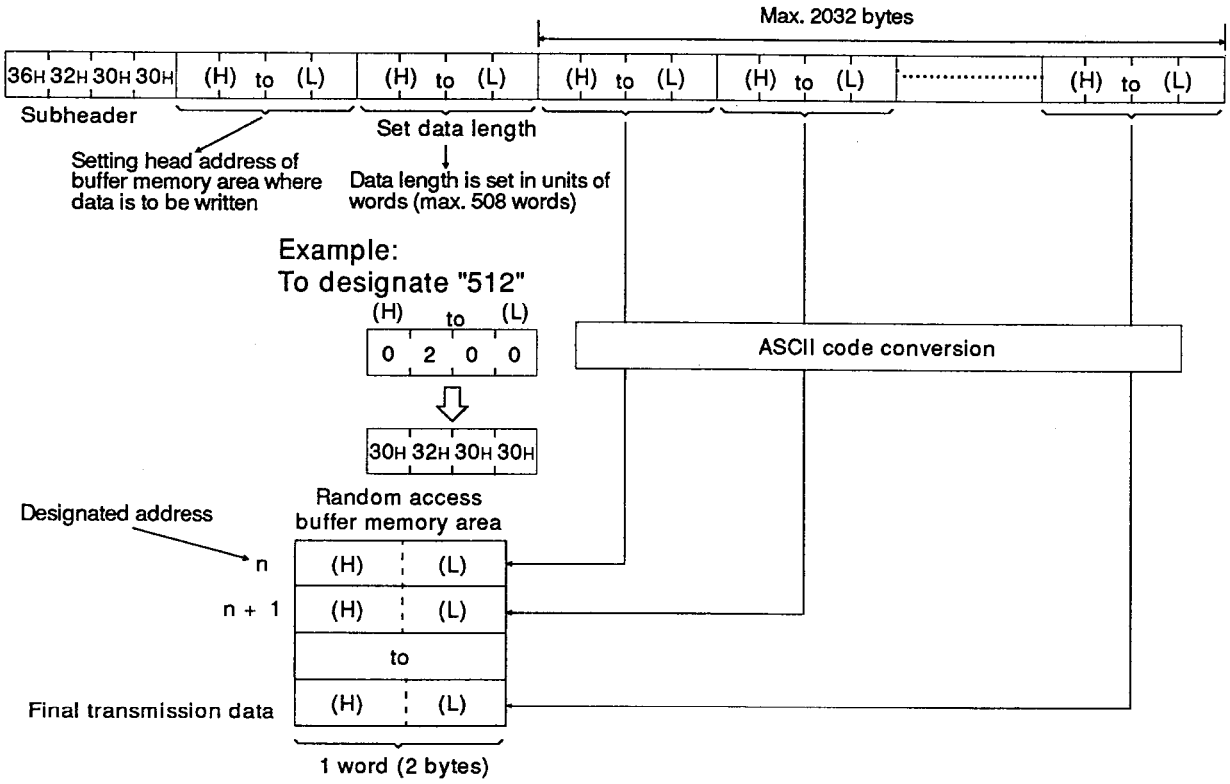
For details, see Section 7.2.4

REMARK

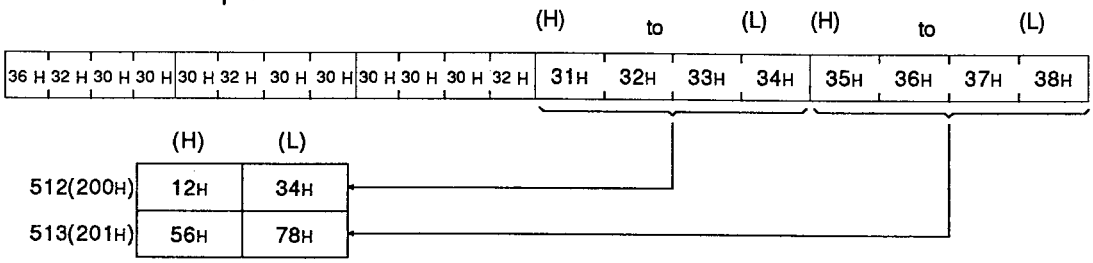
Since the subheader is automatically set by the A1SJ71E71, the user does not have to set the subheader.

ASCII Code Designation

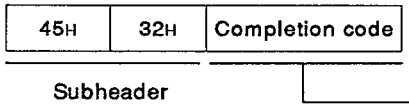
(a) Command format (other node → A1SJ71E71)



Example:



(b) Response format (A1SJ71E71 → other node)



30H30H	Normal complete
35H30H	Command/response type undefined error
35H31H	Illegal head address
35H32H	Illegal data word number
35H34H	ASCII conversion error

For details, see Section 7.2.4

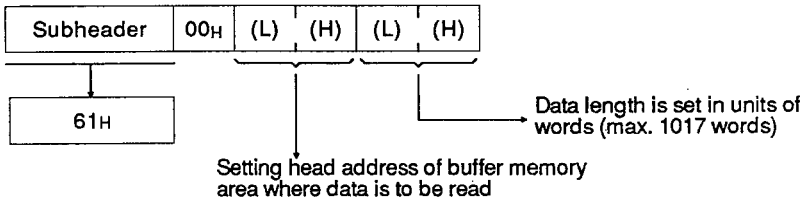
REMARK

Since the subheader is automatically set by the A1SJ71E71, the user does not have to set the subheader.

(2) Transmitting data by read request from other node

Binary Code Designation

(a) Command format

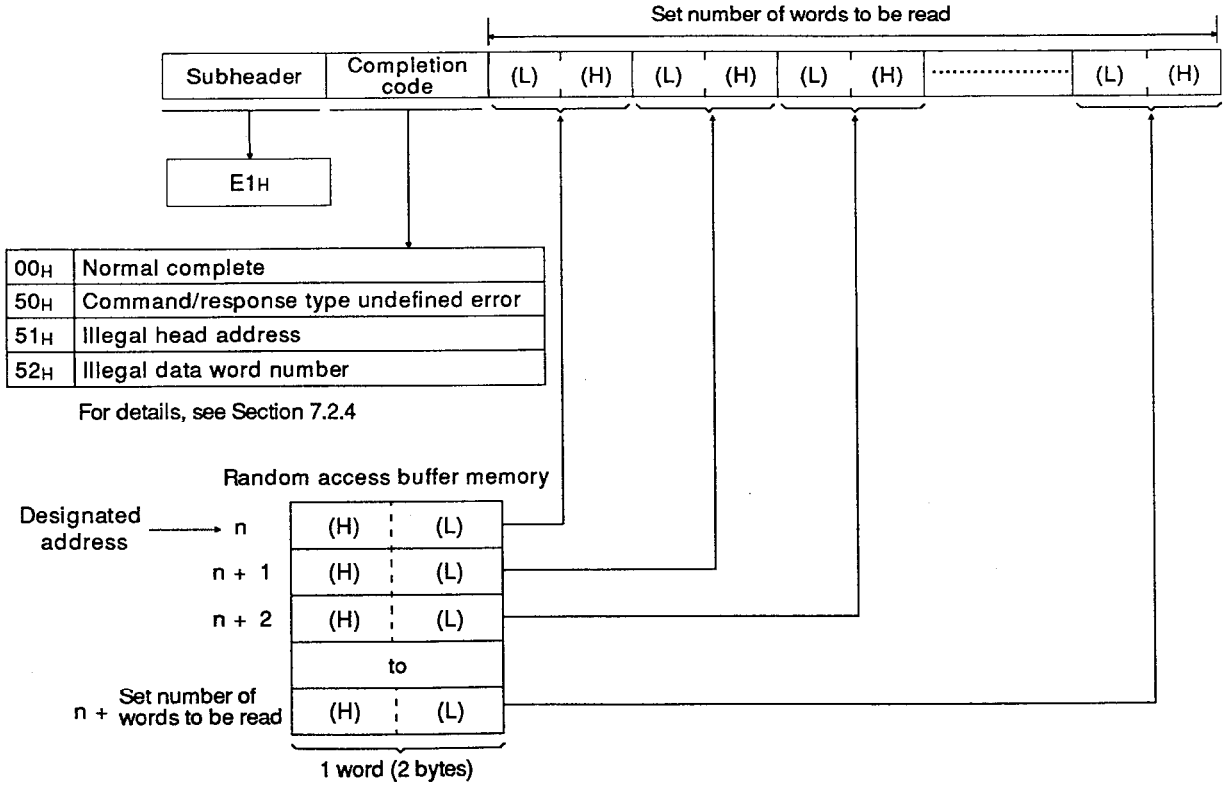


Example: To designate "100"

100 = 64H

(L)	(H)
64H	00H

(b) Response format

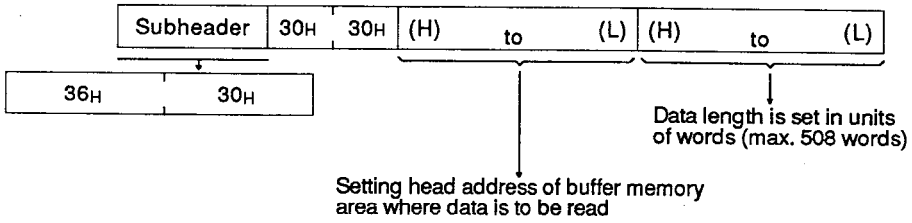


REMARK

Since the subheader is automatically set by the A1SJ71E71, the user does not have to set the subheader.

ASCII Code Designation

(a) Command format

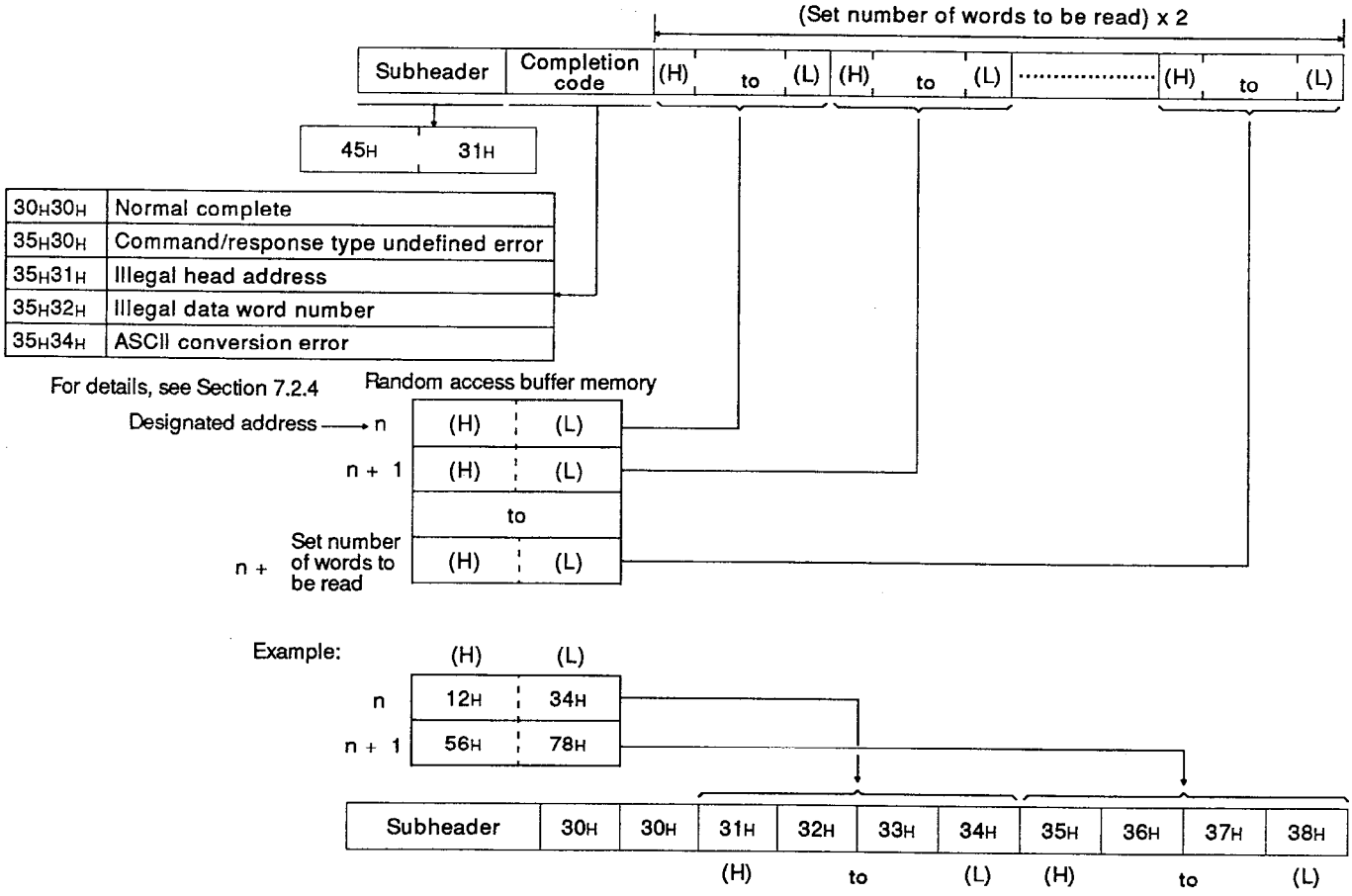


Example: To designate "100"

100 = 64H

(H)		to	(L)
30H	30H	36H	34H

(b) Response format



Example:

	(H)	(L)
n	12H	34H
n + 1	56H	78H

Subheader	30H	30H	31H	32H	33H	34H	35H	36H	37H	38H
			(H)	to		(L)	(H)	to		(L)

REMARK

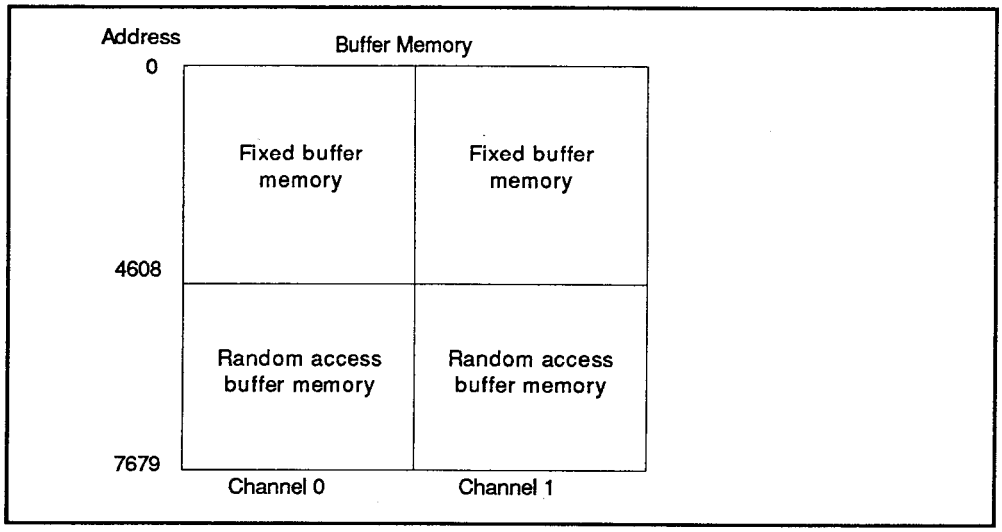
Since the subheader is automatically set by the A1SJ71E71, the user does not have to set the subheader.

7.2.4 List of end codes

<table>
<tr><th>Completion Code</th><th>Description</th><th>Corrective Action</th></tr>
<tr><td>00H</td><td>Normal completion</td><td>–</td></tr>
<tr><td>50H</td><td>The command/response type in the subheader is not allowable code.
<table><tr><th>Communications Processing</th><th>Command/ Response Type</th></tr><tr><td>Communications using fixed buffer memory</td><td>60H</td></tr><tr><td>Communications using random access buffer memory</td><td>61H, 62H</td></tr><tr><td>Read/write of data in PC CPU</td><td>00H to 3CH</td></tr></table></td><td>• Check the command/response type set by the communicating node and make corrections as necessary.
(Since the command/response type is automatically set by the A1SJ71E71, the user does not have to set the command/response type.)
• See the REMARK in Section 9.1.4.</td></tr>
<tr><td>51H</td><td>When random access buffer is used, the head address by the communicating node is outside the range of 0 to 6143.</td><td>Check and correct the head address and the number of data words.</td></tr>
<tr><td>52H</td><td>When random access buffer is used, the data length of the head address set by the communicating node and data words is outside the rang of 0 to 6143.
The number of data words is set when a read operation is done.
Data of the designated number of words cannot be transmitted in one frame.
The following lengths are excessive:
1017 words for binary code
508 words for ASCII code</td><td>Check and correct the number of data words of the communicating station.</td></tr>
<tr><td>54H</td><td>If the A1SJ71E71 code setting is ASCii, an ASCII code that cannot be converted into the binary code is transmitted from the communicating station.</td><td>Check and correct the send data of the communicating station.</td></tr>
</table>

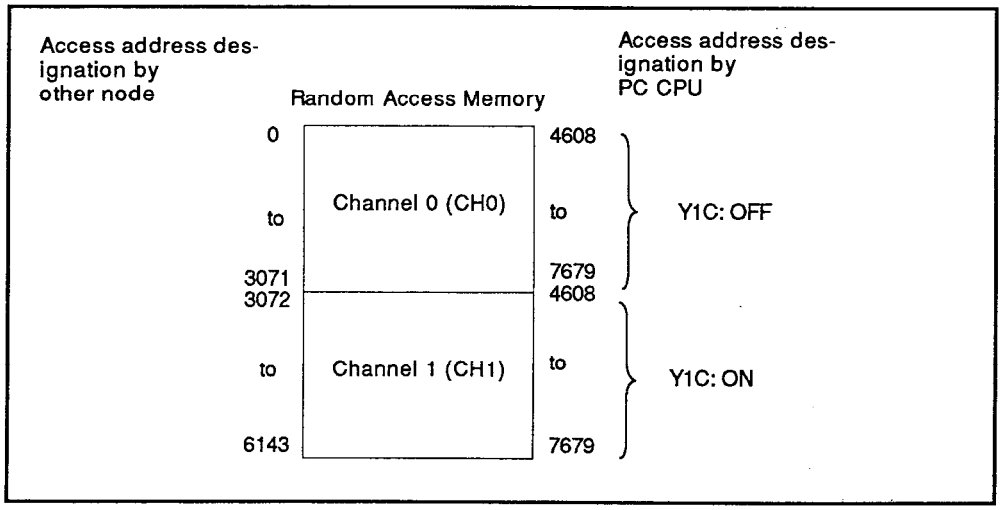
7.3 Data Storing Area

Random access buffer memory area (not battery backed up) of A1SJ71E71 is explained below.



Random access buffer memory address is indicated below.

The address used to designate random access buffer memory area differs between the address designated by other node and the address designated by FROM/TO instruction in a sequence program.



(1) For read/write from other node, it is not necessary to change the channel because continuous 6K word area is used.

For read/write operation using FROM/TO instruction in a sequence program, it is necessary to change the channel by turning ON/OFF the A1SJ71E71's I/O signal Y1C (channel change signal).

Y1C OFF.....Channel 0
Y1C ON .....Channel 1

(2) One address corresponds to 2 bytes.

7.4 Programming

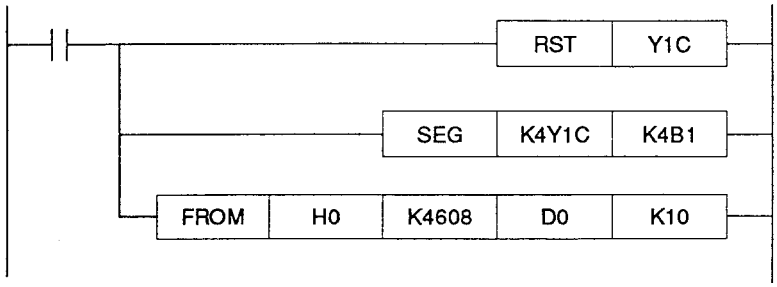
7.4.1 Cautions on programming

(1) Communications using the random access buffer memory area are executed in asynchronous with a sequence program.

Therefore, handshake cannot function between data read/write at random access buffer memory area by PC CPU and data communications.

(2) For communications using random access buffer memory area, address to be designated from other node and the address to be designated with FROM/TO instruction differ from each other. For details, see Section 7.3.

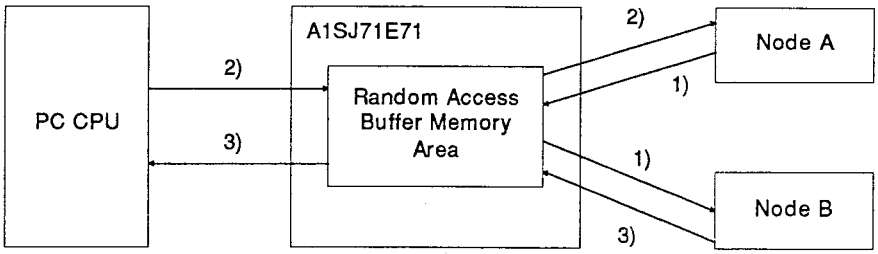
(3) If the ACPU I/O control method used is the refresh method, add a partial refresh (SEG) instruction in order to output the buffer memory channel change signal (Y1C) directly to the A1SJ71E71.



7.4.2 Programming procedure

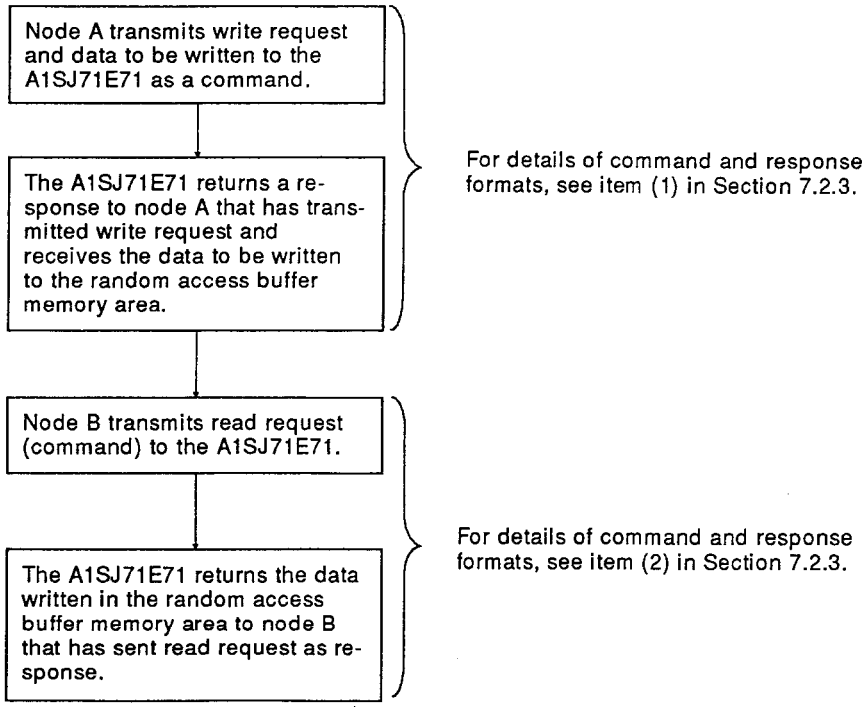
For communications using random access buffer memory area, the following three communication methods are available:

1) Data written to random access buffer memory area by node A is read by node B.
2) Data written to random access buffer memory area by a sequence program is read by a node.
3) Data written to random access buffer memory area by a node is read by a sequence program.

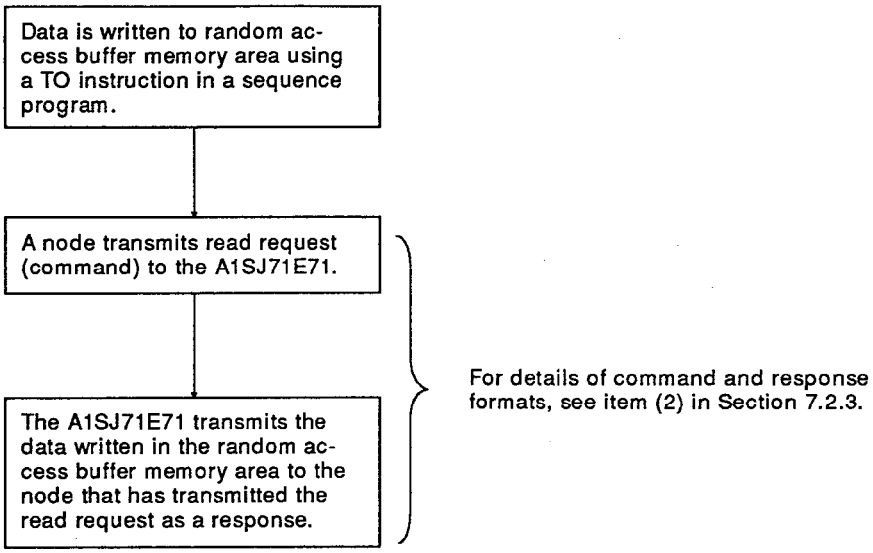


Details of the above indicated three communication methods are described below.

(1) Data written by node A is read by node B:



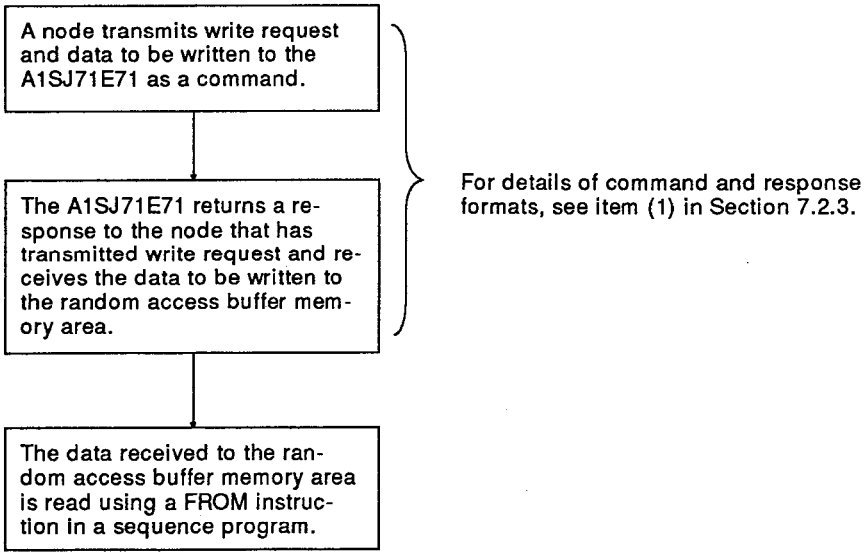
(2) Data written by a sequence program is read by a node:



REMARK

In communications using random access buffer memory area, handshake using the A1SJ71E71 input/output signals.

(3) Data written by a node is read by a sequence program:



REMARK

In communications using random access buffer memory area, handshake using the A1SJ71E71 input/output signals.

8. READING AND WRITING DATA STORED IN THE PC CPU

When data of a device and a program in the PC CPU is read and written through A1SJ71E71 from the communicating node, follow the control method and data format below.

8.1 Control Method

When data in the PC CPU is read and written, use the following control method:

(1) Data in the PC CPU can be read and written. It is not related to the use of the ON/OFF state of the I/O number of A1SJ71E71 and the sequence program.

(2) Writing enable/disable in PC CPU RUN can be selected by a CPU communications timing setting switch in front of A1SJ71E71 in case of the writing from the node to the PC CPU.

Communications timing setting switch (Refer to Section 4.3.3.)

SW3 OFF: Writing cannot be done from the communicating node during PC CPU RUN.

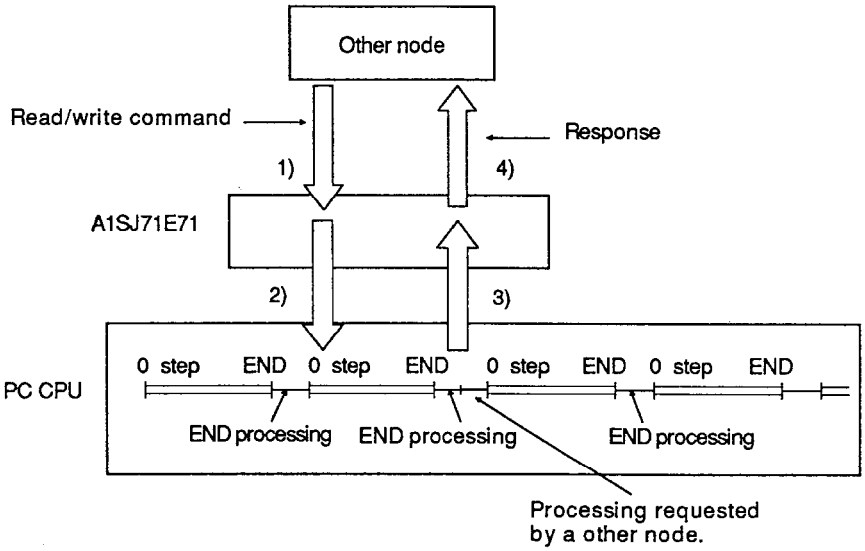
ON : Writing can be done from the node that communicates during PC CPU RUN or STOP.

POINT

When writing is done to the special-function module loaded with a remote I/O station in MELSECNET(II) from the communicating node, set communications timing setting switch (SW3) at ON.
(A remote I/O station is always set to a RUN state. RUN/STOP cannot be switched.)

8.1.1 Communications with a PC CPU which is loaded with an A1SJ71E71

(1) When data in the PC CPU loaded with A1SJ71E71 is read and is written, use the following method of control:



1) A communicating node sends a command to the A1SJ71E71 to read/write data in the PC CPU.

2) Upon receiving the command, the A1SJ71E71 makes a request to read/write the data in the PC CPU.

3) The PC CPU executes the read/write processing according to the request when the END instruction execution of the sequence program is completed.

   Then, the result of the processing is transferred to the A1SJ71E71.

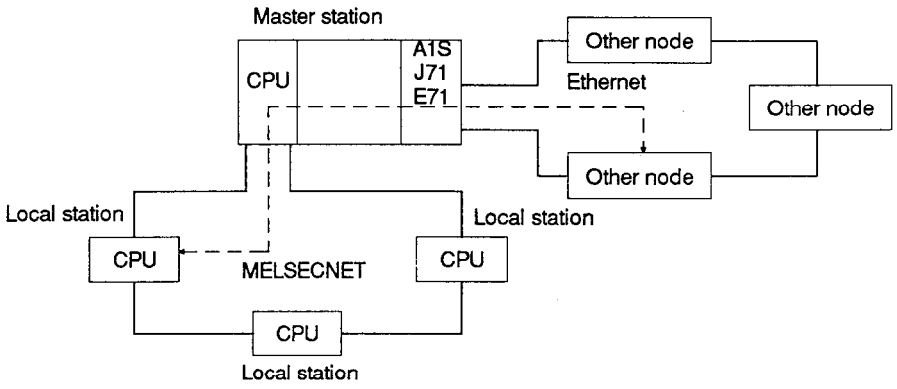
4) Upon receiving the result of the processing from the PC CPU, the AJ71E71 sends a response which contains the result to the other node.

POINT

When read/write from the node is done during PC CPU RUN, time that is necessary for the processing for the command from the node is added in a scan time of a sequence program.

8.1.2 Communications with a PC CPU in MELSECNET

(1) Read/write processing of data in a local station PC CPU can be requested from a mode in other network via the PC CPU loaded with an A1SJ71E71, as shown below.



(2) When data in the PC CPU in MELSECNET is read and written, set the PC number of a PC CPU at a command.

(3) The PC number specifies a communicating PC CPU in MELSECNET.

Set the PC number to FF_H or set it within the range from 00_H to 40_H (0 to 64) as follows.

FF_H ..... When a communications is done for a PC CPU with A1SJ71E71

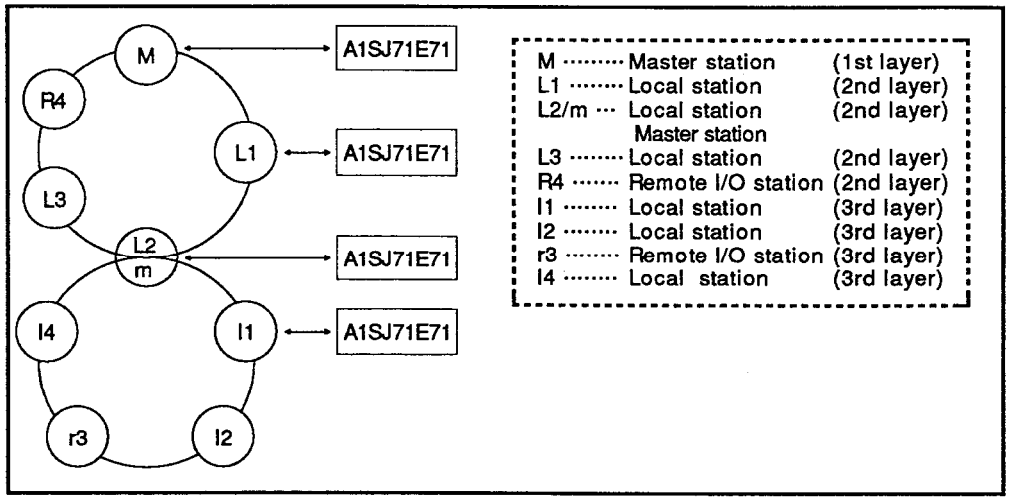
00_H ..... When A1SJ71E71 loaded with the MELSECNET local station communicates with the MELSECNET master station

1 to 40_H ..... When A1SJ71E71 loaded with the MELSECNET master station (1 to 64) communicates with the MELSEC NET local station or a remote I/O station

Set station number (1 to 64) set at a local station or remote I/O station.

(4) Communicating PC CPUs in MELSECNET

The following PC CPUs can be used as communicating nodes in the MELSECNET. Communicating PC CPUs that can be used depend upon the location of stations loaded with an A1SJ71E71.



PC CPU with A1SJ71E71	PC that can be Linked (PC Number)									
	The Self (FF)	M (0)	L1 (1)	L2/m (2/0)	L3 (3)	R4 (4)	I1 (1)	I2 (2)	r3 (3)	I4 (4)
M	o	—	o	o	o	o*1	x	x	x	x
L1	o	o	—	x	x	x	x	x	x	x
L2/m	o	o	x	—	x	x	o	o	o*1	o
I1	o	x	x	o	x	x	—	x	x	x

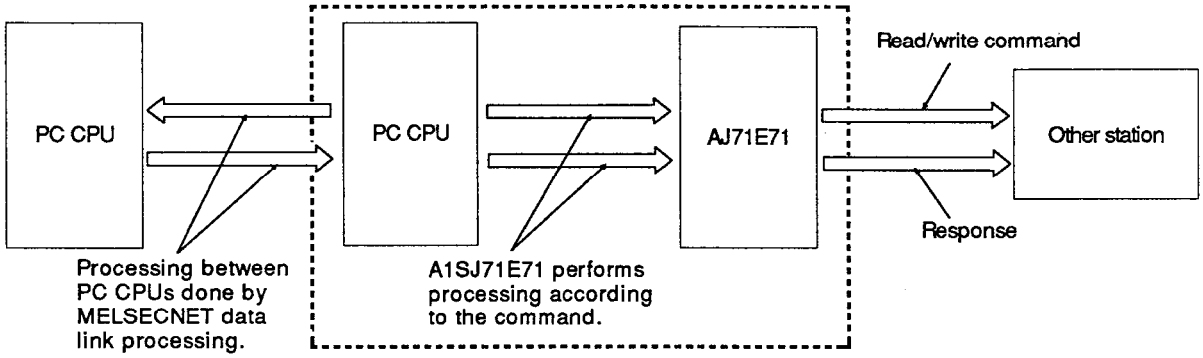
o ..... All devices can be accessed by setting the PC number of a PC.

o*1... A special-function module buffer memory can be acessed by setting the PC number of a PC.

POINT

Communications with A0J2CPUP23/R23 and A0J2P25/R25 cannot be done.

(5) When data in the PC CPU of the other in MELSECNET is read and written, data flows as follows:



(6) Transmission time through MELSECNET

(a) When the PC number is specified for a PC CPU without A1SJ71E71 on MELSECNET, and a data transmission is done, transmission time (T_1) is as follows:

- Local station

  Transmission time (T_1) = (LRDP instruction processing time) + (Scan time with A1SJ71E71) × $\frac{1}{\_\_}$

- Remote I/O station

  Transmission time (T_1) = (1 RFRP instruction processing time + MELSECNET master station scan time) × $\frac{1}{\_\_}$

  As for __ in the above-mentioned formula, when a communications is first done at the time of a power on or after CPU reset, use "3".

- Factor that delays transmission time (T_1)

  When the transmission time of a certain command (writing of device R, etc.) needs 2 scans, the transmission time becomes the double of the above-mentioned value.

  When the other of the link is being monitored by using A6GPP, the transmission time becomes the double per monitor.

- When a device is read through MELSECNET, lengthen the monitoring time of the ACPU watchdog timer of the other.

* As fo details of data link, refer to MELSECNET (II) Reference Manual

Example: When the MELSECNET master station is loaded with A1SJ71E71, and the device memory of a local station is read

(Condition L LS M, M: 80 ms, α1: 10 ms)

Transmission time (T_1) =

$(M \times 4 + \alpha_1 \times 4 + M) = (80 \times 4 + 10 \times 4 + 80) \times 1 = 440$

T_1 is 880 ms.

M : MELSECNET master station scan time

α_1 : MELSECNET master station link refresh time

LS : Link scan time

L : MELSECNET local station scan time

POINT

A condition slows the data transmission time of a PC CPU without A1SJ71E71 on MELSECNET considerably.

Use only station (PC number FF_H) with A1SJ71E71 for the communications between the other and a PC CPU, and use data link (B, W) of MELSECNET for the communications with other PC CPU. As a result a transmission delay time decreases.

8.2 Communicating Data

When read/write of data in the PC CPU is done, table 8.1 shows data that can be read and written from the other.

Table 8.1 Communicating Data

Function			Command/ Response Classification	Description	Number of Point Processed per Communications	PC CPU State			Reference
						During STOP	During RUN SW22 ON	During RUN SW22 OFF	
Device memory	Batch read	Bit units	00H	Reads bit devices (such as X, Y, M) in units of 1 device.	256 points	o	o	o	8.4.2
		Word units	01H	Reads bit devices (such as X, Y, M) in units of 16 devices.	128 words (2048 points)				8.4.3
				Reads word devices (such as D, R, T, C) in units of 1 device. *2	256 points				
	Batch write	Bit units	02H	Writes bit devices (such as X, Y, M) in units of 1 device.	256 points	o	o	x	8.4.4
		Word units	03H	Writes bit devices (such as X, Y, M) in units of 16 devices.	40 words (640 points)				8.4.5
				Writes word devices (such as D, R, T, C) in units of 1 device.	256 points				
	Test (random write)	Bit units	04H	Specifies bit devices (such as X, Y, M) and device number in units of 1device at random and sets/resets the device.	80 points	o	o	x	8.4.6
		Word units	05H	Specifies bit devices (such as X, Y, M) and device number in units of 16 devices at random and sets/resets the device.	40 words (640 points)				8.4.7
				Specifies word devices (such as D, R, T, C) and device number in units of 1 device at random and sets/resets the device.	40 points				
	Monitor data registration	Bit units	06H	Sets bit devices to be monitored (such as X, Y, M) in units of 1 device.	40 points *1	o	o	o	8.4.8
		Word units	07H	Sets bit devices to be monitored (such as X, Y, M) in units of16 devices.	20 words *1 (320 points)				
				Sets word devices to be monitored (such as D, R, T, C) in units of 1 device.	20 points				
	Monitor	Bit units	08H	Reads data from devices for which device data has been registered.	—	o	o	o	
		Word units	09H						
Extension file register	Batch read		17H	Reads extension file registers (R) in units of 1 register.	256 points	o	o	o	8.5.3
	Batch write		18H	Writes extension file registers (R) in units of 1 register.	256 points	o	o	x	8.5.4
	Test (random write)		19H	Specifies the extension file registers (R) in units of 1 register using block or device number and makes a random write.	40 points	o	o	x	8.5.5
	Monitor data registration		1AH	Sets the extension file registers (R) device number and makes a random write.	20 points	o	o	o	8.5.6
	Monitor		1BH	Monitors the extension file register (R) after monitor data registration.	–	o	o	o	
	Direct read		3BH	Reads extension file registers (R) in units of 1 register.	256 points	o	o	o	8.5.7
	Direct write		3CH	Writes extension file registers (R) in units of 1 register.	256 points	o	o	x	8.5.7
Special function module	Batch read		0EH	Reads the contents of the special function module buffer memory.	256 points (128 words)	o	o	o	8.6.3
	Batch write		0FH	Writes the contents of the special function module buffer memory.		o	o	x	8.6.4

o : Available
x : Unavailable

*1 When a CPU except A2AS and A3H, A3M, AnA, AnU is used, the number of device points processed at one time is the half of the above-mentioned value of device X (input).
*2 When read/write of an extensive file register is done, use a dedicated instruction of an extensive register.

Table 8.1 Communicating Data (Continued)

Function				Command/ Response Classification	Description	Number of Point Processed per Communications	PC CPU State			Reference
							During STOP	During RUN SW22 ON	During RUN SW22 OFF	
Sequence Program	Batch read	Main	Sequence program	0A$_H$	Reads main sequence programs.	256 steps	o	o	o	8.8.4
			T/C set value		Reads T/C set values used in main sequence programs.	256 points				
		Sub	Sequence program	0B$_H$	Reads subsequence programs.	256 steps	o	o	o	
			T/C set value		Reads T/C set values used in subsequence programs.	256 points				
	Batch write	Main	Sequence program	0C$_H$	Writes main sequence programs.	256 steps	o	o*3	x	
			T/C set value		Writes T/C set values used in main sequence programs.	256 points	o	o	x	
		Sub	Sequence program	0D$_H$	Writes subsequence programs.	256 steps	o	o*3	x	
			T/C set value		Writes T/C set values used in subsequence programs.	256 points	o	o	x	
Micro comuter program	Batch read		Main	1E$_H$	Reads main micro-computer programs.	256 bytes	o	o	o	8.8.5
			Sub	1F$_H$	Reads submicro-comuter programs.					
	Batch write		Main	20$_H$	Writes main micro-computer programs.		o	o*3	x	
			Sub	21$_H$	Writes submicro-computer programs.					
Comment	Batch read			1C$_H$	Reads comment data.	256 bytes	o	o	o	8.8.6
	Batch write			1D$_H$	Writes comment data.		o	o	x	
Extension file register	Direct read			39$_H$	Reads the extension comment data.	256 bytes	o	o	o	8.8.7
	Direct write			3A$_H$	Writes the extension comment data.	256 bytes	o	o	x	8.8.7
Parameter	Batch read			10$_H$	Reads parameters from PC CPU.	256 bytes	o	o	o	8.8.3
	Batch write			11$_H$	Writes parameters to PC CPU.		o	x	x	
	Analysis request			12$_H$	Causes PC CPU to acknowledge and check rewritten parameters.	—	o	x	x	
PC CPU	Remote RUN			13$_H$	Request remote run/stop of PC CPU.	—	o	o	o	8.7. 2
	Remote STOP			14$_H$						
	PC CPU read			15$_H$	Reads the type of PC CPU: A1N, A2N, A3N, A3H		o	o	o	8.7.8
Loopback test				16$_H$	Echoes unchanged characters back to the computer.	256 bytes	o	o	o	8.9

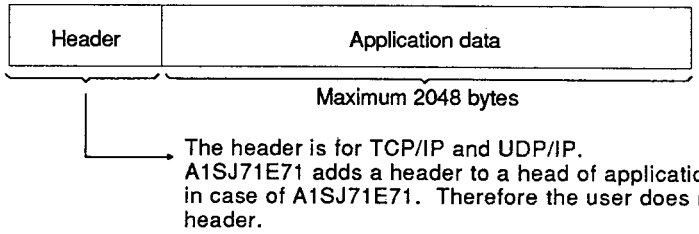
o : Available
x : Unavailable

*3 When all the following conditions are satisfied, a program can be written during RUN:
(1) A PC CPU must be A3, A3N, A3H, A3M, A3A, A3U or A4U.
(2) A program must be the program that is not used at present.
(The main program is a subprogram during RUN.)
(3) A special relay of a PC PCU must be the following state
(a) M9050 (signal flow exchange contact) ..... OFF (only A3CPU)
(b) M9051 (CHG instruction execution disabled) ..... ON

8.3 Data Format

When communications between A1SJ71E71 and a communicating node is done, use the following data format.

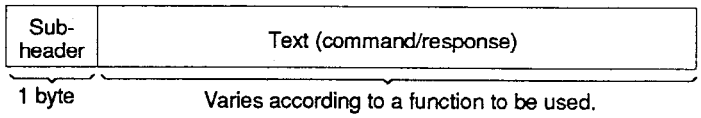
As shown below, communication data is composed of a header and application data.



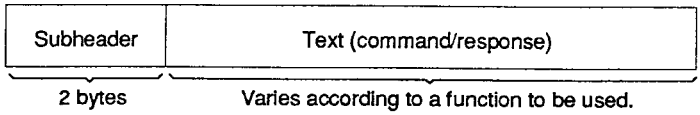
8.3.1 Application data format

As shown below, the application data format is divided into subheader and a text.
A subheader sets the type of each function.
The set value is decided on according to a function to be used.
The text sets request data (command) and reply data (response) for each function.
Set data by using a specified format.
As for the details,. refer to Section 8.4 and after.

Communications in binary code



Communications in ASCII code



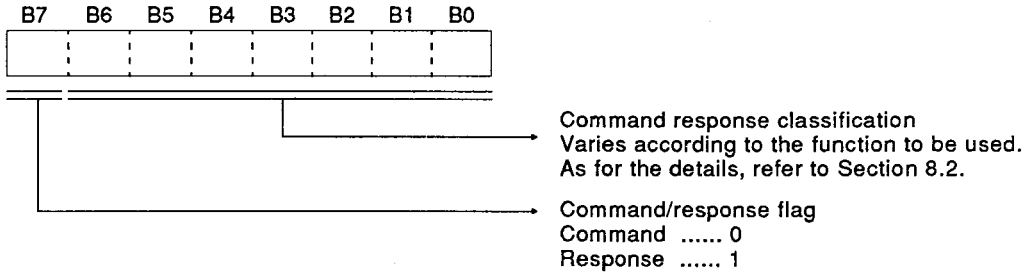
REMARK

The response for the command from the other node is set automatically in A1SJ71E71 in case of data read/write in the PC CPU.
Therefore the user does not have to set this.

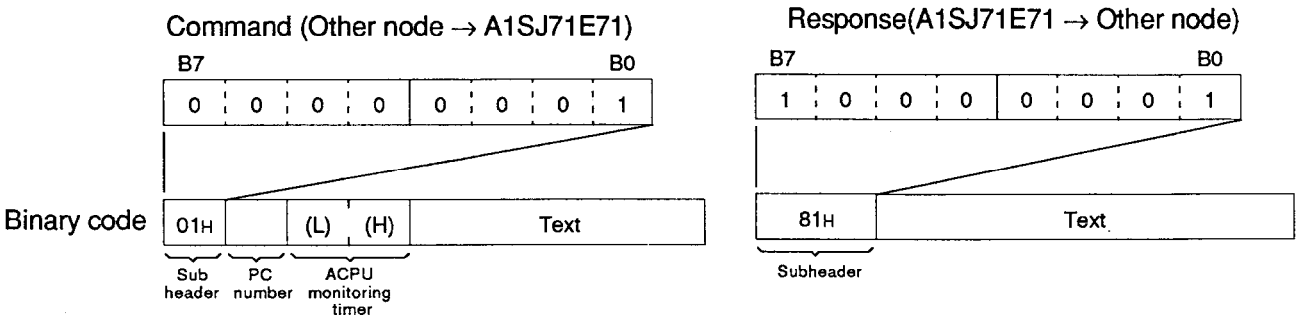
8.3.2 Subheader

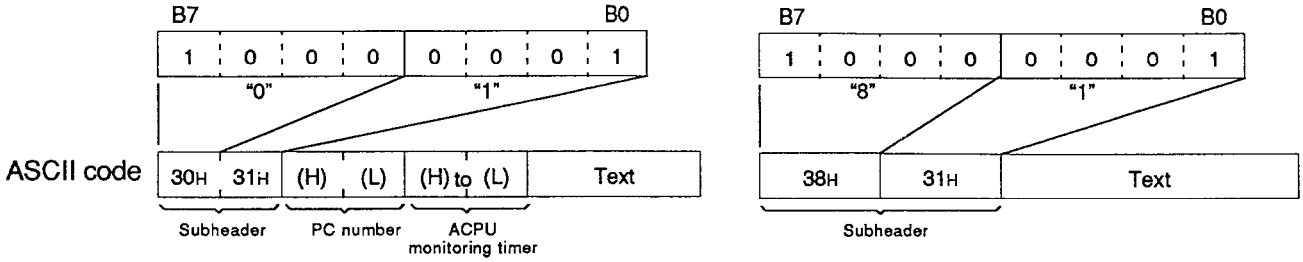
The format of a subheader is composed as follows:

When A1SJ71E71 sends a response back for a communicating node, A1SJ71E71 sets response data automatically. Therefore the user does not have to set this.



Example: When batch read of a device memory is done in a word units (Command/response classification of a device memory batch read (word units) ...01H)





POINTS

- When the ACPU monitoring timer value is set to "0", the CPU becomes an infinite wait state.
- When a data communications with the other in MELSECNET is done, refer to Section 8.1.2.

8.3.3 Completion code

When data in the PC CPU is read and is written, the completion code that A1SJ71E71 adds to a response by a communications between a other node and A1SJ71E71 is shown as follows:

Table 8.2 Completion code

<table>
<tr><th>Completion Code</th><th>Description</th><th>Corrective Action</th></tr>
<tr><td>00H</td><td>Normal completion</td><td>—</td></tr>
<tr><td>50H</td><td>A command/response classification of a subheader is the code that is not specified.
<table><tr><th>Communications Processing</th><th>Command/Response Classification</th></tr><tr><td>Communications of a fixed buffer</td><td>60H</td></tr><tr><td>Communications of a random access buffer</td><td>61, 62H</td></tr><tr><td>Read/write of data in the PC CPU</td><td>00H to 3CH</td></tr></table></td><td>Confirm and modify a command/response classification set in the other node.
(A1SJ71E71 adds a command/response classification automatically. Therefore setting by a user is unnecessary.)
See the REMARK in Section 9.1.14.</td></tr>
<tr><td>54H</td><td>The ASCII code that cannot be translated into a binary code when setting the data code of A1SJ71E71 in the ASCII code has been transmitted from the other node.</td><td>Confirm and correct the send data of a other node.</td></tr>
<tr><td>55H</td><td>• When a DIP switch SW3 in front of A1SJ71E71 was OFF state (it is write-disabled during RUN), data was written in a PC CPU during PC CPU RUN from the other node.
• A parameter, a sequence program and a microcomputer program were written during PC CPU RUN from the other node.
(The ON/OFF state of a DIP switch (SW3) in front of A1SJ71E71 is not related.)</td><td>• Set SW3 to the ON (write-enabled during RUN). Then, write data. However, a parameter, a sequence program and a microcomputer program cannot be written in a CPU during RUN.
• Stop a PC CPU. Then, write data.</td></tr>
<tr><td>56H</td><td>• There is an error in device specification done from the other node.</td><td>Refer to Section 8.4.1(2), and correct device specification.</td></tr>
<tr><td rowspan="4">57H</td><td>• A number of device points processed specified by a command transmitted from the other node is larger than the maximum number of device points processed of the processing.
• The sum of a head address (head device number and head step number) and a specified number of device points processed is larger than the maximum address (device number and step number) of the processing.</td><td>• Correct a number of specified points or a head address (device number and step number).</td></tr>
<tr><td>• Length of the byte of a command is not specified length.
• When data is written, a set number of write data points is different from a specified number of device points processed.</td><td>• Confirm the data length of a command, and set data again.</td></tr>
<tr><td>• Though monitoring data is not registered, monitoring was done.</td><td>• Register monitoring data.</td></tr>
<tr><td>• The final address of a parameter setting range was specified at the time of read/write of a microcomputer program.</td><td>• Read/write of a final address cannot be done.
Correct a specified address.</td></tr>
<tr><td></td><td>• When specifying block No. of an extensive file register, block No. of the range that exceeds a memory cassette capacity was specified.</td><td>Correct specification of block No.</td></tr>
</table>

Table 8.2 Completion code (Continued)

Completion Code	Description	Corrective Action
58H	• Specification of a head address (head address number and head step number) of a command transmitted from the other node has exceeded the range that can be specified. • When a microcomputer program and file register (R) were read and were written, a value outside the parameter setting range of a PC CPU was specified.	• Correct a head address to the value within a range that is possible to set each processing.
	• Block No. of an extensive file register is set at block No. that does not exist.	• Correct specification of block No..
	• File register (R) was specified for the A1(N) CPU.	The file register cannot be used in the A1(N) CPU.
	• A word device was specified by using a bit device command. • A head number of a bit device was specified for a value except the multiple of 16 by using a word device command.	• Correct a command or a specified device.
59H	• Read/write of an extensive file register was done for the A1(N) CPU.	An extensive file register cannot be used in the A1(N) CPU.
5BH	• The PC CPU and A1SJ71E71 cannot be • communicated. A PC CPU could not be processed for the request from the other node.	• Confirm an error code to be added after a completion code, and correct an error place. Refer to Section 8.3.4.
60H	• The communicating time of A1SJ71E71 and a PC CPU exceeded the ACPU watchdog timer value.	Lengthen the ACPU watchdog timer value.

8.3.4 Error codes

When the PC CPU fails to execute processing in request to from a communicating node, the following error codes are detected by the PC CPU.

As shown below, the A1SJ71E71 adds an error code after the completion code of a response and transmits it to the requesting node.

The error code is added only when a completion code is "5B".

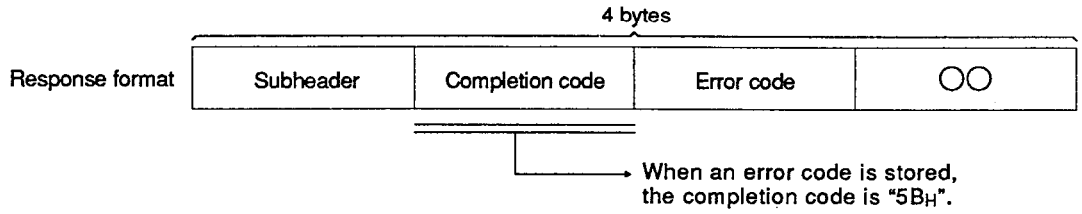


Table 8.3 Error Codes

Error Code (Hexadecimal)	Error Item	Error Description	Corrective Action
10_H	PC number error	The station of a specified number PC does not exist. (1) The PC number specified using a command is except "FF" for the self or except a station number set in the MELSECNET link parameter.	(1) Change the PC number to "FF" for the self or a station number set in a link parameter, and retry communications.
11_H	Mode error	Communications fault between A1SJ71E71 and PC CPU (1) After the A1SJ71E71 received a request from a communicating station normally, communications between the A1SJ71E71 and the PC CPU was not done normally by some causes (noise, etc.).	(1) Retry communications again. When an error occurs again, check noise, etc., and replace the A1SJ71E71. Then, retry communications.
12_H	Special-function module specification error	Special-function module error (1) A special-function module which has buffer memory for communications was not specified. (For example, an input/output module is loaded in the specified slot or the specified slot is a vacant slot.)	(1) Change the specification of the protocol. Or change the loading position of a special-function module, and retry communications.
13_H	Program step No. specification error	Sequence program step No. specification error (1) Specified step No. exceeds the range of a program capacity set in the parameter of a PC CPU.	(1) Specify step No. within a set range. Or change the parameter setting of the PC CPU, and retry communications.
18_H	Remote error	Remote RUN/STOP is disabled. Other modules (other A1SJ71E71s, etc.) have already done a remote STOP/PAUSE.	(1) Check of any other module is not doing a remote STOP/PAUSE. Cancel a remote STOP/PAUSE, and retry communications.
20_H	Link error	The CPU module at the request destination is disconnected from the data link.	Check if the PC CPU with the station number set for the PC number is disconnected. Eliminate the cause of the disconnection and reattempt communications.
21_H	Special-function module bus error	The memory of a special-function module is not accessible. (1) Control bus error with a special-function module (2) A special-function module is malfunctioning.	Hardware error of the PC CPU, base unit, special-function module or A1SJ71E71. Consult Mitsubishi representatives.

8.4 Command/Response Format for Read/Write of Device Memory

The following explains the method of control when a device memory is read and written.

8.4.1 Command and device range

(1) Functions to be used for the read/write of device memory

Table 8.4 Functions

Item		Command/ Response Classification	Description	Number of Points Processed per Communications	PC CPU State		
					During STOP	During RUN	
						SW22 ON	SW22 OFF
Batch read	Bit units	00H	Reads bit devices (such as X, Y, M) in units of 1 device.	256 points	o	o	o
	Word units	01H	Reads bit devices (such as X, Y, M) in units of 16 devices.	128 words (2048 points)			
			Reads word devices (such as D, R, T, C) in units of 1 device.	256 points			
Batch write	Bit units	02H	Writes bit devices (such as X, Y, M) in units of 1 device.	256 points	o	o	x
	Word units	03H	Writes bit devices (such as X, Y, M) in units of 16 devices.	40 words (640 points)			
			Writes word devices (such as D, R, T, C) in units of 1 device.	256 points			
Test (random write)	Bit units	04H	Specifies bit devices (such as X, Y, M) and device number in units of 1 device at random and sets/resets the device.	80 points	o	o	x
	Word units	05H	Specifies bit devices (such as X, Y, M) and device number in units of 16 devices at random and sets/resets the device.	40 words (640 points)			
			Specifies word devices (such as D, R, T, C) and device number in units of 1 device at random and sets/resets the device.	40 points			
Monitor data registration	Bit units	06H	Sets bit devices to be monitored (such as X, Y, M) in units of 1 device.	*40 points	o	o	o
	Word units	07H	Sets bit devices to be monitored (such as X, Y, M) in units of 16 devices.	*20 words (320 points)			
			Sets word devices to be monitored (such as D, R, T, C) in units of 1 device.	20 points			
Monitor	Bit units	08H	Reads data from devices for which device data has been registered.	—	o	o	o
	Word units	09H					

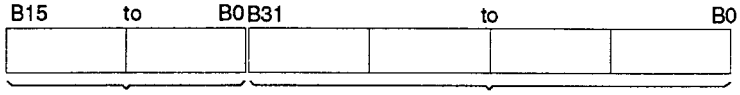
Note : o ... Executable
x ... Not executable

Number of device points processed marked with *
The number of device points processed per point of device X (input) is 2 points in case of a CPU except A2AS, A3H, A3M, AnA and AnU is used.
For example, when device X is included in the specified devices when the monitoring data registration is done with bit unit.

(Number of specified points of X) × 2) +
(Number of specified points of other devices) ≤ 40 points

(2) Method of specifying a device and range

(a) As for the method of setting each device, use a device code and a device number as follows:



(set in hexadecimal.)

Example: D100

D100 = 4420 00000064H

Device number (hexadecimal)

Device code

(b) Table 8.5 shows device codes and device numbers.

Table 8.5 Device

Device		Device Code	Device Range	Device Number	Applicable CPU							+
					A1S A1SJ A1 A1N	A1S-S1 A2S A2 A2N A2C A0J2H	A2S-S1 A2-S1 A2N-S1	A3 A3N	A3M	A2AS A2A A2U	A2AS-S1 A2A-S1 A2U-S1	A3A A3AU A4U
Data register		D0 (44_H, 20_H)	D0 to D1023	0000h to 03FFh	o	o	o	o	o			
			D0 to D6143	0000h to 17FFh						o	o	o
			D9000 to D9255	2328h to 2427h	o	o	o	o	o	o	o	o
Link register		W0 (57_H, 20_H)	W0 to W3FF	0000h to 03FFh	o	o	o	o	o			
			W0 to WFFF	0000h to 0FFFh						o	o	o
File register		R0 (52_H, 20_H)	R0 to R4095	0000h to 0FFFh		o	o	o	o	o	o	o
			R4096 to R8191	1000h to 1FFFh				o	o	o	o	o
Timer	Present value	TN (54_H, $4E_H$)	T0 to T255	0000h to 00FFh	o	o	o	o	o			
			T0 to T2047	0000h to 07FFh						o	o	o
	Contact	TS (54_H, 53_H)	T0 to T255	0000h to 00FFh	o	o	o	o	o			
			T0 to T2047	0000h to 07FFh						o	o	o
	Coil	TC (54_H, 43_H)	T0 to T255	0000h to 00FFh	o	o	o	o	o			
			T0 to T2047	0000h to 07FFh						o	o	o

Table 8.5 Device (Continued)

Device		Device Code	Device Range	Device Number	Applicable CPU							+
					A1S A1SJ A1 A1N	A1S-S1 A2S A2 A2N A2C A0J2H	A2S-S1 A2-S1 A2N-S1	A3 A3N	A3M	A2AS A2A A2U	A2AS-S1 A2A-S1 A2U-S1	A3A A3AU A4U
Counter	Present value	CN (43_H, $4E_H$)	C0 to C255	0000h to 00FFh	o	o	o	o	o			
			C0 to C1023	0001h to 03FFh						o	o	o
	Contact	CS (43_H, 53_H)	C0 to C255	0000h to 00FFh	o	o	o	o	o			
			C0 to C1023	0000h to 03FFh						o	o	o
	Coil	CC (43_H, 43_H)	C0 to C255	0000h to 00FFh	o	o	o	o	o			
			C0 to C1023	0000h to 03FFh						o	o	o
Input		X0 (58_H, 20_H)	X0 to X0FF	0000h to 00FFh	o							
			X0 to X1FF	0000h to 01FFh		o				o		
			X0 to X3FF	0000h to 03FFh			o				o	
			X0 to X7FF	0000h to 07FFh			o	o				o
Output		Y0 (59_H, 20_H)	Y0 to Y0FF	0000h to 00FFh	o							
			Y0 to Y1FF	0000h to 01FFh		o				o		
			Y0 to Y3FF	0000h to 03FFh			o				o	
			Y0 to Y7FF	0000h to 07FFh			o	o				o
Internal relay		M0 ($4D_H$, 20_H)	M(L, S)0 to M(L, S) 2047	0000h to 07FFh	o	o	o	o	o			
			M(L, S)0 to M(L, S) 8191	0000h to 01FFh						o	o	o
			M9000 to M9255	2328h to 2427h	o	o	o	o	o	o	o	o
Link relay		B0 (42_H, 20_H)	B0 to B3FF	0000h to 03FFh	o	o	o	o	o			
			B0 to BFFF	0000h to 0FFFh						o	o	o
Annunciator		F0 (46_H, 20_H)	F0 to F255	0000h to 00FFh	o	o	o	o	o			
			F0 to F2047	0000h to 07FFh						o	o	o

POINTS

(1) Bit devices and word devices are classified as follows:

Bit device·····X, Y, M, L, B, F, T (contact), T (coil),C (contact), and C (coil)

Word device···T (present value), C (present value), D, W, and R

(2) The device number of a bit device must be divisible by 16 in case of word unit specification.

(3) Special relay (M9000 to M9255) and special registers (D9000 to D9255) are classified into the read, write and system areas.

When writing is done to outside a write-enabled range, an error of the PC CPU sometimes occurs.

As for the details of special relays and special registers, see the ACPU Programming Manual.

(4) When doing read/write of a file register with a PC CPU using an extension file register

Use a command explained in Section 8.5 Command/Response Format for Read/Write of Extension File Registers.

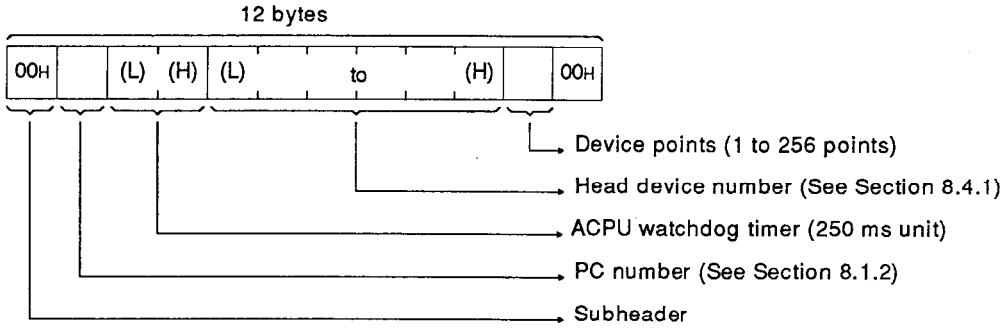
When an extension file register is used and if device batch read/write processing is done with a file register, read/write may not be done normally.

8.4.2 Batch read in bit unit

When batch read of bit device memory is done, the command and response format are as follows:

(1) Communications in binary code

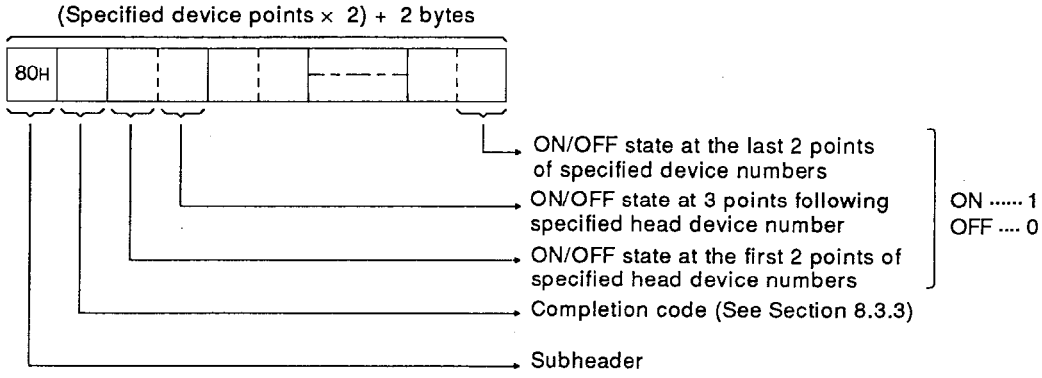
Command format



REMARK

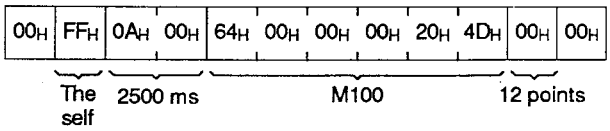
When device points is specified to 256 points, set "00_H".

Response format

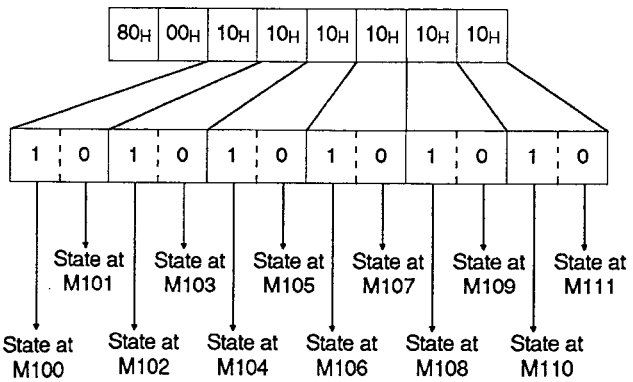


Example: When reading the ON/OFF state of M100 to M111 of a PC CPU loaded with A1SJ71E71

Command (Other node → A1SJ71E71)

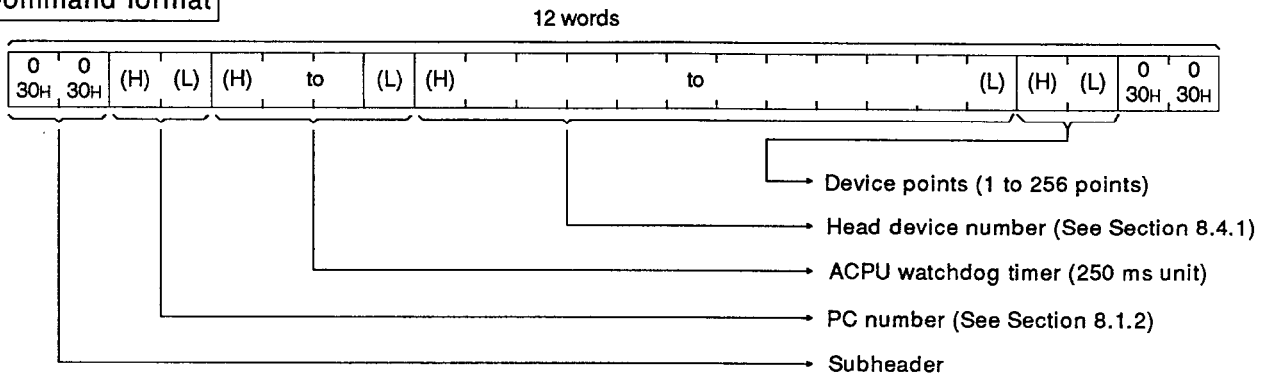


Response (A1SJ71E71 → Other node)



(2) Communications in ASCII code

Command format

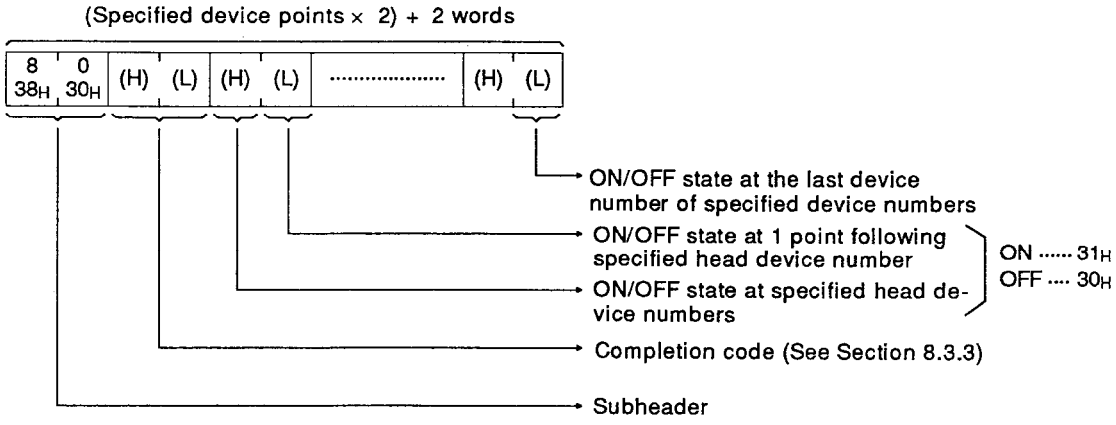


REMARKS

(1) When device points is specified to 256 points, set "3030H".

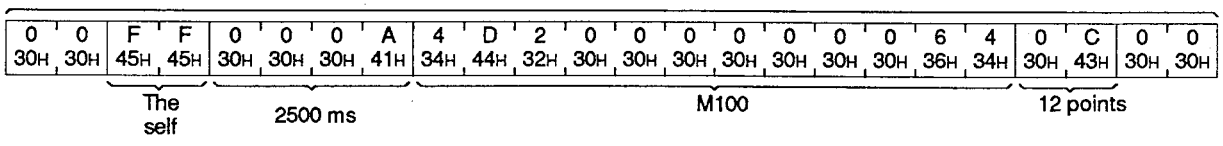
(2) If the number of device points specified is an odd number, 1 byte of dummy data (30H) will be added to the response data. For example, if three points are read, data equivalent to 4 points will be returned. The final byte of this data will be dummy data.

Response format

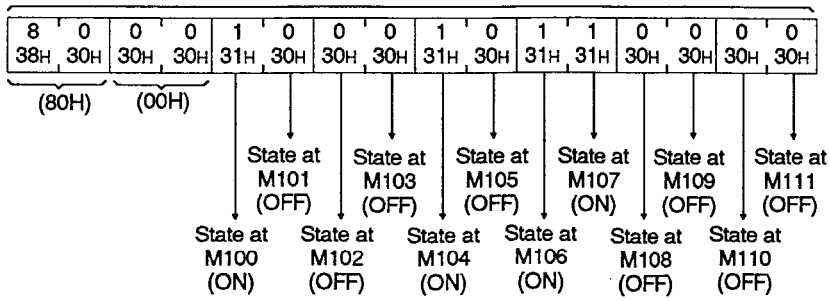


Example: When reading the ON/OFF state of M100 to M111 of a PC CPU loaded with A1SJ71E71

Command (Other node → A1SJ71E71)



Response (A1SJ71E71 → Other node)

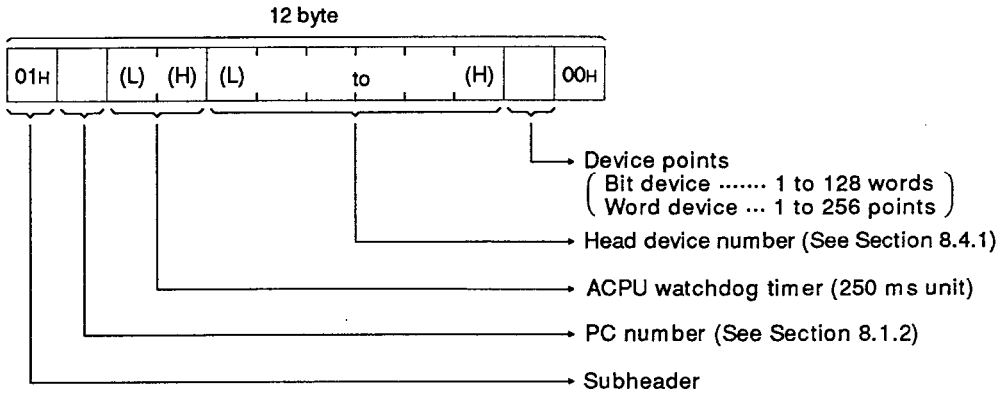


8.4.3 Batch read in word unit

When batch read of word device memory and batch read of bit device memory (16 points unit) is done, the command and response format are as follows:

(1) Communications in binary code

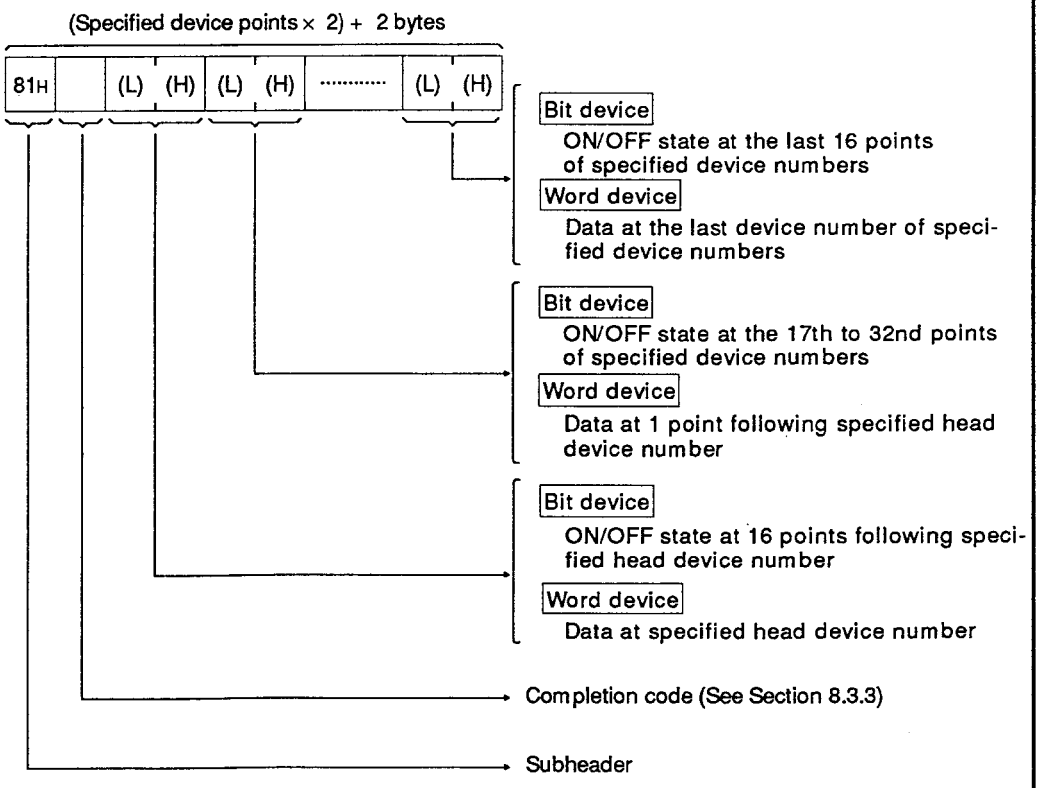
Command format



REMARK

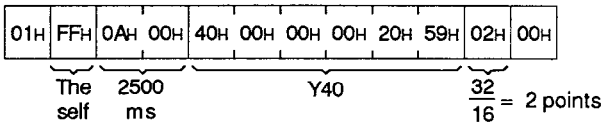
When device points is specified to 256 points, set "00_H".

Response format

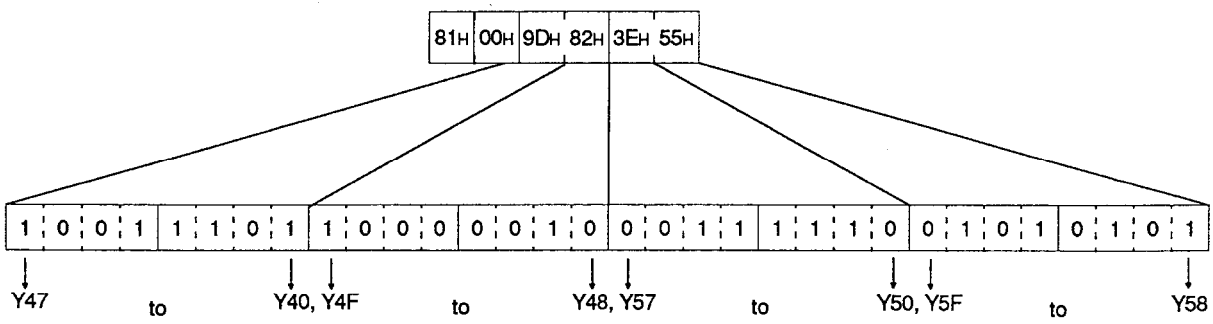


Example 1: When reading the ON/OFF state of Y40 to 5F (32 points) of a PC CPU loaded with A1SJ71E71

Command (Other node → A1SJ71E71)

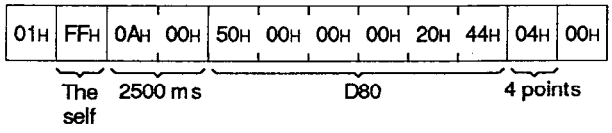


Response (A1SJ71E71 → Other node)

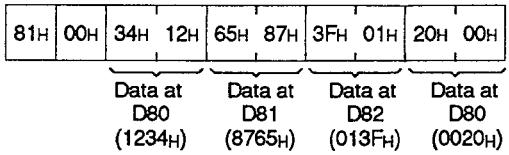


Example 2: When reading the data of D80 to D83 of a PC CPU loaded with A1SJ71E71

Command (Other node → A1SJ71E71)

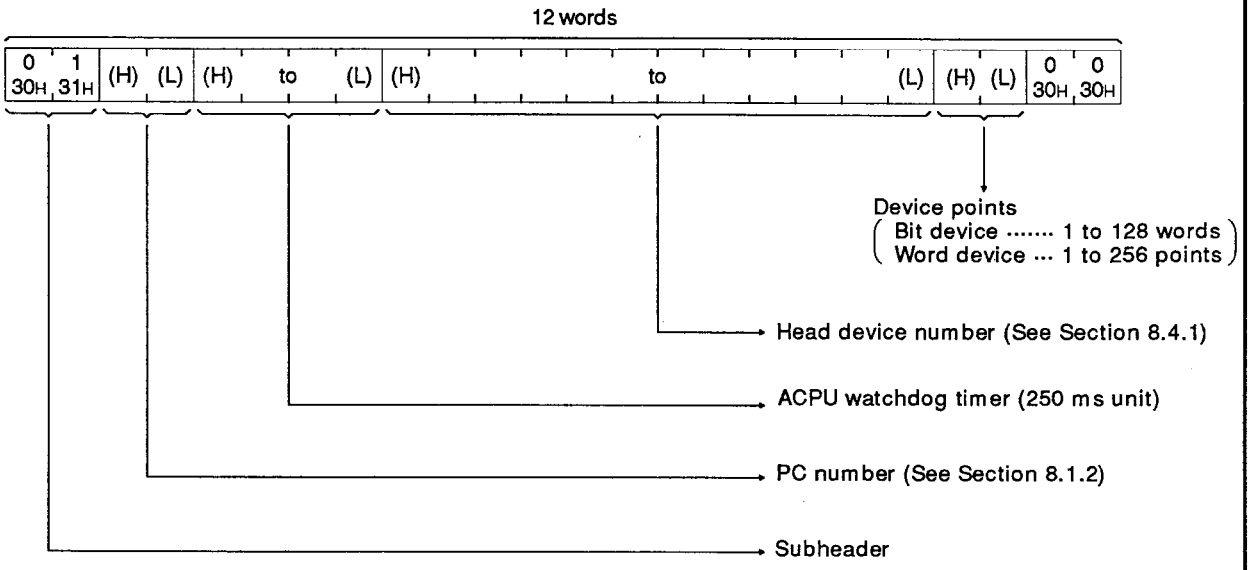


Response (A1SJ71E71 → Other node)



(2) Communications in ASCII code

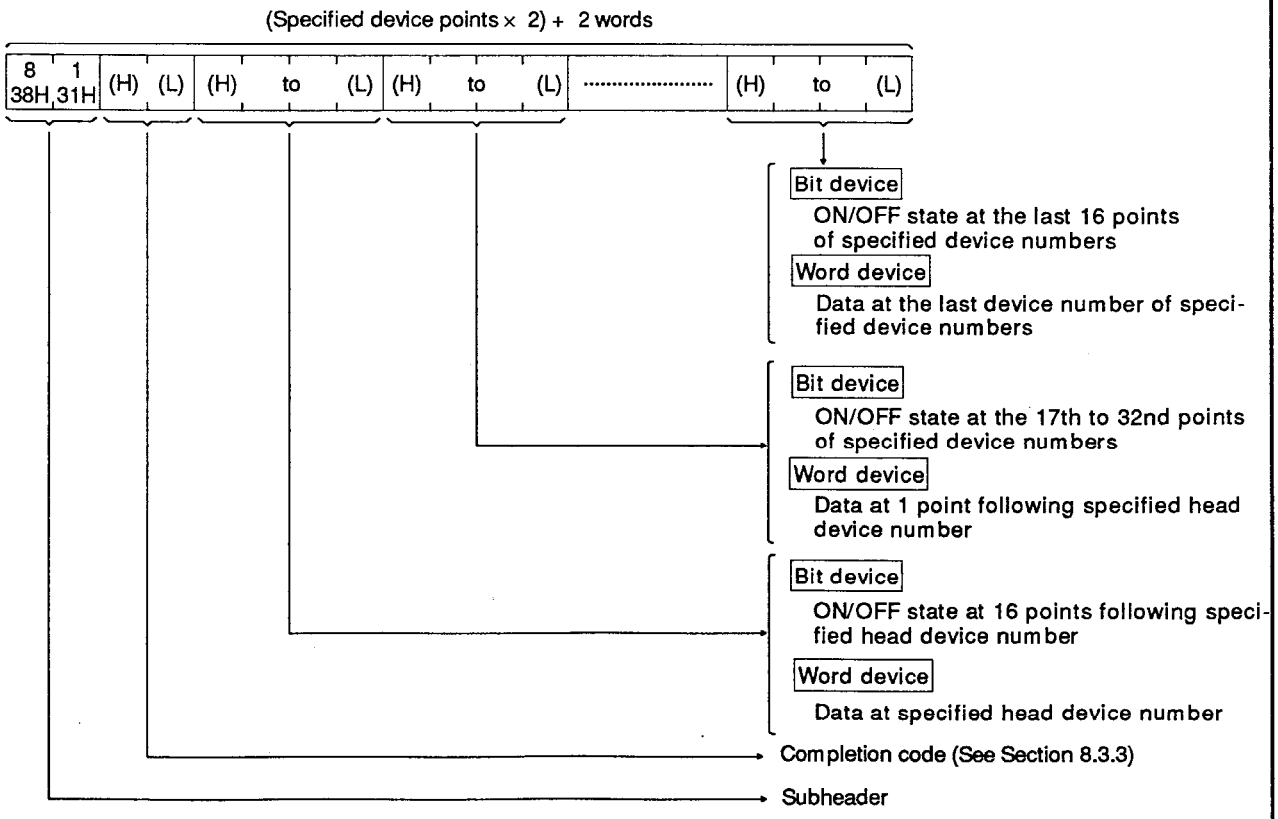
Command format



REMARK

When device points is specified to 256 points, set "3030_H".

Response format



Example 1: When reading the ON/OFF state of Y40 to 5F (32 points) of a PC CPU loaded with A1SJ71E71

Command (Other node → A1SJ71E71)

0	1	F	F	0	0	0	A	5	9	2	0	0	0	0	0	0	0	4	0	0	2	0	0
30H	31H	46H	46H	30H	30H	30H	41H	35H	39H	32H	30H	30H	30H	30H	30H	30H	30H	34H	30H	30H	32H	30H	30H

The self — 2500 ms — Y40 — $\frac{32}{16}$ = 2 points

Response (A1SJ71E71 → Other node)

8	1	0	0	8	2	9	D	5	5	3	E
38H	31H	30H	30H	38H	32H	39H	44H	35H	35H	33H	45H

"8"				"2"				"9"				"D"				"5"				"5"				"3"				"E"			
1	0	0	0	0	0	1	0	1	0	0	1	1	1	0	1	0	1	0	1	0	1	0	1	0	0	1	1	1	1	1	0

Y4F to Y48, Y47 to Y40, Y5F to Y58, Y57 to Y50

Example 2: When reading the data of D80 to D82 of a PC CPU loaded with A1SJ71E71

Command (Other node → A1SJ71E71)

0	1	F	F	0	0	0	A	4	4	2	0	0	0	0	0	0	0	5	0	0	3	0	0
30H	31H	46H	46H	30H	30H	30H	41H	34H	34H	32H	30H	30H	30H	30H	30H	30H	30H	35H	30H	30H	33H	30H	30H

The self — 2500 ms — D80 — 3 points

Response (A1SJ71E71 → Other node)

8	1	0	0	1	2	3	4	8	7	6	5	0	1	3	F
38H	31H	30H	30H	31H	32H	33H	34H	38H	37H	36H	35H	30H	31H	33H	46H

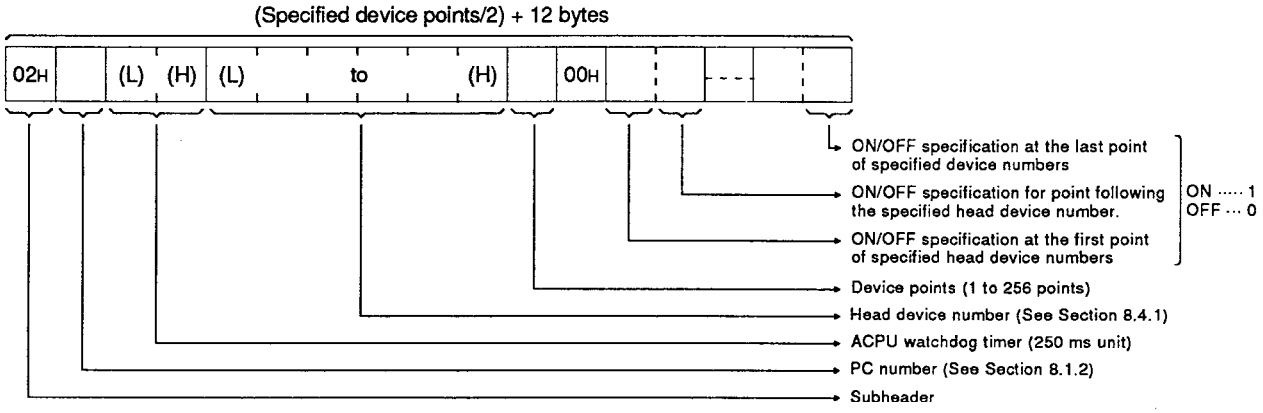
Data at D80 (1234_H) — Data at D81 (8765_H) — Data at D82 ($013F_H$)

8.4.4 Batch write in bit units

When batch write of bit device memory is done, the command and response format are as follows:

(1) Communications in binary code

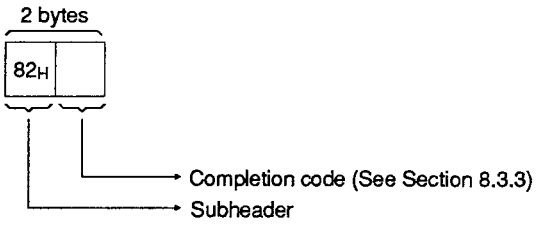
Command format



REMARK

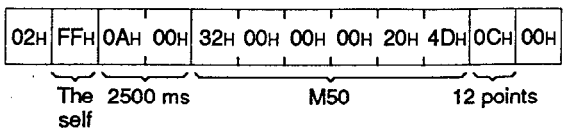
When device points is specified to 256 points, set "00_H".

Response format

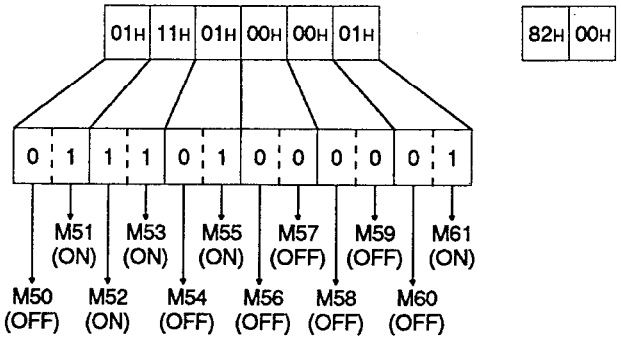


Example: When writing the ON/OFF data to M50 to M61 of a PC CPU loaded with A1SJ71E71

Command (Other node → A1SJ71E71)

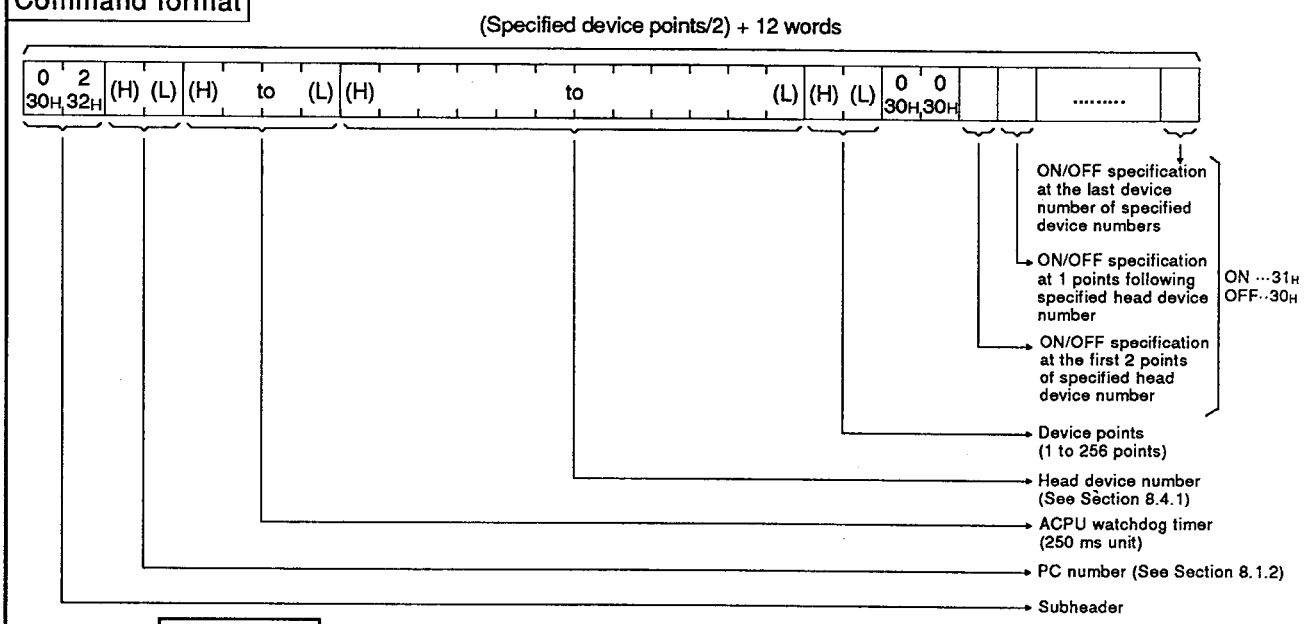


Response (A1SJ71E71 → Other node)



(2) Communications in ASCII code

Command format

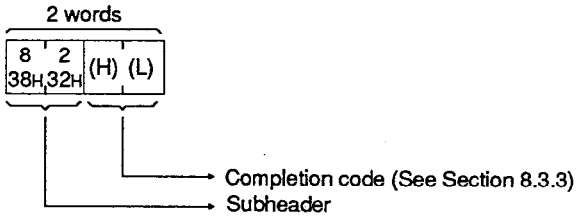


REMARKS

(1) When device points is specified to 256 points, set "3030_H".

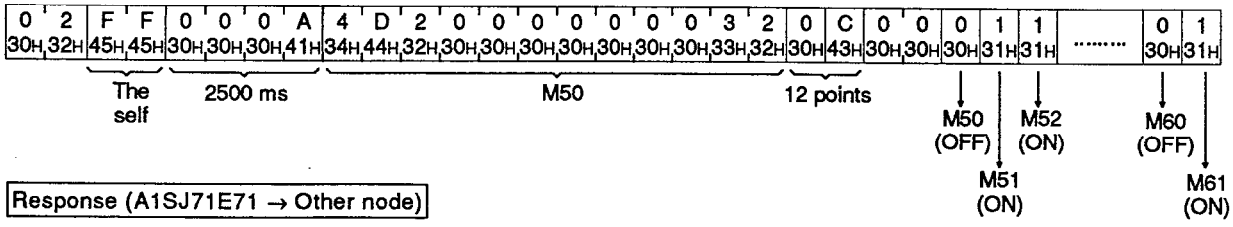
(2) If the number of device points specified is an odd number, 1 byte of dummy data (30_H) will be added as the last byte of the write data. For example, if three points are written, one byte of dummy data (30_H) will be added.

Response format

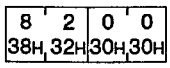


Example: When writing the ON/OFF data to M50 to M61 of a PC CPU loaded with A1SJ71E71

Command (Other node → A1SJ71E71)



Response (A1SJ71E71 → Other node)

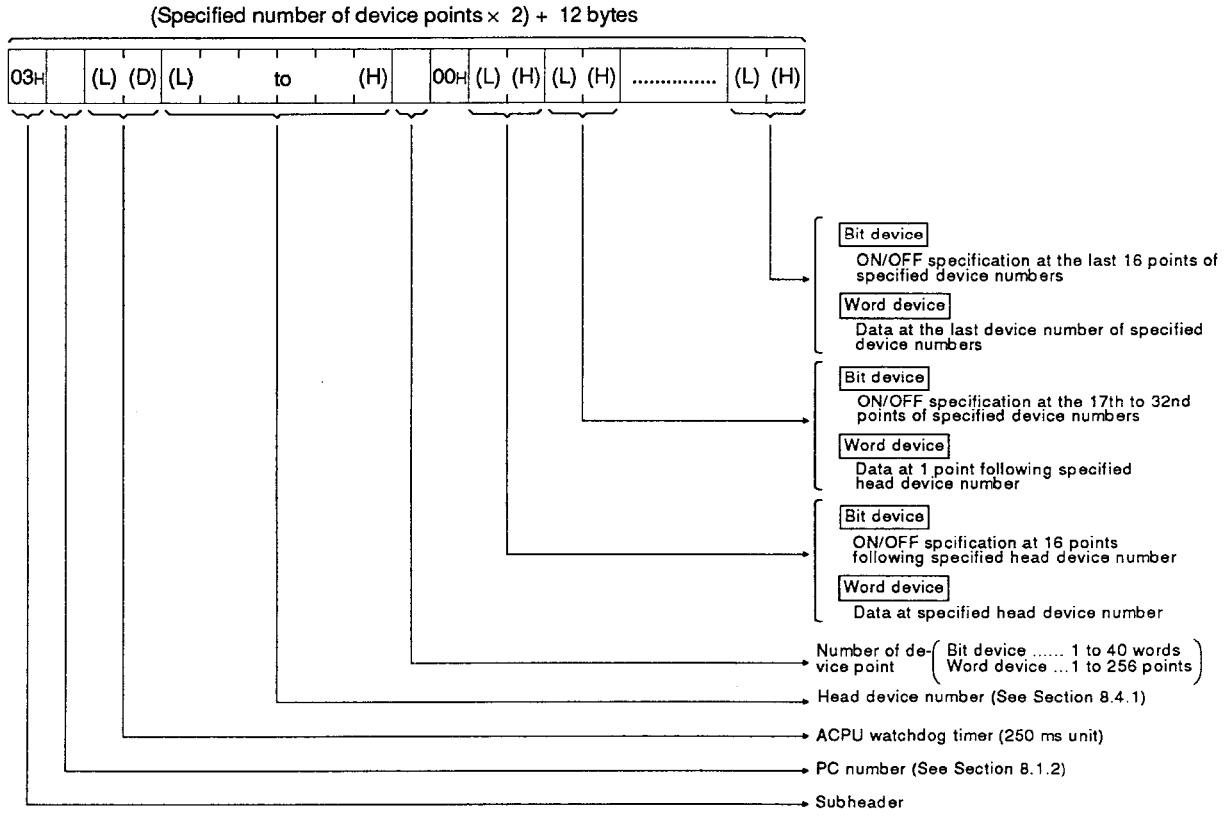


8.4.5 Batch write in word units

The command and response formats are as follows when batch write of word device memory and batch write of bit device memory (16 points unit) is done.

(1) Communications in binary code

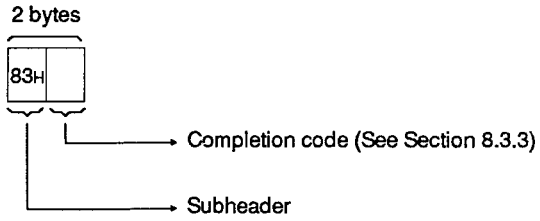
Command Format



REMARK

When number of device points is specified to 256 points, set "00_H".

Response Format



Example 1: When writing the ON/OFF data to X60 to 7F (32 points) of a PC CPU loaded with A1SJ71E71

Command (Other node → A1SJ71E71)

03H	FFH	0AH	00H	60H	00H	00H	00H	20H	58H	02H	00H	3AH	7CH	A6H	65H

The self | 2500 ms | X60 | $\frac{32}{16}$ = 2 points

Response (A1SJ71E71 → Other node)

83H	00H

0	0	1	1	1	0	1	0	0	1	1	1	1	1	0	0	1	0	1	0	0	1	1	0	0	0	1	1	0	1	0	1
X67		to			X60	X6F		to			X68	X77		to			X70	X7F		to			X78								

Example 2: When writing data to D100 to D102 of a PC CPU loaded with A1SJ71E71

Command (Other node → A1SJ71E71)

00H	FFH	0AH	00H	64H	00H	00H	00H	20H	44H	03H	00H	34H	12H	76H	98H	09H	01H

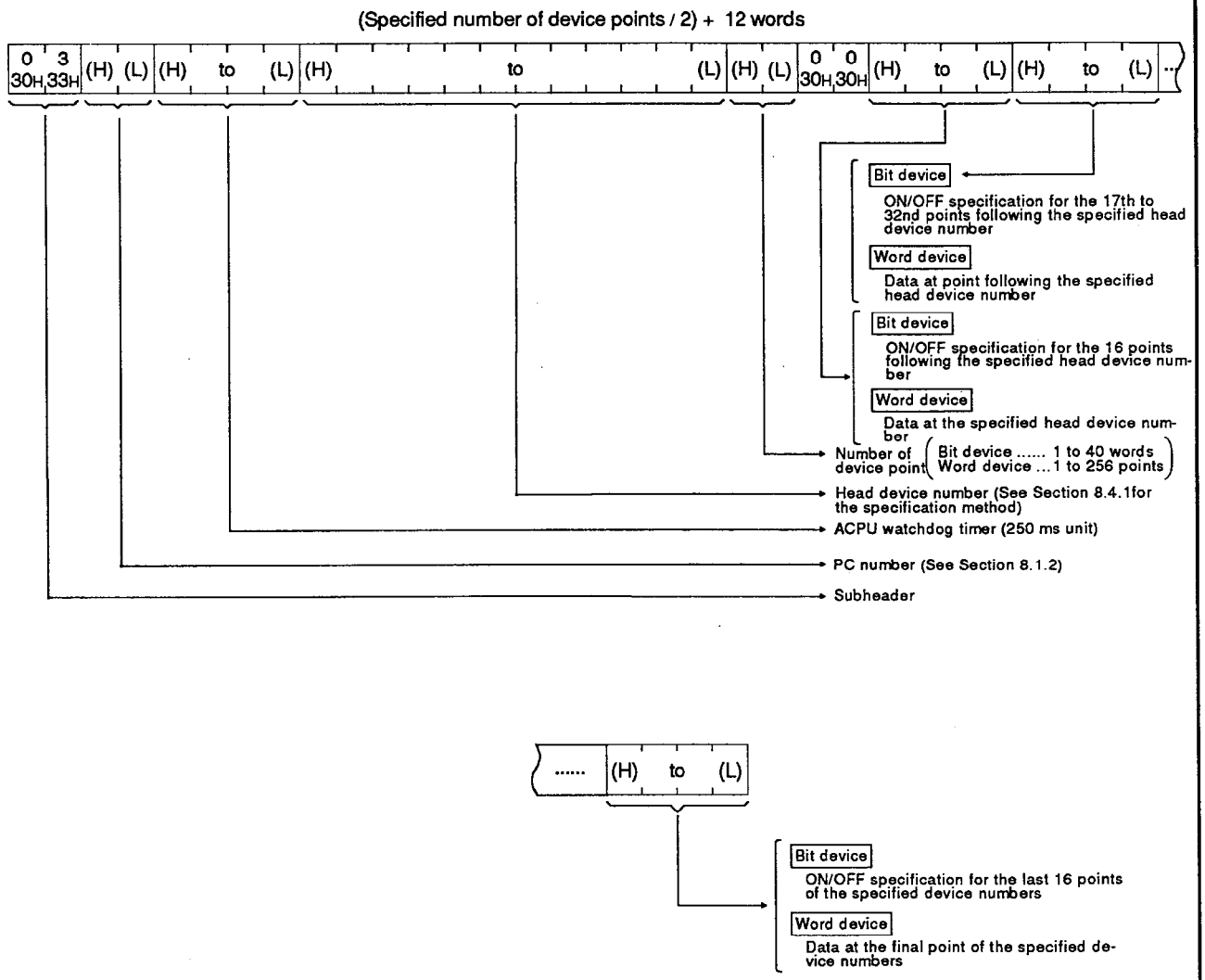
The self | 2500 ms | D100 | 3 points | Write data to D100 (1234_H) | Write data to D101 (9876_H) | Write data to D102 (0109_H)

Response (A1SJ71E71 → Other node)

83H	00H

(2) Communications in ASCII code

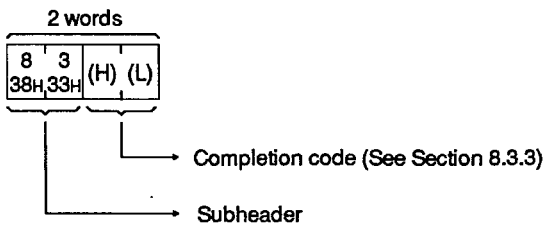
Command Format



REMARK

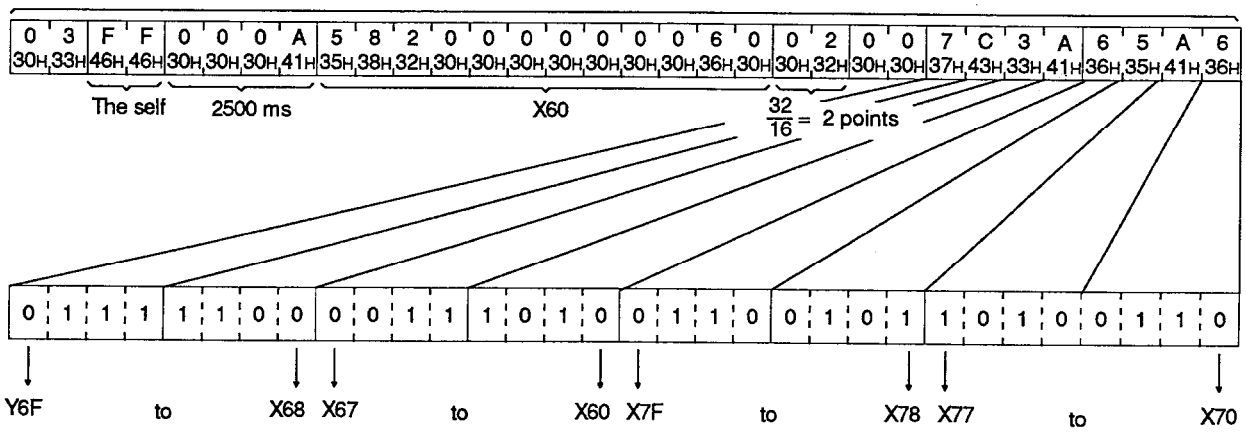
When number of device points is specified to 256 points, set "3030_H".

Response Format

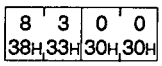


Example 1: When writing the ON/OFF to X60 to 7F (32 points) of a PC CPU loaded with A1SJ71E71

Command (Other node → A1SJ71E71)

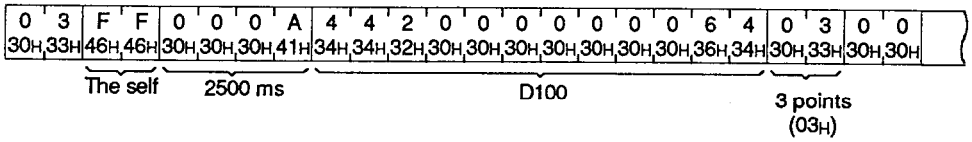


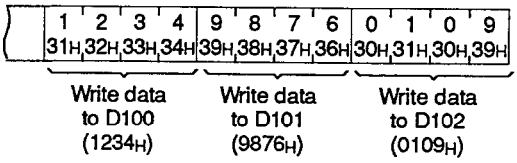
Response (A1SJ71E71 → Other node)



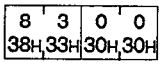
Example 2: When writing data to D100 to D102 of a PC CPU loaded with A1SJ71E71

Command (Other node → A1SJ71E71)





Response (A1SJ71E71 → Other node)

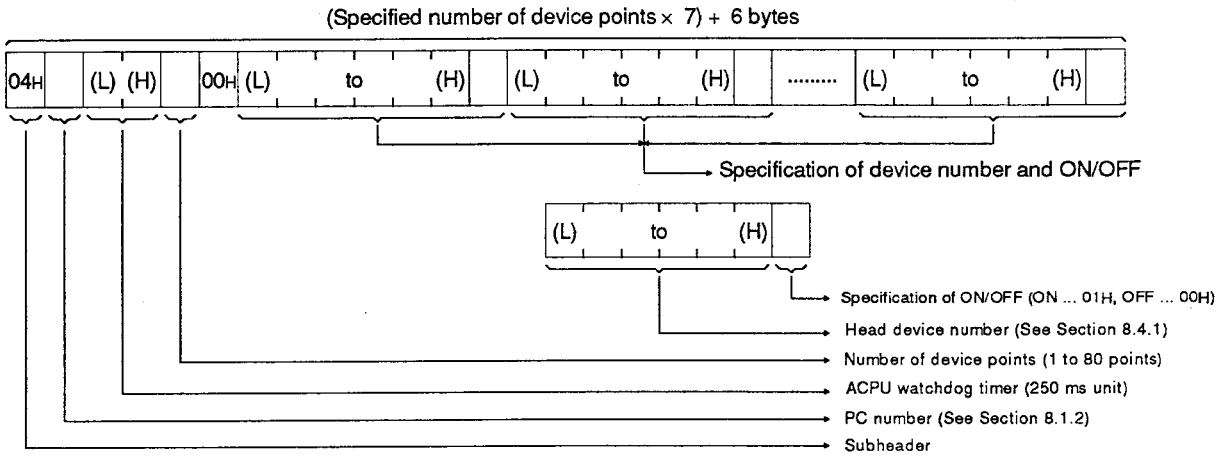


8.4.6 Test (random write) in bit units

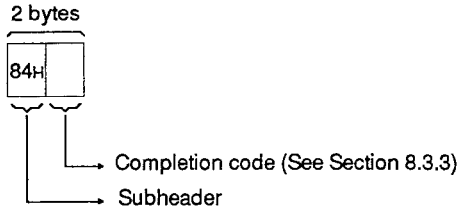
The command and response formats are as follows when bit device memory is specified at random and is written.

(1) Communications in binary code

Command Format

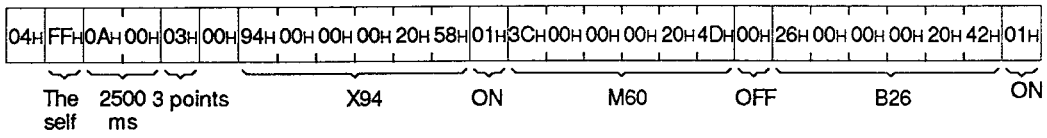


Response Format



Example: When turning ON X94 and B26 and turning OFF M60 of a PC CPU loaded with A1SJ71E71

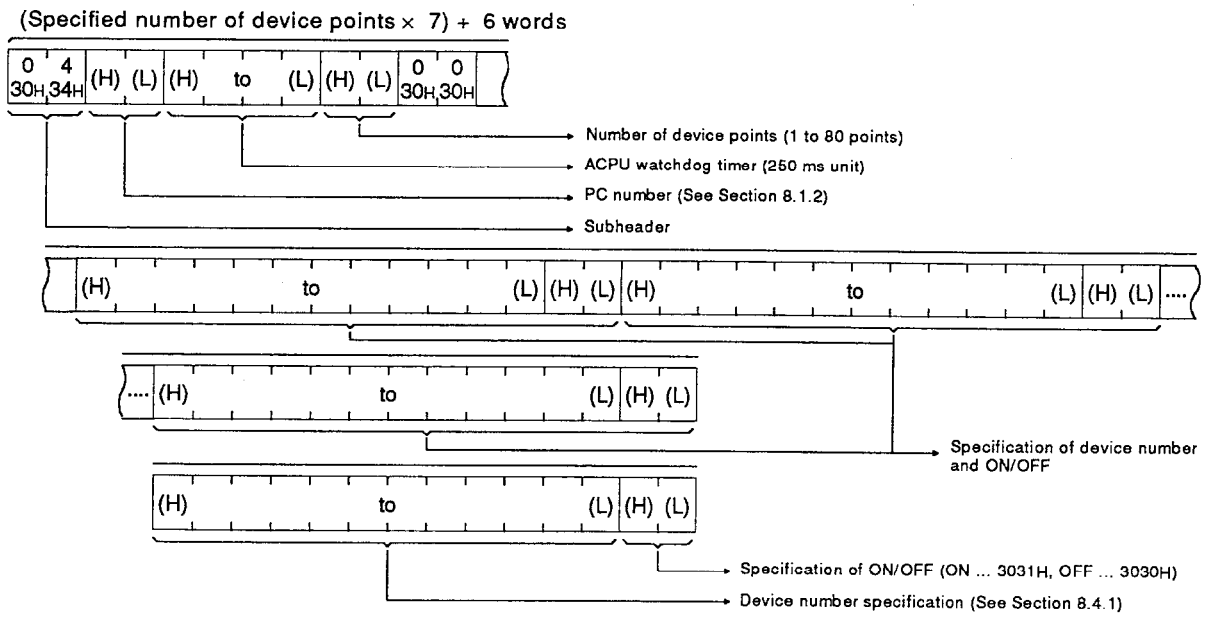
Command (Other node → A1SJ71E71)



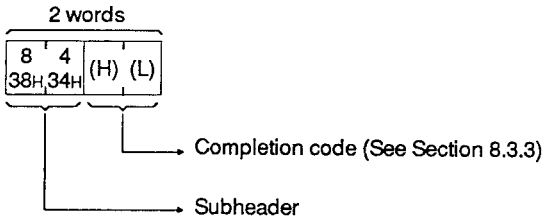
Response (A1SJ71E71 → Other node)

(2) Communications in ASCII code

Command Format

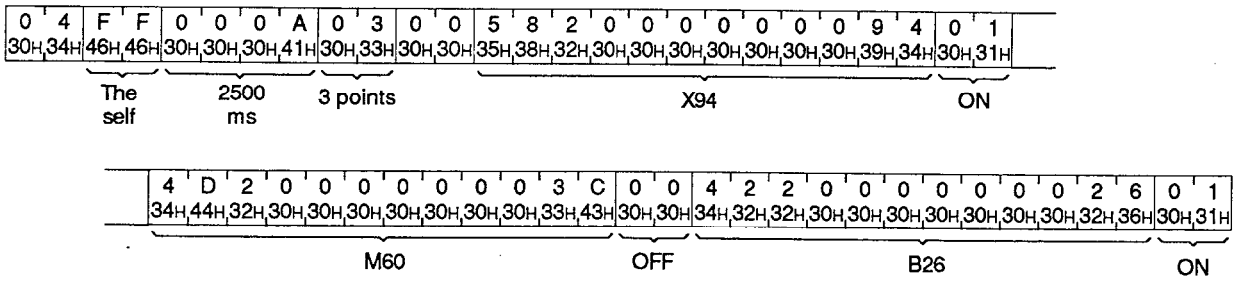


Response Format

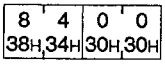


Example: When turning ON X94 and B26 and tuning OFF M60 of a PC CPU loaded with A1SJ71E71

Command (Other node → A1SJ71E71)



Response (A1SJ71E71 → Other node)



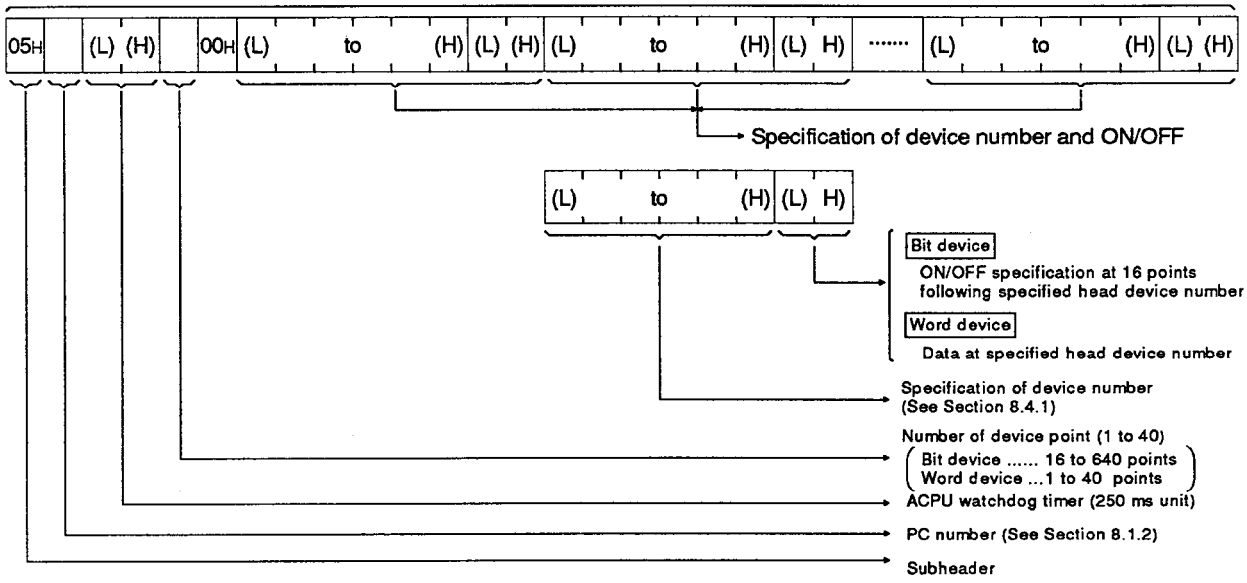
8.4.7 Test (random write) in word unit

The command and response formats are as follows when word device memory and bit device memory (16 points unit) is specified at random and is written.

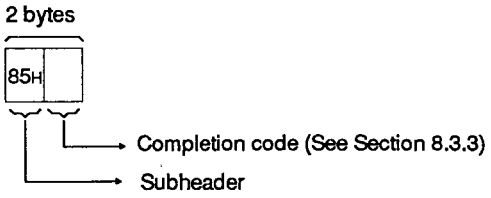
(1) Communications in binary code

Command Format

(Specified number of device points × 8) + 6 bytes

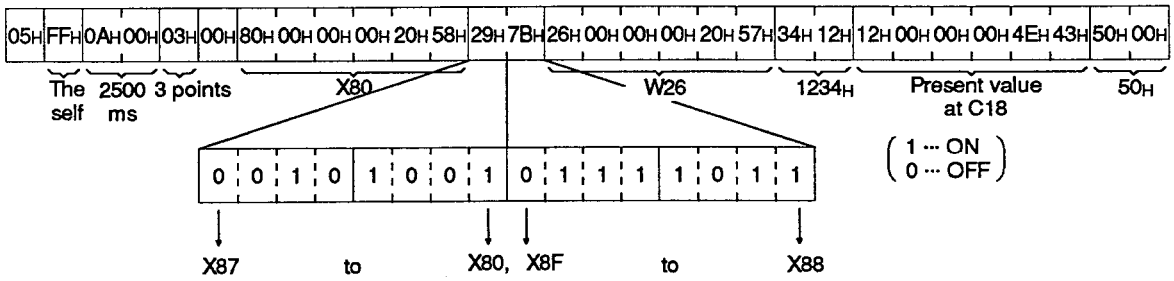


Response Format



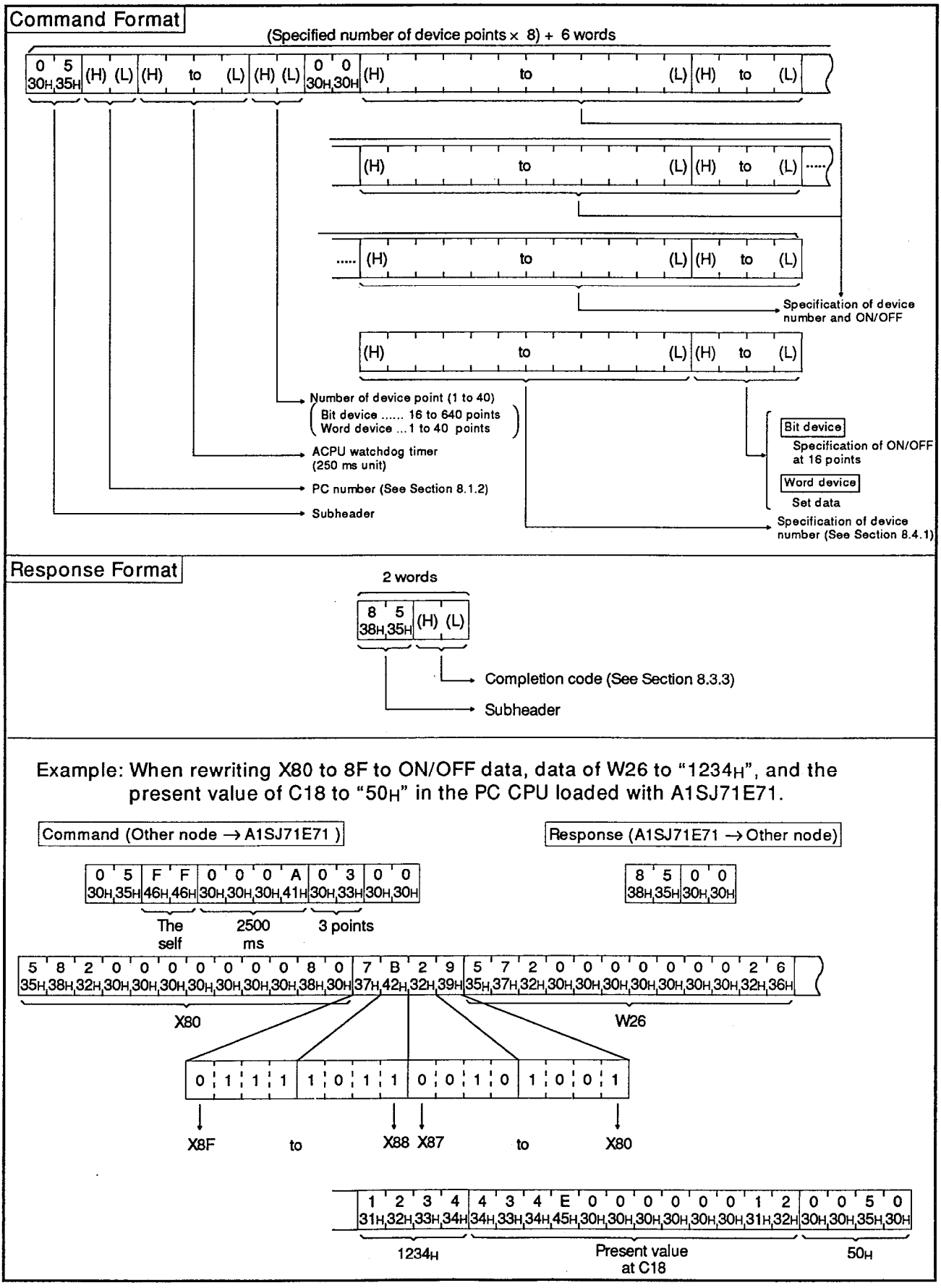
Example: When rewriting X80 to 8F to ON/OFF data, data of W26 to "1234H", and the present value at C18 to "50H" in the PC CPU loaded with A1SJ71E71.

Command (Other node → A1SJ71E71)



Response (A1SJ71E71 → Other node)

(2) Communications in ASCII code



Example: When rewriting X80 to 8F to ON/OFF data, data of W26 to "1234H", and the present value of C18 to "50H" in the PC CPU loaded with A1SJ71E71.

8.4.8 Monitoring device memory

The ON/OFF state or data of a device in the PC CPU can be monitored by a communicating station by transmitting a monitoring command from the communicating station.

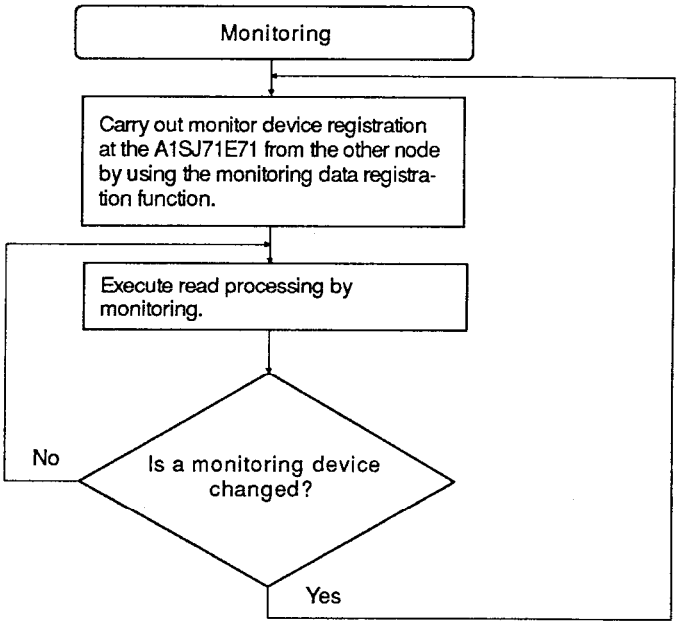
The type and the number of a device to be monitored must be registered in advance to A1SJ71E71.

When device memory is read by using batch read processing, the devices are read by serial device numbers.

The type and the number of a device can be specified at random in case of the read by the monitoring function.

(1) Procedure for monitoring

The chart below shows the operation procedure to execute monitoring.



POINTS

(1) Be sure to execute monitoring after doing monitoring data registration according to the above-mentioned operation procedure.

When monitoring is executed without doing monitoring data registration, an error (completion code 57_H) occurs.

(2) When a power supply is turned OFF, and the PC CPU is reset, the contents of monitoring data registration are cleared.

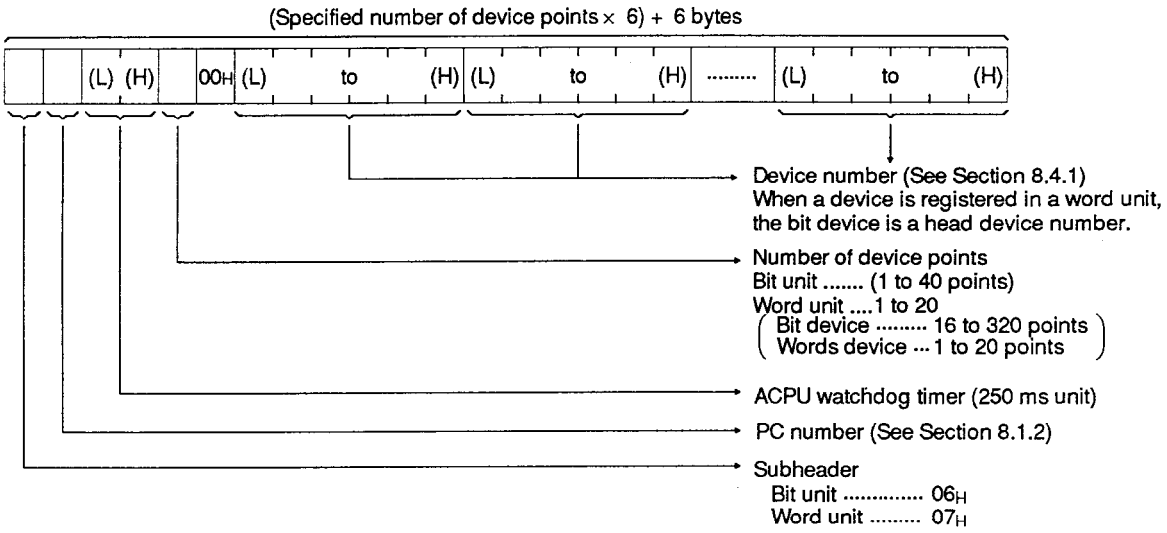
(3) The bit unit and the word unit of a device memory and the extension file register can be registered in case of monitoring data registration.

(2) Monitoring data registration

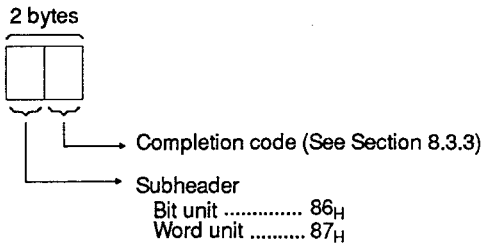
The command and the response formats are as follows when a monitoring device is registered:

(a) Communications in binary code

Command Format

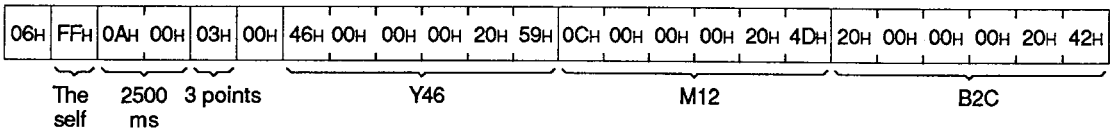


Response Format



Example 1: Monitoring data registration in bit units
When setting Y46, M12 and B2C of a PC CPU loaded with A1SJ71E71

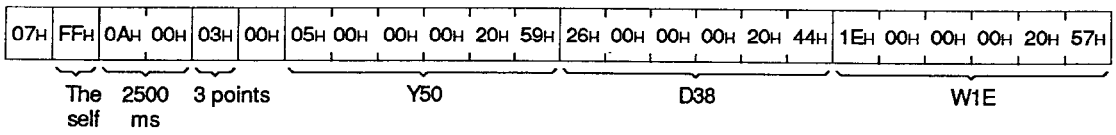
Command (Other node → A1SJ71E71)



Response (A1SJ71E71 → Other node)

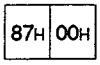
Example 2: Monitoring data registration in bit units
When setting Y50 to 5F, D38 and W1E of a PC CPU loaded with A1SJ71E71

Command (Other node → A1SJ71E71)



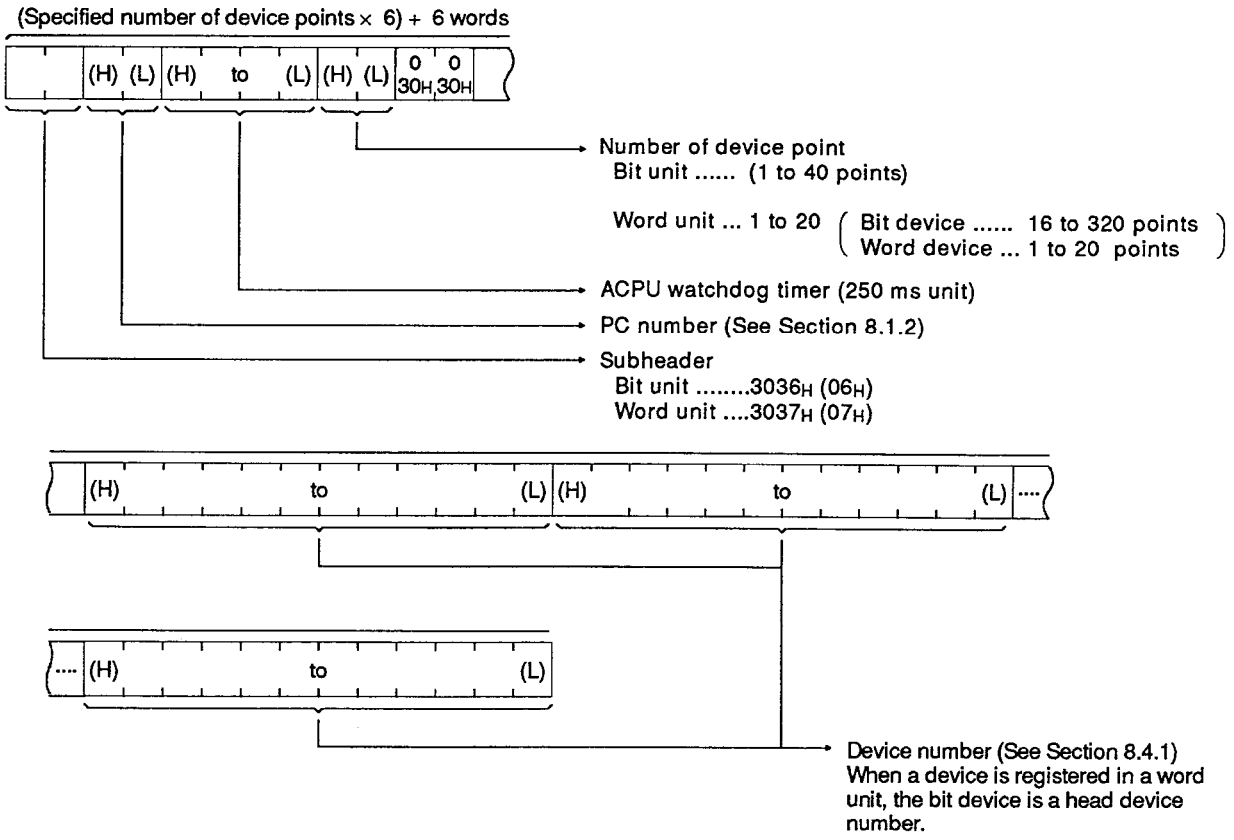
(specification of Y50 to 5F)

Response (A1SJ71E71 → Other node)

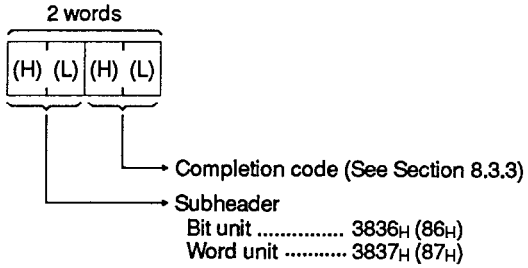


(b) Communications in ASCII code

Command Format



Response Format



Example 1: Monitoring data registration in bit units
When setting Y46, M12 and B2C of a PC CPU loaded with A1SJ71E71

Command (Other node → A1SJ71E71)

0	6	F	F	0	0	0	A	0	3	0	0	5	9	2	0	0	0	0	0	0	0	4	6	...
30H	36H	46H	46H	30H	30H	30H	41H	30H	33H	30H	30H	35H	39H	32H	00H	00H	00H	00H	00H	00H	00H	34H	36H	

The self | 2500 ms | 3 points (03H) | Y46

...	4	D	2	0	0	0	0	0	0	0	0	C	4	2	2	0	0	0	0	0	0	0	2	C
	34H	44H	32H	30H	30H	30H	30H	30H	30H	30H	30H	43H	34H	32H	32H	30H	30H	30H	30H	30H	30H	30H	32H	43H

M12 | B2C

Response (A1SJ71E71 → Other node)

8	6	0	0
38H	36H	30H	30H

Example 2: Monitoring data registration in bit units
When setting Y50 to 5F, D38 and W1E of a PC CPU loaded with A1SJ71E71

Command (Other node → A1SJ71E71)

0	7	F	F	0	0	0	A	0	3	0	0	5	9	2	0	0	0	0	0	0	0	5	0	...
30H	37H	46H	46H	30H	30H	30H	41H	30H	33H	30H	30H	35H	39H	32H	30H	30H	30H	30H	30H	30H	30H	35H	30H	

The self | 2500 ms | 3 points (03H) | Y50

...	4	4	2	0	0	0	0	0	0	0	2	6	5	7	2	0	0	0	0	0	0	0	1	E
	34H	34H	32H	30H	30H	30H	30H	30H	30H	30H	32H	36H	36H	37H	32H	30H	30H	30H	30H	30H	30H	30H	31H	45H

D38 | W1E

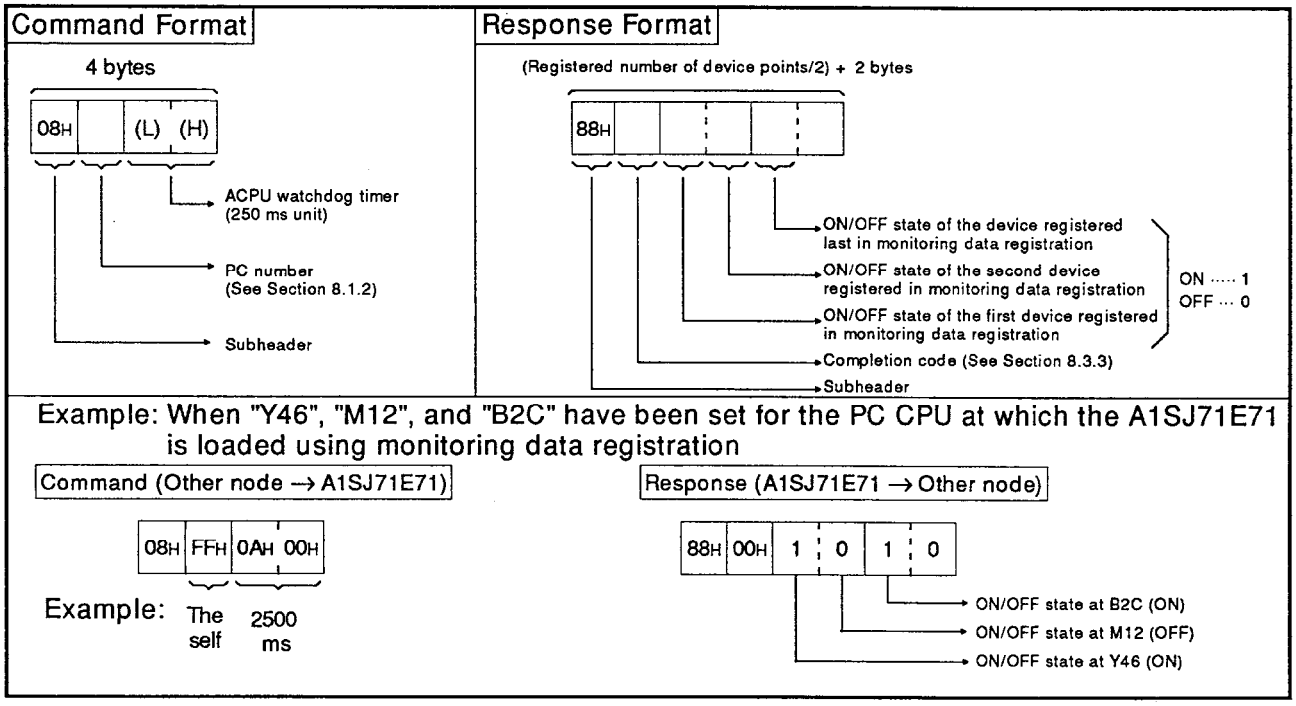
Response (A1SJ71E71 → Other node)

8	7	0	0
38H	37H	30H	30H

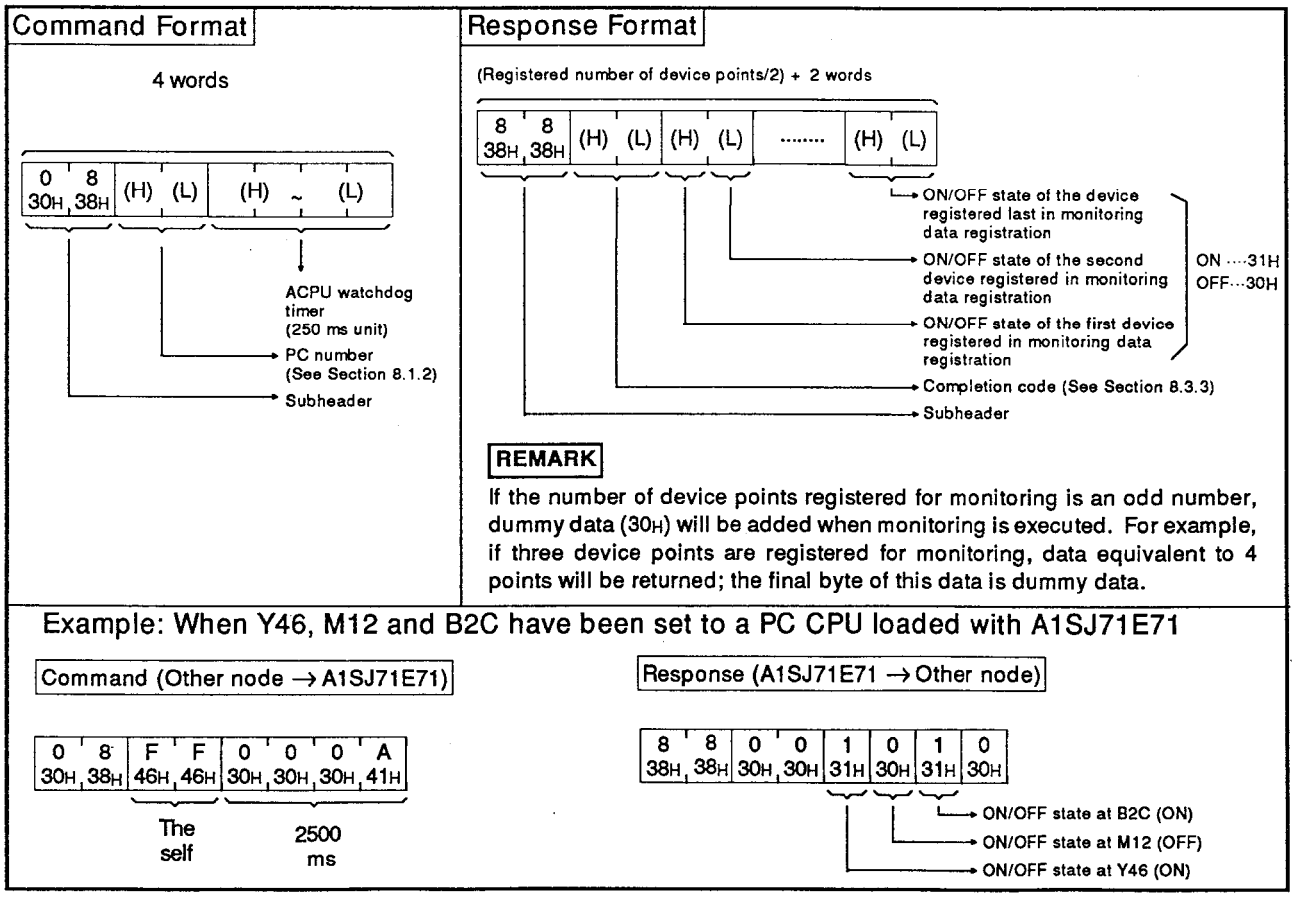
(3) Monitoring in bit units

The command and response formats are as follows when a bit device set by monitoring data registration is monitored:

(a) Communications in binary code



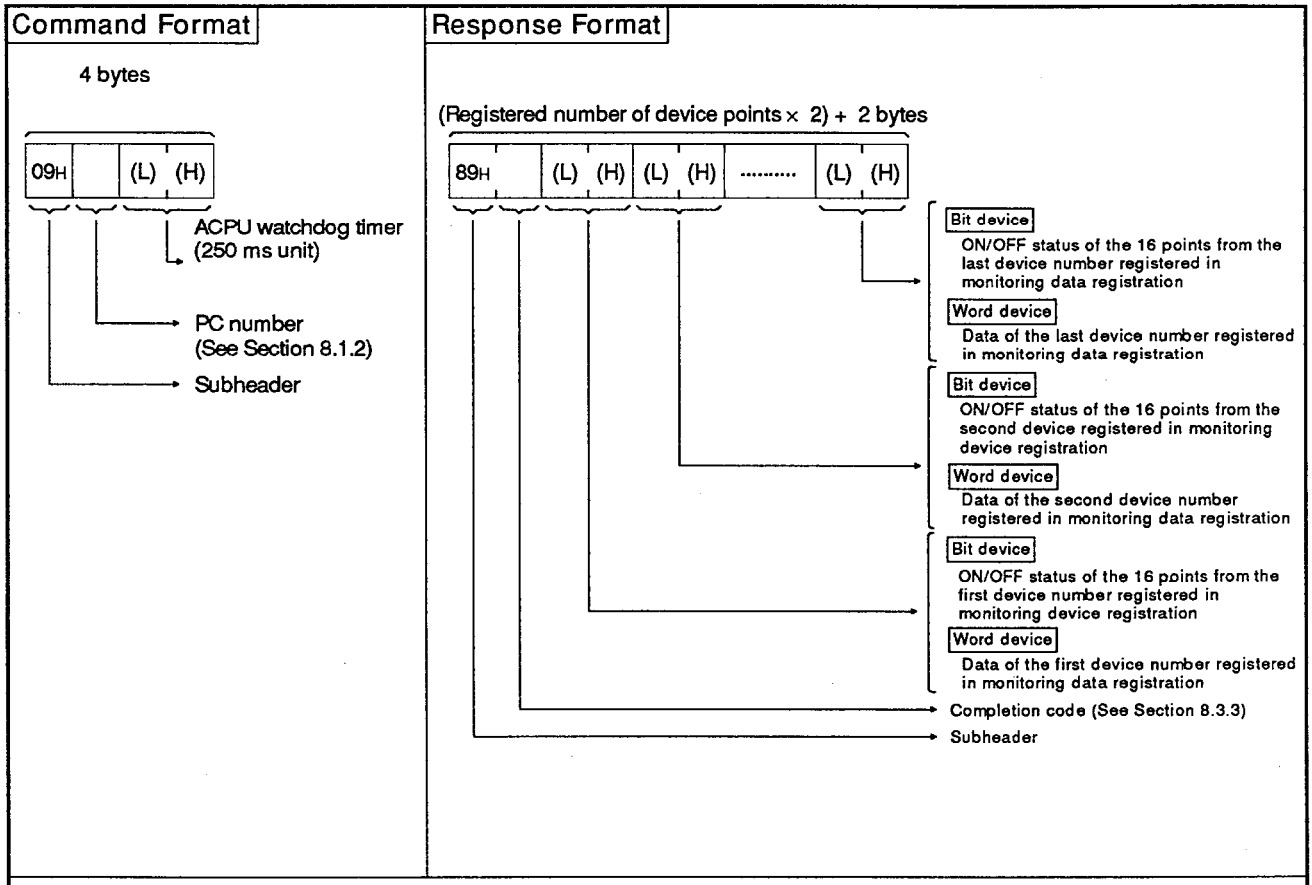
(b) Communications in ASCII code



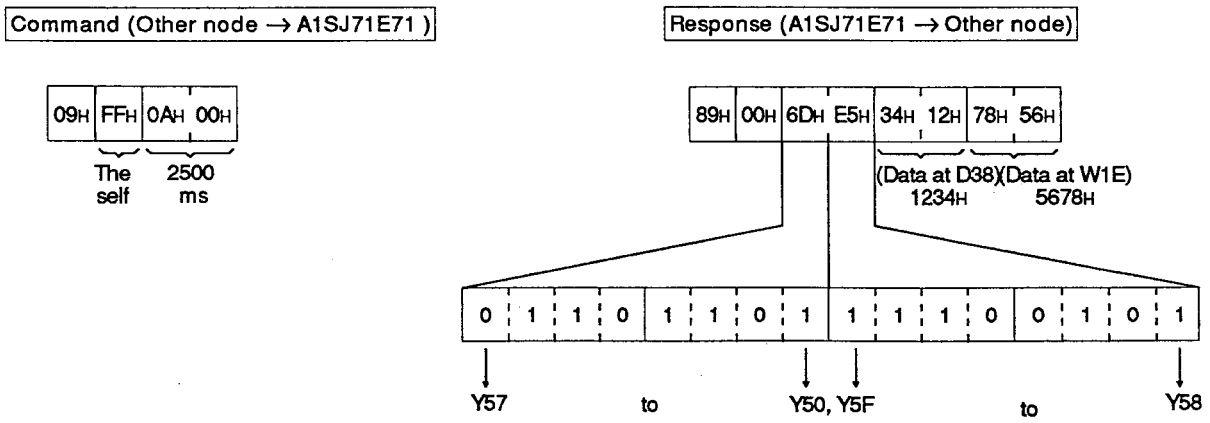
(4) Monitoring in word units

The command and response formats are as follows when a word device and a bit device (16 points unit) set by monitoring data registration are monitored:

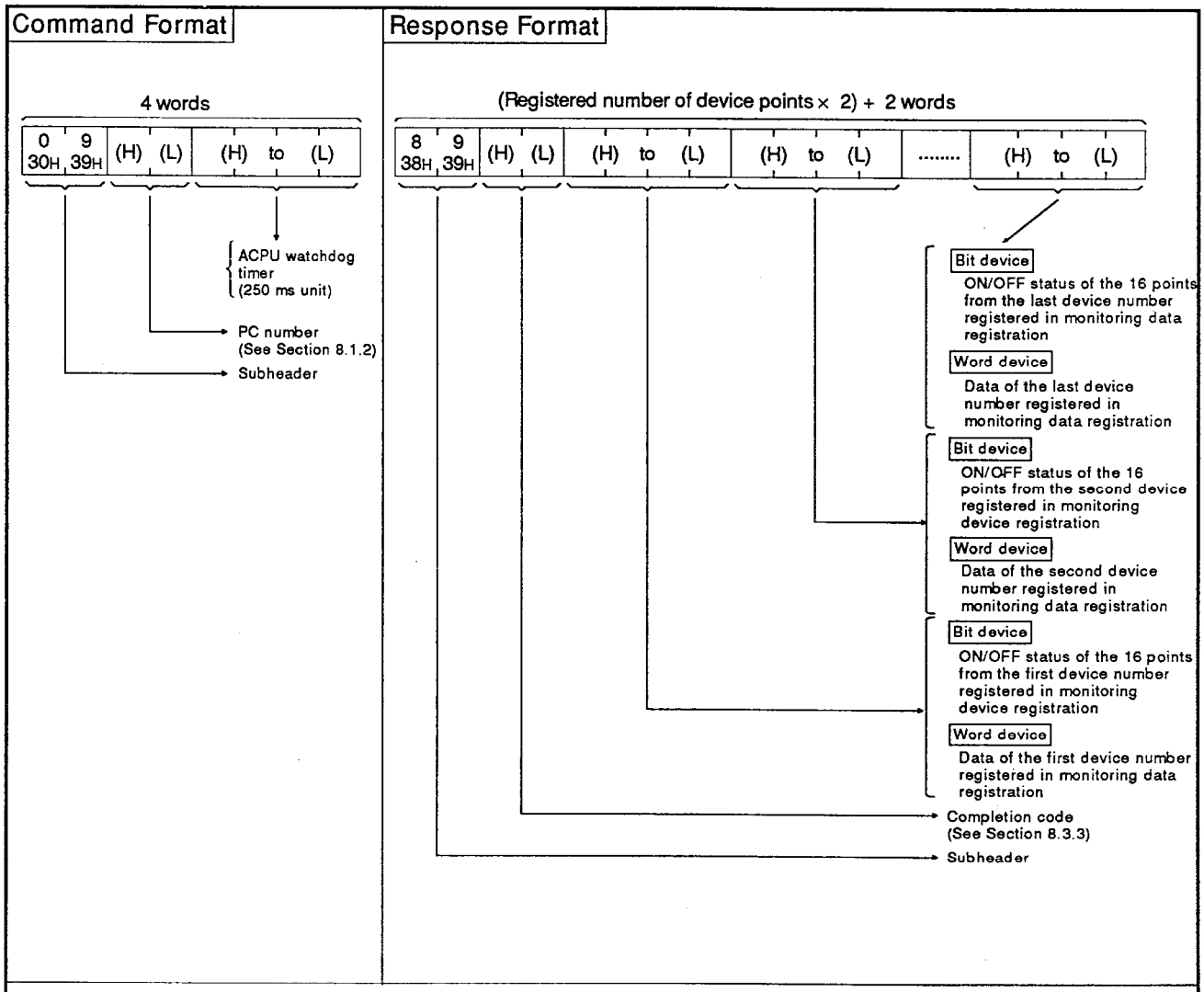
(a) Communications in binary code



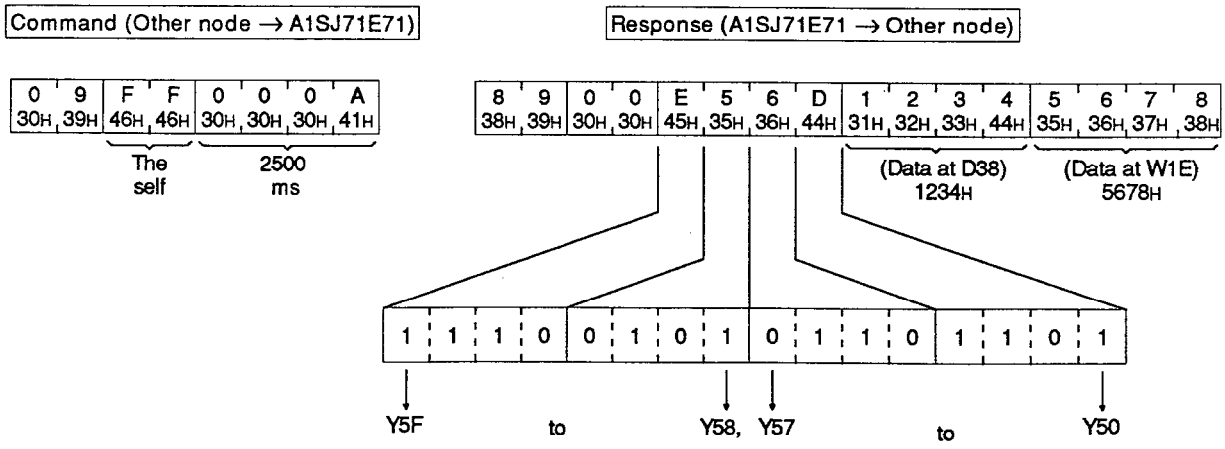
Example: When Y50 to 5F, D38 and W1E have been set to a PC CPU loaded with A1SJ71E71



(b) Communications in ASCII code



Example: When Y50 to 5F, D38 and W1E have been set to a PC CPU loaded with A1SJ71E71



8.5. Command and Response Formats for Read/Write of Extension File Register

An extension file register uses an empty area as a file register in the user memory area of a PC CPU.
Necessary data and operation results are stored in this area in case of various data processing done by using software package "SW[ ]GHP-UTLPC-FN1 utility package" (UTLP-FN1).

The following explains the contents, the method and the example of the control protocol to be used for read/write of extension file registers.

8.5.1 Commands and addresses

Table 8.6 shows the functions to be used for read/write of extension file registers.

Table 8.6 Function

Function	Command/ Response Classification	Description	Number of Point Processed per Communi-cations	PC CPU State		
				During STOP	During RUN	
					SW22 ON	SW22 OFF
Batch read	17_H	Reads extension file registers (R) in units of 1 register.	256 points	o	o	o
Batch write	18_H	Writes extension file registers (R) in units of 1 register.	256 points	o	o	x
Test (random write)	19_H	Specifies the extension file registers (R) in units of 1 register using block or device number and makes a random write.	40 points	o	o	x
Monitor data registration	$1A_H$	Registers device numbers for monitoring in 1 point units.	20 points	o	o	o
Monitor	$1B_H$	Monitors the extension file register after monitor data registration.	–	o	o	o

Note : o .....Executable
x.....Not executable

Addresses of extension file registers

(1) Numbers of points of extension file registers

Block No. 0..... Number of extension file registers corresponding to the number of points set in the PC CPU parameters
Block Nos. 1 onward..... Each block has 8192 points of registers.

(2) The range of block numbers that can be specified varies depending on the PC CPU memory capacity (type of memory cassette) and the PC CPU parameter settings.

For details, refer to the UTLP-FN1 Operating Manual or the AnA/AnU Programming Manual (Dedicated Instructions)

8.5.2 Precautions for read/write of extension file registers

The following explains precautions for read/write of extension file registers.

(1) The extension file register cannot be used with A1 and A1NCPU.

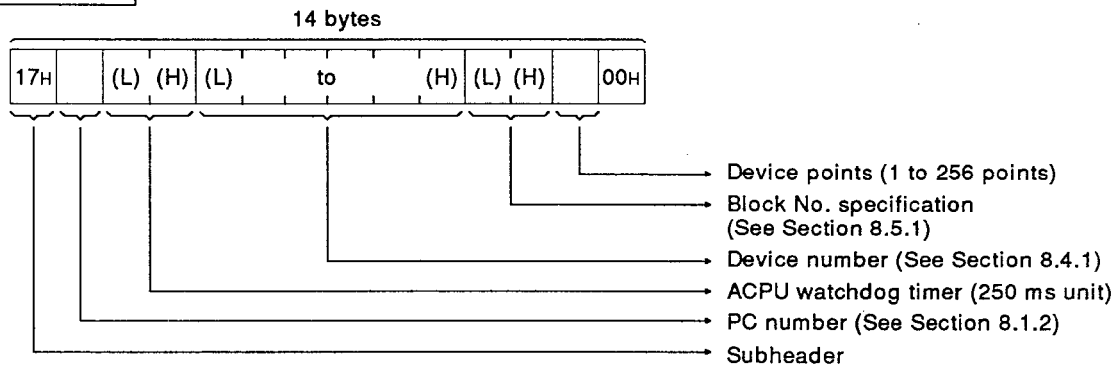
(2) The range of block numbers that can be specified varies depending on the PC CPU memory capacity (type of memory cassette) and the PC CPU parameter settings.

8.5.3 Batch read of extension file register

When batch read of extension file register is done, the command and response format are as follows:

(1) Communications in binary code

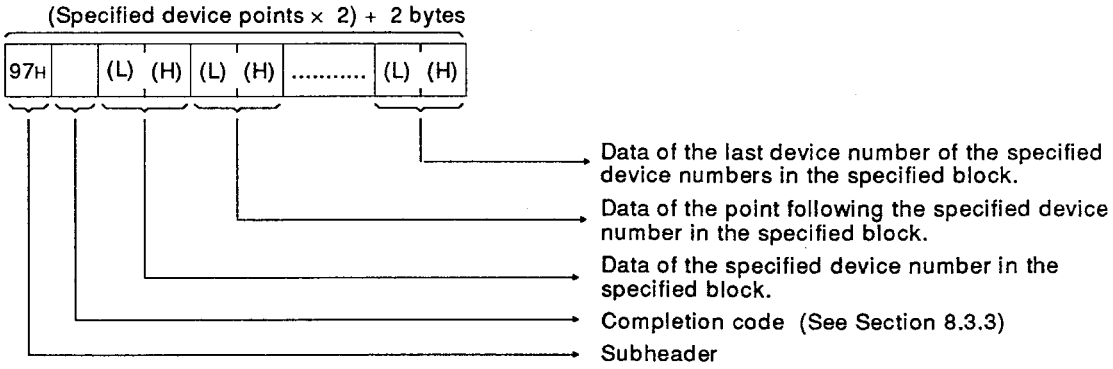
Command Format



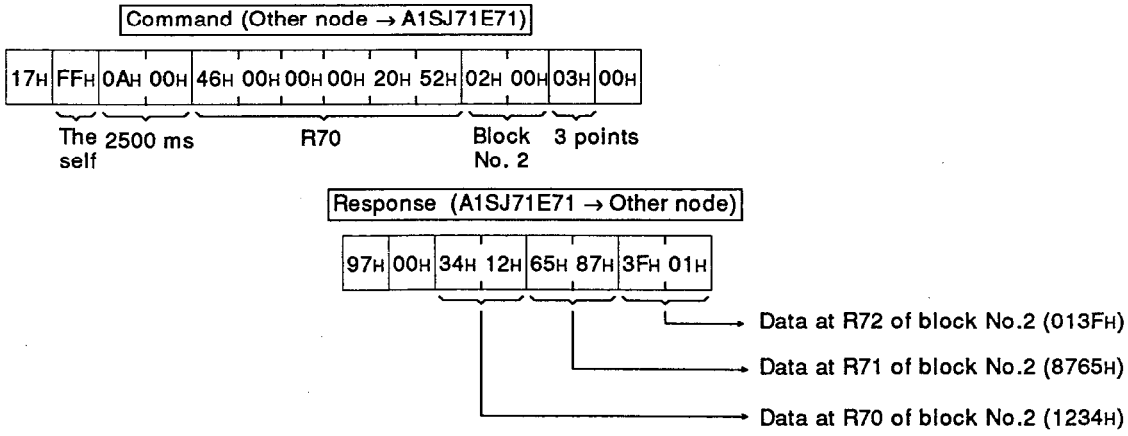
REMARK

When device points is specified to 256 points, set "00_H".

Response Format

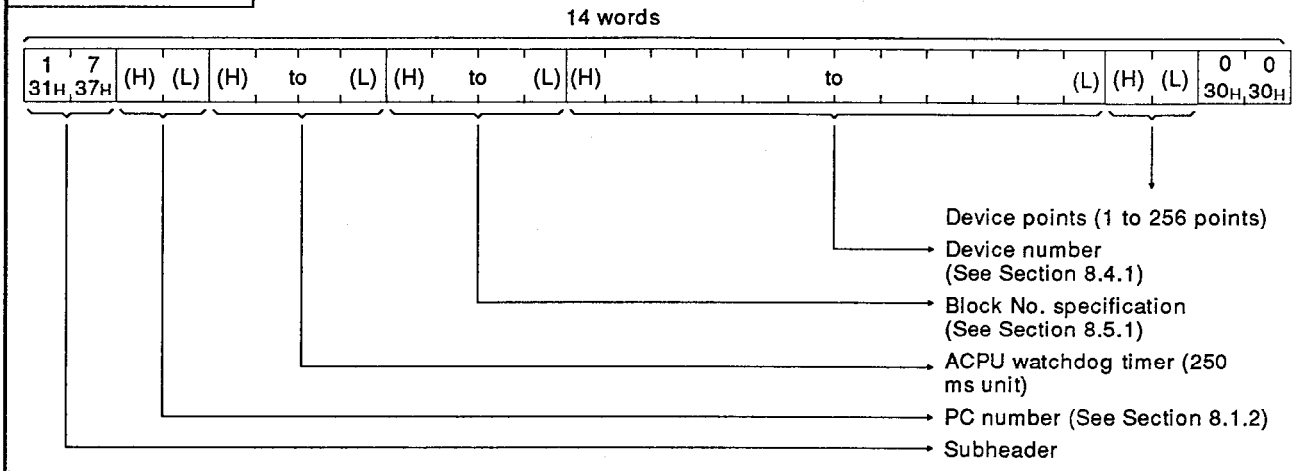


Example: When reading data at R70 to 72 of the extension file register block No.2 of a PC CPU loaded with A1SJ71E71



(2) Communications in ASCII code

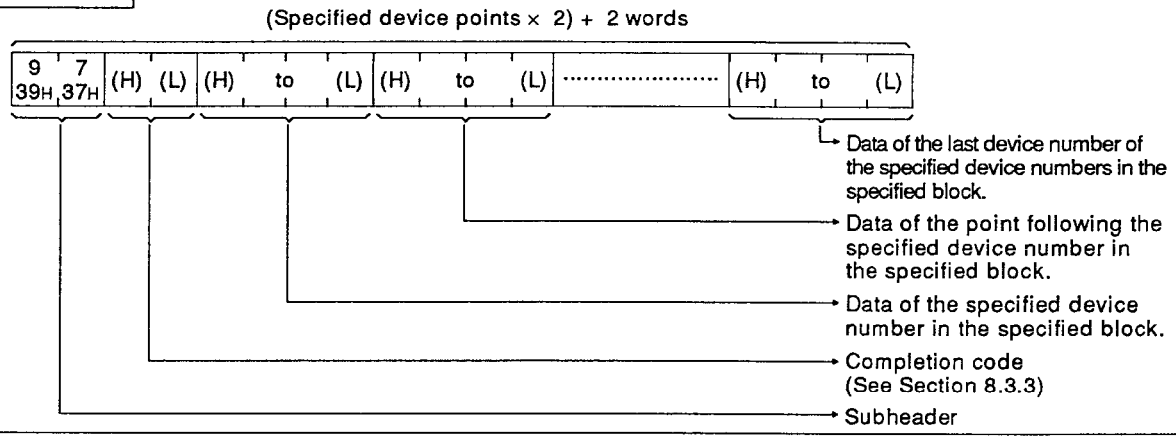
Command Format



REMARK

When device points is specified to 256 points, set "00_H".

Response Format



Example : When reading data at R70 to 72 of the extension file register block No.2 of a PC CPU loaded with A1SJ71E71

Command (Other node → A1SJ71E71)

1 7	F F	0 0 0 A	0 0 0 2	5 2 2 0 0 0 0 0 0 0 4 6	0 3
31H,37H	46H,46H	30H,30H,30H,41H	30H,30H,30H,32H	35H,32H,32H,30H,30H,30H,30H,30H,30H,30H,34H,36H	30H,33H
The self	2500 ms	Block No.2	R70		3 points

Response (A1SJ71E71 → Other node)

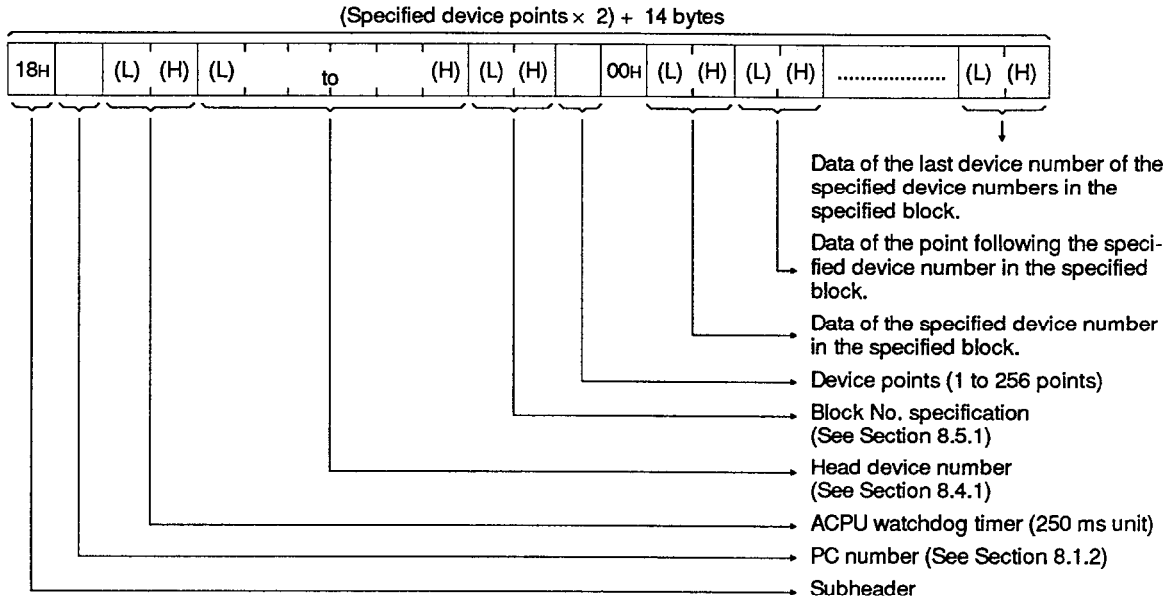
9 7	0 0	1 2 3 4	8 7 6 5	0 1 3 F
39H,37H	30H,30H	31H,32H,33H,34H	38H,37H,36H,35H	30H,31H,33H,46H
		Data at R70 of block No.2 (1234H)	Data at R71 of block No.2 (8765H)	Data at R72 of block No.2 (013FH)

8.5.4 Batch write of extension file register

When batch write of extension file register is done, the command and response format are as follows:

(1) Communications in binary code.

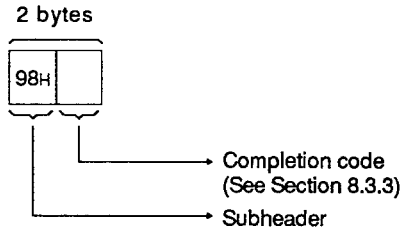
Command Format



REMARK

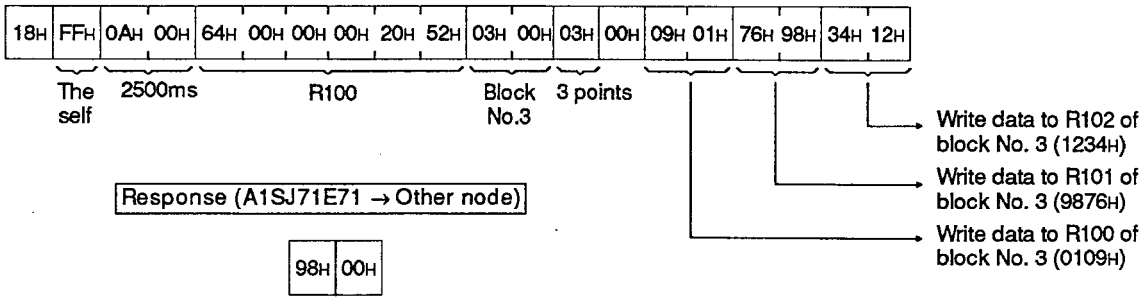
When device points is specified to 256 points, set "00H".

Response Format



Example: When writing data to R100 to 102 of the extension file register block No.3 of a PC CPU loaded with A1SJ71E71

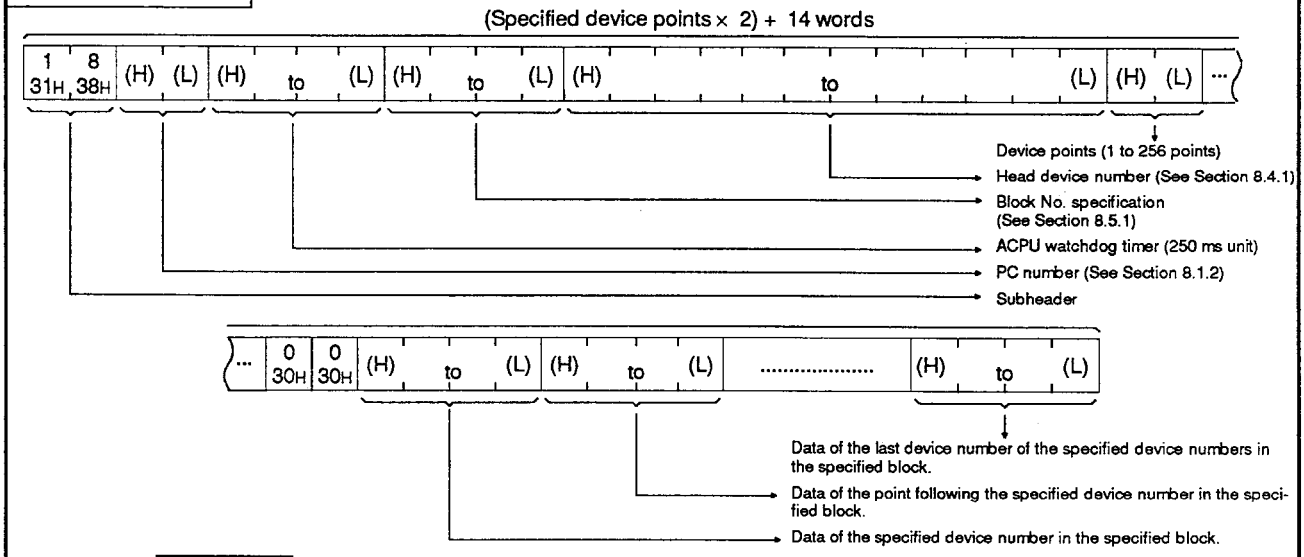
Command (Other node → A1SJ71E71)



Response (A1SJ71E71 → Other node)

(2) Communications in ASCII code

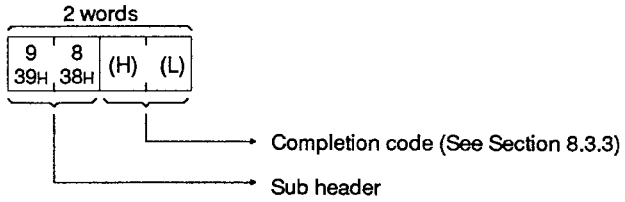
Command Format



REMARK

When device points is specified to 256 points, set "3030_H".

Response Format



Example: When writing data to R100 to 102 of the extension file register block No.3 of a PC CPU loaded with A1SJ71E71

Command (Other node → A1SJ71E71)

1 8	F F	0 0 0 A	0 0 0 3	5 2 2 0 0 0 0 0 0 0 0 6 4	0 3	...
31H 38H	46H 46H	30H 30H 30H 41H	30H 30H 30H 33H	35H 32H 32H 30H 30H 30H 30H 30H 30H 30H 36H 34H	30H 33H	
(18_H)	The self	2500ms	Block No. 3	R100	3 points	

...	0 0	0 1 0 9	9 8 7 6	1 2 3 4
	30H 30H	30H 31H 30H 39H	39H 38H 37H 36H	31H 32H 33H 34H
		Write data to R100 of block No. 3 (0109_H)	Write data to R101 of block No. 3 (9876_H)	Write data to R102 of block No. 3 (1234_H)

Response (A1SJ71E71 → Other node)

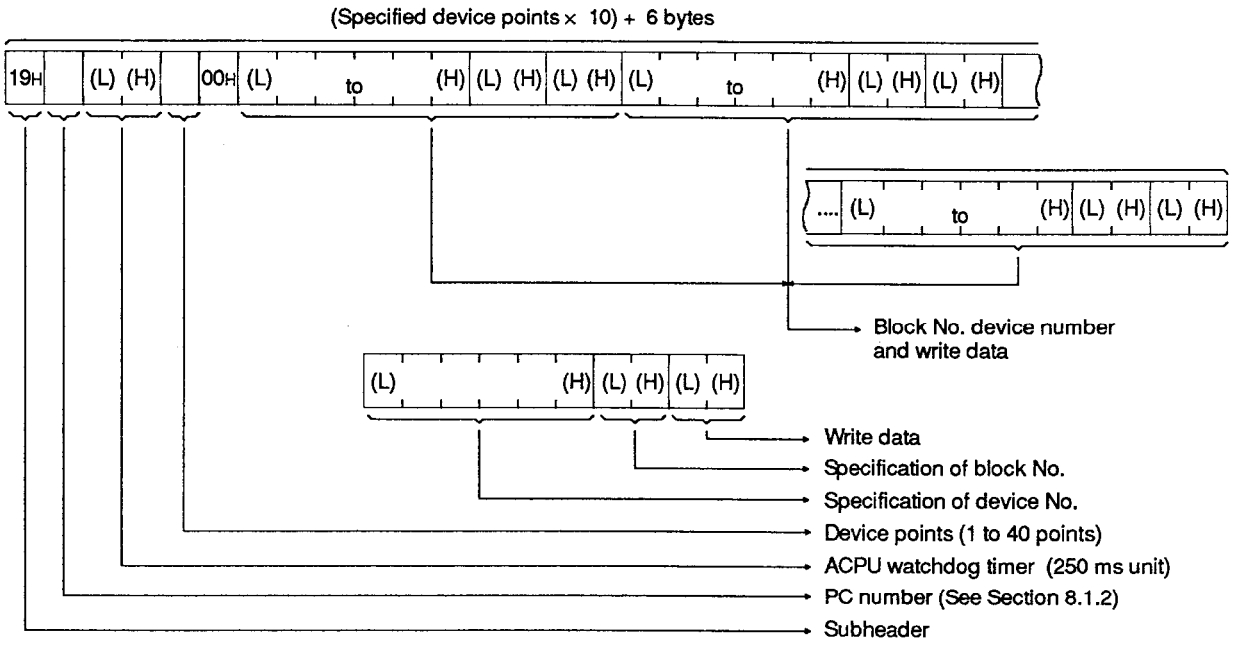
9 8	0 0
39H 38H	30H 30H

8.5.5 Test (random write) of extension file register

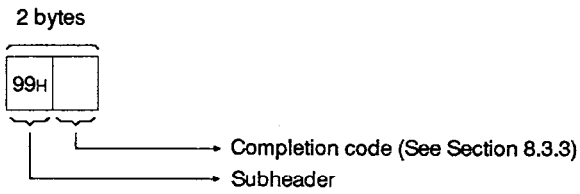
When an extension file register is specified at random and is written, the command and response format are as follows:

(1) Communications in binary code

Command Format



Response Format



Example: When writing data to R26 of block No.2 and R19 of block No.3 of a PC CPU loaded with A1SJ71E71

Command (Other node → A1SJ71E71)

19H	FFH	0AH	00H	02H	00H	1AH	00H	00H	00H	20H	52H	02H	00H	34H	12H	13H	00H	00H	00H	20H	52H	03H	00H	09H	01H

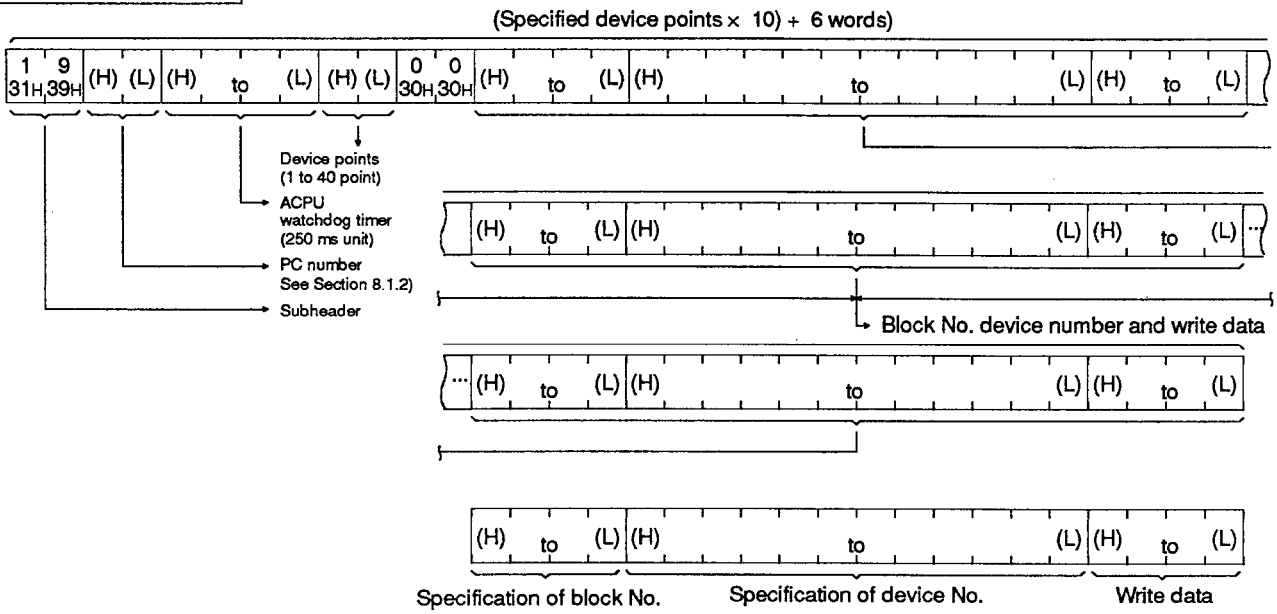
The 2500msec self
2 points
Data (1234_H) is written to R26 of block No. 2
Data (0109_H) is written to R19 of block No. 3

Response (A1SJ71E71 → Other node)

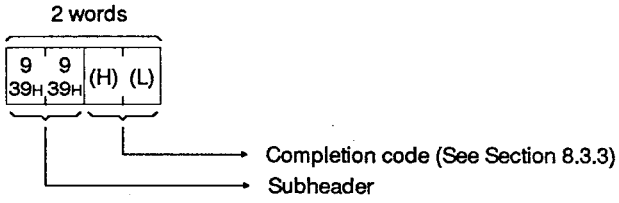
99H	00H

(2) Communications in ASCII code

Command Format



Response Format



Example: When writing data to R26 of block No. 2 and R19 of block No. 3 of a PC CPU loaded with A1SJ71E71.

Command (Other node → A1SJ71E71)

1 9	F F	0 0 0 A	0 2	0 0	0 0 0 2	5 2 2 0 0 0 0 0 0 0 1 A	1 2 3 4
31H,39H	46H,46H	30H,30H,30H,41H	30H,32H	30H,30H	30H,30H,30H,32H	35H,32H,32H,30H,30H,30H,30H,30H,30H,30H,31H,41H	31H,32H,33H,34H
(19H)	The self	2500ms	2 points		Data (1234H) is written to R26 of block No. 2		

0 0 0 3	5 2 2 0 0 0 0 0 0 0 1 3	0 1 0 9
30H,30H,30H,33H	35H,32H,32H,30H,30H,30H,30H,30H,30H,30H,31H,33H	30H,31H,30H,39H
	Data (0109H) is written to R19 of block No. 3	

Response (A1SJ71E71 → Other node)

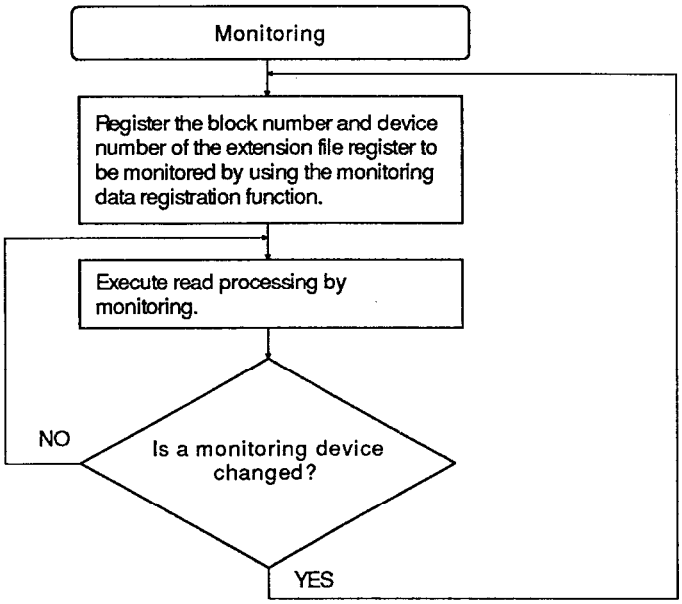
9 9	0 0
39H,39H	30H,30H

8.5.6 Monitoring extension file register

By registering the block number and device numbers of extension file registers to be monitored at another node in advance, then issuing a monitoring command from the other node, the contents of extension file registers (device numbers registered at the A1SJ71E71) in the PC CPU can be monitored at the other node.
In reading by batch reading of extension file registers, consecutive device numbers are processed, but in reading by monitoring, the file register corresponding to any specified block number and device number can be read.

(1) Procedure for monitoring

The chart below shows the operation procedure to execute monitoring.



POINTS

(1) Be sure to execute monitoring after monitoring data registration according to the above-mentioned operation procedure.

When monitoring is executed without completing monitoring data registration, an error (completion code 57_H) occurs.

(2) When a power supply is turned OFF, and the PC CPU is reset, the contens of monitoring data registration are cleared.

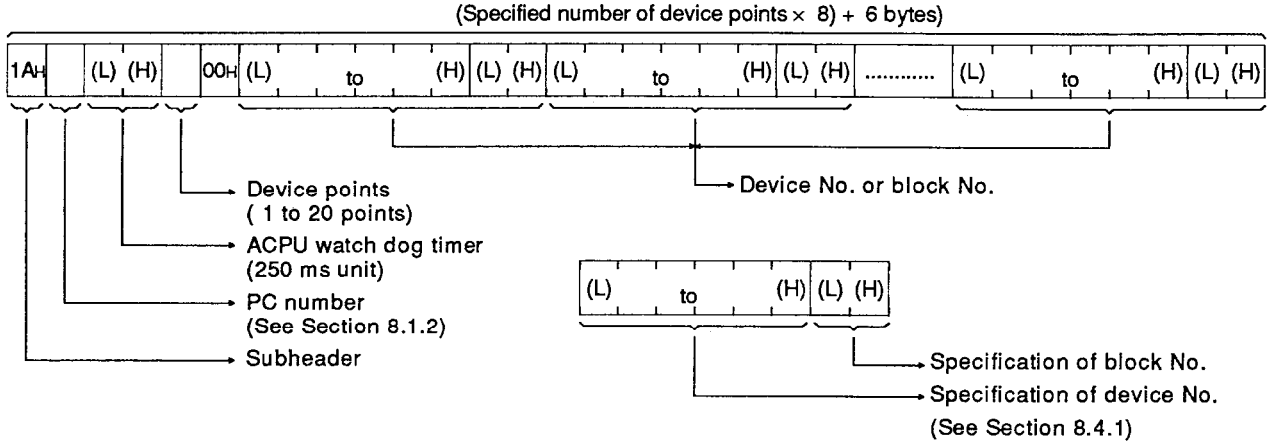
(3) The bit unit and the word unit of device memory and the extension file register can be registered by monitoring data registration.

(2) Monitoring data registration

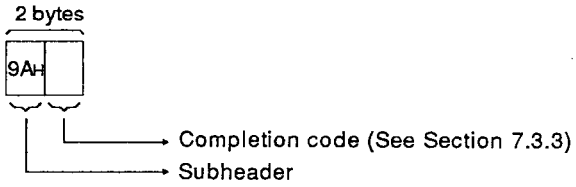
The command and response formats are as follows when a monitoring device number of extension file register is registered:

(a) Communications in binary code

Command Format

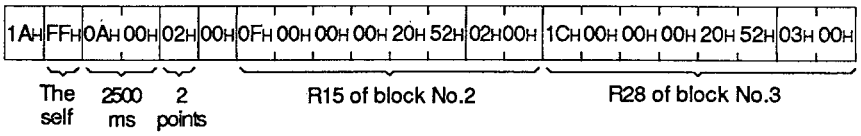


Response Format



Example: When setting R15 of block No. 2 and R28 of block No. 3 of a PC CPU loaded with A1SJ71E71.

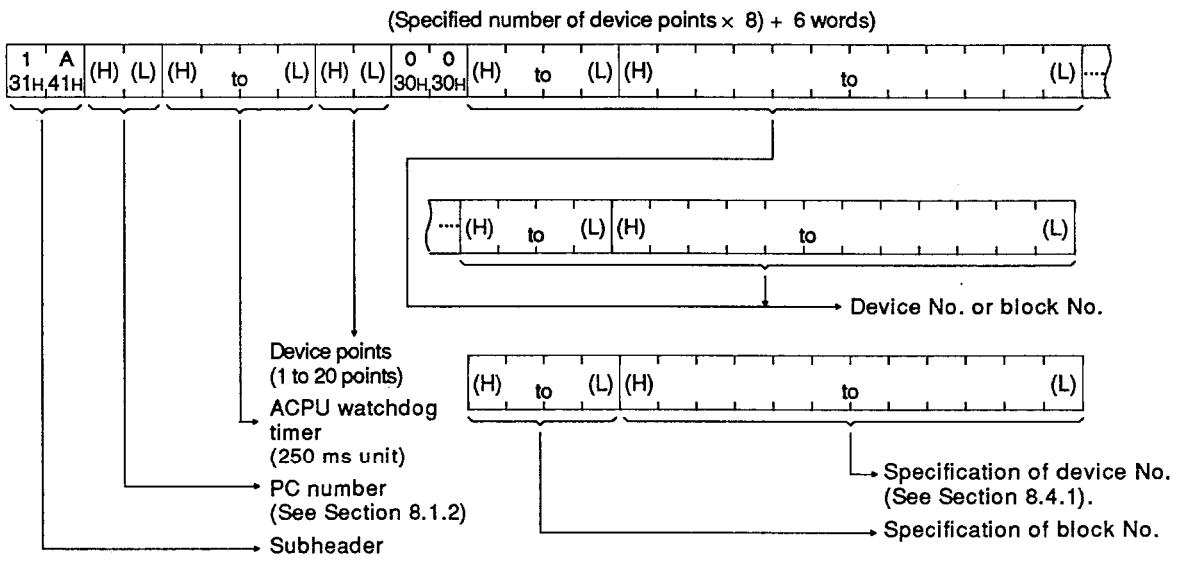
Command (Other node → A1SJ71E71)



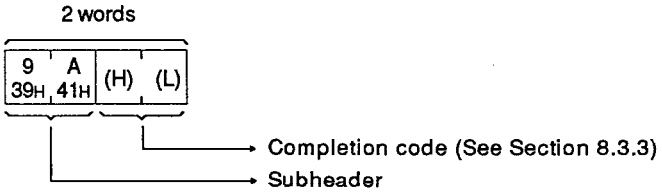
Response (A1SJ71E71 → Other node)

(b) Communications in ASCII code

Command Format

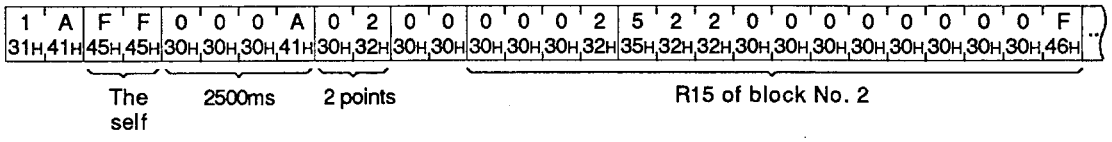


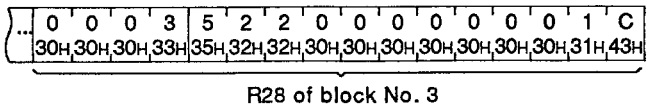
Response Format



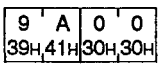
Example: When setting R15 of block No.2 and R28 of block No.3 of a PC CPU loaded with A1SJ71E71

Command (Other node → A1SJ71E71)





Response (A1SJ71E71 → Other node)

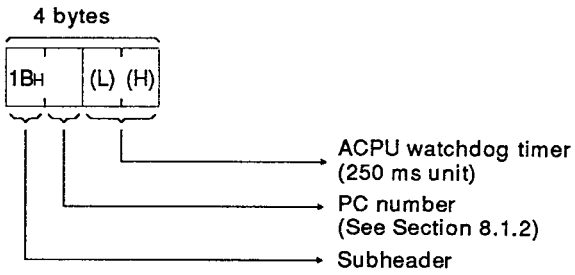


(3) Monitoring

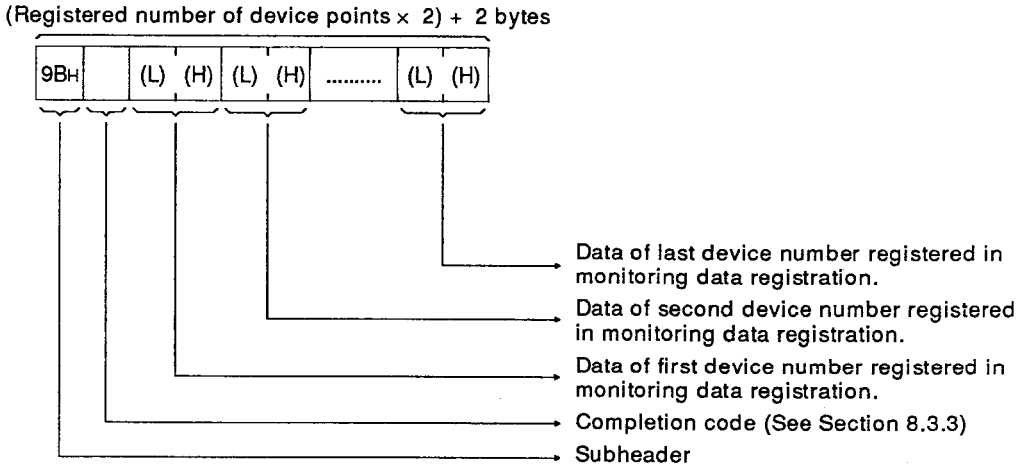
The command and response formats are as follows when an extension file register set by monitoring data registration is registered:

(a) Communications in binary code

Command Format

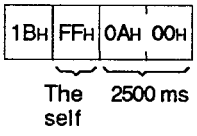


Response Format

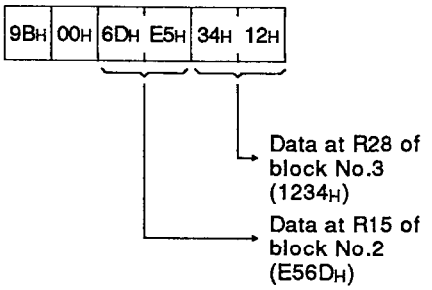


Example: When R15 of block No.2 and R28 of block No.3 have been set to a PC CPU loaded with A1SJ71E71

Command (Other node → A1SJ71E71)

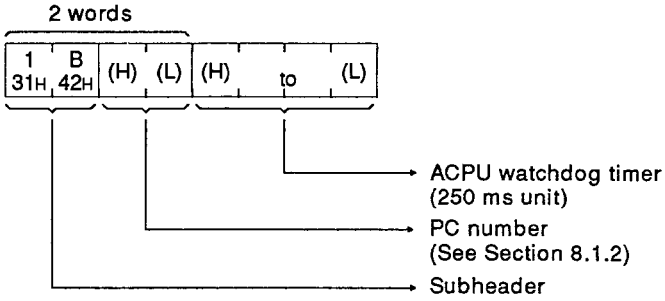


Response (A1SJ71E71 → Other node)

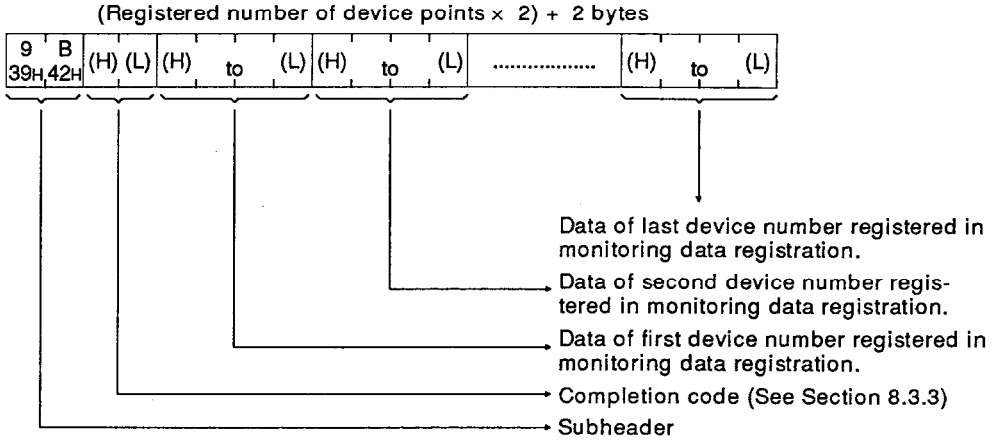


(b) Communications in ASCII code

Command Format

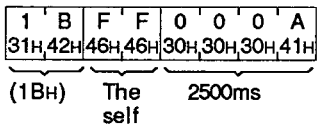


Response Format

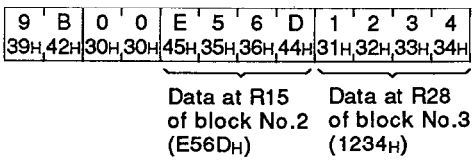


Example: When R15 of block No.2 and R28 of block No.3 have been set to a PC CPU loaded with A1SJ71E71

Command (Other node → A1SJ71E71)



Response (A1SJ71E71 → Other node)



8.5.7 Direct read/write of extension file registers

(1) The AnACPU dedicated commands used for direct read and direct write of extension file registers are described below.

These dedicated commands are used to access the extension file registers of block numbers 0 to 256 by ignoring block numbers and directly designating an address as the device number: the address range starts with address 0 in block number 1. (Extension file registers corresponding to "usable number of blocks x 8192" points are accessed as consecutive device numbers).

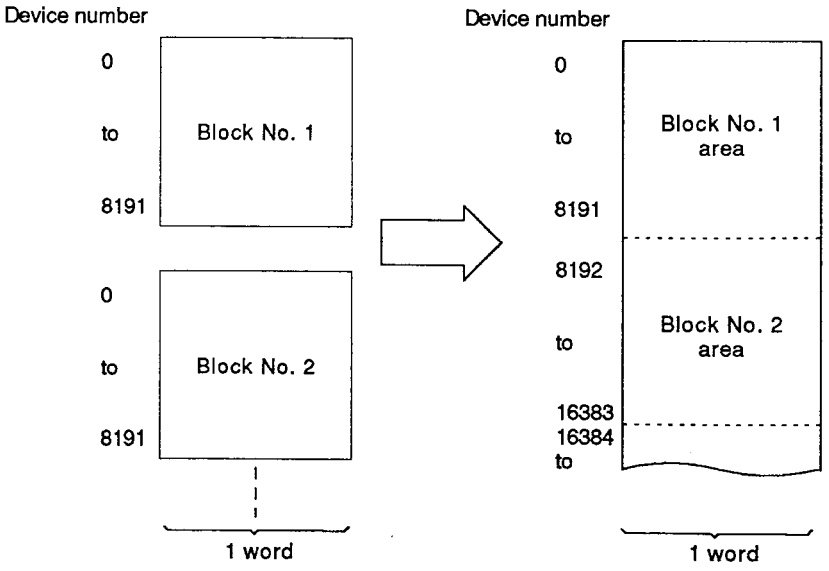
Item	Command	Processing	Number of Points Processed per Communications	State of PC CPU		
				During STOP	During RUN	
					SW22 ON	SW22 OFF
Direct read	$3B_H$	Reads extension file registers (R) in 1-point units.	256 points	o	o	o
Direct write	$3C_H$	Writes extension file registers (R) in 1-point units.	256 points	o	o	x

Note : o .....Executable
x.....Not executable

(2) Device numbers of extension file registers

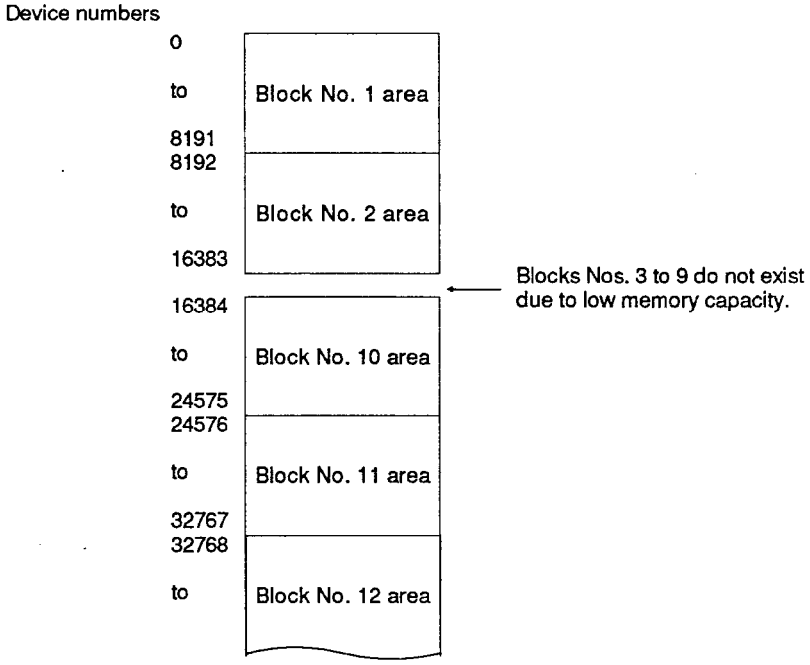
(a) Device number range

Range: 0 through [(the number of usable blocks × 8192) – 1]



The device numbers to be used for direct read/write are assigned automatically in ascending order of the block numbers of devices, from block No.1 upward.
The range of device numbers that can be specified varies depending on the PC CPU memory capacity (type of memory cassette) and the PC CPU parameter settings.

And, the device numbers that can be specified change according to the type of memory cassete and parameter settings of the PC CPU.



POINTS

(1) Only when doring read/write of data using the extension file registers of block Nos. 0 to 256, the AnACPU dedicated commands can be used.

And, the AnACPU dedicated commands can be used independently of the existence of file register setting done with parameter.

(2) Use command shown in Section 8.5.6 when access is made to file registers (R) set with parameter or the specified block numbers.

(3) A head device number to be specified by the AnACPU dedicated command is calculated as follows.

When the device number in the nth block from the head is m (0 to 8191)

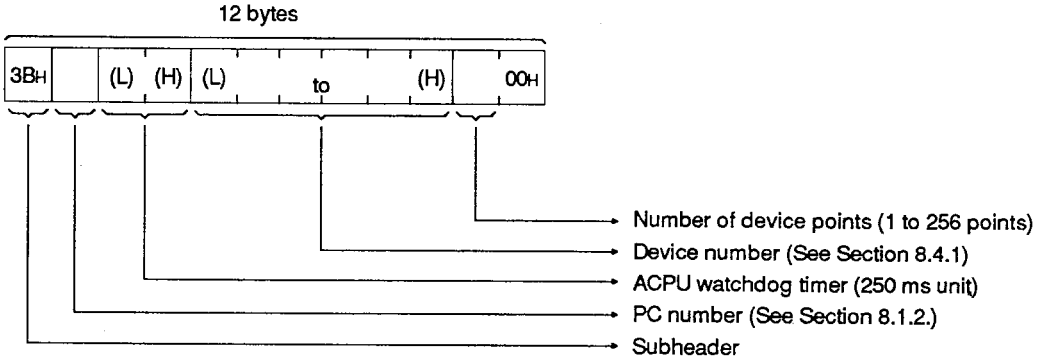
$$\text{Head device number} = (n-1) \times 8192 + m$$

(3) Batch read of extension file register

The comand and response formats are as follows when batch read fo extension file register is done:

(a) Communications in binary code

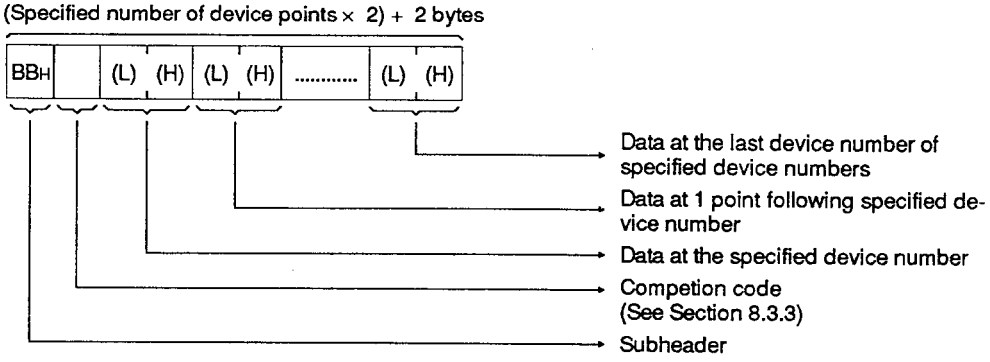
Command Format



REMARK

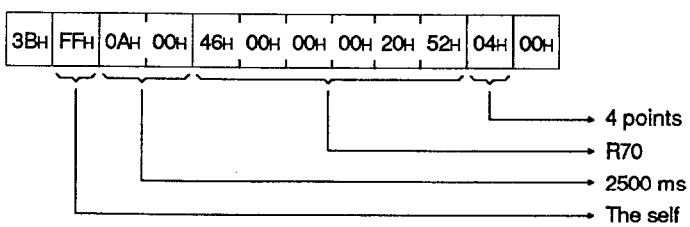
When number of device points is specified to 256 points, set "00H"

Response Format

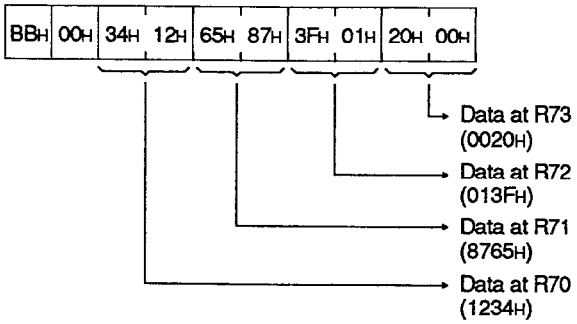


Example: When reading the data in extension file registers R70 to 73 of the PC CPU to which the A1SJ71E71 is loaded.

Command (Other node → A1SJ71E71)

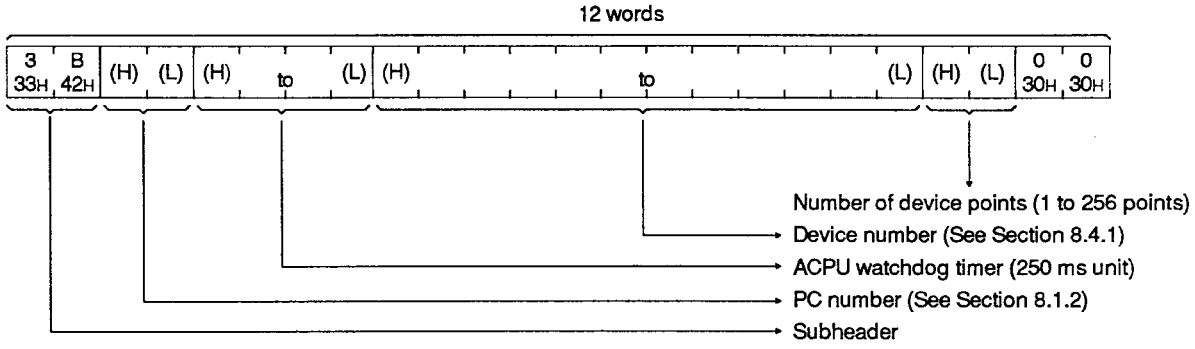


Response (A1SJ71E71 → Other node)



(b) Communications in ASCII code

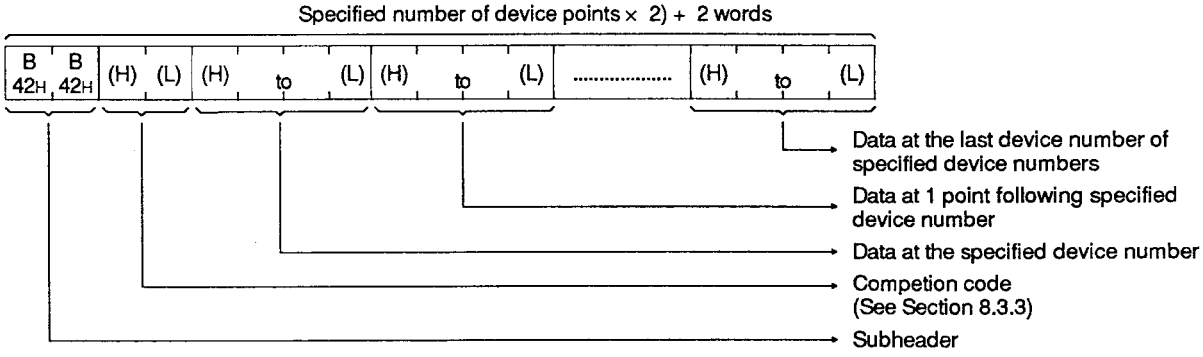
Command Format



REMARK

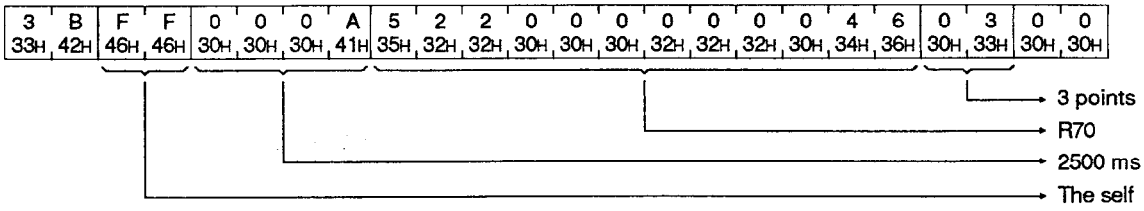
When number of device points is specified to 256 points, set "00H".

Response Format

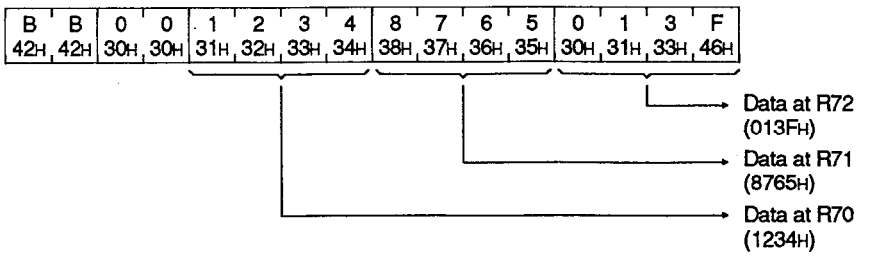


Example: When reading the data in extension file registers R70 to 72 of the PC CPU to which the A1SJ71E71 is loaded.

Command (Other node) → A1SJ71E71



Response (A1SJ71E71 → Other node)

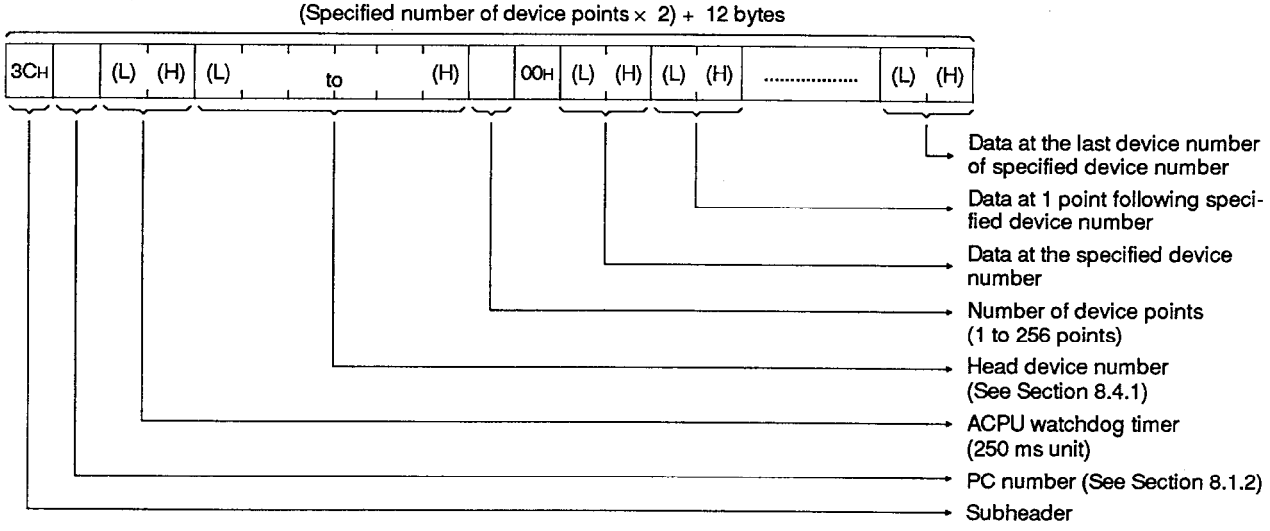


(4) Direct writ of extension file register

The command and response formats are as follows when write of extension file register is done:

(a) Communications in binary code

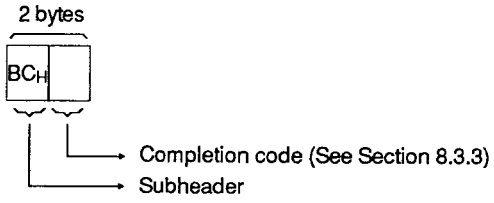
Command Format



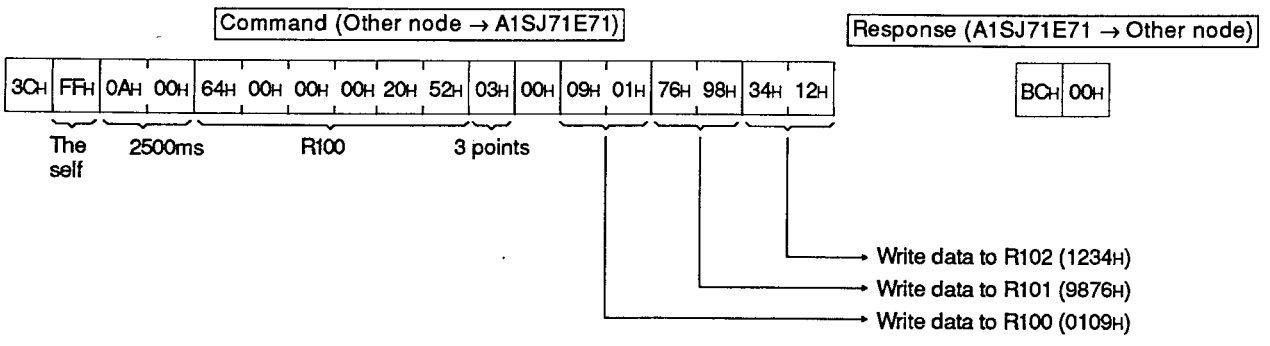
REMARK

When number of device points is specified to 256 points, set "00H".

Response Format

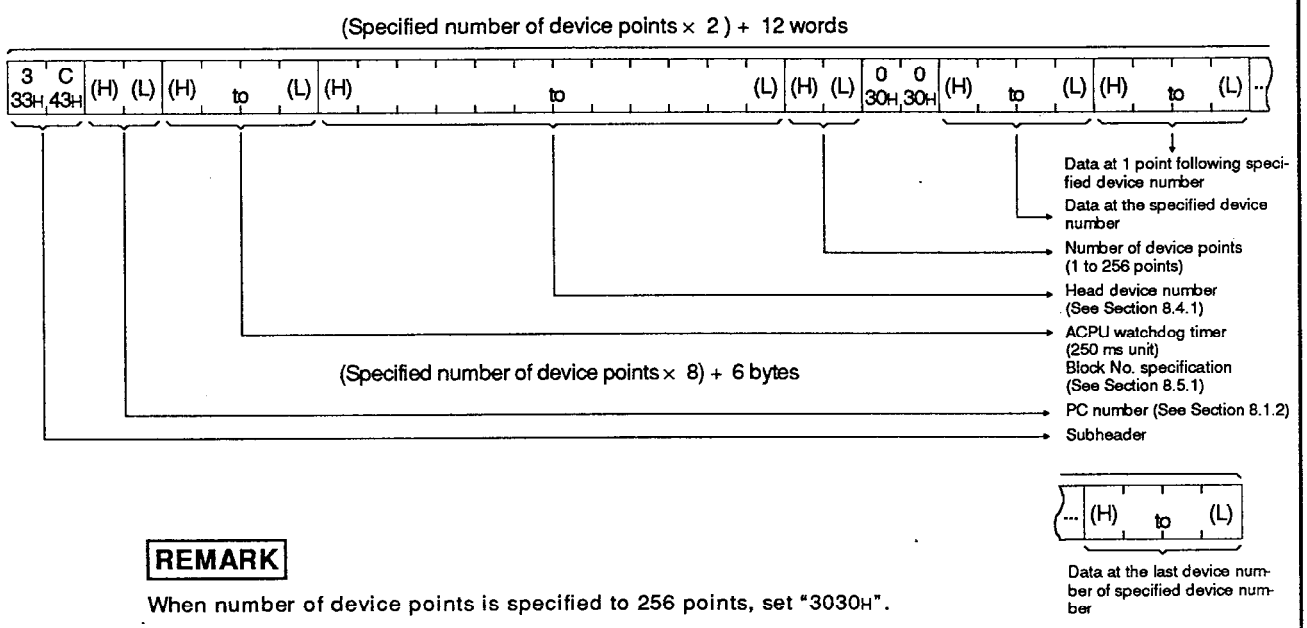


Example: When writing data to extension file registers R100 to 102 of the PC CPU to which the A1SJ71E71 is loaded.



(b) Communications in ASCII code

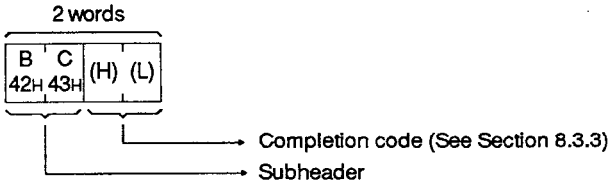
Command Format



REMARK

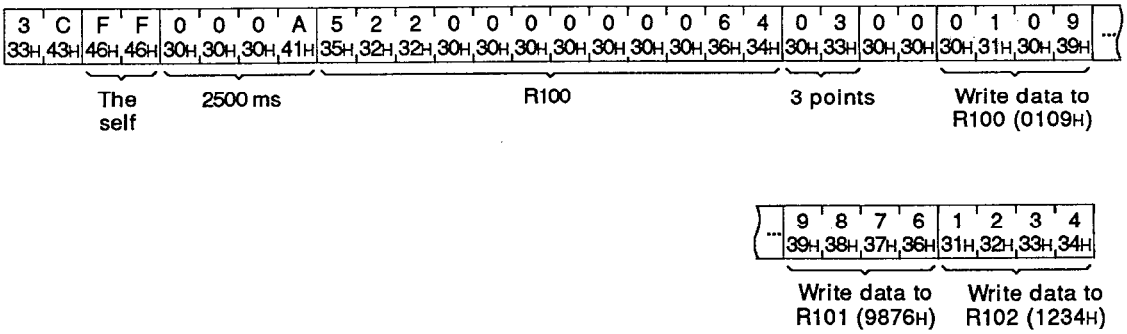
When number of device points is specified to 256 points, set "3030H".

Response Format

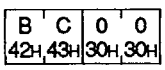


Example: When writing data to extension file registers R100 to 102 of the PC CPU to which the A1SJ71E71 is loaded.

Command (Other node → A1SJ71E71)



Response (A1SJ71E71 → Other node)



8.6 Command/Response Format for Read/Write of Special-Function Module Data

This section explains the specification and method of control protocols, and the examples, when data of the buffer memory area of a special-function module is read, or when data is written to the buffer memory area.

If this command is used, the buffer memory of a special-function module is accessed in byte units.

8.6.1 Commands and method of specification

(1) Table 8.7 shows the function of read/write of special-function module data.

Table 8.7 Function

Item	Command/ Response Classification	Processing	Number of Points Processed per Communication	State of PC CPU		
				During STOP	During RUN	
					SW22 ON	SW22 OFF
Batch read	$0E_H$	Reads from special function module buffer memory.	256 words (128 bytes)	o	o	o
Batch write	$0F_H$	Writes to special function module buffer memory.		o	o	x

Note : o.....Executable
x.....No

(2) Model names of the special-function modules that can be linked, buffer memory head addresses and module numbers.

Special Function Module Name	Buffer Memory Head Address (Hexadecimal)	Module Number When Loaded in Slot No. 0
AD61(S1) high-speed counter module	80_H	01_H
A616AD analog-digital converter module	10_H	01_H
A616DAI digital-analog converter module	10_H	01_H
A616DAV digital-analog converter module	10_H	01_H
A616TD temperature-digital converter module	10_H	01_H
A62DA(S1) digital-analog converter module	10_H	01_H
A68AD(S2) analog-digital converter module	80_H	01_H
A68ADN analog-digital converter module	80_H	01_H
A68DAV/DAI digital-analog converter module	10_H	01_H
A68RD3/4 temperature-digital converter module	10_H	01_H
A84AD analog-digital converter module	10_H	02_H
A81CPU PID control module	200_H	03_H
A61LS position detection module	80_H	01_H
A62LS position detection module	80_H	02_H

Special Function Module Name	Buffer Memory Head Address (Hexadecimal)	Module Number When Loaded in Slot No. 0
AJ71PT32(S3) MELSECNET/MINI master module	20_H	01_H
AJ71C22(S1) multidrop link module	1000_H	01_H
AJ71C24(S3/S6/S8) computer link module	1000_H	01_H
AJ71UC24 computer link module	400_H	01_H
AD51(S3) intelligent communication module	800_H	02_H
AD51H(S3) intelligent communication module	800_H	02_H
AJ71C21(S1) terminal interface module	400_H	01_H
AJ71B62 B/NET interface module	20_H	01_H
AJ71P41 SUMINET interface module	400_H	01_H
AJ71E71 Ethernet interface module	400_H	01_H
AD51FD(S3) external fault diagnosis module	280_H	02_H
AD57G(S3) graphic controller module	280_H	02_H
AD70(D)(S2) positioning module	80_H	01_H
AD71(S1) positioning module	200_H	01_H
AD71-S2 positioning module	200_H	01_H
AD71-S7 positioning module	200_H	01_H
AD72 positioning module	200_H	02_H
A1SD61 high-speed counter module	10_H	01_H
A1S62DA digital-analog converter module	10_H	01_H
A1S62RD3/4 temperature-digital converter module	10_H	01_H
A1S64AD analog-digital converter module	10_H	01_H
A1SJ71C24-R2 computer link module	400_H	01_H
A1SJ71C24-PRF computer link module	400_H	01_H
A1SJ71C24-R4 computer link module	400_H	01_H
A1SD70 single axis positioning module	80_H	02_H
A1SD71-S2 positioning module	200_H	02_H
A1SD71-S7 positioning module	200_H	02_H
A1S63ADA analog input module	10_H	01_H
A1SJ71PT32-S3 MELSECNET/MINI master module	20_H	01_H

(3) Special-function module buffer memory

Each address of the special-function module buffer memory consists of 16 bits (1 word), and read/write between a PC CPU and a special-function module is executed using FROM/TO instructions.
When another node writes to or reads from the buffer memory of a special-function module via an A1SJ71E71, processing is executed in 1 address = 8 bits (1 byte) units.

The address (hexadecimal) to be specified by the other node is calculated as follows from the address for the FROM/TO instruction.

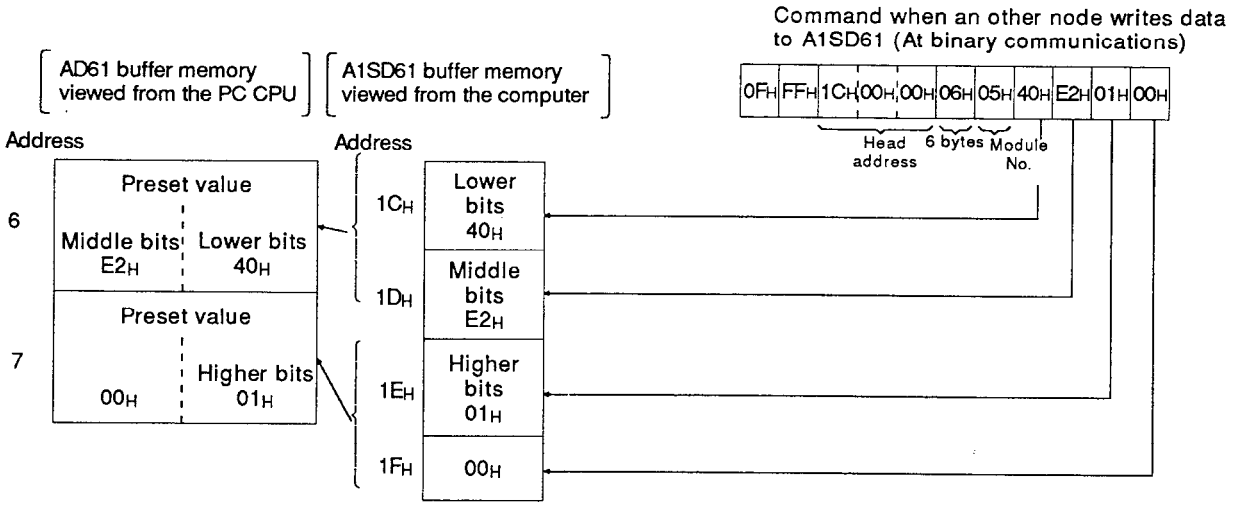
Address to be specified (Hexadecimal) = (Address for the FROM/TO instruction × 2) is converted into hexadecimal + Head address of each module

Example When address 6 (preset value) of the FROM/TO instruction of the A1SD61 high-speed counter module is specified:

Specified address = FROM/TO instruction address 6 × 2 + Head address

$1C_H$ C_H 10_H

The data format used when accessing the buffer memory of a special-function module from the other node via an A1SJ71E71 is explained below, using A1SD61 as an example.

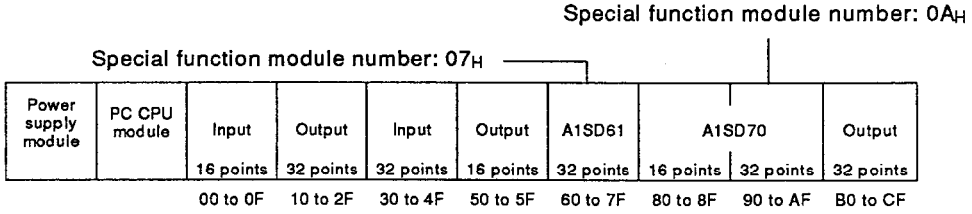


POINT

Buffer memory of each special-function module has a read/write area, a read-only area, a write-only area and a system-use area.
Execute this function according to the instructions of the operation manual of each module.
When wrong read/write is done, an error sometimes occurs in the PC CPU or in each special-function module.

(4) Special-function module number to be used with a command

When the head I/O address of a special-function module is expressed in 3 digits, the special-function module number specified by the control protocol is calculated by adding the "Module Number When Loaded in Slot 0" in the table in item (2) above to the upper two digits.
The system example below indicates the special-function module numbers.

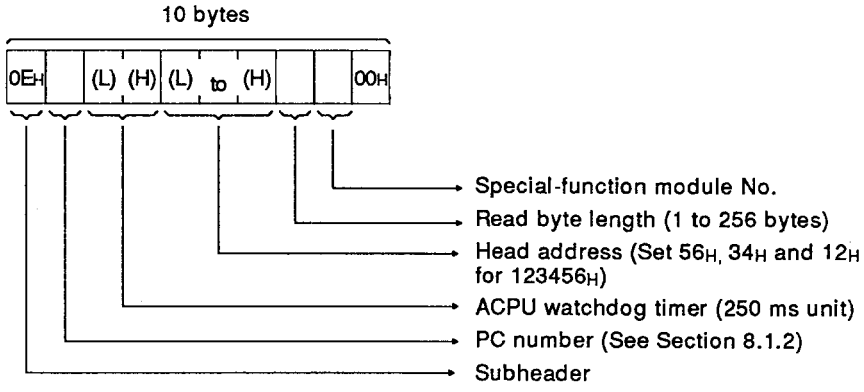


8.6.2 Reading special-function module buffer memory

The command and response formats as follows when data is read from the buffer memory of a special-function module:

(1) Communications in binary code

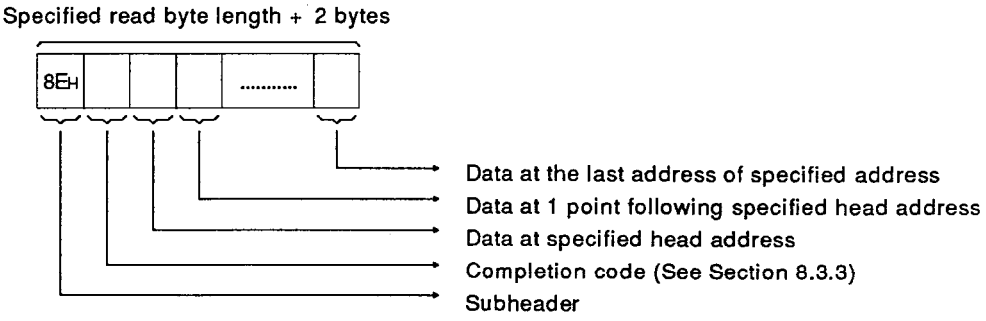
Command Format



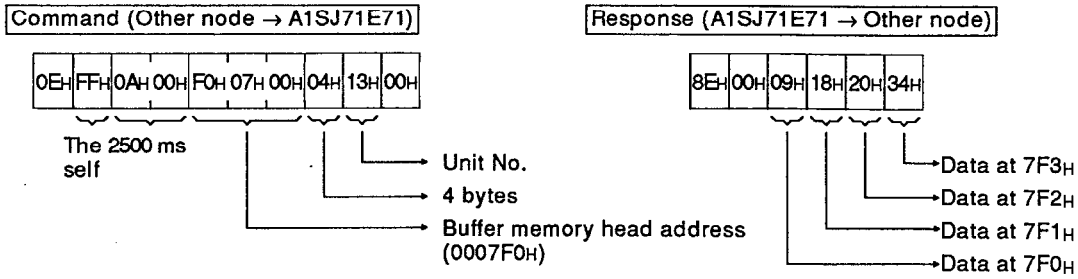
REMARK

When byte length is specified to 256 bytes, set "00_H".

Response Format

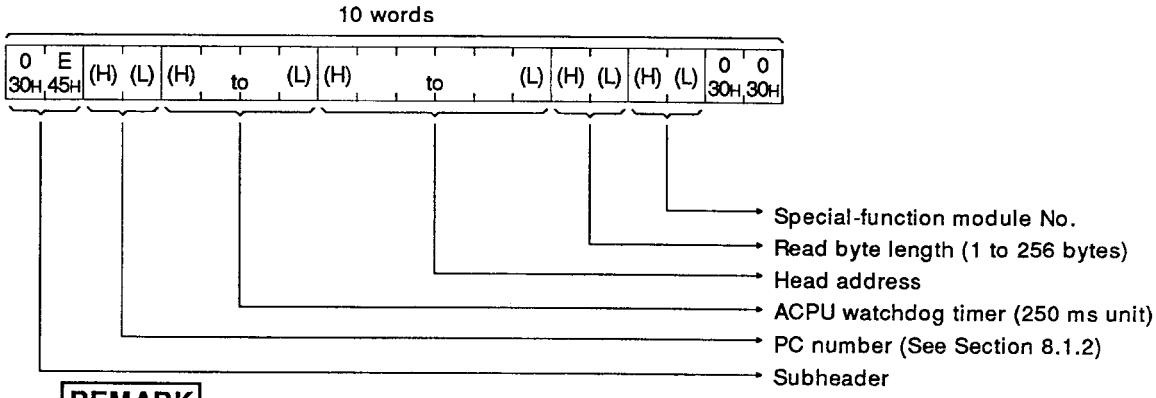


Example: When reading data of $7F0_H$ to $7F3_H$ of special-function modules (X and Y120 to 13F (module No. 13 H)) of a same station loaded with A1SJ71E71



(2) Communications in ASCII code

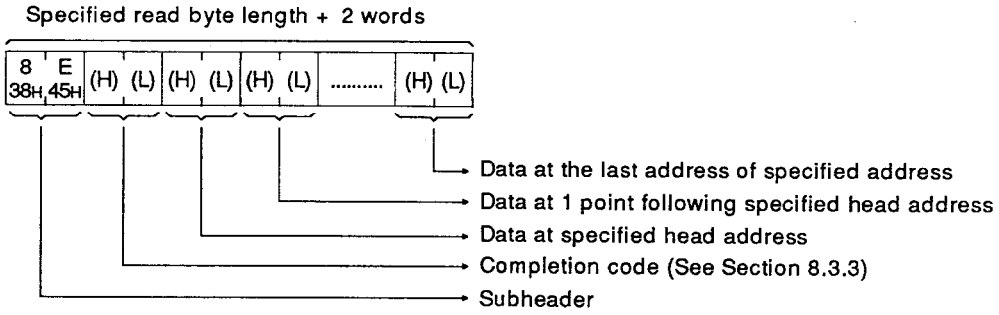
Command Format



REMARK

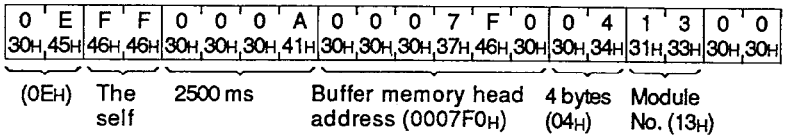
When byte length is specified to 256 bytes, set "3030H".

Response Format

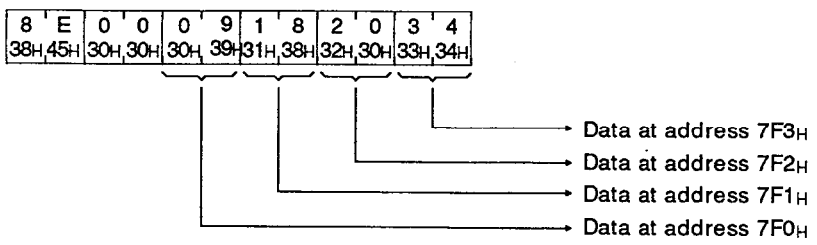


Example: When reading data of 7F0H to 7F3H of special-function modules (X and Y120 to 13F (module No. 13H)) of a same station loaded with A1SJ71E71

Command (Other node → A1SJ71E71)



Response (A1SJ71E71 → Other node)

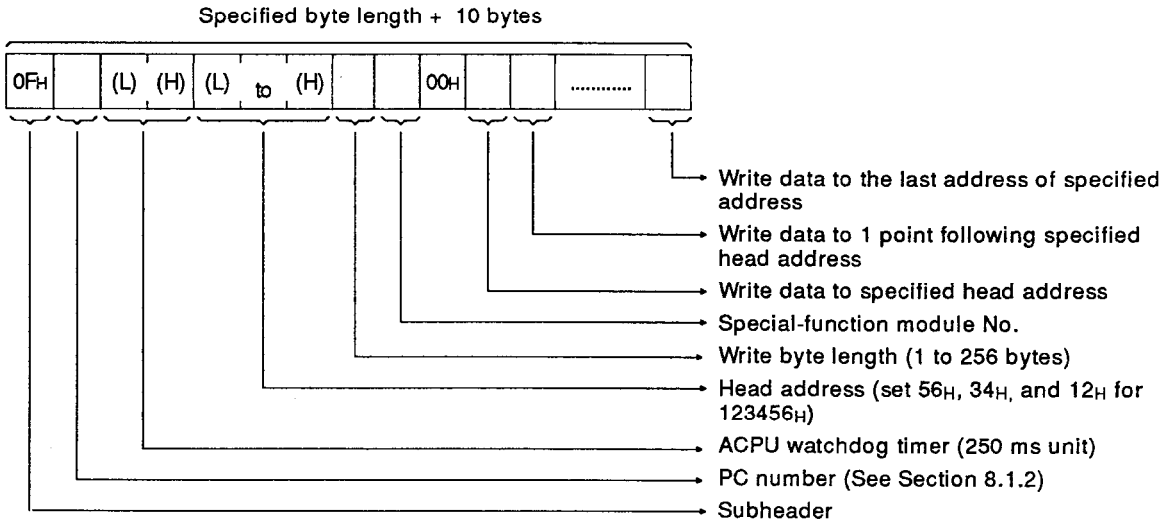


8.6.3 Writing special-function module buffer memory

The command and response formats are as follows when data is written to the buffer memory of a special-function module:

(1) Communications in binary code

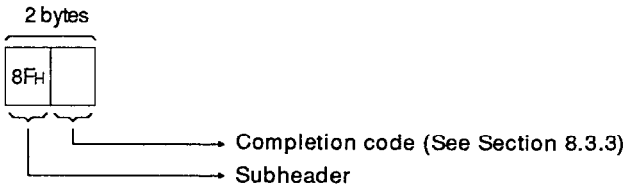
Command Format



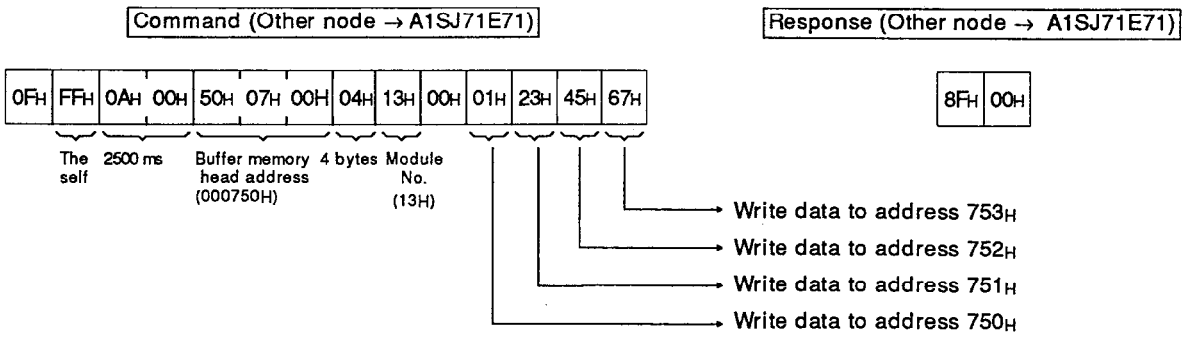
REMARK

When byte length is specified to 256 bytes, set "00H".

Response Format



Example: When writing data to 750H to 753H of special-function modules (X and Y120 to 13F (module No. 13H)) of a same station loaded with A1SJ71E71



(2) Communications in ASCII code

Command Format

Specified byte length + 10 words

0 F	(H) (L)	(H) to (L)	(H) to (L)	(H) (L)	(H) (L)	0 0	(H) (L)	(H) (L)	.....	(H) (L)
30H,46H						30H,30H				

- Write data to the last address of specified address
- Write data to 1 point following specified head address
- Write data to specified head address
- Special-function module No.
- Write byte length (1 to 256 bytes)
- Head address
- ACPU watchdog timer (250 ms unit)
- PC number (See Section 8.1.2)
- Subheader

REMARK

When byte length is specified to 256 bytes, set "3030_H".

Response Format

2 words

8 F	(H) (L)
38H,46H	

- Completion code (See Section 8.3.3)
- Subheader

Example: When writing data to 750_H to 753_H of special-function modules (X and Y120 to 13F (module No. 13_H)) of a same station loaded with A1SJ71E71

Command (Other node → A1SJ71E71)

0 F	F F	0 0 0 A	0 0 0 7 5 0	0 4	1 3	0 0	0 1	2 3	4 5	6 7
30H,46H	46H,46H	30H,30H,30H,41H	30H,30H,30H,37H,35H,30H	30H,34H	31H,33H	30H,30H	30H,31H	32H,33H	34H,35H	36H,37H
(0FH)	The self	2500 ms	Buffer memory head address (000750H)	4 bytes (04H)	Module No. (13H)					

- Write data to address 753H
- Write data to address 752H
- Write data to address 751H
- Write data to address 750H

Response (A1SJ71E71 → Other node)

8 F	0 0
38H,46H	30H,30H

8.7 Command and Response Formats for the Remote RUN/STOP and CPU Model Name Read

This function is used to make a remote RUN/STOP request to a PC CPU from another node and also to read the model name of the PC CPU that is linked with the node.
This section describes the specification contents and specification method for the control protocol used for this function, and gives an example of control protocol specicication.

8.7.1 Function

Table 8.8 shows the functions to be used for remote RUN/STOP and reading of CPU model name.

Table 8.8 Functions

Item	Command/ Response Classification	Description	PC CPU State		
			During STOP	During RUN	
				SW22 ON	SW22 OFF
Remote RUN	13_H	Requests remote RUN of PC CPU.	o	o	o
Remote STOP	14_H	Requests remote STOP of PC CPU.	o	o	o
PC CPU model read mode	15_H	Reads if the PC CPU is model A1N, A2N, A3N, A3H or AJ72P25/R25.	o	o	o

Note : o .....Executable
x.....Not executable

POINT

Remote RUN and remote STOP are enabled only for the CPU of a communicating station.
Remote STOP cannot be done for the CPU of the self. This is because, when the host station is set in the STOP status, the initial processing request signal (Y19) and open processing request signal (Y8 to F) go OFF, making communication between other nodes and the A1SJ71E71 impossible.

8.7.2 Remote RUN/STOP

(1) Control states by remote RUN/STOP

(a) The state of a PC CPU changes according to the specification of remote RUN/STOP from other stations and setting of the RUN/STOP keyswitch on the front of the PC CPU as shown below.

		Keyswitch Setting on the PC CPU Front			
		RUN	STOP	PAUSE	STEP-RUN
Specification from other node	Remote RUN	RUN	STOP	PAUSE	STEP-RUN
	Remote STOP	STOP	STOP	STOP	STOP

REMARKS

(a) If the relevant PC CPU has already been set in the remote STOP state via a special-function module such as another A1SJ71E71 or an A1SJ71C24-R2, it cannot be set in the RUN state by requesting remote RUN via the A1SJ71E71 at the host station.

(b) The state of special relays M9016 and M9017 decides whether a data memory is cleared or not before executing remote RUN.

Special Relay		State of Data Memory
M9016	M9017	
OFF	OFF	RUN is enabled without clearing memory.
OFF	ON	Data outside a latched range set in the parameter is cleared. (X image data for link is not cleared)
ON	ON/OFF	All data are cleared, and RUN is enabled.

REMARK

Be sure to set special relays M9016 and M9017 to the reset state not to clear data memory before executing remote RUN as shown above.

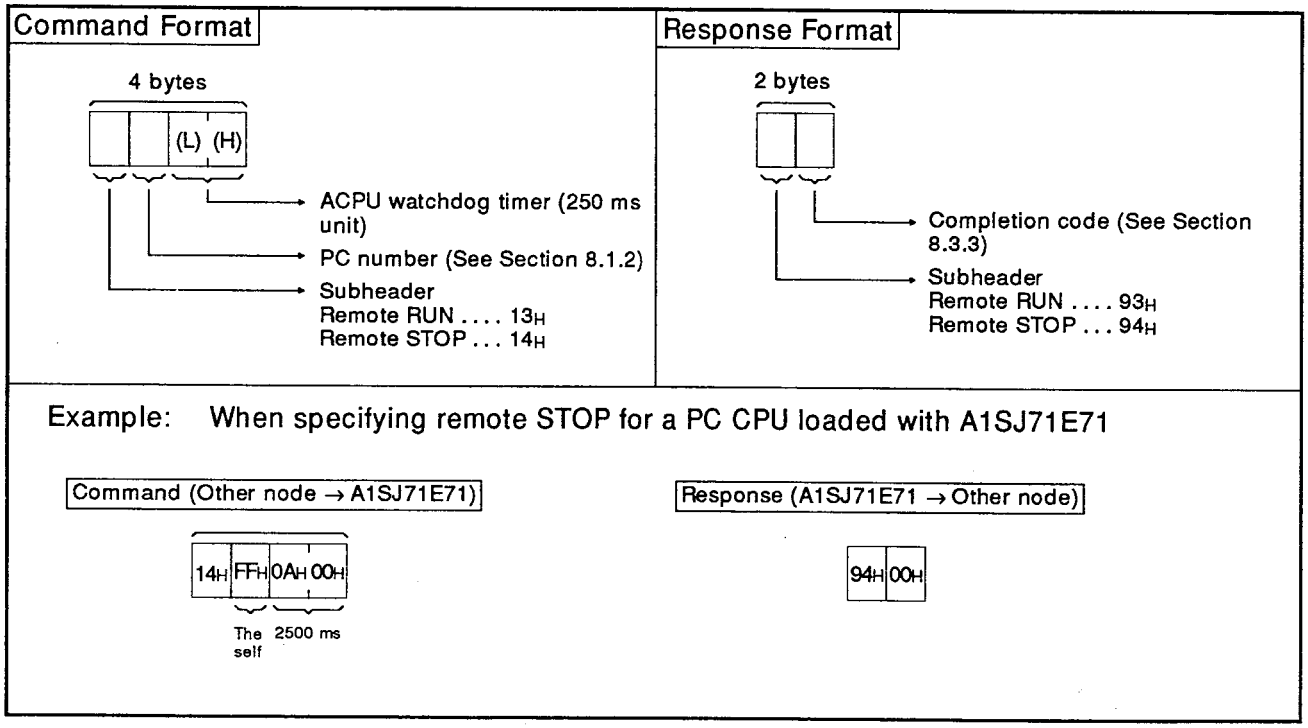
POINT

After operations remote RUN/STOP control from the other station are completed, the remote data will be lost if the power supply is turned OFF or the PC CPU is reset.

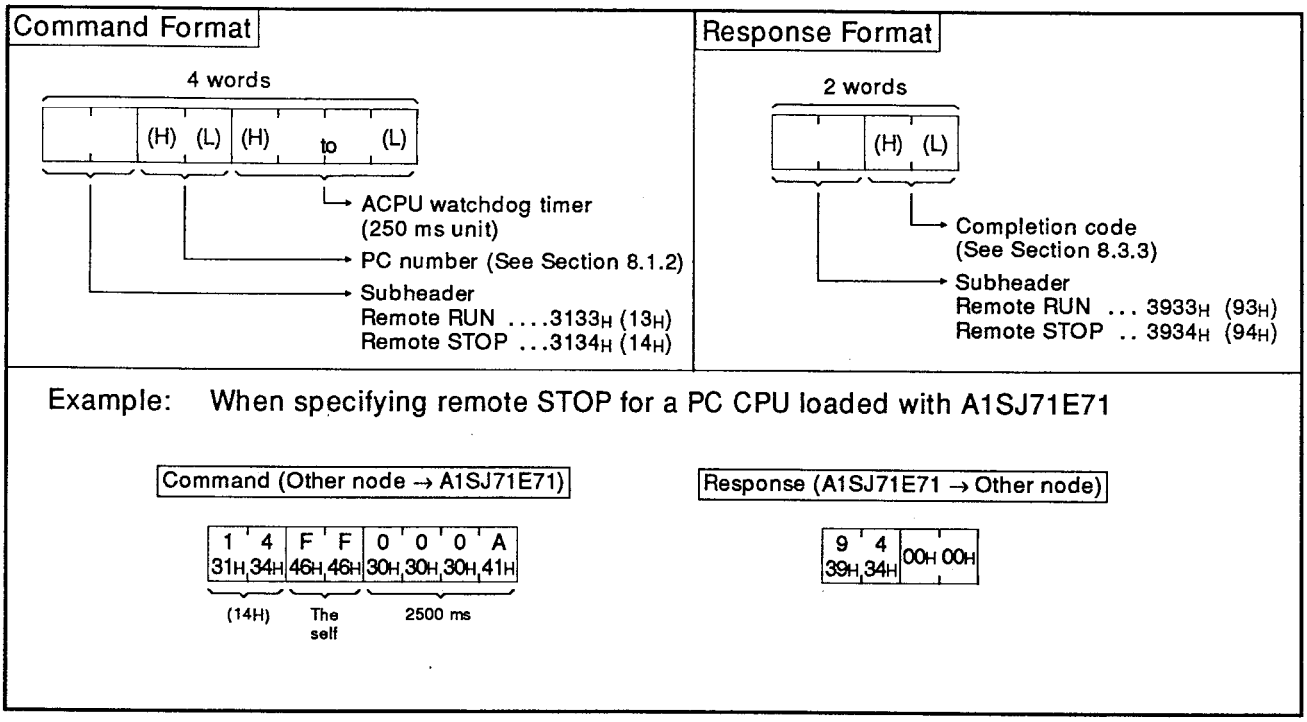
(2) Command/response formats

When remote RUN/STOP of a PC CPU is specified in the other, the command and response formats are as follows:

(a) Communications in binary code



(b) Communications in ASCII code



8.7.3 Read of PC CPU model name

This function reads the model name of the PC CPU that is communicating with another through an A1SJ71E71.

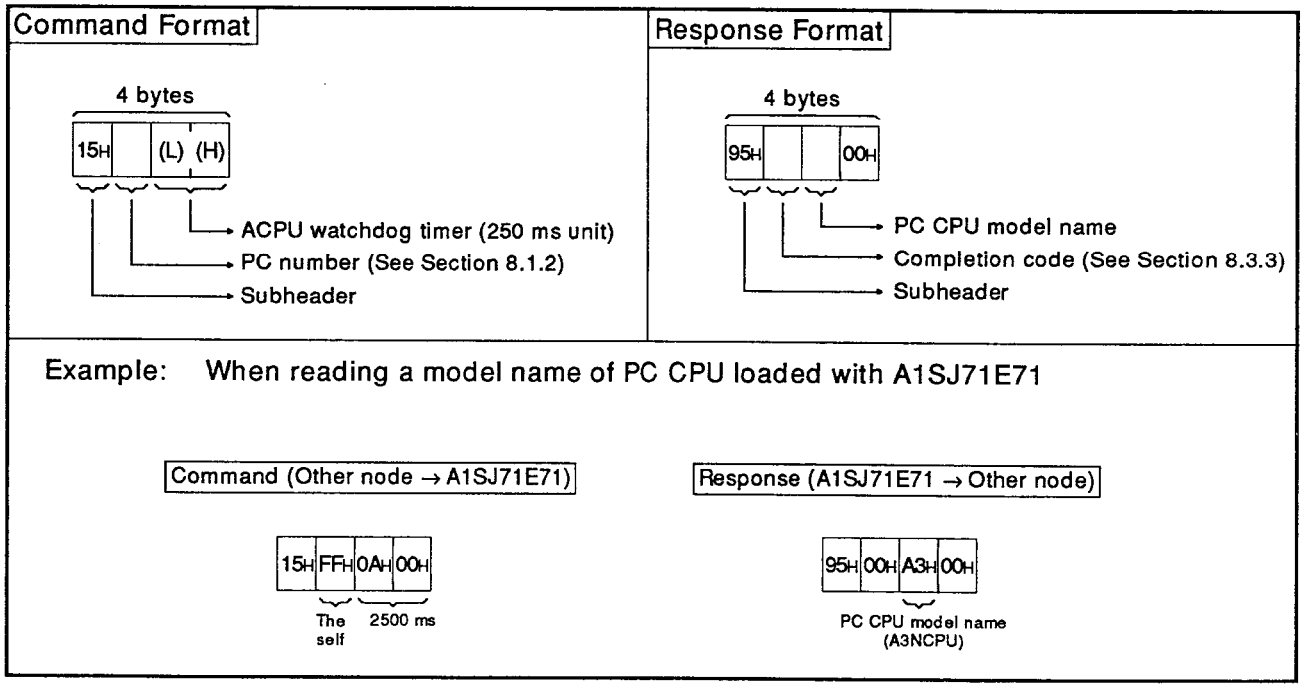
(1) PC CPU model names and read codes

PC CPU Model Name	Read Code (Hexadecimal)
A1CPU, A1NCPU	$A1_H$
A2SCPU, A2SCPU-S1, A2CPU-S1, A2NCPU, A2CPU, A2NCPU-S1	$A2_H$
A3CPU, A3NCPU, A3CPU	$A3_H$
A3HCPU, A3MCPU	$A4_H$
A2ASCPU, A2ACPU, A2UCPU	92_H
A2ASCPU-S1, A2ACPU-S1, A2UCPU-S1	93_H
A3ACPU, A3UCPU, A4UCPU	94_H
A1SCPU, A1SJCPU, A0J2HCPU	98_H
AJ72P25/R25	AB_H
A1SCPU-S1	None

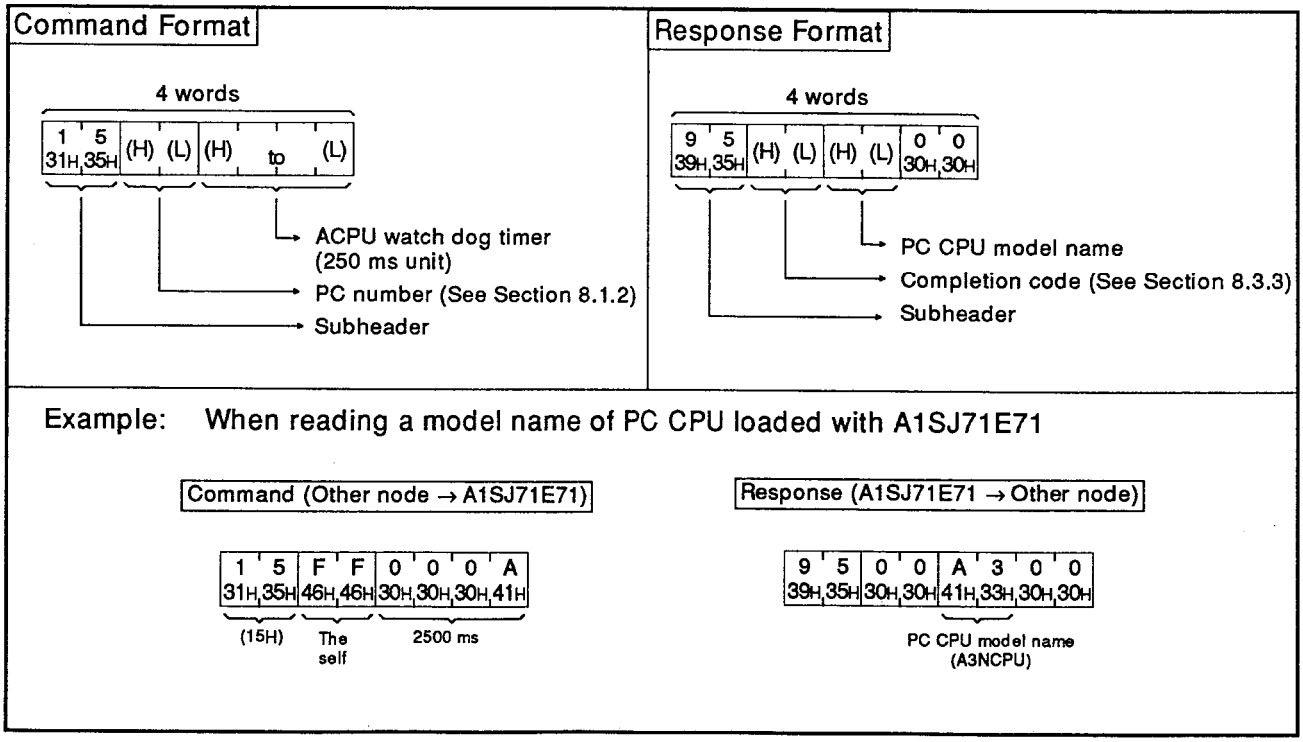
(2) Command/response formats

When the model name of a PC CPU is read from another, the command and the response formats are as follows:

(a) Communications in binary code



(b) Communications in ASCII code



8.8 Command/Response Format for Read/Write of a Program

When another node reads and stores several kinds of programs (main subsequence program and main submicrocomputer program), parameter data and comment data from the PC CPU and when another writes a program, parameter data and comment data in a PC CPU according to a control data this function is used.

8.8.1 Precautions for read/write of a program

The following explains precautions at the time of read/write of a program:

(1) When a program is read, read all areas of a sequence program, a microcomputer program, parameter data and comment data written by a PC CPU.

When a program is written, write all data that are read and stored in a PC CPU.

Be sure to write parameter data before writing a program. Then, execute an analysis request.

(2) If they are not executed, while the parameter of a user memory is changed, the parameter stored in the work area of a PC CPU is not changed.

Therefore even if it is loaded with a peripheral device, and a CPU is operated, after changing a parameter, the parameter is processed in the state of set contents before changing it (contents stored in a work area).

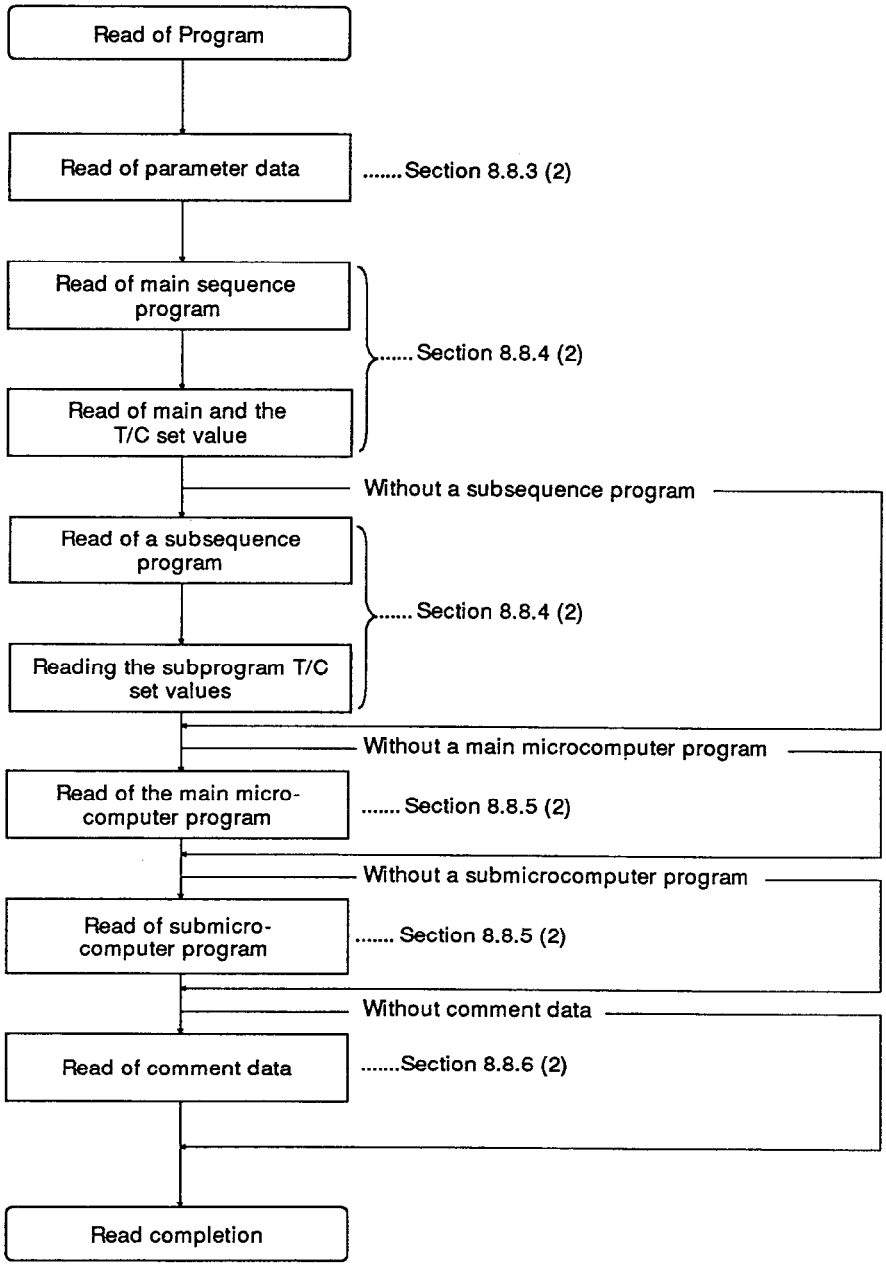
(3) The number of device points that can be processed in the communications at one time has been arranged for each command.

When read/write is done, divide data, and read/write of all areas.

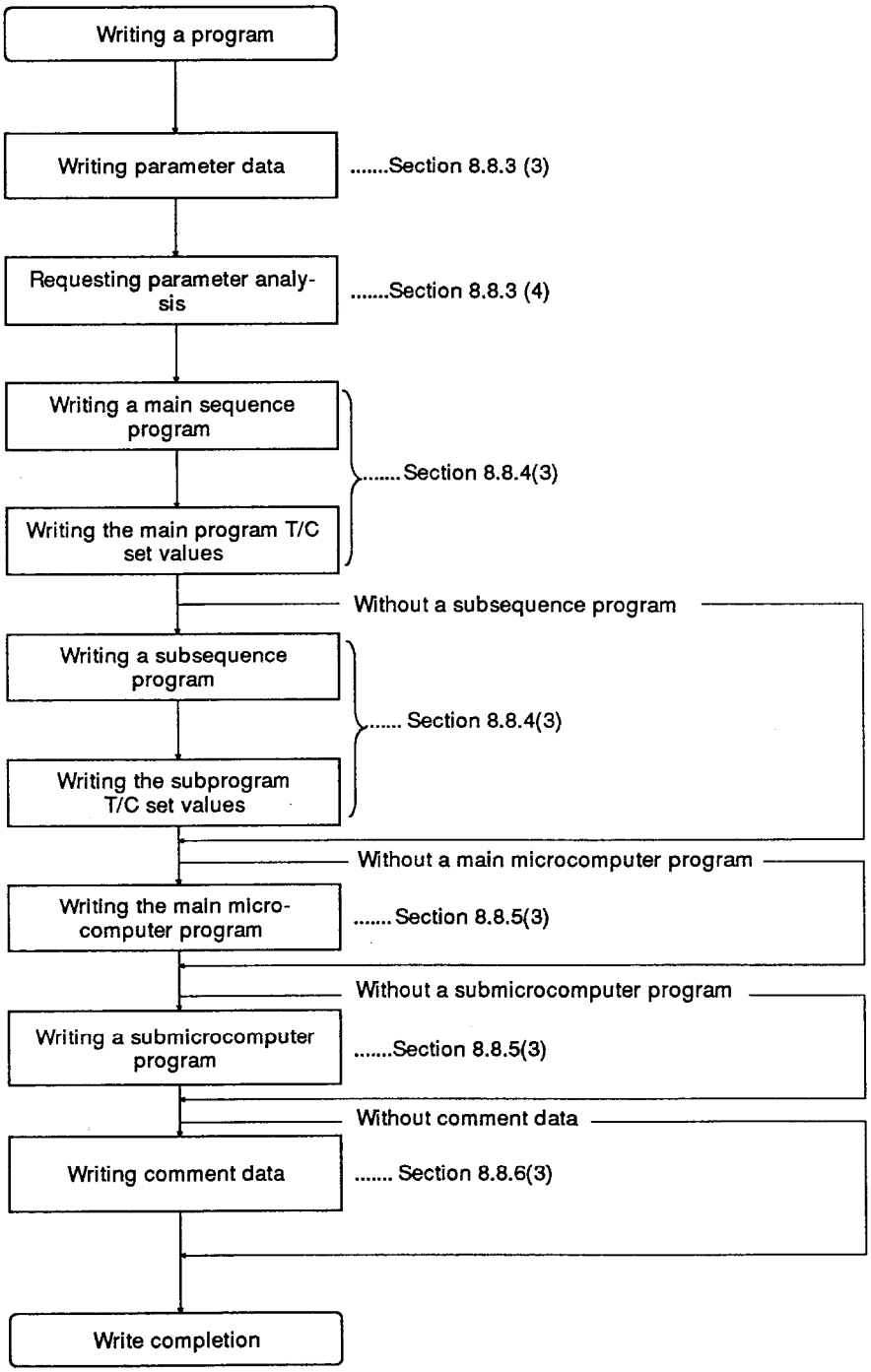
8.8.2 Operation procedure

The operation procedure to be used for read/write of a program is as follows:

(1) Read operation



(2) Write operation



8.8.3 Read/write of parameter memory

When the contents of the parameter memory of a PC CPU is read, or data is written in a parameter memory. The specification contents and the method and the specification example of a control protocol are as follows:

(1) Commands and addresses

(a) Table 8.9 shows a function to be used for parameter read/write.

Table 8.9 Function

Item	Command/ Response Classification	Processing	Number of Points Processed per Communication	PC CPU State		
				During STOP	During RUN	
					SW22 ON	SW22 OFF
Batch read	10_H	Read parameters from PC CPU	256 bytes	o	o	o
Batch write	11_H	Writes parameters to PC CPU.		o	x	x
Analysis request	12_H	Causes PC CPU to acknowledge and check rewritten parameters.		o	x	x

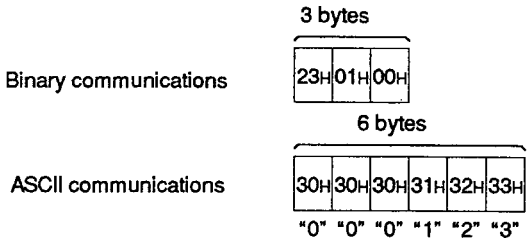
Note : o.....Executable
x.....Unavailable

(b) Address of a parameter

The parameter memory area has 3k bytes from addresses 0_H to BFF_H.

As shown in the examples below, the address is specified as 3 bytes for binary communications and as 6 bytes for ASCII communications.

When specifying example address 123_H



POINT

After writing all data that requires to change, change a parameter memory. And then, execute a parameter analysis request.
If it is not written, the parameter in the user memory is changed, but the parameter stored in the work area of the PC CPU is not changed.
Therefore, even if the CPU is loaded with a peripheral device and it is operated after changing the parameter, the CPU executes processing with the parameter setting before it is changed (contents stored in the work area).

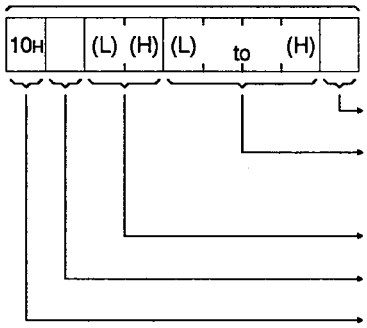
(2) Batch read

When a parameter memory contents of a PC CPU is read, the command and response formats are as follows:

(a) Communications in binary code

Command Format

8 bytes

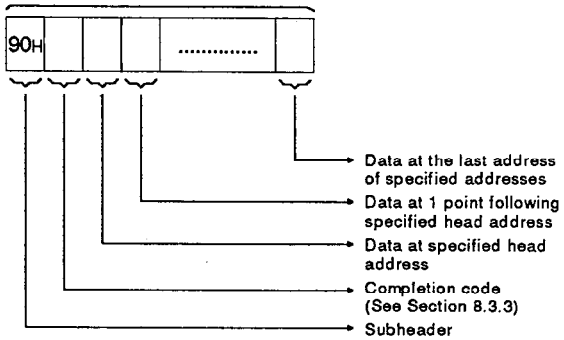


REMARK

When byte length is specified to 256 bytes, set "00_H".

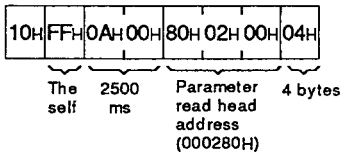
Response Format

Specified read byte length + 2 bytes

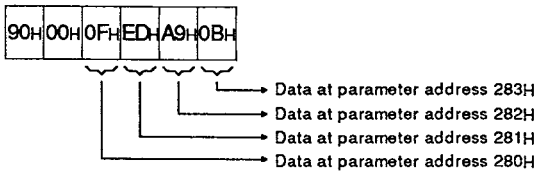


Example: When reading parameter data (280_H to 283_H) of a PC CPU loaded with A1SJ71E71

Command (Other node → A1SJ71E71)



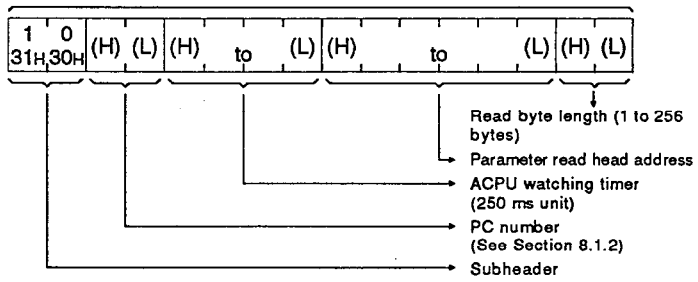
Response (A1SJ71E71 → Other node)



(b) Communications in ASCII code

Command Format

8 words

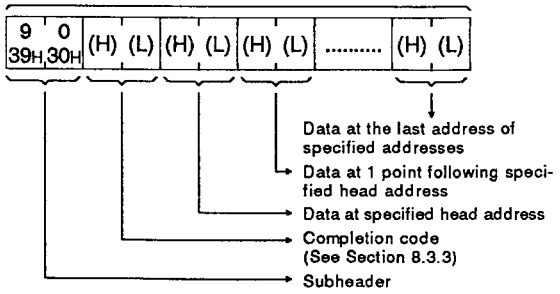


REMARK

When byte length is specified to 256 bytes, set "3030_H".

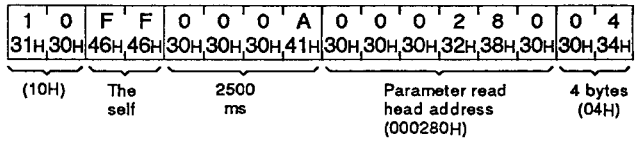
Response Format

Specified read byte length + 2 words

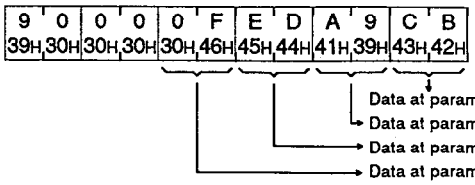


Example: When reading parameter data (280_H to 283_H) of a PC CPU loaded with A1SJ71E71

Command (Other node → A1SJ71E71)



Response (A1SJ71E71 → Other node)

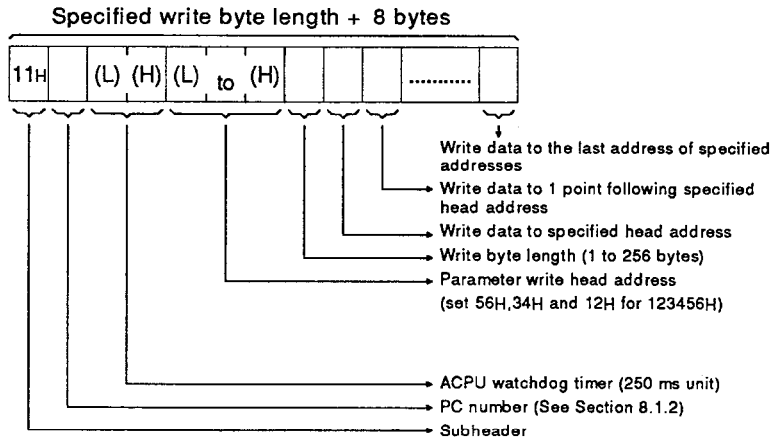


(3) Batch write

When data is written to a parameter memory contents of a PC CPU, the command and response formats are as follows:

(a) Communications in binary code

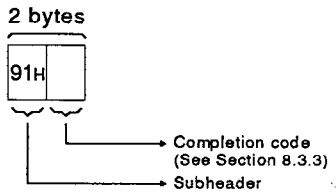
Command Format



REMARK

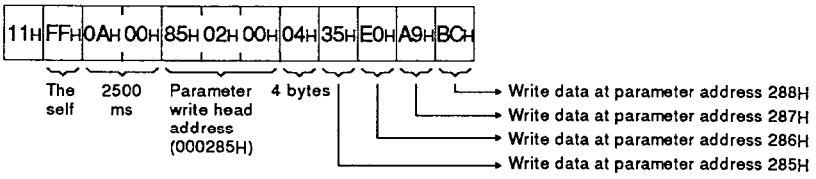
When byte length is specified to 256 bytes, set "00_H".

Response Format



Example: When reading parameter data (285_H to 288_H) of a PC CPU loaded with A1SJ71E71

Command (Other node → A1SJ71E71)



Response (A1SJ71E71 → Other node)

(b) Communications in ASCII code

Command Format

Specified write byte length + 8 words

1 1	(H) (L)	(H) to (L)	(H) to (L)	(H) (L)	(H) (L)	(H) (L)	.....	(H) (L)
31H,31H								

- Write data to the last address of specified addresses
- Write data to 1 point following specified head address
- Write data to specified head address
- Write byte length (1 to 256 bytes)
- Parameter write head address
- ACPU watchdog timer (250 ms unit)
- PC number (See Section 8.1.2)
- Subheader

REMARK

When byte length is specified to 256 bytes, set "3030H".

Response Format

2 words

9 1	(H) (L)
39H,31H	

Subheader Completion code (See Section 8.3.3)

Example: When reading parameter data (285H to 288H) of a PC CPU loaded with A1SJ71E71

Command (Other node → A1SJ71E71)

1 1	F F	0 0 0 A	0 0 0 2 8 5	0 4	3 5	E 0	A 9	B C
31H,31H	46H,46H	30H,30H,30H,41H	30H,30H,30H,32H,38H,35H	30H,34H	33H,35H	45H,30H	41H,39H	42H,43H
(11H)	The self	2500 ms	Parameter write head address (000285H)	4 bytes 04H				

- Write data (BCH) at parameter address 288H
- Write data (A9H) at parameter address 287H
- Write data (E0H) at parameter address 286H
- Write data (35H) at parameter address 285H

Response (A1SJ71E71 → Other node)

9 1	0 0
39H,31H	30H,30H

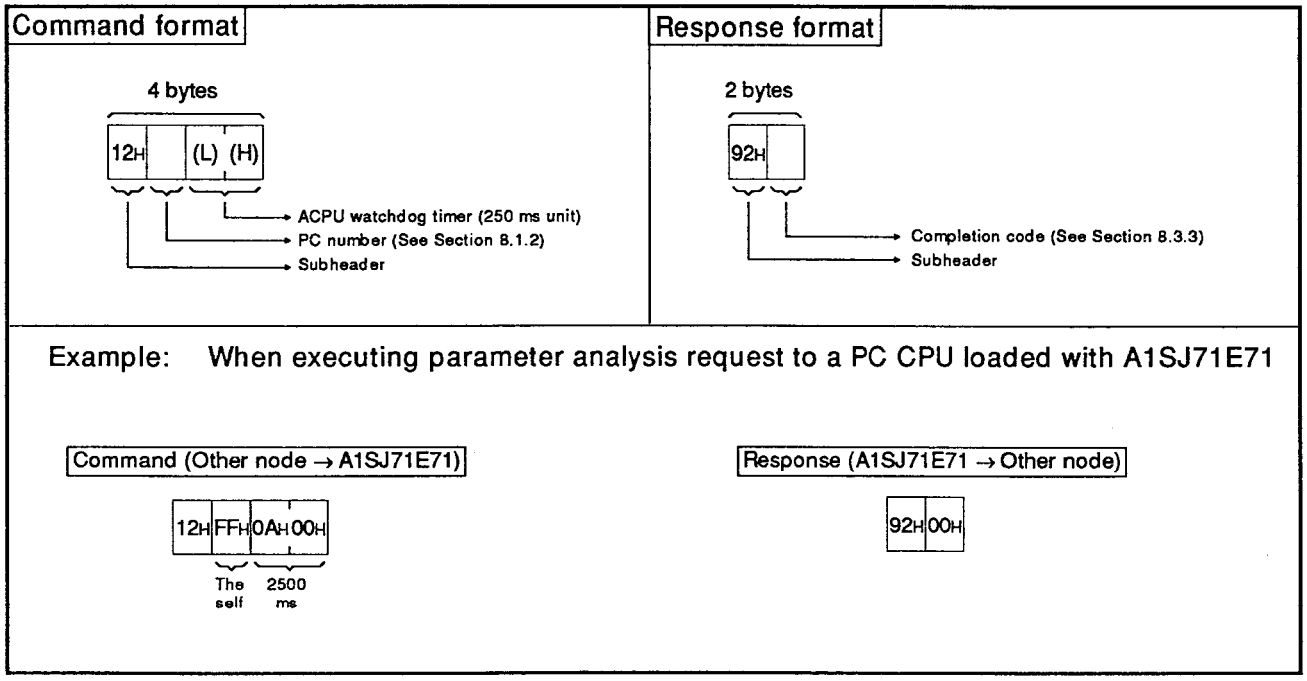
(4) Analysis request

When an analysis request of parameter data is made to a PC CPU, the command and the response formats are as follows:

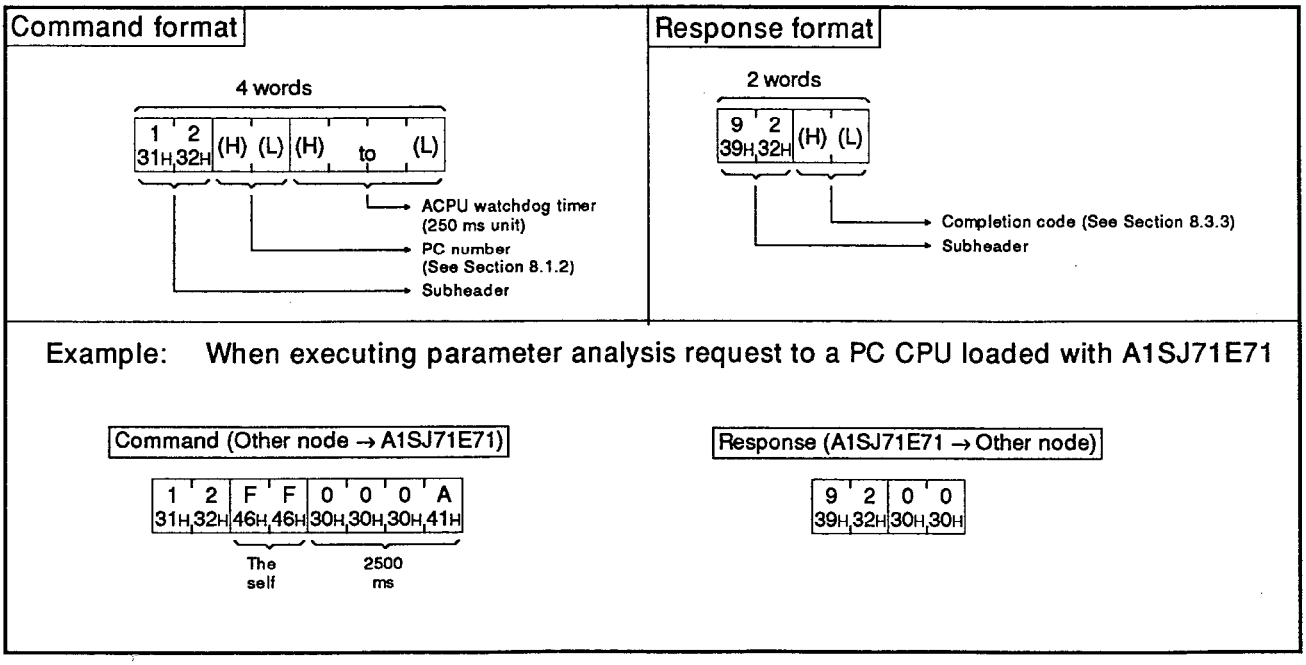
When a parameter is changed, the PC CPU is made to recognize the change of parameter by making an analysis request.

If an analysis request is not executed, the PC CPU cannot operate with changed parameter.

(a) Communications in binary code



(b) Communications in ASCII code



8.8.4 Read/write of sequence programs

When the sequence program of a PC CPU is read and written, the specification contents, the method and the specification example of a protocol are as follows:

(1) Command and method of setting

(a) Table 8.10 shows the functions to be used for read/write of sequence programs.

Table 8.10 Functions

Item			Command/ Response Classification	Processing	Number of Points Processed per Communication	PC CPU State		
						During STOP	During RUN	
							SW22 ON	SW22 OFF
Batch read	Main	Sequence program	$0A_H$	Reads main sequence programs.	256 steps	o	o	o
		T/C set value		Reads T/C set values used in main sequence programs.	256 points			
	Sub	Sequence program	$0B_H$	Reads subsequence programs.	256 steps	o	o	o
		T/C set value		Reads T/C set values used in subsequence programs.	256 points			
Batch write	Main	Sequence program	$0C_H$	Writes main sequence programs.	256 steps	o	o *	x
		T/C set value		Writes T/C set values used in main sequence programs.	256 points	o	o	x
	Sub	Sequence program	$0D_H$	Writes subsequence programs.	256 steps	o	o *	x
		T/C set value		Writes T/C set values used in subsequence programs.	256 points	o	o	x

Note : o.....Executable
x.....Not executable

* Writing during a program run may execute if all the following conditions are met:

1) The PC CPU is A3, A3N, A3H, A3M, A3A, A3U or A4U.
2) The program is not the currently running a program (indicates a subprogram called by the main program, if the main program is being run).
3) The PC CPU special relay is in the following state:
   i) M9050 (signal flow conversion contact).....OFF (A3CPU only)
   ii) M9051 (CHG instruction disable).....ON

(b) Step number specification of a sequence program

Specify the step number of a sequence program in hexadecimal as shown in Table 8.11.

Table 8.11 Step Number Specification

Step Number	Set Value
Step 0	0000_H
Step 1 to Step 30719 (30K)	0001_H to $77FE_H$

(c) Device number specification for read/write of the T/C set values

Set the device number for read/write of the T/C set values using the codes shown in Table 8.12.
Read/write of the T/C set values can be done within the range from T0 to T255 and from C0 to C255.
Read/write of T/C set values is executed in the range T0 to 255, and C0 to 255.
Set values in the ranges T256 to 2047 and C256 to 1023 cannot be read or written. Read/write set values by using device memory read/write.

Table 8.12 Specification of the T/C Set Values

Device Number	Setting Code
T0 set value T1 set value to T255 set value	$FE00_H$ $FE01_H$ to $FEFF_H$
C0 set value C1 set value to C255 set value	$FF00_H$ $FF01_H$ to $FFFF_H$

The formulas for the relationship between device numbers and setting codes are given below.

Timer : Tm = $FE00_H$ + n
Counter : Cm = $FF00_H$ + n
where, m = device number
n = hexadecimal value of device number

(d) Contents of the T/C set values

The T/C set values are stored in hexadecimals as shown in Table 8.13.

When the T/C set values are rewritten through an A1SJ71E71 from an other node, specify set data shown in Table 8.13.

Examples) Setting data to rewrite K10 of T10 to K20: 0014_H
Setting data to rewrite D30 of T11 to D10: 8014_H

Table 8.13 T/C Set Value Data Specification

Ladder Example in Program	Setting in Program	Setting Data
K[ ][ ][ ][ ][ ] T[ ][ ][ ] K[ ][ ][ ][ ][ ] C[ ][ ][ ]	K0 K1 to K9 K10 to K32767	0000_H 0001_H to 0009_H $000A_H$ to $7FFF_H$
D[ ][ ][ ][ ][ ] T[ ][ ][ ] D[ ][ ][ ][ ][ ] C[ ][ ][ ]	D0 D1 D2 to D1023	8000_H 8002_H 8004_H to $87FE_H$

Relationship between setting contents in the program and set data is as follows.

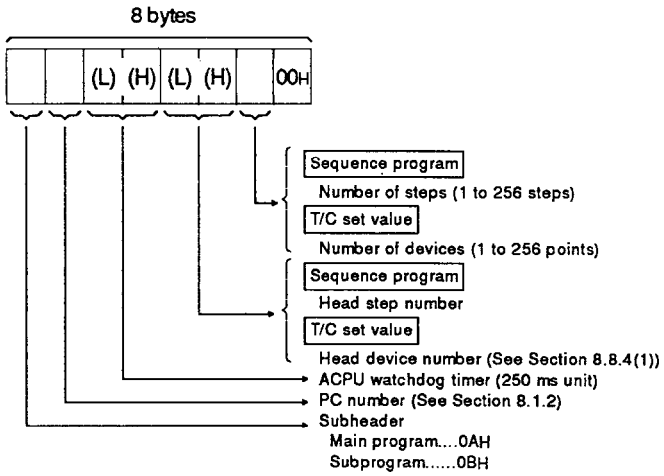
Km = 0000_H + n
Dm = 8000_H + 2n
m : Device number
n : Device number converted to hexadecimal

(2) Batch read

When batch read is done for the contents (machine language) of a sequence program or the set value of timer (T) and counter (C), the command and the response formats are as follows:

(a) Communications in binary code

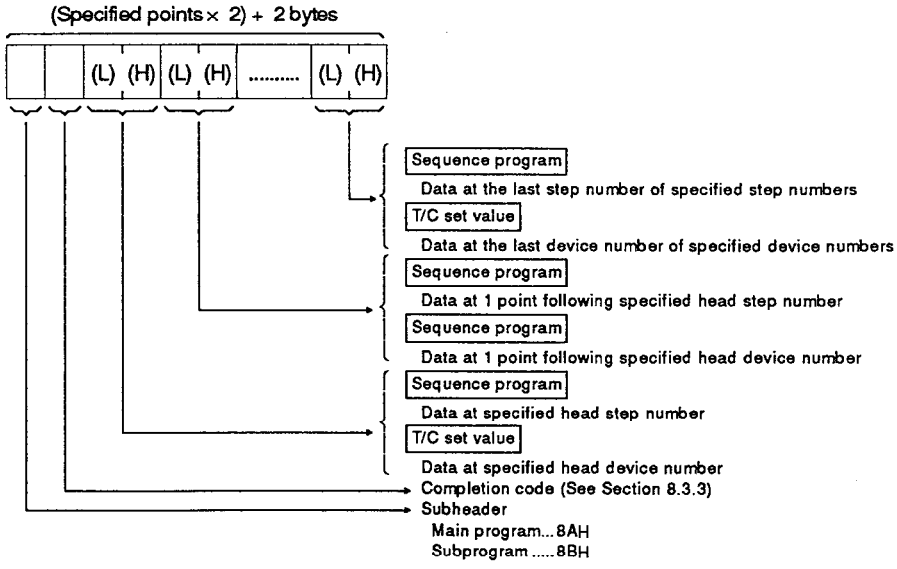
Command Format



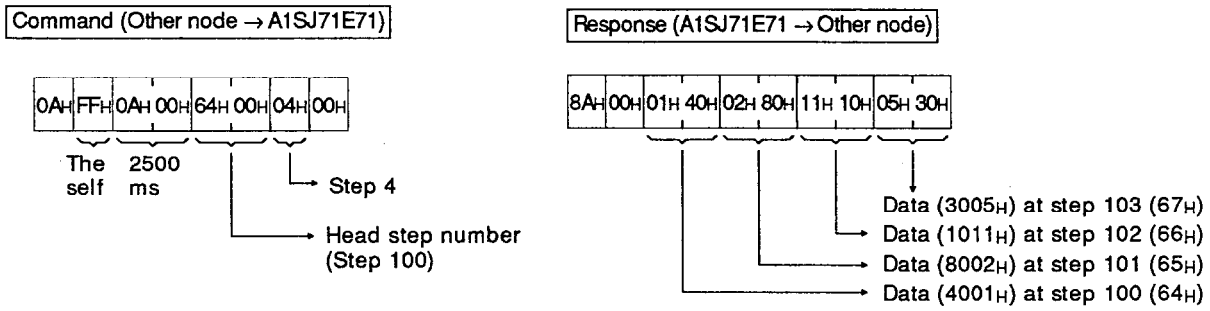
REMARK

When the number of steps or the number of devices is specified to 256 points, set "00_H".

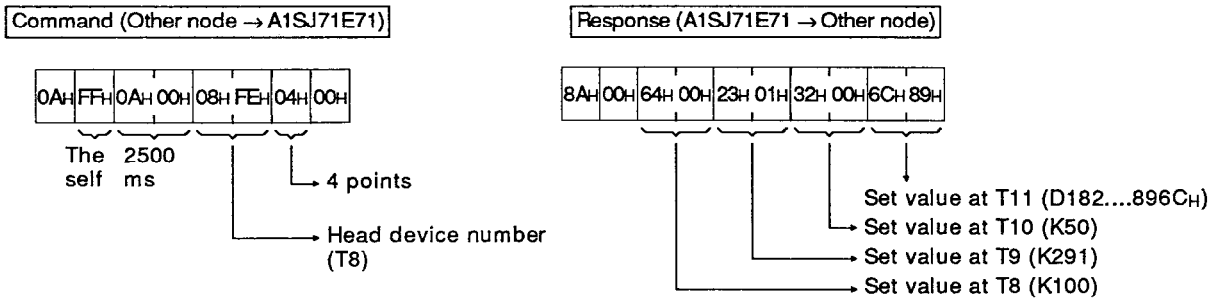
Response Format



Example 1: When reading a main sequence program (steps 100 to 103) of a PC CPU loaded with A1SJ71E71

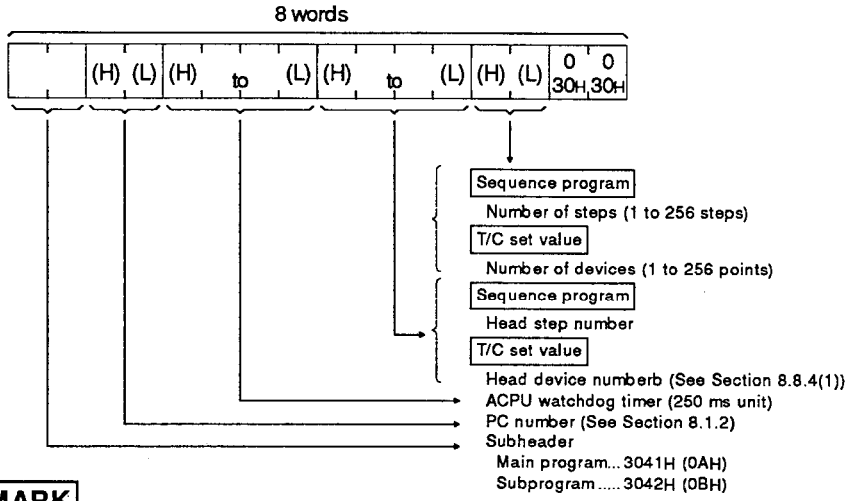


Example 2: When reading setting values of timer T8 to T11 used in a main sequence program of a PC CPU loaded with A1SJ71E71



(b) Communications in ASCII code

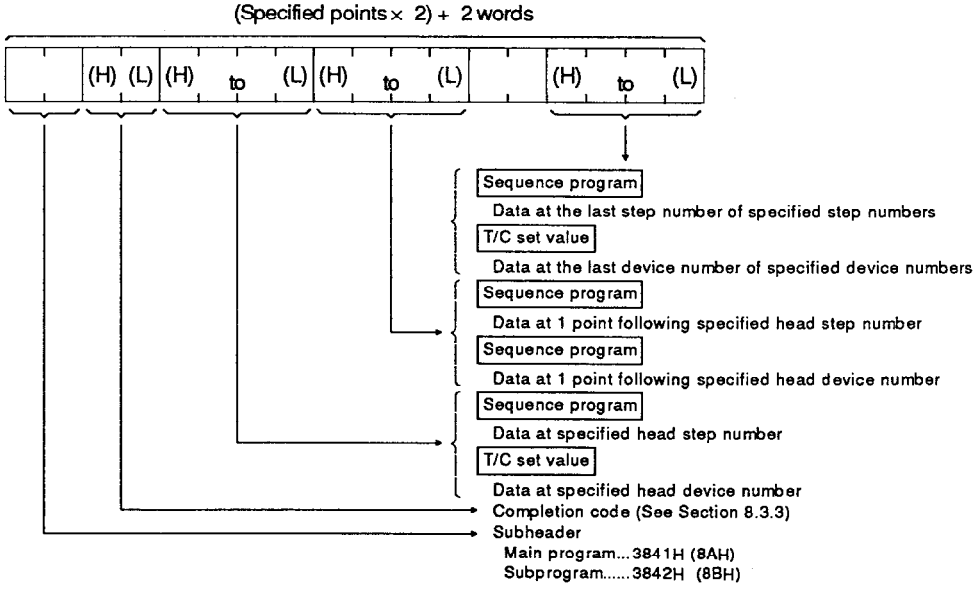
Command Format



REMARK

When the number of steps or the number of devices is specified to 256 points, set "3030_H".

Response Format



Example 1: When reading a main sequence program (steps 100 to 103) of a PC CPU loaded with A1SJ71E71

Command (Other node → A1SJ71E71)

0 A	F F	0 0 0 A	0 0 6 4	0 4	0 0
30H,41H	46H,46H	30H,30H,30H,41H	30H,30H,36H,34H	30H,34H	30H,30H
(0AH)	The self	2500 ms	Head step number (Step 100) (0064H)	Step 4 (04H)	

Response (A1SJ71E71 → Other node)

8 A	0 0	4 0 0 1	8 0 0 2	1 0 1 1	3 0 0 5
38H,41H	30H,30H	34H,30H,30H,31H	38H,30H,30H,32H	31H,30H,31H,31H	33H,30H,30H,35H
		Data (4001H) at step 100 (64H)	Data (8002H) at step 101 (65H)	Data (1011H) at step 102 (66H)	Data (3005H) at step 103 (67H)

Example 2: When reading set value of timer (T8 to T11) used in a main sequence program of a PC CPU loaded with A1SJ71E71

Command (Other node → A1SJ71E71)

0 A	F F	0 0 0 A	F E 0 8	0 4	0 0
30H,41H	46H,46H	30H,30H,30H,41H	46H,45H,30H,38H	30H,34H	30H,30H
(0AH)	The self	2500 ms	Head device number (T8) (FE08H)	Step 4 (04H)	

Response (A1SJ71E71 → Other node)

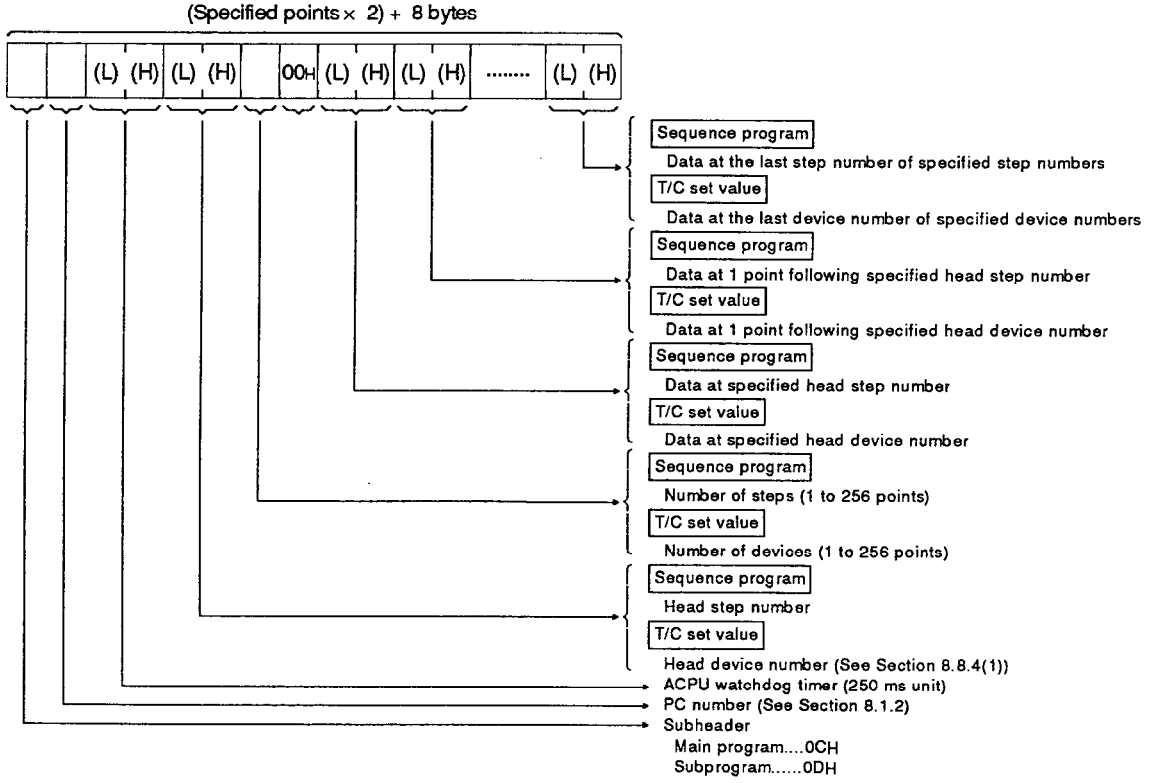
8 A	0 0	0 0 6 4	0 1 2 3	0 0 3 2	8 9 6 C
38H,41H	30H,30H	30H,30H,36H,34H	30H,31H,32H,33H	30H,30H,33H,32H	38H,39H,36H,43H
		Set value at T8 (0064H.....K100)	Set value at T9 (0123H.....K291)	Set value at T10 (0032H....K50)	Set value at T11 (896CH....D182)

(3) Batch write

When batch write is executed for the contents (machine language) of a sequence program or the set value of timer (T) and counter (C), the command and the response formats are as follows:

(a) Communications in binary code

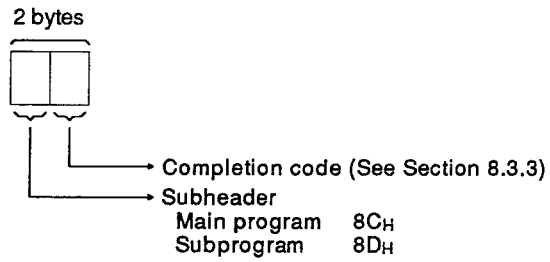
Command Format



REMARK

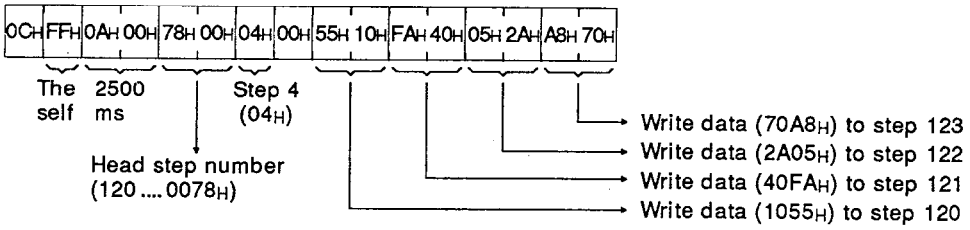
When the number of steps or the number of devices is specified to 256 points, set "00_H".

Response format



Example 1: When writing a main sequence program (steps 120 to 123) to a PC CPU loaded with A1SJ71E71

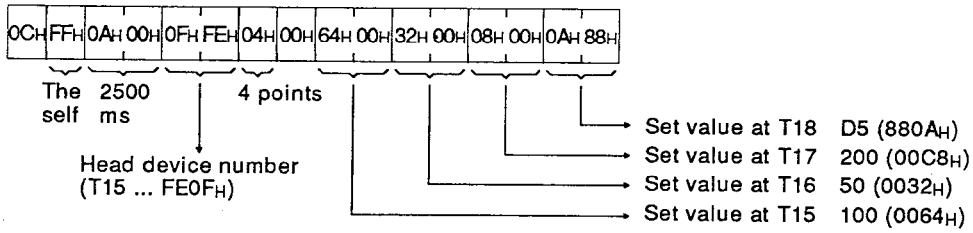
Command (Other node → A1SJ71E71)



Response (A1SJ71E71 → Other node)

Example 2: When changing set value of timer (T15 to T18) used in a main sequence program of a PC CPU loaded with A1SJ71E71

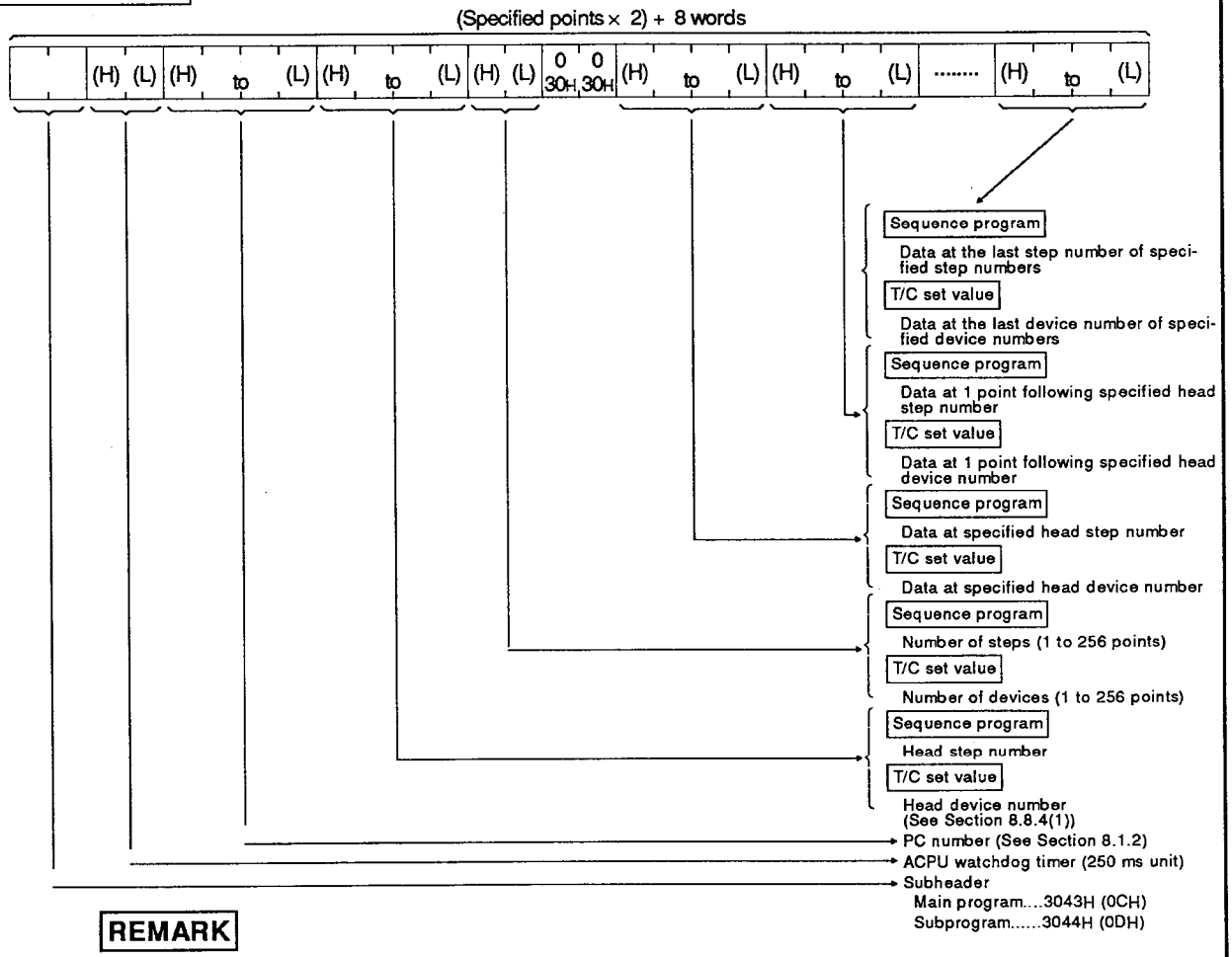
Command (Other node → A1SJ71E71)



Response (A1SJ71E71 → Other node)

(b) Communications in ASCII code

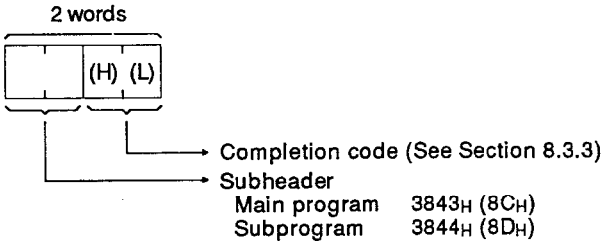
Command Format



REMARK

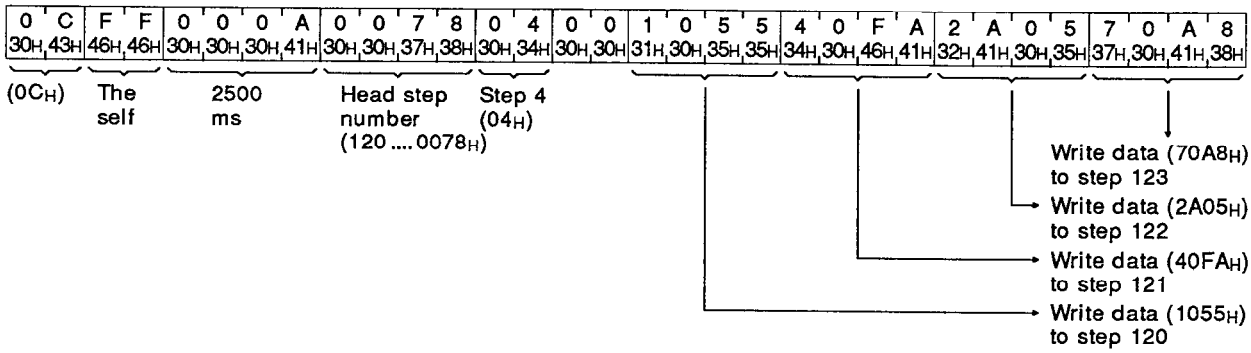
When the number of steps or the number of devices is specified to 256 points, set "3030_H".

Response format

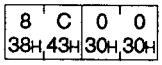


Example 1: When writing a main sequence program (steps 120 to 123) to a PC CPU loaded with A1SJ71E71

Command (Other node → A1SJ71E71)

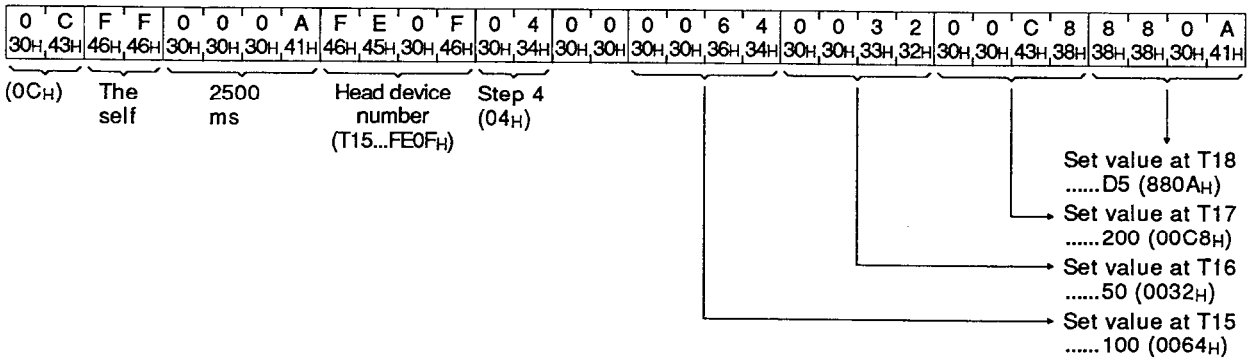


Response (A1SJ71E71 → Other node)

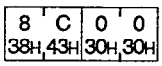


Example 2: When changing set value of timer (T15 to T18) used in a main sequence program of a PC CPU loaded with A1SJ71E71

Command (Other node → A1SJ71E71)



Response (A1SJ71E71 → Other node)



8.8.5 Read/write of a microcomputer program

This section describes the specification contents and specification method for the control protocol used for reading/writing the microcomputer program of a PC CPU, and gives an example of control protocol specification.

(1) Command and address

The command/response classification and the program addresses are as follows when read/write of a microcomputer program is done:

(a) Table 8.14 shows the functions for the read/write of a microcomputer program.

Table 8.14 Functions

Item		Command/ Response Classification	Processing	Number of Points Processed per Communication	State of PC CPU		
					During STOP	During RUN	
						SW22 ON	SW22 OFF
Batch read	Main	$1E_H$	Reads main microcomputer programs.	256 bytes	o	o	o
	Sub	$1F_H$	Reads submicrocomputer programs.				
Batch write	Main	20_H	Writes main microcomputer programs.		o	o*	x
	Sub	21_H	Writes submicrocomputer programs.				

Note : o.....Executable
x.....Not executable

* Writing during a program run may be executed if all the following conditions are met:
  1) The PC CPU is A3, A3N, A3H, A3M, A3A, A3U or A4U.
  2) The program is not a currently running program (indicates a subprogram called by the main program, if the main program is running).
  3) The PC CPU special relay is in the following state:

     M9050 signal flow conversion contact : OFF (A3CPU only)
     M9051 (CHG instruction disable) : ON

(b) Microcomputer program address

1) The address ranges which can be specified in each CPU are as shown in the following table:

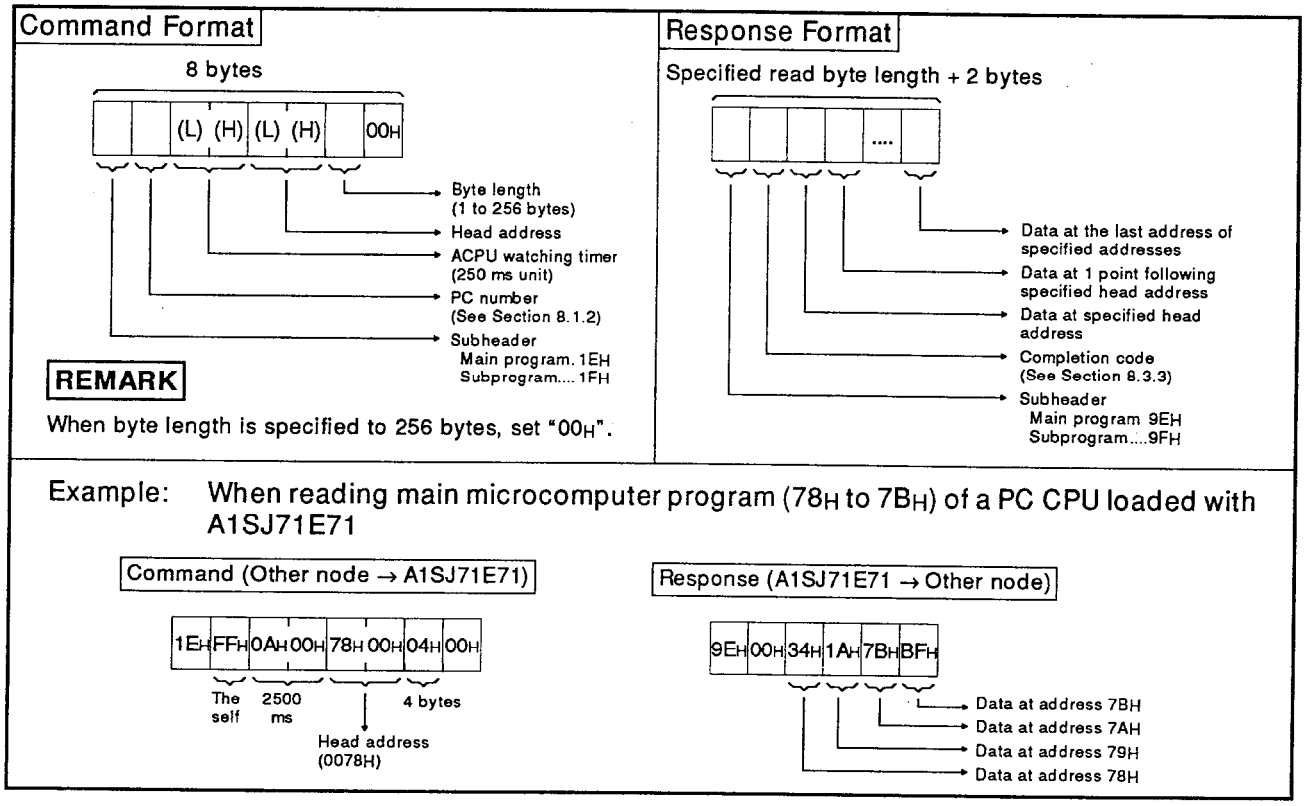
CPU Model Name	Microcomputer Program Capacity	Microcomputer Program Address
A1CPU, A1NCPU	Max. 10k bytes	0000_H to $27FF_H$
A2CPU (S1), A2NCPU (S1)	Max. 26k bytes	0000_H to $67FF_H$
A3CPU, A3NCPU, A3HCPU	Main/Sub, Max. 58 byte	0000_H to $E7FE_H$

2) When the sum of a head address and the number of bytes is larger than microcomputer program capacity, an error (completion code 57_H) occurs.

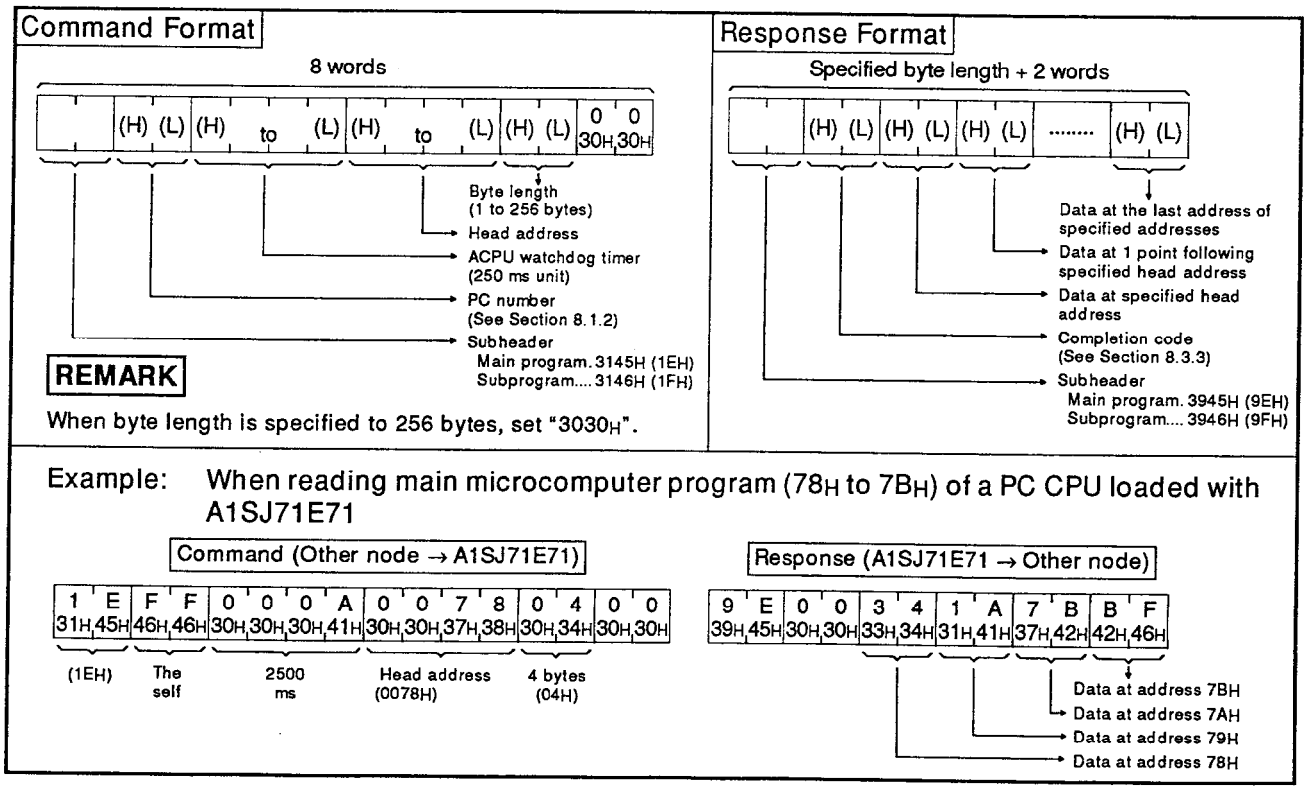
(2) Batch read

The command and the response formats are as follows when batch read of the contents of a microcomputer program is done:

(a) Communications in binary code



(b) Communications in ASCII code

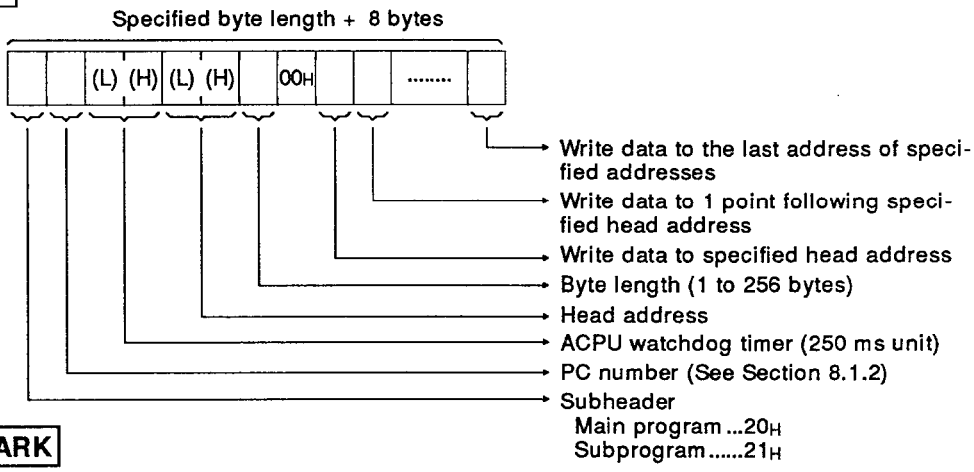


(3) Batch write

The command and response formats are as follows when the batch wirte of contents of microcomputer program is done:

(a) Communications in binary code

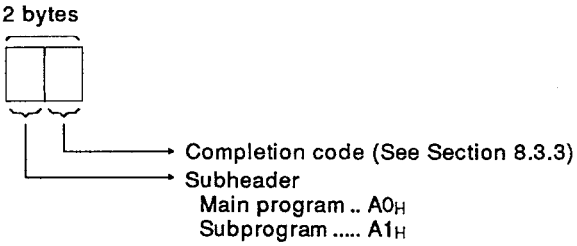
Command Format



REMARK

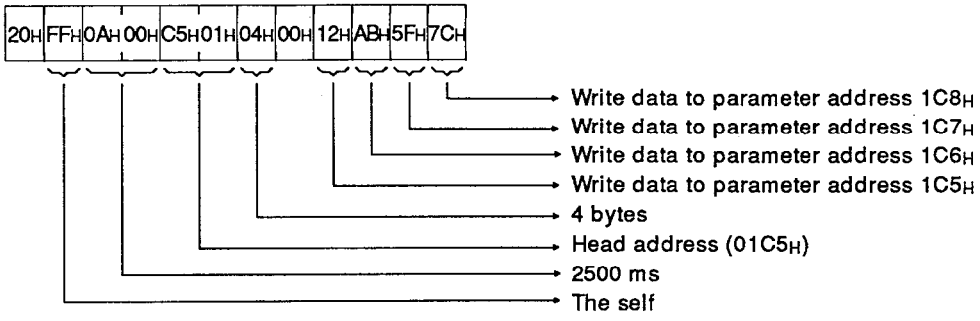
When byte length is specified to 256 bytes, set "00H".

Response Format



Example: When writing a main microcomputer program (1C5H to 1C8H) of a PC CPU loaded with A1SJ71E71

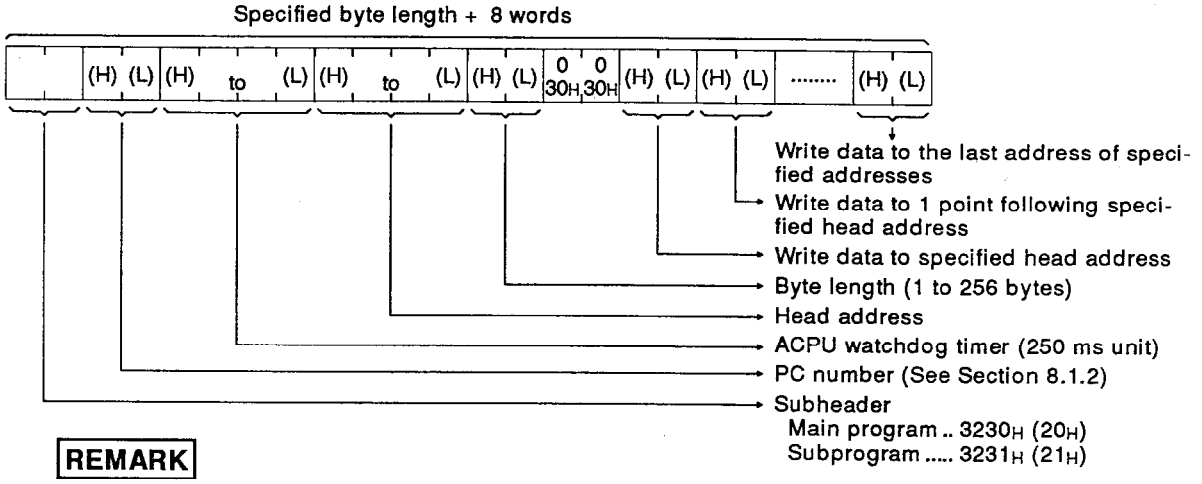
Command (Other node → A1SJ71E71)



Response (A1SJ71E71 → Other node)

(b) Communications in ASCII code

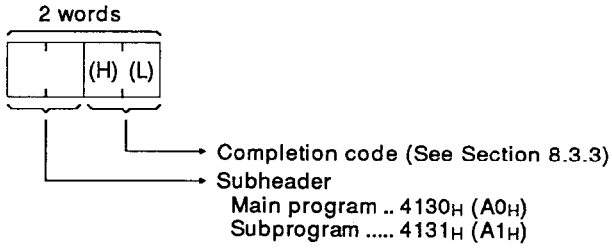
Command Format



REMARK

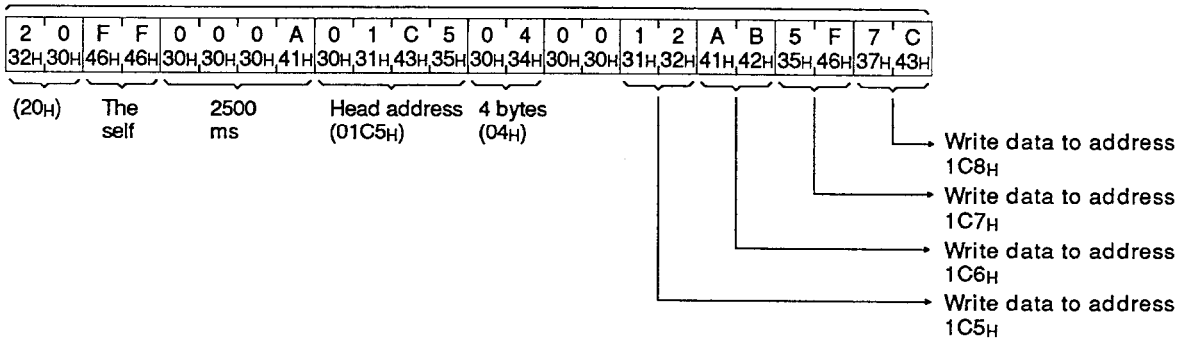
When byte length is specified to 256 bytes, set "3030H".

Response Format

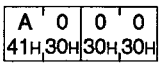


Example: When writing a main microcomputer program (1C5H to 1C8H) of a PC CPU loaded with A1SJ71E71

Command (Other node → AJ71E71)



Response (A1SJ71E71 → Other node)



8.8.6 Read/write of comment

This section describes the specification contents and specification method for the control protocol used for reading/writing the comment data of a PC CPU, and gives an example of control protocol specification.

(1) Commands and addresses

The command/response classification and the program addresses are as follows when read/write of comment data is done:

(a) Table 8.15 shows the functions for the read/write of comment data.

Table 8.15 Functions

Item	Command/ Response Classification	Processing	Number of Points Processed per Communication	State of PC CPU		
				During STOP	During RUN	
					SW22 ON	SW22 OFF
Batch read	$1C_H$	Reads from comment data.	256 bytes	o	o	o
Batch write	$1D_H$	Writes to comment data.		o	o	x

Note : o.....Executable
x.....Not executable

(b) Comment memory addresses

The comment data storage area is managed by the relative address which begins with the head address of 00_H.

For example, when the comment capacity of parameter is 2k bytes, the range from 00_H to $7FF_H$ can be specified for the head address.

1) The comment memory can be set up to 64k bytes.

The address range of comment data is fixed according to the set capacity of parameter.

2) Specify a comment memory address in hexadecimal.

3) When the sum of a head address and the number of bytes is larger than comment memory capacity, an error (completion code 57_H) occurs.

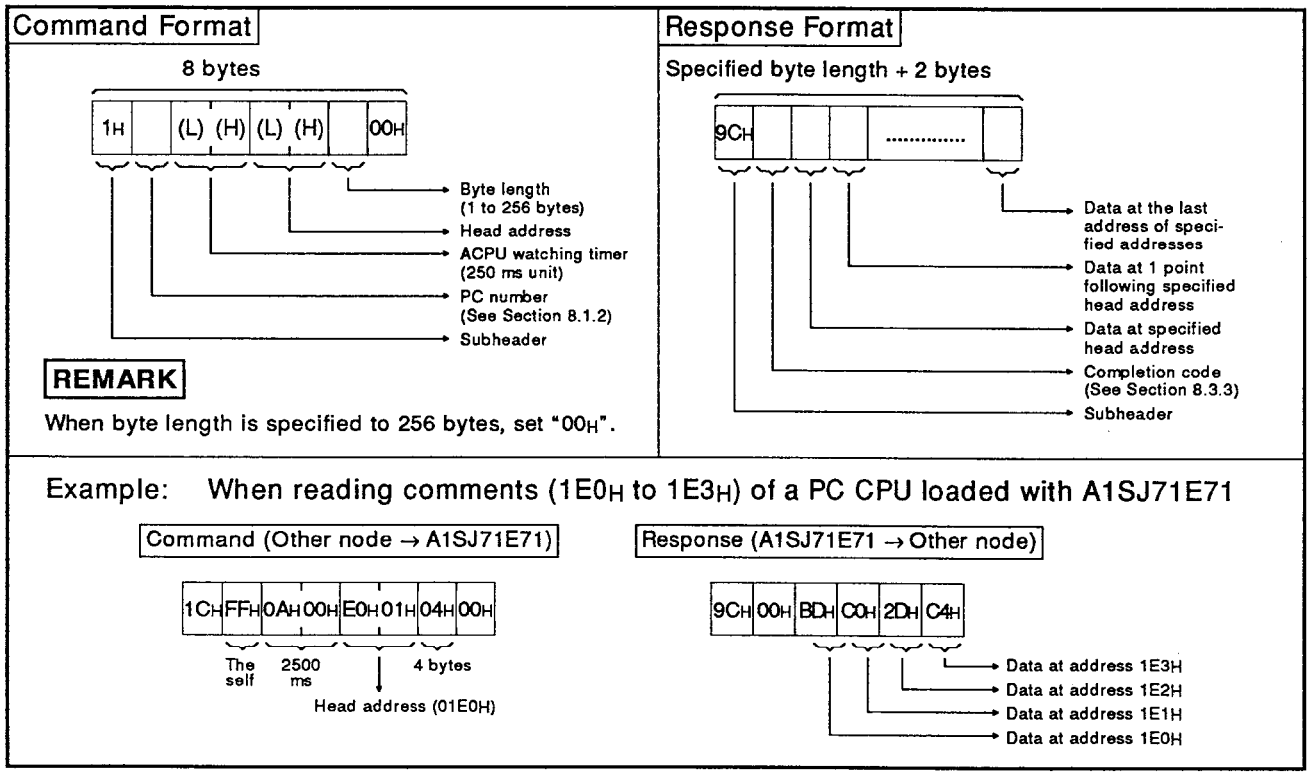
POINT

Comment data cannot be read or written by setting a specific device and device number.
Start reading or writing beginning with address 0_H.

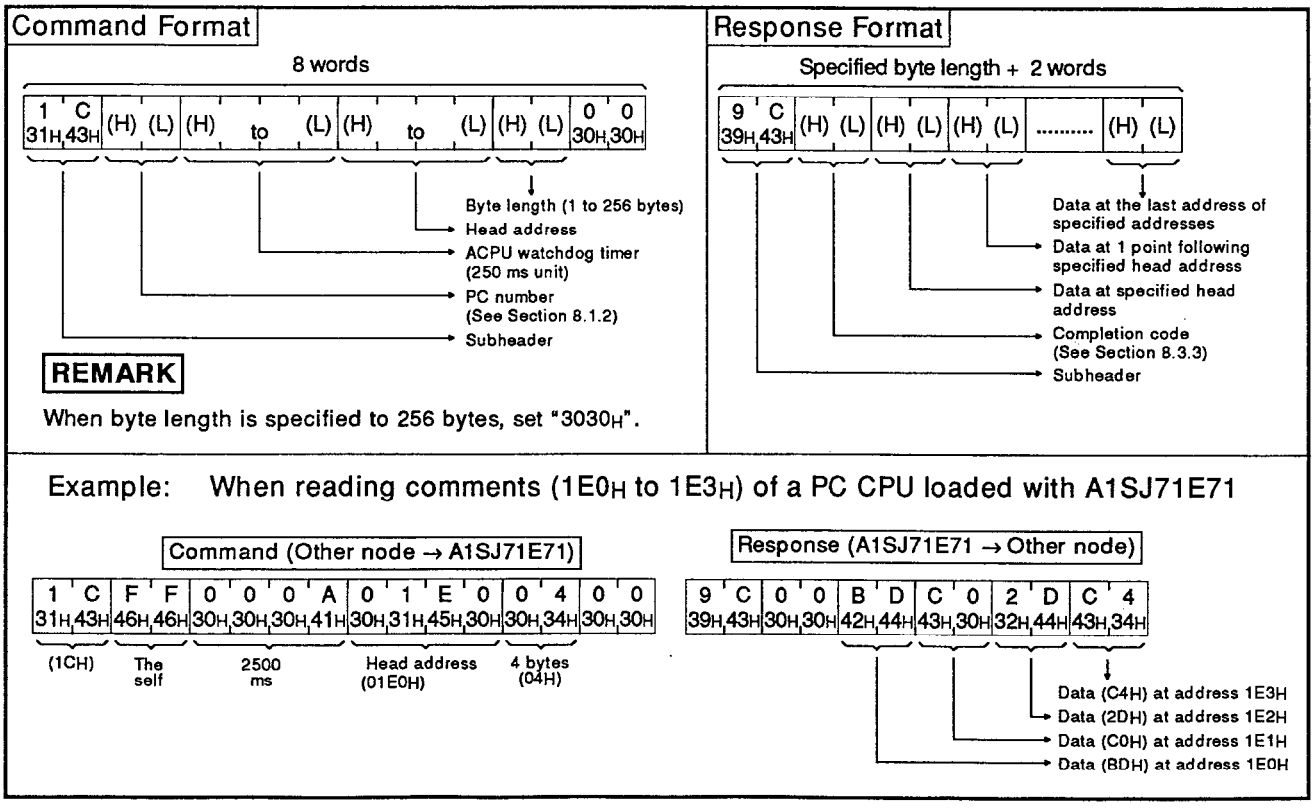
(2) Batch read

The command and the response formats are as follows when batch read of comment memory is done:

(a) Communications in binary code



(b) Communications in ASCII code

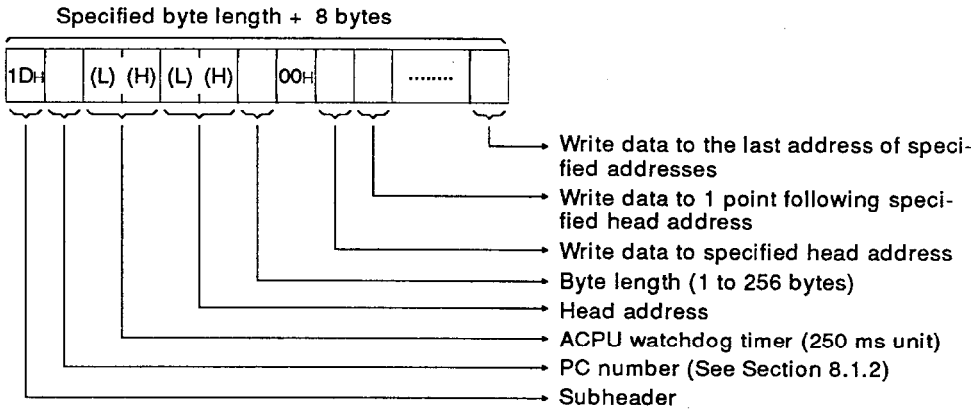


(3) Batch write

The command and response formats are as follows when the batch write of comment memory is done:

(a) Communications in binary code

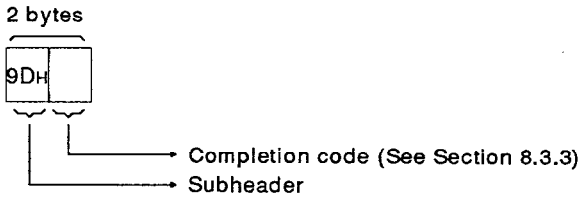
Command Format



REMARK

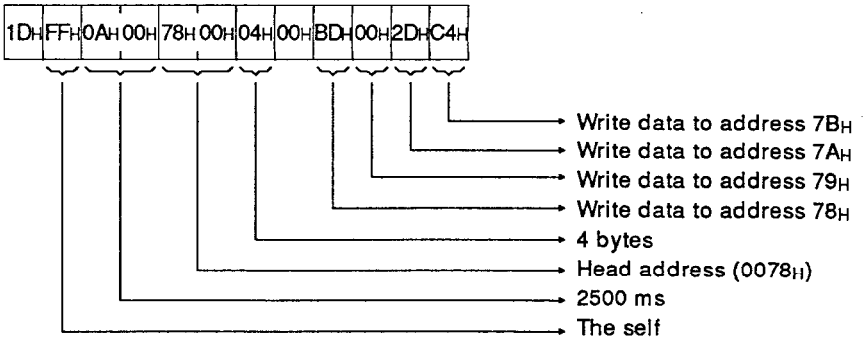
When byte length is specified to 256 bytes, set "00_H".

Response Format



Example: When writing data to the comment area (78_H to $7B_H$) of a PC CPU loaded with A1SJ71E71

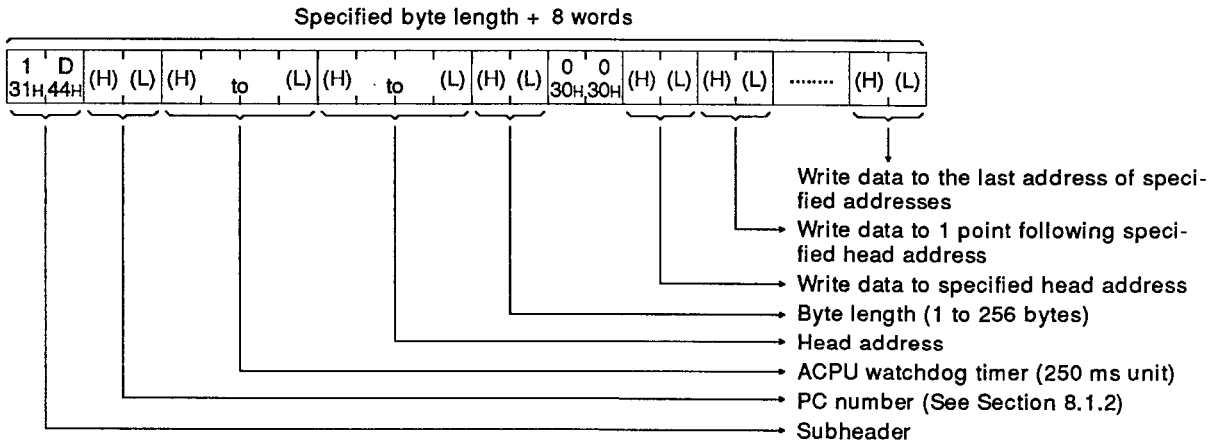
Command (Other node → A1SJ71E71)



Response (A1SJ71E71 → Other node)

(b) Communications in ASCII code

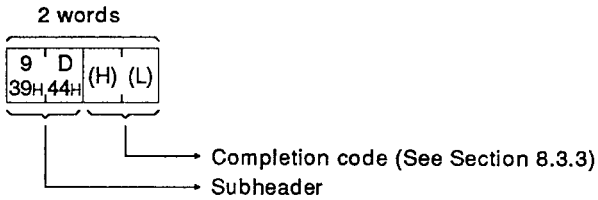
Command Format



REMARK

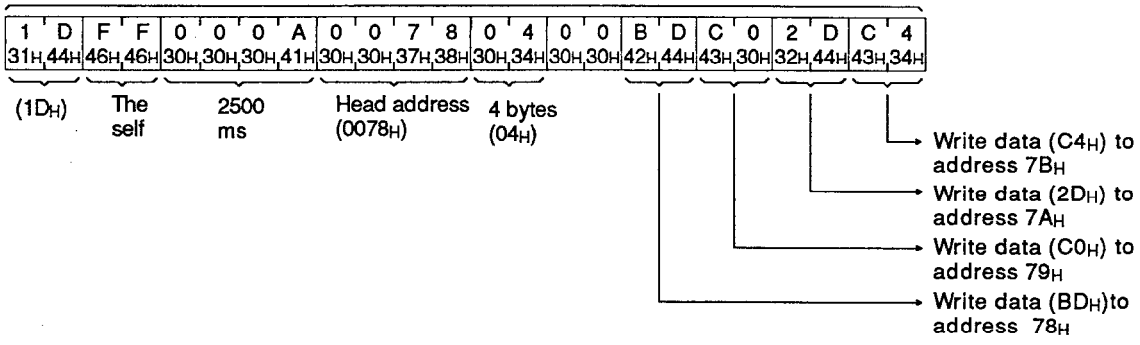
When byte length is specified to 256 bytes, set "3030H".

Response Format

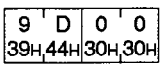


Example: When writing data to the comment area (78H to 7BH) of a PC CPU loaded with A1SJ71E71

Command (Other node → A1SJ71E71)



Response (A1SJ71E71 → Other node)



8.8.7 Read/write of extension comment

This section describes the specification contents and specification method for the control protocol used for reading/writing the extension comment data of a PC CPU, and gives an example of control protocol specification.

(1) Extension command and address

The command/response classification and comment data addresses when reading/writing comment data area as follows.

(a) Table 8.16 shows the functions for the read/write of AnACPU dedicated command comment data.

Table 8.16 Functions

Item	Command/ Response Classification	Processing	Number of Points Processed per Communication	State of PC CPU		
				During STOP	During RUN	
					SW22 ON	SW22 OFF
Batch read	39_H	Reads the extension comment data.	256 bytes	o	o	o
Batch write	$3A_H$	Writes the extension comment data.		o	o	x

Note : o.....Executable
x.....Not executable

(b) Extension comment memory address

The extension comment data storage area is managed by the relative address which geins with the head address of 00_H.

For example, when the comment capacity of parameter is 2k bytes, the range from 00_H to $7FF_H$ can be specified for the head address.

1) The extension comment memory can be set up to 64k bytes.

The address range of comment data is fixed according to the set capacity of parameter.

2) Specify an extension comment memory address in hexadecimal.

3) When the sum of a head address and the number of bytes is larger than extension commnet memory capacity, an error (completion code 57_H) occurs.

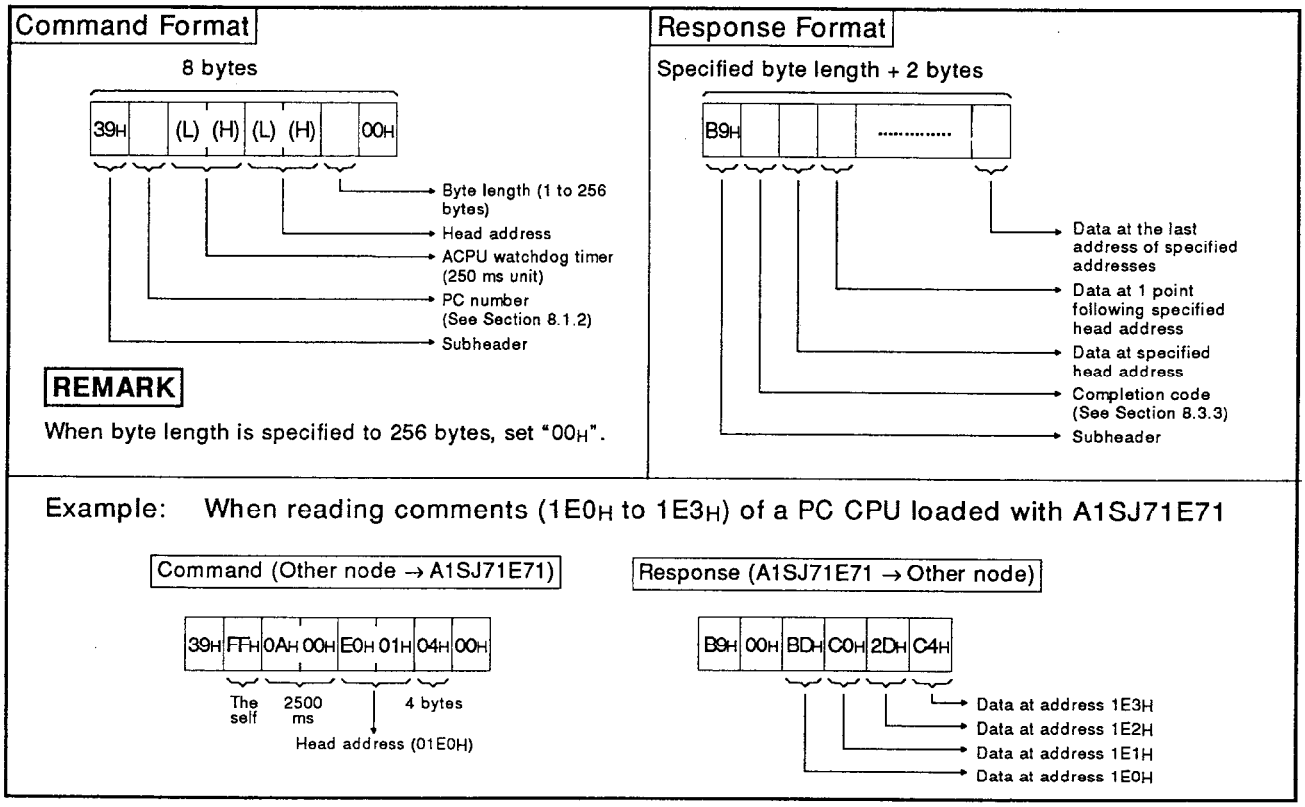
POINT

Comment data cannot be read or written by setting a specific device and device number.
Start reading or writing beginning with address 0_H.

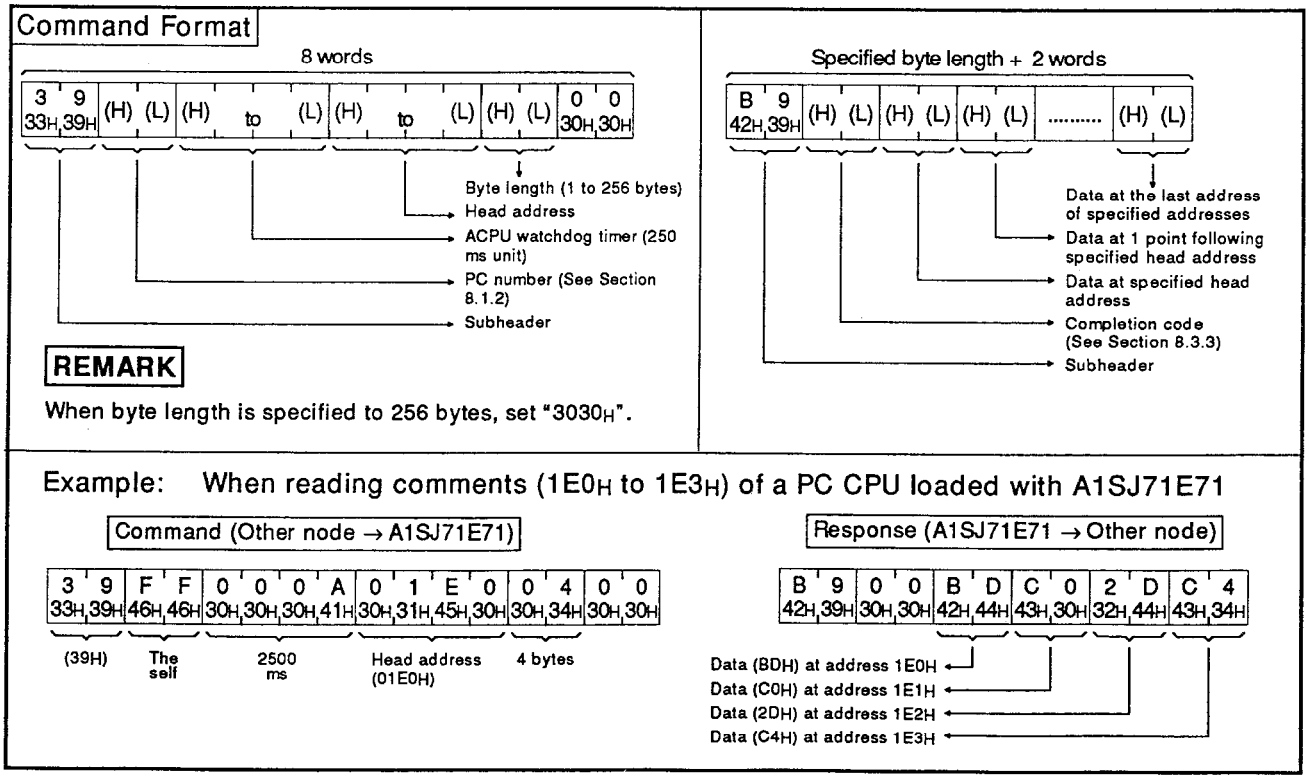
(2) Batch read

The command and the response formats are as follows when batch read of comment memory is done:

(a) Communications in binary code



(b) Communications in ASCII code



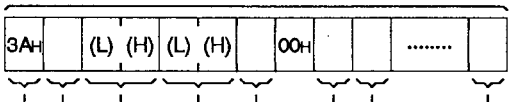
(3) Batch write

The command and response formats are as follows when the batch write of comment memory is done:

(a) Communications in binary code

Command format

Specified byte length + 8 bytes

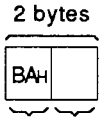


- Write data to the last address of specified addresses
- Write data to 1 point following specified head address
- Write data to specified head address
- Byte length (1 to 256 bytes)
- Head address
- ACPU watchdog timer (250 ms unit)
- PC number (See Section 8.1.2)
- Subheader

REMARK

When byte length is specified to 256 bytes, set "00_H".

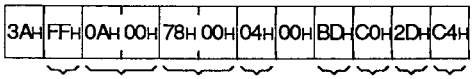
Response Format



- Completion code (See Section 8.3.3)
- Subheader

Example: When writing data to the comment area (78_H to $7B_H$) of a PC CPU loaded with A1SJ71E71

Command (Other node → A1SJ71E71)

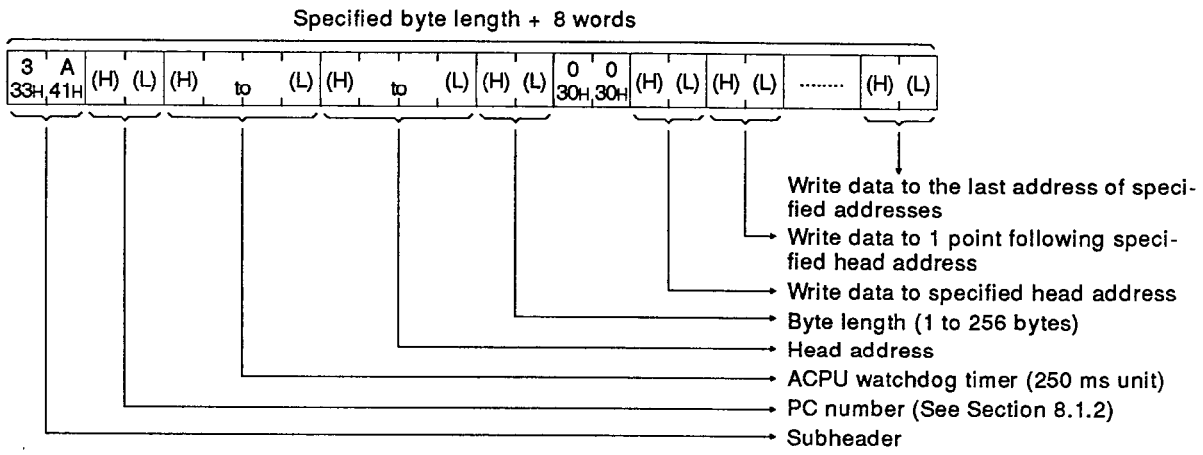


- Write data ($C4_H$) to address $7B_H$
- Write data ($2D_H$) to address $7A_H$
- Write data ($C0_H$) to address 79_H
- Write data (BD_H) to address 78_H

Response (A1SJ71E71 → Other node)

(b) Communications in ASCII code

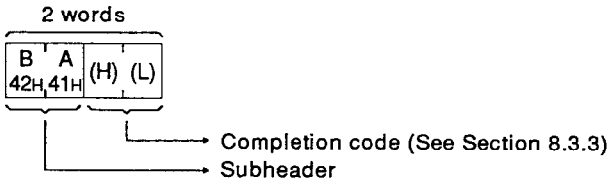
Command Format



REMARK

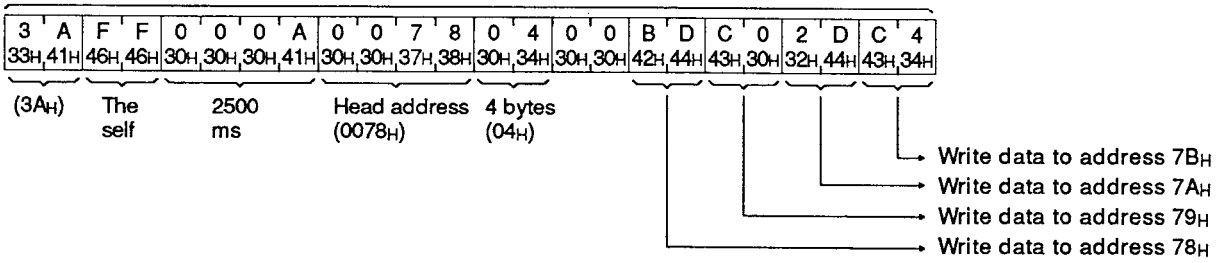
When byte length is specified to 256 bytes, set "3030_H".

Response Format

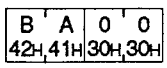


Example: When writing data to the comment area (78_H to $7B_H$) of a PC CPU loaded with A1SJ71E71

Command (Other node → A1SJ71E71)



Response (A1SJ71E71 → Other node)



8.9 Command and Response Formats for the Loopback Test

The loopback test is used to check normal communications between a communicating station and A1SJ71E71.
Data transmitted from a station is sent back to the same station as a response from the A1SJ71E71.

(1) Table 8.17 shows the function of the loopback test.

Table 8.17 Functions

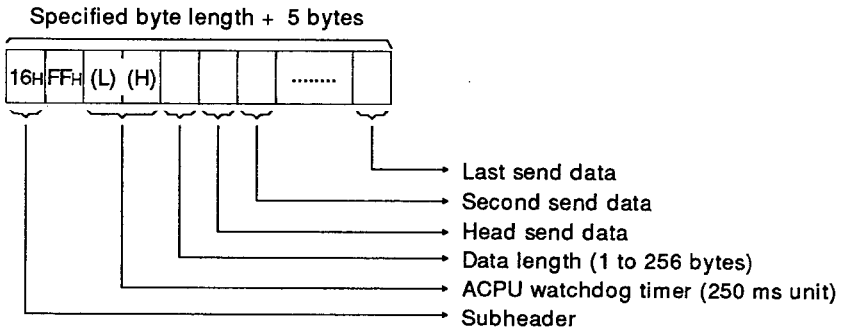
Item	Command/ Response Classification	Processing	Number of Points Processed per Communication	State of PC CPU		
				During STOP	During RUN	
					SW22 ON	SW22 OFF
Loopback test	16_H	Echoes unchanged characters back to the computer.	256 bytes	o	o	o

Note : o .....Executable

The command and response formats are as follows when the loopback test is executed:

(a) Communications in binary code

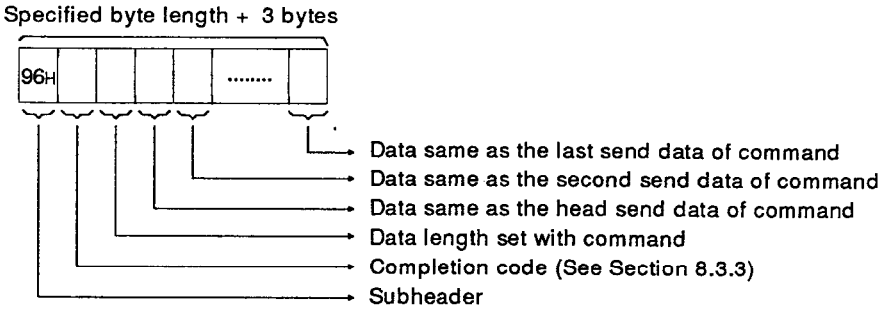
Command Format



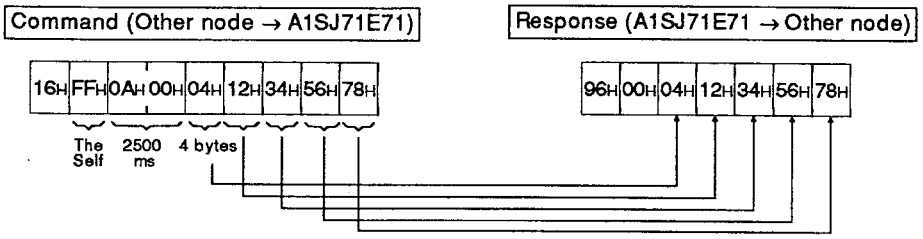
REMARK

When data length is specified to 256 bytes, set "00H".

Response Format

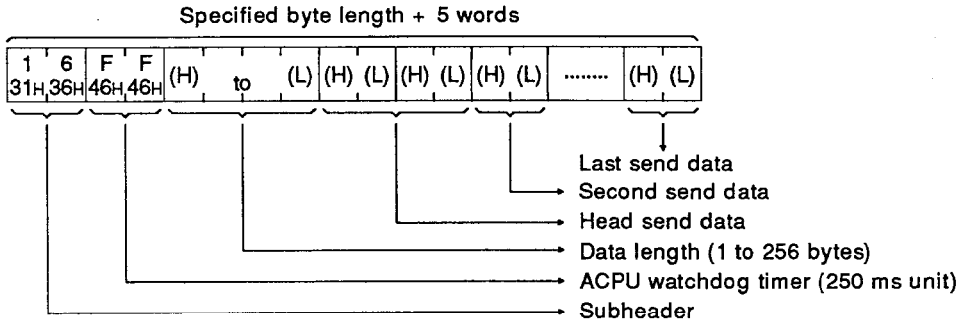


Example: When doing a loopback test by using data (12H, 34H, 56H, 78H)



(b) Communications in ASCII code

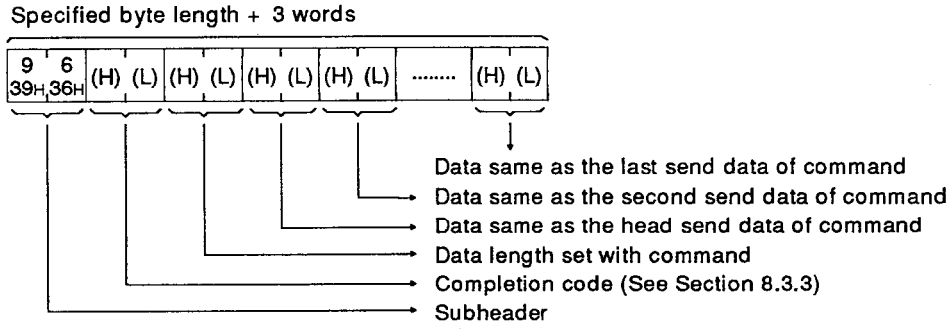
Command Format



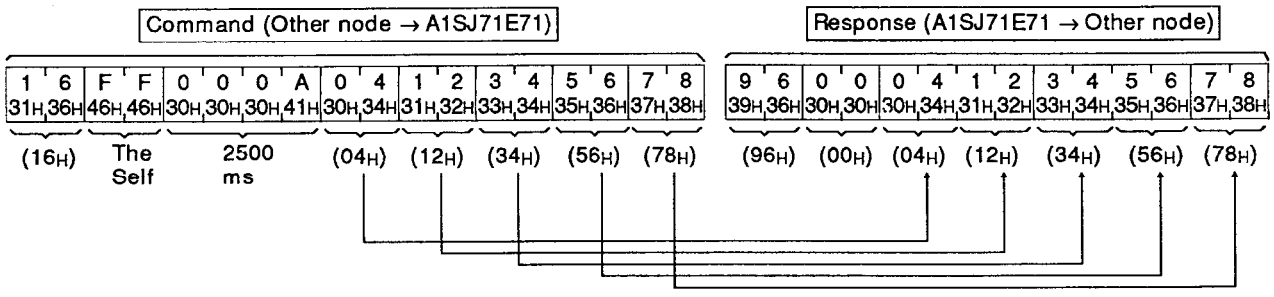
REMARK

When data length is specified to 256 bytes, set "3030H".

Response Format



Example: When doing a loopback test by using data (12H, 23H, 45H, 67H)



9. TROUBLESHOOTING

9.1 Error Code List

Errors occurring during communications between the A1SJ71E71 and a communicating node are classified into the following four types:

(1) Initialization error codes

Initialization parameter setting error code is stored.

(2) Open error codeLine opening parameter setting error code is stored.

(3) Fixed buffer send error code

The code of communication error occurring in sending data using fixed buffer is stored.

(4) Error log area storing error code

The code of communication error occurring in other than sending data using fixed buffer is stored.

9.1.1 Initialization error code list

Initialization error code is stored in the initialization error storing area (buffer address 80_H).

Error Code	Description	Corrective Action
201_H	The IP address setting of the A1SJ71E71 is either "0" or "$FFFFFFFF_H$".	Correct the IP address of the A1SJ71E71.
8004_H	System error	Initialize the A1SJ71E71.
8005_H	Initialization error due to an error in initialization parameter.	Check and correct the initialization parameter setting.

9.1.2 Communication line opening error code list

Communication line opening error code is stored in the communication line opening error code storing area (buffer addresses 93, 103, ...) of each connection.

Error Code	Description	Corrective Action
71_H	Data of the set length has not been received within the duration set by a response watchdog timer. Actual data is shorter than the set data length.	Correct the set data length.
101_H	The port number for the A1SJ71E71 is set within a range of 0_H to $00FF_H$.	Correct the port number. A port number should be set within a range of 0100_H to $FFFF_H$.
102_H	The port number for the communicating node is set within a range of 0_H to $00FF_H$.	Correct the port number. A port number should be set within a range of 0100_H to $FFFF_H$.
103_H	In communications using the TCP/IP protocol, the same combination of the A1SJ71E71 port number and communicating node port number as the connection that has been opened is set.	Correct the communication address setting area of each connection.
104_H	In communications using the UDP/IP protocol, the same A1SJ71E71 port number is set for more than one connection.	Correct the communication address setting area of each connection.
105_H	The A1SJ71E71 has not been initialized.	Initialize the A1SJ71E71.
106_H	The IP address setting of the communicating node is either "0" or "$FFFFFFFF_H$"	Correct the IP address of the communicating node.
7004_H	In the communication line opening processing of the TCP connection, connection has not been established.	Correct the value in the application setting area of each connection. Check the connection processing of the communicating node.
8001_H	20 or more values other than defaults ($FFFFFFFFFFFF_H$) have been registered as other node Ethernet addresses (buffer memory addresses 28 to 30, ..., 77 to 79).	If using the ARP function, do not change the default settings.
8004_H	System error	Initialize the A1SJ71E71.
9002_H	Fixed buffer sending has been executed with close processing still in progress after the open request signal (Y8 to YF) has been switched OFF (but before the open completed signal (X10 to X17) has gone OFF).	Execute fixed buffer sending when both the open request signal (Y8 to YF) and the open completed signal (X10 to X17) for the relevant connection are ON.
9059_H	In communications using the TCP/IP protocol, the TCP ULP time-out error occurred. (ACK is not returned from the communication node in communications using the TCP protocol.)	Check the Ethernet cable if it is connected securely. Check the operating status of the communicating node. Correct the TCP UPL time setting of the initial parameters.
$A009_H$	The Ethernet address that corresponds to the designated IP address does not exist.	Check the IP address. Set the Ethernet address if the ARP function is not used (see Section 5.3).
$B000_H$	Send error	Check the cable, A1SJ71E71 hardware, etc.

9.1.3 Fixed buffer send error code list

Fixed buffer send error code is stored in the fixed buffer send error code storing area (buffer addresses 94, 104, ...) of each connection.

Error Code	Description	Corrective Action
01H	Communication data length exceeds the maximum allowable length: Binary code .....1018 words ASCII code .....509 words	Correct the data length of the data to be sent.
62H	A code other than "00H" (normal completion) is returned from the communication node as the response end code in communications using fixed buffer.	Read the response end code returned from the communicating node stored in buffer memory addresses 95, 105, ...
70H	A response is not returned within a duration set by the response watchdog timer.	Check the operation of the communicating node. If the setting of the response watch dog timer is too small, change it to a larger one.
71H	Data of the set data length has not been received within the time set for the response watchdog timer. The actual data length is shorter than the value set for data length. The remainder of a message divided at the TCP/UDP level has not been received within the time set for the response watchdog timer.	Change the data length of the communication data. In TCP communication, if there is a possibility that packets are colliding in the line, change the set data for initial processing. In UDP communication, retry with the program at the sending side.
80H	Communication line opening processing for the connection in question is not completed.	Execute opening processing.
81H	An Ethernet address that does not exist is designated. (Only when the UDP I/P is used.)	Check the Ethernet address of the communicating node. If the APR function is used, do not set the Ethernet address using the IP address.
105H	The A1SJ71E71 has not been initialized.	Initialize the A1SJ71E71. Execute opening processing.
8004H	System error	Initialize the A1SJ71E71.
9002H	Fixed buffer sending has been executed with close processing still in progress after the open request signal (Y8 to F) has been switched OFF (but before the open completed signal (X10 to X17) has gone OFF).	Execute fixed buffer sending when both the open request signal (Y8 to F) and the open completed signal (X10 to X17) for the relevant connection are ON.
9005H	Internal resource is insufficient for the TCP request. Send buffer size is insufficient.	Send the same data again. Next data might be sent without confirming the response from the communicating node. Send the next data only after the response has been received.
9008H	Internal resource is insufficient for the UDP request. Send buffer size is insufficient.	Send the same data again. Next data might be sent without confirming the response from the communicating node. Send the next data only after the response has been received.
9059H	In communications using the TCP/IP protocol, the TCP ULP time-out error occurred. (ACK is not returned from the communication node in communications using the TCP protocol.)	Check the Ethernet cable if it is connected securely. Check the operating status of the communicating node. Correct the TCP UPL time setting of the initial parameters.
B000H	Send error	Check the cable, A1SJ71E71 hardware, etc.

9.1.4 List of error codes stored in error log area

Minor errors (physical layer to transport layer, see Section 9.2) occurred in data receiving using fixed buffer, communications using random access buffer, and general data communications are stored in the error log area. (buffer addresses: 168 to 178)
When executing read/write of data in the PC CPU, apart from the error codes in the table below, the completion codes described in Section 8.3.3 and the error codes described in Section 8.3.4 are also stored.

Error Code	Description	Corrective Action
71H	Data of the set data length has not been received within the time set for the response watchdog timer. The actual data length is shorter than the value set for data length. The remainder of a message divided at the TCP/UDP level has not been received within the time set for the response watchdog timer.	Change the data length of the communication data. In TCP communication, if there is a possibility that packets are colliding in the line, change the set data for initial processing. In UDP communication, retry with the program at the sending side.
81H	An Ethernet address that does not exist is designated. (Only when the UDP I/P is used.)	Check the Ethernet address of the communicating node. If the ARP function is used, do not set the Ethernet address using the IP address.
105H	The A1SJ71E71 has not been initialized.	Initialize the A1SJ71E71.
8004H	System error	Initialize the A1SJ71E71.
9001H	Communication line opening processing for the connection in question is not completed	Execute opening processing.
9002H	Fixed buffer sending has been executed with close processing still in progress after the open request signal (Y8 to F) has been switched OFF (but before the open completed signal (X10 to X17) has gone OFF).	Execute fixed buffer sending when both the open request signal (Y8 to F) and the open completed signal (X10 to X17) for the relevant connection are ON.
9005H	Internal resource is insufficient for the TCP request. Send buffer size is insufficient.	Send the same data again. Next data might be sent without confirming the response from the communicating node. Send the next data only after the response has been received.
9006H	Check sum error in received data in communications using the TCP protocol	Check the check sum calculation in the communicating node.
9008H	Internal resource is insufficient for the UDP request. Send buffer size is insufficient.	Send the same data again. Next data might be sent without confirming the response from the communicating node. Send the next data only after the response has been received
9009H	Check sum error in received data in communications using the UDP protocol.	Check the check sum calculation in the communicating node.
9059H	In communications using the TCP/IP protocol, the TCP ULP time-out error occurred. (ACK is not returned from the communication node in communications using the TCP protocol.)	Check the Ethernet cable if it is connected securely. Check the operating status of the communicating node. Correct the TCP ULP time setting of the initial parameters.
A001H	An illegal IP address (network number) is used. (ICMP error packet is received while the IP address of the IP packet sent to the communicating node is incorrect.)	Check the IP address of the communicating node set in the A1SJ71E71. Check the IP address of the communicating node.
A002H	An illegal IP address (host number) is used. (ICMP error packet is received while the IP address of the IP packet sent to the communicating node is identical.)	Check the IP address of the set in the A1SJ71E71. Check the IP address of the communicating node.
A004H	An illegal port number is used. (ICMP error packet is received while the port number of the IP packet sent to the communicating node is not registered in it.)	Check and correct the port number of the communicating node.

Error Code	Description	Corrective Action
A006H	ICMP error packet is received when an assemble time-out error occurred in the communicating node.	Check the Ethernet cable if it is connected securely and the termination processing at the transceiver. If the IP assemble timer setting is too small, change it to a larger value
A007H	IP assemble time-out error. (Time-out occurred with the remaining portion of divided data not received.	Check the Ethernet cable if it is connected securely and the termination processing at the transceiver. Check if the IP assemble timer setting in the initial parameters is too small. Check the operation of the communicating node.
A00BH	ICMP error packet that cannot be analyzed by the system is received.	Check the reason the communicating node has sent that ICMP packet.
A00CH	ICMP error packet not supported by the system is received.	The system supports only the echo, time-stamp, and the response to information request.
A00DH	Check sum error with the header of the IP packet received.	Check the check sum calculation at the communicating node.
A00EH	Sending data is impossible due to full internal buffer like IP header buffer.	Send the same data again.
B000H	Send error	Check the cable, A1SJ71E71 hardware, etc.

REMARK

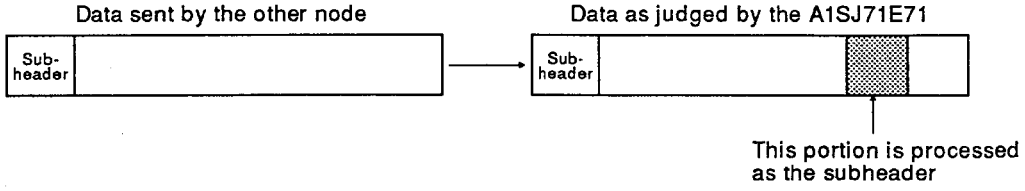
Communicated data is divided into portions due to the restrictions on buffer size of the host station and/or communicating station. The data received in portions is reassembled by the A1SJ71E71 and communicated using the fixed buffer or random access buffer. Data reassembling is executed based on the data length of the communicated data.
An error occurs if the set data length and actual data length differ.

(a) If the actual data length is shorter than the set data length:

Since data reception is delayed until data of the set length is received, a response watchdog timeout error occurs and the connection is automatically closed.

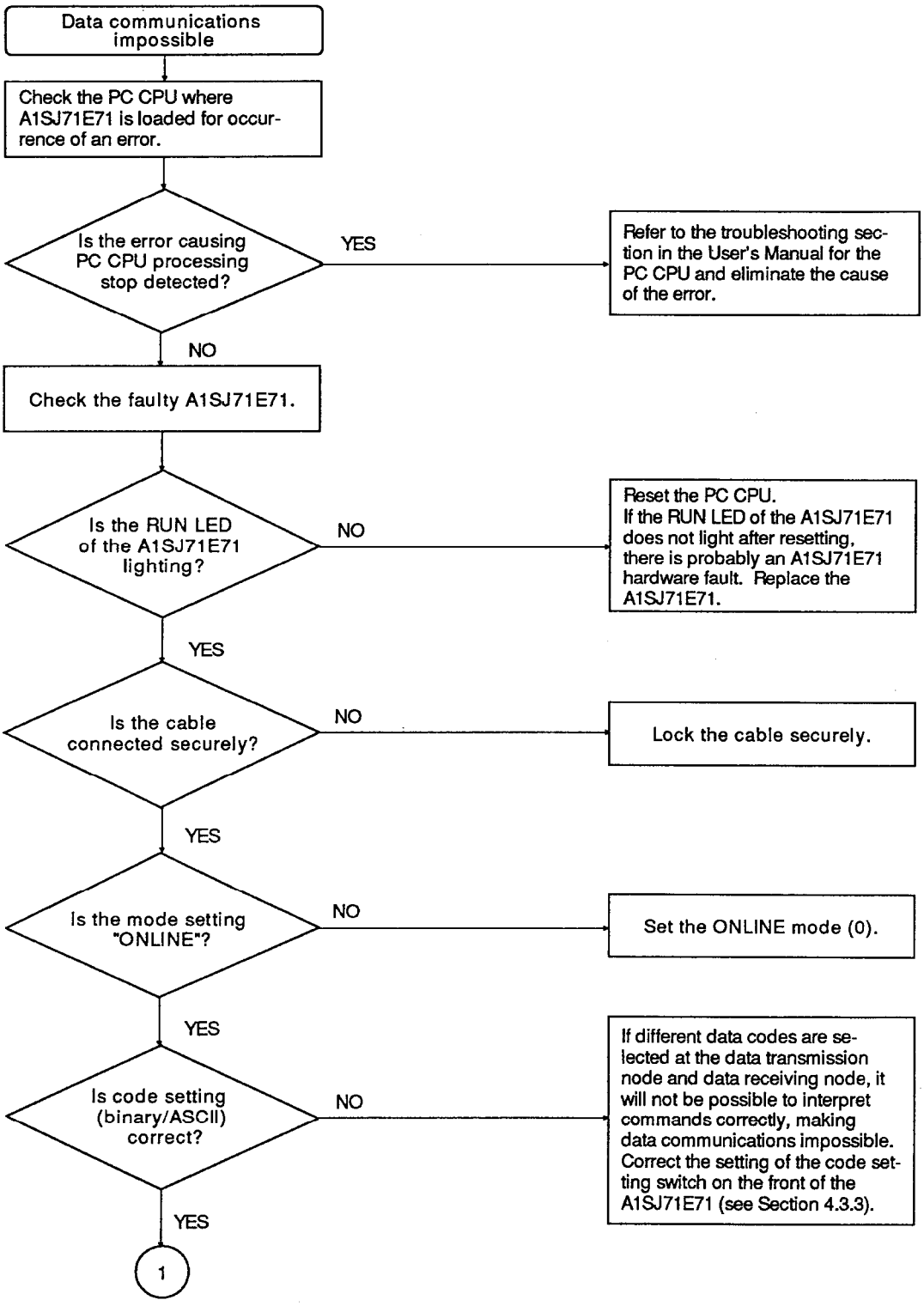
(b) If the actual data length is larger than the set data length:

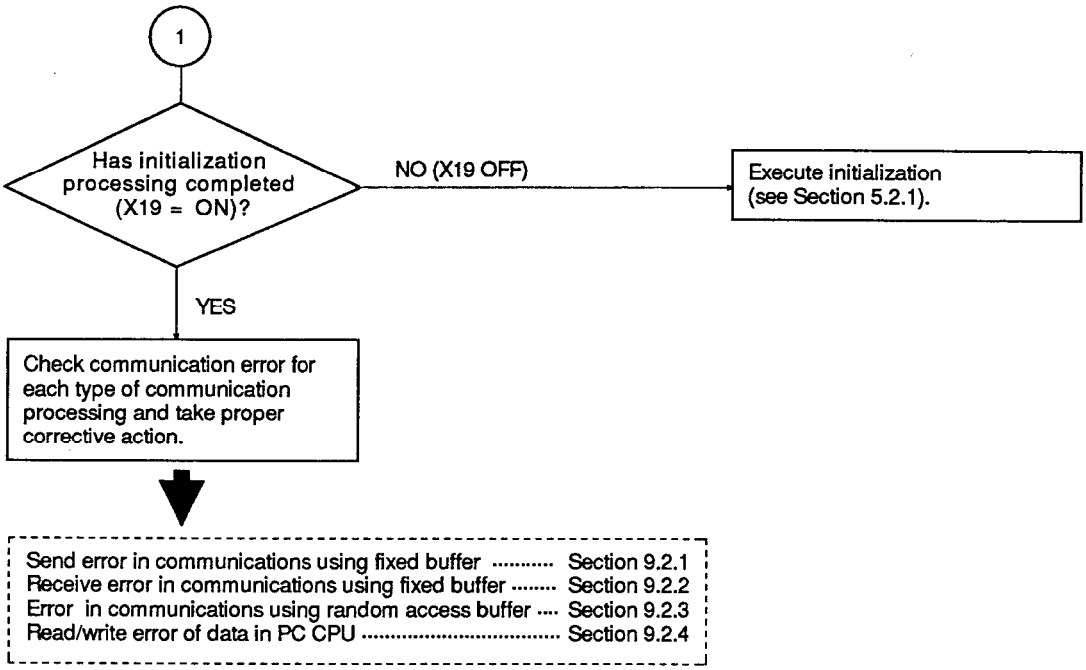
The data of the set data length is processed as the first data and an attempt is made to process the remaining data as the second data. Consequently, the second data does not have a subheader and the command/response type undefined error occurs.



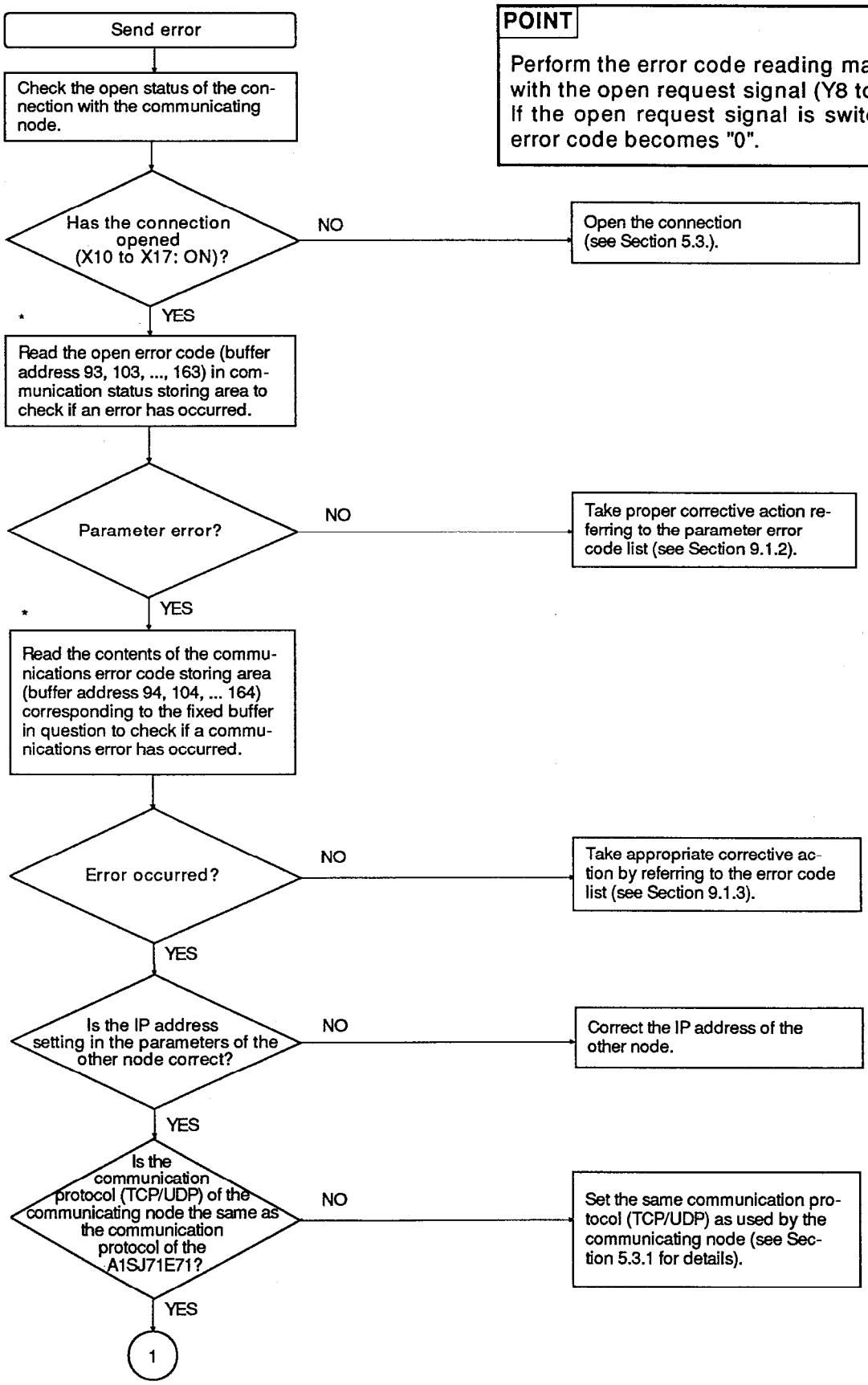
As the response in this case, the code processed as the subheader is returned with the most significant bit set to "1".
For example, if the subheader part of the command was 65H, the subheader of the response is E5H.

9.2 Troubleshooting Flowchart



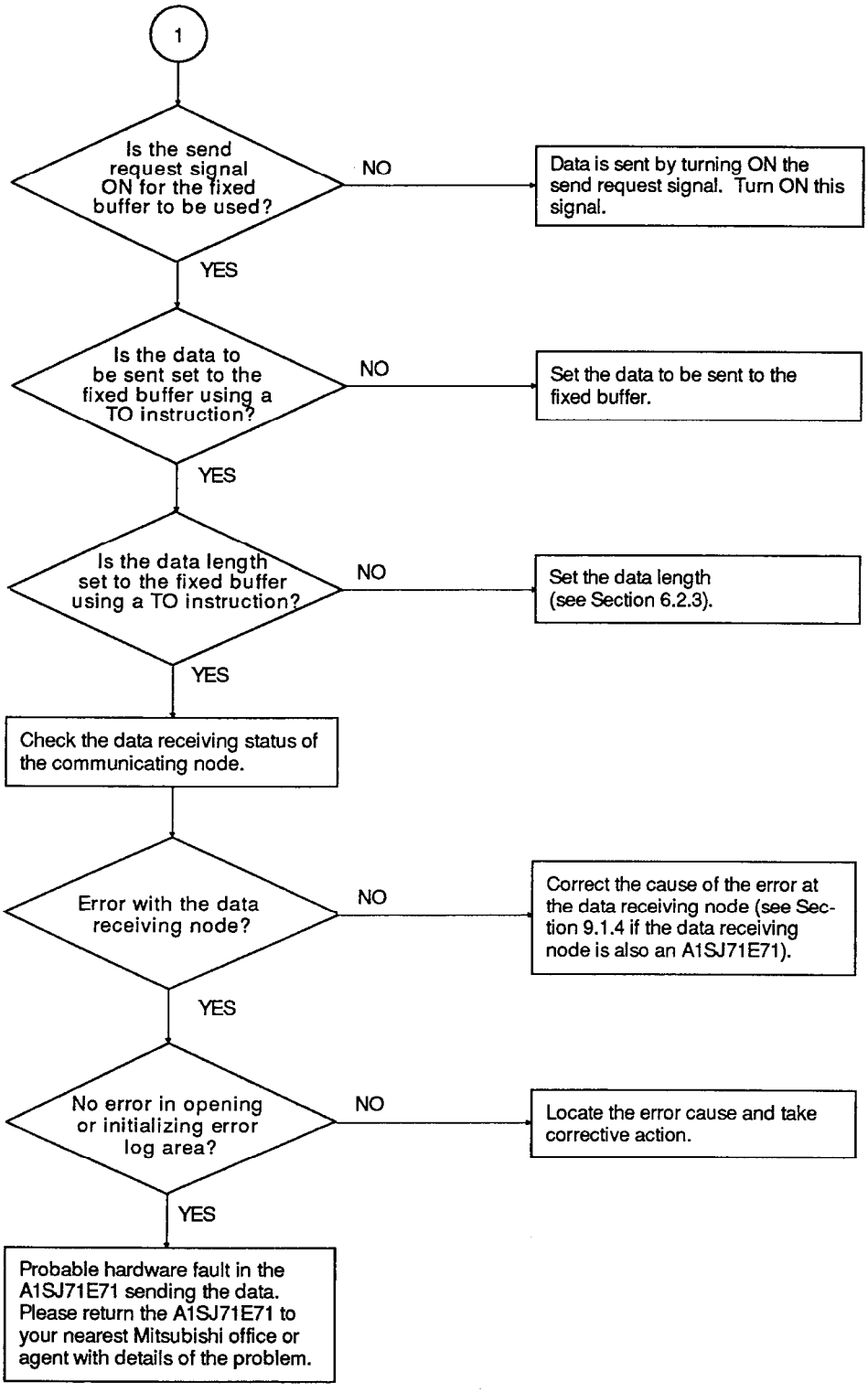

1
Has initialization processing completed (X19 = ON)?
NO (X19 OFF)
Execute initialization (see Section 5.2.1).
YES
Check communication error for each type of communication processing and take proper corrective action.
Send error in communications using fixed buffer ..... Section 9.2.1
Receive error in communications using fixed buffer ..... Section 9.2.2
Error in communications using random access buffer .... Section 9.2.3
Read/write error of data in PC CPU ..... Section 9.2.4

9.2.1 Send error in communications using fixed buffer

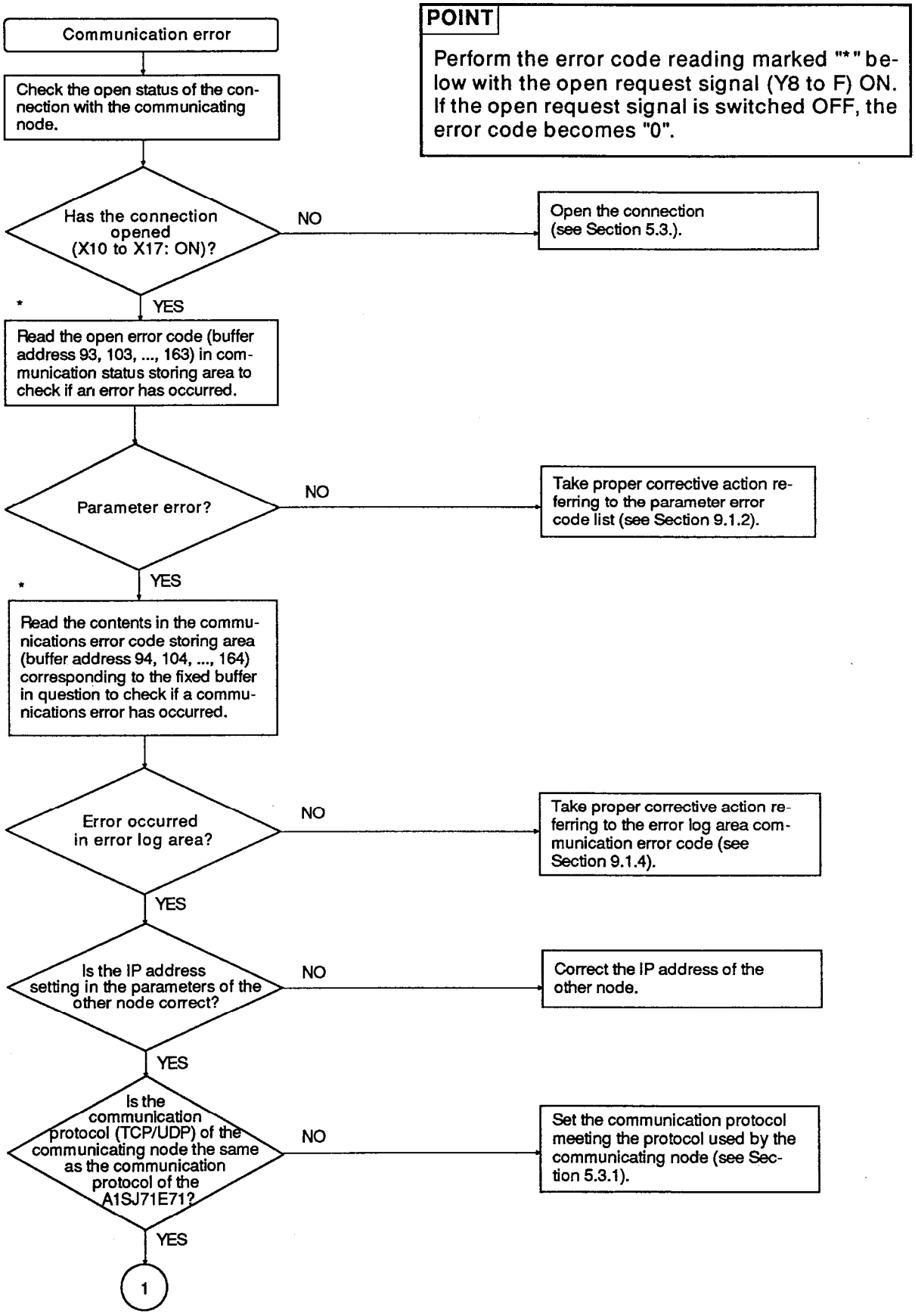


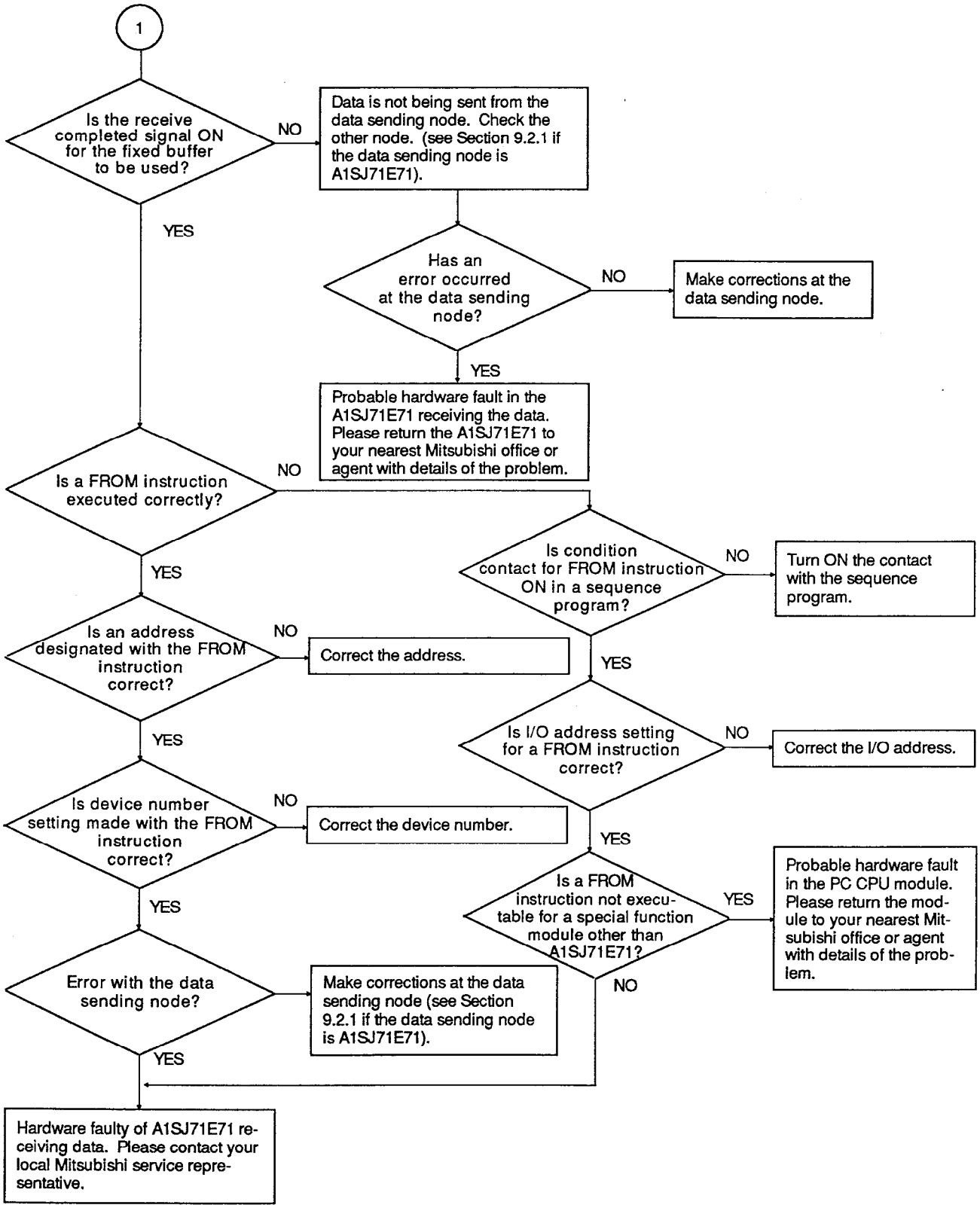
POINT

Perform the error code reading marked "*" below with the open request signal (Y8 to F) ON.
If the open request signal is switched OFF, the error code becomes "0".

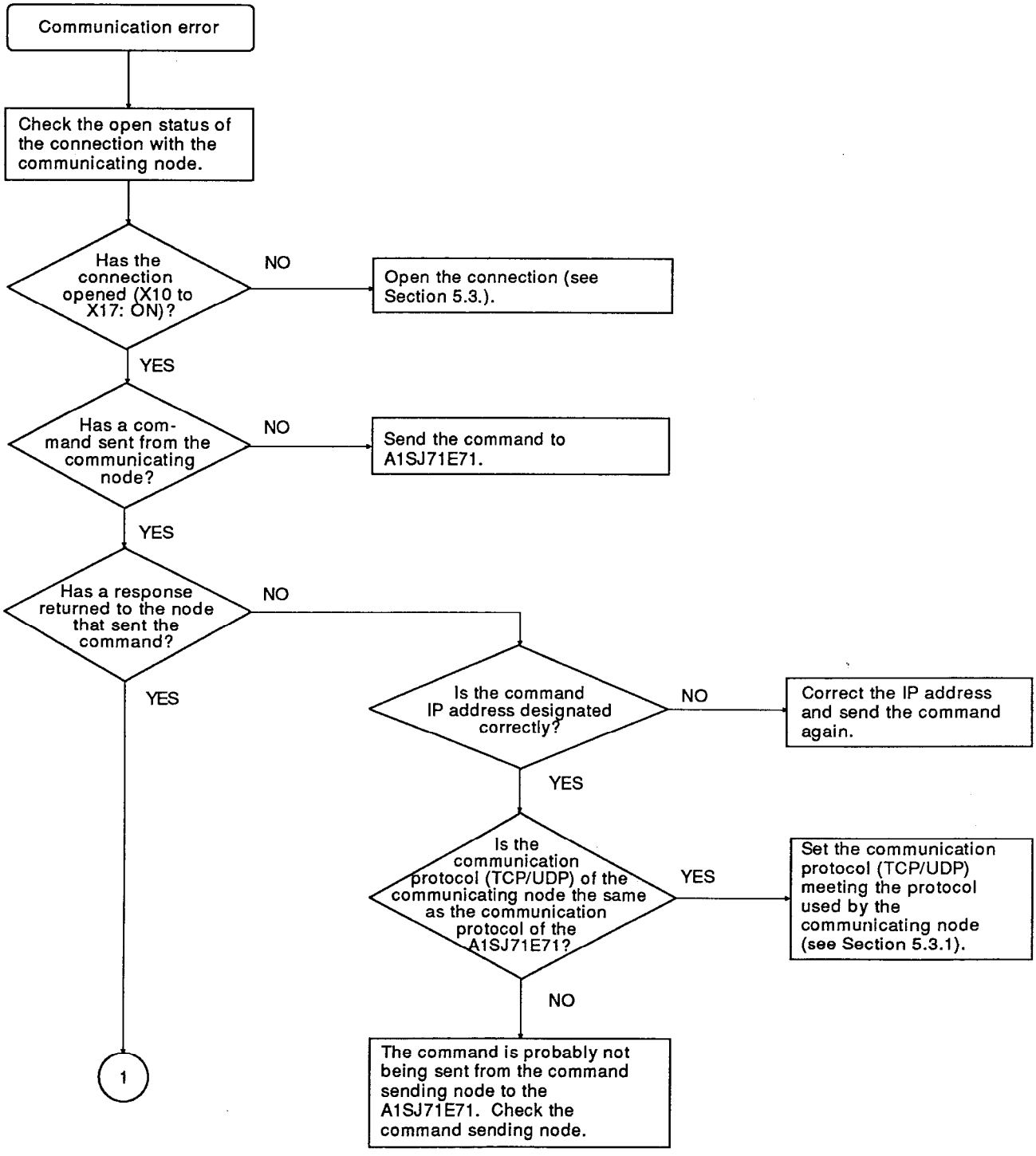

1
Is the send request signal ON for the fixed buffer to be used?
NO
Data is sent by turning ON the send request signal. Turn ON this signal.
YES
Is the data to be sent set to the fixed buffer using a TO instruction?
NO
Set the data to be sent to the fixed buffer.
YES
Is the data length set to the fixed buffer using a TO instruction?
NO
Set the data length (see Section 6.2.3).
YES
Check the data receiving status of the communicating node.
Error with the data receiving node?
NO
Correct the cause of the error at the data receiving node (see Section 9.1.4 if the data receiving node is also an A1SJ71E71).
YES
No error in opening or initializing error log area?
NO
Locate the error cause and take corrective action.
YES
Probable hardware fault in the A1SJ71E71 sending the data. Please return the A1SJ71E71 to your nearest Mitsubishi office or agent with details of the problem.

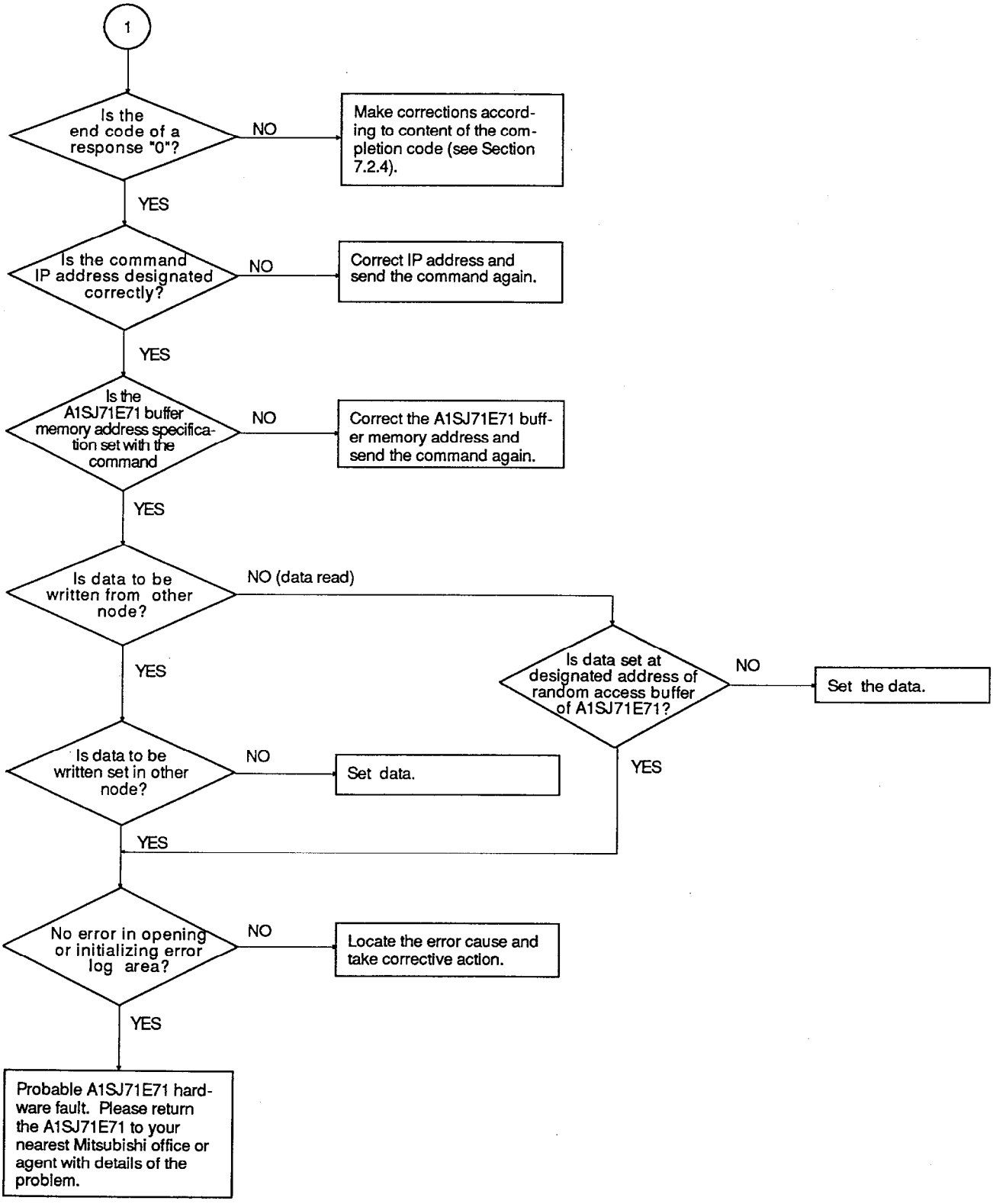
9.2.2 Receive error in communications using fixed buffer



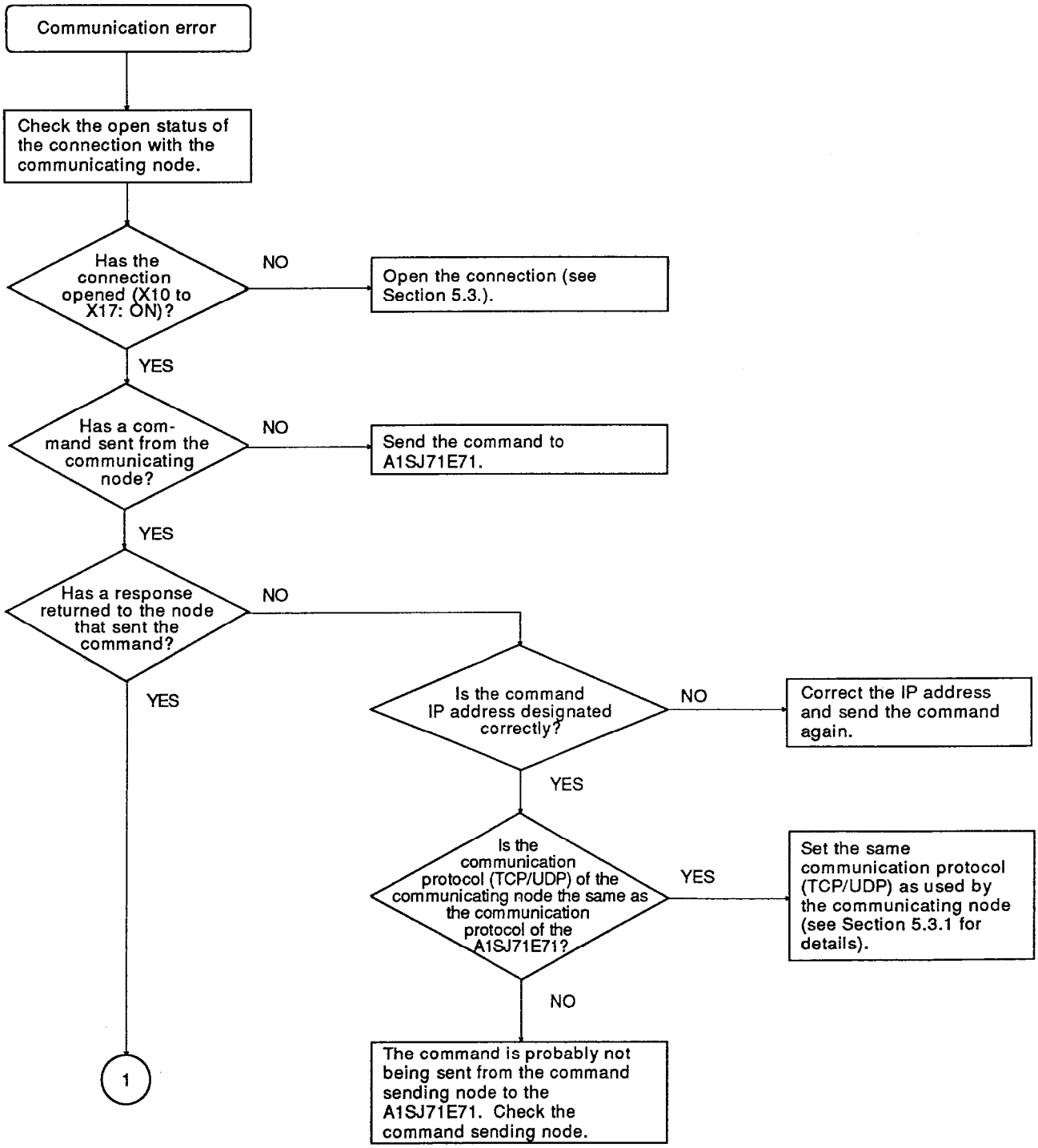

1
Is the receive completed signal ON for the fixed buffer to be used?
NO
Data is not being sent from the data sending node. Check the other node. (see Section 9.2.1 if the data sending node is A1SJ71E71).
YES
Has an error occurred at the data sending node?
NO
Make corrections at the data sending node.
YES
Probable hardware fault in the A1SJ71E71 receiving the data. Please return the A1SJ71E71 to your nearest Mitsubishi office or agent with details of the problem.
Is a FROM instruction executed correctly?
NO
YES
Is condition contact for FROM instruction ON in a sequence program?
NO
Turn ON the contact with the sequence program.
YES
Is an address designated with the FROM instruction correct?
NO
Correct the address.
YES
Is I/O address setting for a FROM instruction correct?
NO
Correct the I/O address.
YES
Is device number setting made with the FROM instruction correct?
NO
Correct the device number.
YES
Is a FROM instruction not executable for a special function module other than A1SJ71E71?
YES
Probable hardware fault in the PC CPU module. Please return the module to your nearest Mitsubishi office or agent with details of the problem.
NO
Error with the data sending node?
Make corrections at the data sending node (see Section 9.2.1 if the data sending node is A1SJ71E71).
YES
Hardware faulty of A1SJ71E71 receiving data. Please contact your local Mitsubishi service representative.

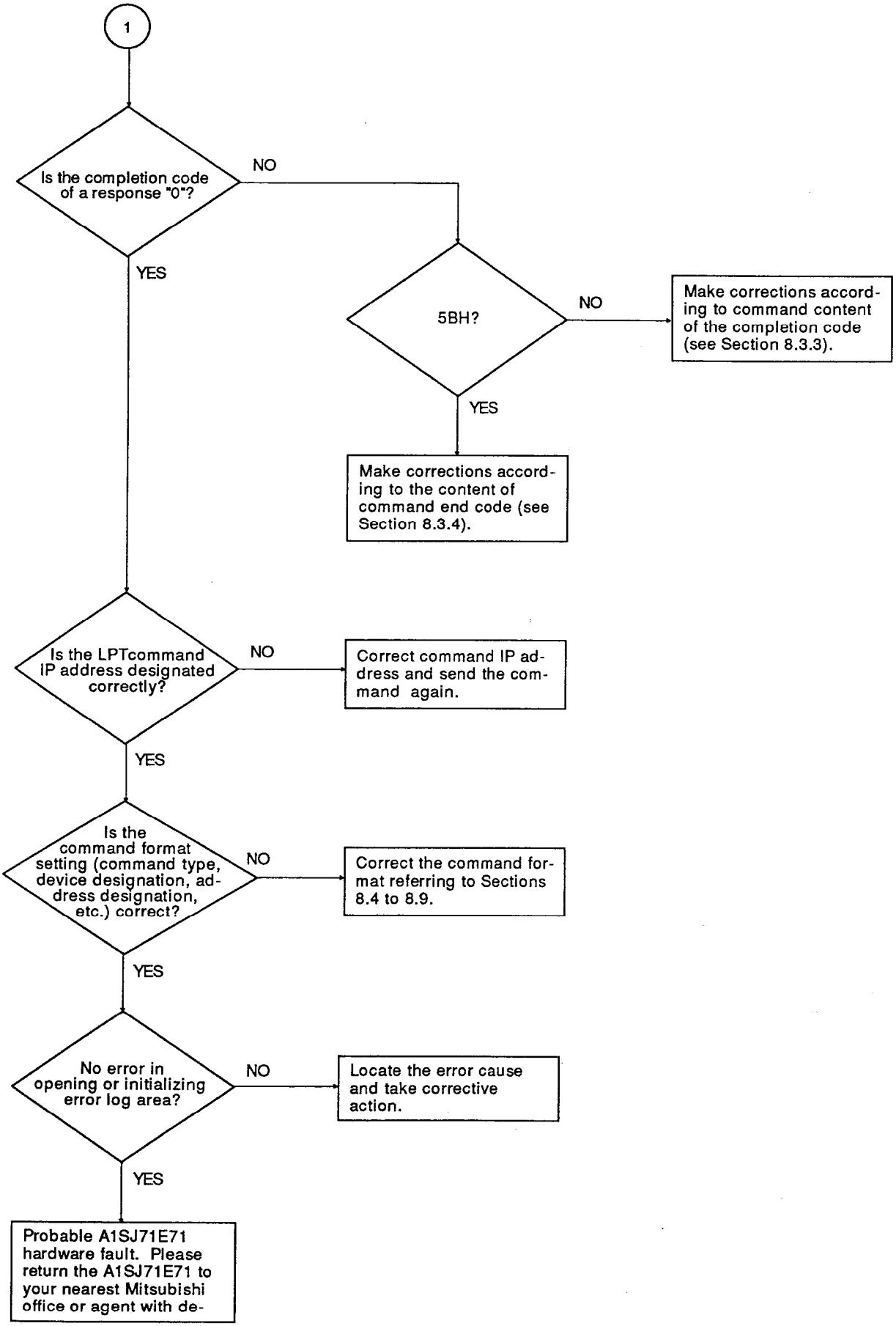
9.2.3 Error in communications using random access buffer




1
Is the end code of a response "0"?
NO
Make corrections according to content of the completion code (see Section 7.2.4).
YES
Is the command IP address designated correctly?
NO
Correct IP address and send the command again.
YES
Is the A1SJ71E71 buffer memory address specification set with the command
NO
Correct the A1SJ71E71 buffer memory address and send the command again.
YES
Is data to be written from other node?
NO (data read)
YES
Is data set at designated address of random access buffer of A1SJ71E71?
NO
Set the data.
YES
Is data to be written set in other node?
NO
Set data.
YES
No error in opening or initializing error log area?
NO
Locate the error cause and take corrective action.
YES
Probable A1SJ71E71 hardware fault. Please return the A1SJ71E71 to your nearest Mitsubishi office or agent with details of the problem.

9.2.4 Read/write error of data in PC CPU




1
Is the completion code of a response "0"?
NO
YES
5BH?
NO
Make corrections according to command content of the completion code (see Section 8.3.3).
YES
Make corrections according to the content of command end code (see Section 8.3.4).
Is the LPTcommand IP address designated correctly?
NO
Correct command IP address and send the command again.
YES
Is the command format setting (command type, device designation, address designation, etc.) correct?
NO
Correct the command format referring to Sections 8.4 to 8.9.
YES
No error in opening or initializing error log area?
NO
Locate the error cause and take corrective action.
YES
Probable A1SJ71E71 hardware fault. Please return the A1SJ71E71 to your nearest Mitsubishi office or agent with de-

APPENDICES

APPENDIX 1 PROCESSING TIME

Calculate the minimum transmission delay time of each function using the following formulas:

It should be noted that the minimum delay time could be longer than the calculation. This is because it is influenced by the network load ratio (to the extent the line is utilized), window size of individual nodes, number of connections used simultaneously, and system configuration. Use the value obtained from the formulas as a reference value when communications are executed with only one connection.

(1) Minimum transmission delay time in communications using fixed buffer (communications between two A1SJ71E71 modules)

(a) TCP/IP

$$47 + (0.025 \times \frac{\text{(Command data length)}}{\text{In byte}}) + (0.025 \times \frac{\text{(Response data length)}}{\text{In byte}}) +$$

(Scan time at the data receiving node) + (Scan time at the data sending node) (ms)

(b) UDP/IP

$$47 + (0.023 \times \frac{\text{(Command data length)}}{\text{In byte}}) + (0.023 \times \frac{\text{(Response data length)}}{\text{In byte}}) +$$

(Scan time at the data receiving node) + (Scan time at the data sending node) (ms)

Command data length:	This includes subheader, data length, and text data. This data is set in the command application data area when data stored in the fixed buffer memory is transmitted. The command data length is processed by the byte. In binary data, command data length is "4 + (data length) × 2". In ASCII data, command data length is "4 + (data length) × 4".
Response data length:	This includes subheader and completion code. This data is set in the response application data area when data stored in the fixed buffer memory is stored. The command data length is processed by the byte. In binary data, response data length is "2". In ASCII data, response data length is "4".

[Calculation example]

The minimum transmission delay time when 1017 words data (binary data) is sent from an A1SJ71E71 to another A1SJ71E71 by using the TCP/IP protocol:

(Assume that scan time is 100 ms at the data sending node and 80 ms at the data receiving node.)

$$47 + (0.025 \times (4 + (1017 \times 2))) + (0.025 \times 2) + 100 + 80 = 278 \text{ (ms)}$$

(2) Minimum transmission delay time in communications using random access buffer memory

(a) TCP/IP

$$30 + (0.018 \times \frac{\text{(Command data length)}}{\text{In byte}}) + (0.007 \times \frac{\text{(Response data length)}}{\text{In byte}} +$$

(ACK processing time at the communicating node) (ms)

(b) UDP/IP

$$30 + (0.017 \times \frac{\text{(Command data length)}}{\text{In byte}}) + (0.006 \times \frac{\text{(Response data length)}}{\text{In byte}} \text{ (ms)}$$

Command data length: This includes subheader, data length, and text data. This data is set in the command application data area when read/write to the random access buffer memory is executed. The data length is processed by the byte.
To read binary data, command data length is "6".
To write binary data, command data length is "6 + ((data length) × 2)".
To read ASCII data, command data length is "12".
To write ASCII data, command data length is "12 + ((data length) × 4)".

Response data length: This includes subheader and completion code.
This data is set in the application data area when read/write operation to the random access buffer memory is executed. Data length is processed by the byte.
To read binary data, response data length is "2+ ((data length) × 2)".
To write binary data, response data length is "2".
To read ASCII data, response data length is "4 + ((data length) × 4)".
To write of ASCII data, response data length is "4".

ACK processing time by communicating node:

The time in which ACK is returned from the communicating node for read/write operation using random access buffer.

[Calculation example 1]

The minimum transmission delay time when reading 508 words of data (ASCII data) using the UDP/IP protocol.

$30 + (0.017 \times 12 + (0.006 \times (4 + (508 \times 4))) = 43$ (ms)

[Calculation example 2]

The minimum transmission delay time when writing 508 words of data (ASCII data) using the UDP/IP protocol.

$30 + (0.017 \times (12 + (508 \times 4))) + (0.006 \times 4) = 65$ (ms)

(3) Minimum transmission delay time for read/write operation of data in PC CPU

(a) TCP/IP

$$30 + (0.018 \times \frac{\text{(Command data length)}}{\text{In byte}}) + (0.007 \times \frac{\text{(Response data length)}}{\text{In byte}}) +$$

(PC CPU processing time) +
(Time to receive ACK from the communicating node) (ms)

(b) UDP/IP

$$30 + (0.017 \times \frac{\text{(Command data length)}}{\text{In byte}}) + (0.006 \times \frac{\text{(Response data length)}}{\text{In byte}}) +$$

(PC CPU processing time) (ms)

Command data length: This includes subheader, data length, and text data. This data is set in the command application data portion when read/write operation of data in the PC CPU is done. The data length is processed by the byte. The command data length varies according to the command to be used.
See Section 8.

Response data length: This includes subheader, data length, and text data. This data is set in the response application data area when read/write operation of data in the PC CPU is done. The data length is processed by the byte. The response data length varies according to the command to be used.
See Section 8.

PC CPU processing time:

The time in which read/write request of data in the PC CPU is processed. This is determined by the type of data to be read/written, number of processing points, and PC CPU scan time. See table 1.1.

PC CPU processing time =
(Designated number of points) ÷ (Number of processing points per sequence program scan) × (scan time)

Round off to the nearest decimal point

Time to receive ACK from the communicating node:

The time in which ACK is returned from the communicating node after the completion of read/write operation of the data in the PC CPU.

[Calculation example 1]

The minimum transmission delay time for reading data (ASCII) at 100 points of data registers (D) using the TCP/IP protocol.
(Assume that scan time is 100 ms.)

Command data length = 24 bytes
Response data length = 404 bytes
PC CPU processing time = (100 ÷ 64) × 100 = 200 (ms)
Minimum transmission delay time = 30 + (0.018 × 24) + (0.007 × 404) + 200 + (time to receive ACK from the communicating node)
= 234 + (time to receive ACK from the communicating node) (ms)

[Calculation example 2]

The minimum transmission delay time for writing data (ASCII) at 100 points of data registers (D) using the TCP/IP protocol.
(Assume that scan time is 100 ms.)

Command data length = 424 bytes
Response data length = 4 bytes
PC CPU processing time = (100 ÷ 64) × 100 = 200 (ms)
Minimum transmission delay time = 30 + (0.018 × 424) + (0.007 × 4) + 400 + (time to receive ACK from the communicating node)
= 238 + (time to receive ACK from the communicatingnode) (ms)

Table 1.1 PC CPU Communication Time

Item					PC CPU Processing Time (Time to Scan) AnSCPU A1SJCPU A0J2HCPU AnNCPU	PC CPU Processing Time (Time to Scan) A3H A3MCPU	PC CPU Processing Time (Time to Scan) A2ASCPU AnACPU AnUCPU	A1SJ71E71 Max. Processing Data between Communicating Nodes	Processing Data with 1 Scan in Sequence Program	Scan Times for Processing
Device data	Device memory	Batch read	Unit of bit		0.76 ms	0.57 ms	1.38 ms	256 points	256 points	1 scan
			Unit of words	Bit device	1.13 ms	0.81 ms	2.42 ms	128 words (2048 points)	32 words (512 points)	(Specified numbers/32) scan Round up decimal fractions (Max. 4 scans)
				Word device	1.13 ms	0.81 ms	2.42 ms	256 points	64 points	Except device R (Specified numbers/64) scan Round up decimal fractions (Max. 4 scans) Device R (Specified numbers/64) Round up decimal fractions + 1 scan (Max. 5 scans)
		Batch write	Unit of bit		1.13 ms	0.94 ms	1.06 ms	256 points	256 points	2 scans (1 scan when "Enabled during RUN" is set)
			Unit of words	Bit device	1.13 ms	0.84 ms	2.60 ms	40 words (640 points)	10 words (160 points)	(Specified numbers/64) Round up decimal fractions + 1 scan "0" when "Enabled during RUN" is set. (Max. 5 scans)
				Word device	1.13 ms	0.84 ms	2.60 ms	256 points	64 points	Except device R (Specified numbers/64) Round up decimal fractions + 1 scan "0" when "Enabled during RUN" is set. (Max. 5 scans) Device R (Specified numbers/64) Round up decimal fractions + 1 scan (Max. 5 scans)
		Test (random write)	Unit of bit		1.13 ms	0.90 ms	1.06 ms	80 points	20 points	(Specified numbers/20) Round up decimal fractions + 1 scan "0" when "Enabled during RUN" is set. (Max. 5 scans)
			Unit of words	Bit device	1.13 ms	0.90 ms	1.06 ms	40 words (640 points)	10 words (160 points)	(Specified numbers/10) Round up decimal fractions + 1 scan "0" when "Enabled during RUN" is set. (Max. 5 scans)

Table 1.1 PC CPU Communication Time (Continued)

Item					PC CPU Processing Time (Time to Scan) AnSCPU A1SJCPU A0J2HCPU AnNCPU	PC CPU Processing Time (Time to Scan) A3H A3MCPU	PC CPU Processing Time (Time to Scan) A2ASCPU AnACPU AnUCPU	A1SJ71E71 Max. Processing Data between Communicating Nodes	Processing Data with 1 Scan in Sequence Program	Scan Times for Processing
Device data	Device memory	Test (random write)	Unit of words	Word device	1.13 ms	0.90 ms	1.06 ms	40 points	10 points	Except device R (Specified numbers/10) Round off to the nearest decimal point. + 1 scan "0" when "Enabled during RUN" is set. (Max.5 scans) Device R (Specified numbers/10) Round off to the nearest decimal point. + 1 scan (Max.5 scans)
		Monitor data registration	Unit of bit		—	—	—	—	—	—
			Unit of words					—	—	1 scan for device R
		Monitor	Unit of bit		2.02 ms	0.93 ms	1.46 ms	40 points	40 points	1 scan
			Unit of words	Bit device	2.08 ms	0.96 ms	1.47 ms	320 points (20 words)	320 points (20 words)	1 scan
				word device	2.08 ms	0.96 ms	1.47 ms	20 points	20 points	
Device data	Extension file register	Batch read			1.27 ms	0.76 ms	2.42 ms	256 points	64 points	(Specified numbers/64) Round off to the nearest decimal point. (Max.5 scans)
		Batch write			1.27 ms	0.76 ms	2.60 ms	256 points	64 points	
		Direct read			—	—	2.30 ms	256 points	64 points	
		Direct write			—	—	2.57 ms	256 points	64 points	
		Test (random write)			1.31 ms	0.87 ms	0.97 ms	40 points	10 points	
		Monitor data registration			—	—		—	—	—
		Monitor			1.75 ms	0.98 ms	1.42 ms	20 points	20 points	1 scan
Special function module buffer memory		Batch read			FROM instruction processing time + 1.13 ms	FROM instruction processing time + 0.81 ms	FROM instruction processing time + 0.75 ms	256 bytes	128 bytes	(Specified numbers/128) Round off to the nearest decimal point. Scan (Max.2 scans)
		Batch write								(Specified numbers/128) Round off to the nearest decimal point. + 1 scan "0" when "Enabled during RUN" is set. (Max.3 scans)

Table 1.1 PC CPU Communication Time (Continued)

Item				PC CPU Processing Time (Time to Scan)			A1SJ71E71 Max. Processing Data between Communicating Nodes	Processing Data with 1 Scan in Sequence Program	Scan Times for Processing
				AnSCPU A1SJCPU A0J2HCPU AnNCPU	A3H A3MCPU	A2ASCPU AnACPU AnUCPU			
Program	Sequence program	Batch read	Main	1.20 ms	0.78 ms	0.70 ms	256 steps	64 steps	(Specified numbers/64) Round off to the nearest decimal point. Scan (Max.4 scans)
			Sub	120 ms	0.84 ms	0.70 ms			
		Batch write	Main	0.67 ms	0.55 ms	0.49 ms	256 steps	64 steps	(Specified numbers/64) Round off to the nearest decimal point. + 1 scan ("0" when "Enabled during RUN" is set.) (Max.4 scans)
			Sub	0.67 ms	0.55 ms	0.49 ms			
	Microcomputer program	Batch read	Main	1.35 ms	0.76 ms	—	256 bytes	128 bytes	(Specified numbers/128) Round off to the nearest decimal point. + 1 scan (Max.3 scans)
			Sub	1.35 ms	0.76 ms				
		Batch write	Main	1.35 ms	0.76 ms				
			Sub	1.53 ms	0.73 ms				
	Comment	Batch read		1.35 ms	0.76 ms	2.42 ms	256 bytes	128 bytes	(Specified numbers/128) + 1 scan (Max.3 scans)
		Batch write		1.53 ms	0.73 ms	2.60 ms			
	Extension comment	Batch read		—	—	2.31 ms	256 bytes	128 bytes	(Specified numbers/128) Round off to the nearest decimal point. + 1 scan (Max.3 scans)
		Batch write		—	—	2.59 ms			
	Parameter	Batch read		0.68 ms	0.50 ms	2.42 ms	256 bytes	128 bytes	(Specified numbers/128) Round off to the nearest decimal point. + 1 scan (Max.3 scans)
		Batch write		—	—	—	—	—	—
		Analyze request							
PC CPU		Remote RUN		—	—	—	—	—	—
		Remote STOP		—	—	—	—	—	
		Read for individual PC		—	—	—	—	—	1 scan

POINTS

(1) The PC CPU processes any one of the above operations for each END. So, if the A6GPP and A1SJ71E71 access the PC CPU simultaneously, one operation is suspended until the other one is completed. In this case, the scan time for processing could be longer.

(2) Even if the PC CPU is not linked to the A1SJ71E71, the scan time would remain approximately 0.2 ms longer (for an A2AS, A3H, A3M, AnA, or AnUCPU, 0.1 ms longer).

APPENDIX 2 ASCII CODE TABLE

LSD \ MSD		0	1	2	3	4	5	6	7
		000	001	010	011	100	101	110	111
0	0000	NUL	DLE	SP	0	@	P		p
1	0001	SOH	DC1	!	1	A	Q	à	q
2	0010	STX	DC2	!!	2	B	R	b	r
3	0011	ETX	DC3	#	3	C	S	c	s
4	0100	EOT	DC4	$	4	D	T	d	t
5	0101	ENQ	NAK	%	5	E	U	e	u
6	0110	ACK	SYN	&	6	F	V	f	v
7	0111	BEL	ETB	/	7	G	W	g	w
8	1000	BS	CAN	(	8	H	X	h	x
9	1001	HT	EM	)	9	I	Y	i	y
A	1010	LF	SUB	*	:	J	Z	j	z
B	1011	VT	ESC	+	;	K	[	k	{
C	1100	FF	FS	,	<	L	\	l	|
D	1101	CR	GS	−	=	M	]	m	}
E	1110	SO	RS	.	>	N	↑	n	~
F	1111	SI	VS	/	?	O	←	o	DEL

APPENDIX 3 REFERENCE

The "DDN Protocol Handbook" (a three-volume set) gives details on the TCP/IP.

Publisher:

DDN Network Information Center

SRI International

333 Ravenswood Avenue, EJ291

Menlo Park, California 94025

RFC Numbers:

TCP RFC793

UDP RFC768

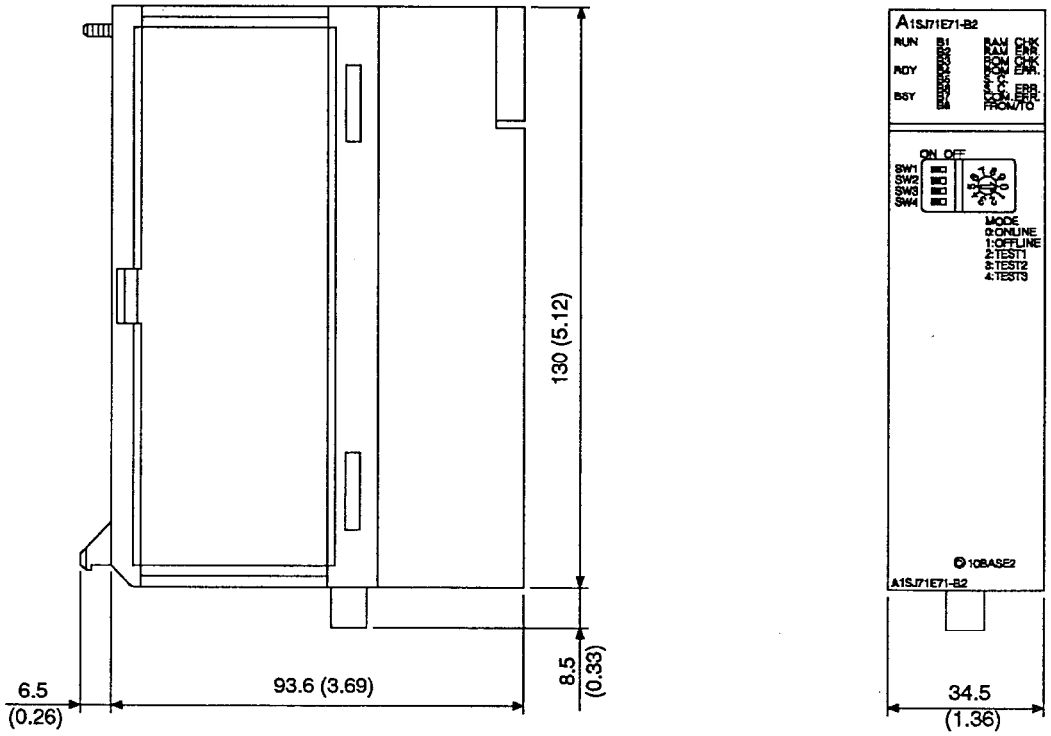
IP RFC791

ICMP RFC792

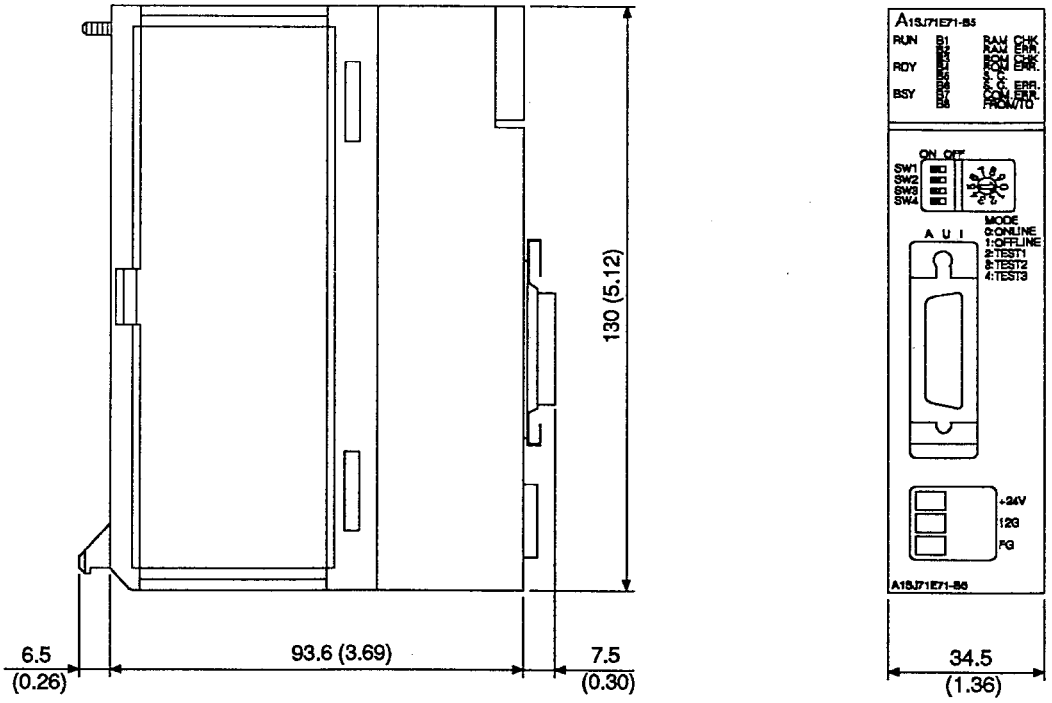
ARP RFC826

APPENDIX 4 EXTERNAL DIMENSIONS DIAGRAM

A1SJ71E71-B2



A1SJ71E71-B5



Unit : mm (inch)

APPENDIX 5 SAMPLE PROGRAMS

This sample program is used to perform a connection test on the connection between the A1SJ71E71 and the LM7000.
This program is presented as an example: it accesses the data registers (D) and extension file registers (R) of the ACPU on which the A1SJ71E71 is loaded, and reads the random buffer of the A1SJ71E71.
Use an Ethernet board made by Digital Equipment Corporation for the LM7000.

Ethernet board made by Digital Equipment Corporation: EB-10M/AX
Library made by Digital Equipment Corporation: LSOCK.LIB

(1) Access range

Data registers: D100 to D121
Extension file registers Block No.1: R10 to R20
Random buffer reading: Addresses 100 to 121

(2) Modification method

By modifying E71INC.H, it is possible to access other station and other devices.

< Changing devices>

The devices to be accessed are set at the "D_TYPEL" and "D_TYPEH" device code entries.
To access data registers (D), set the entries to D_TYPEH, D_TYPEL = 44H, 20H.
To access link registers (W), set the entries to D_TYPEH, D_TYPEL = 57H, 20H.
For other devices, see section 8.4.1.

< Changing device numbers>

The head device number is set at the "D_NO" entry.
In this program the setting is "D_NO" = 100.

< Changing PC numbers>

The communication destination ACPU is set at the "PC_NO" entry.
Self station: "PC_NO" = FF, Other station: "PC_NO" = station number

< Changing the extension file register block number>

The extension file register block number is set at the "R_BLOCKL" and "D_TYPEH" entries.
In this program the setting is for block number 1.

< Number of communications>

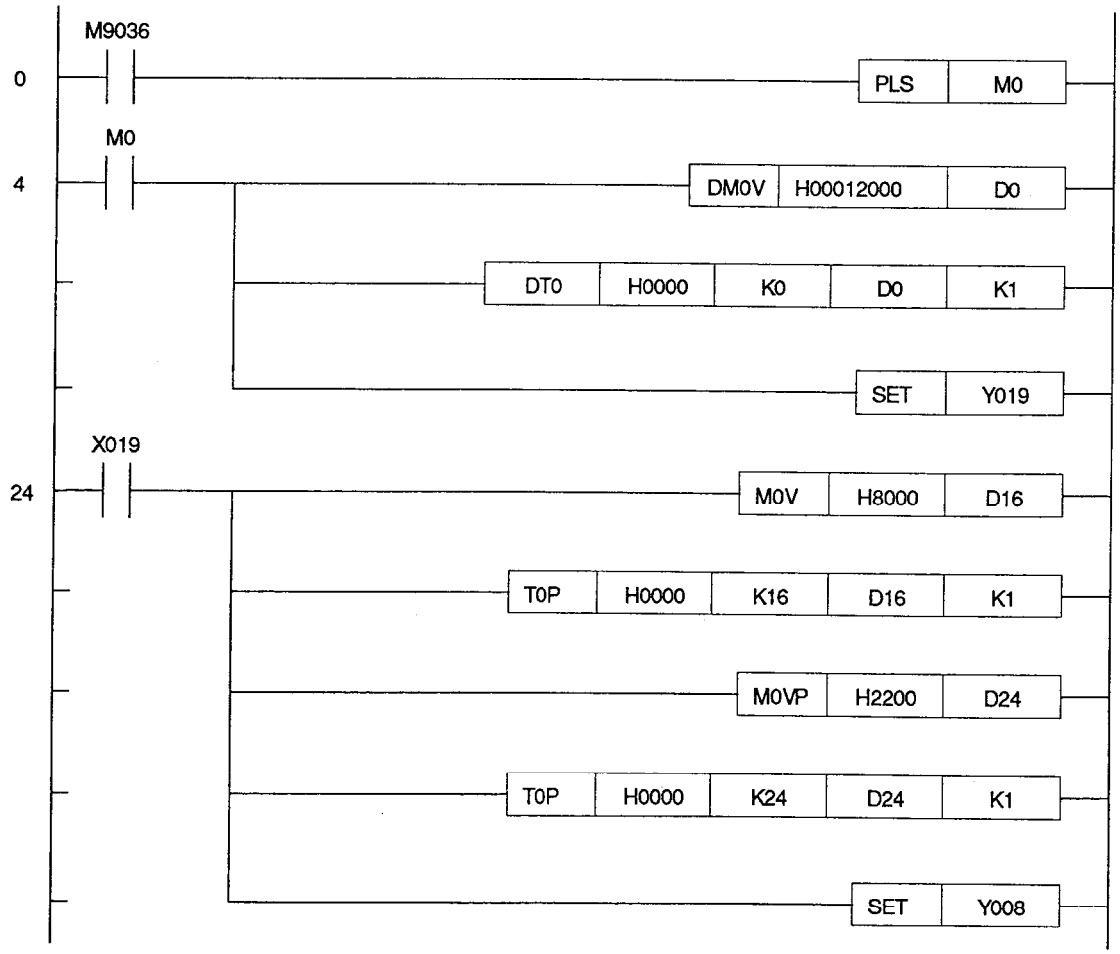
The number of communications with the A1SJ71E71 is set at the "ACLOOP" entry.
Inthis program the setting is "ACLOOP" = 10.

< Changing the port number & IP address>

The LM7000 port number and IP address are set at the "MYPORT" and MY_IP" entries.
The A1SJ71E71 port number and IP address are set at the "DST_PORT" and "DST_IP" entries.
When an LM7000 is added to an existing Ethernet system, its port number and IP address must be set in accordance with the system.
Confirmation of the port number and IP address must be obtained from the "super user" (network manager).

(3) Sequence program

An example sequence program for the ACPU installed with the communicating A1SJ71E71 is shown below.
This program does only the bare minimum. For details on action to take in the event of errors etc., see Chapter 5.



(4) Header file (E71INC.H)

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/* *     Definition for A1SJ71E71                                                  * */
/* *                                                                               * */
/* *     By modifying E71INC.H, it is possible to access other stations            * */
/* *     and other devices.                                                        * */
/* *                                                                               * */
/* *                                                  Date 19/01/91                * */
/* *                                                                               * */
/* *                                                                               * */
/* *     Copyright (C) 1991 Mitsubishi Electric Corporation                        * */
/* *     All Rights Reserved                                                       * */
/* *                                                                               * */
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/* *     < Program modify list>                                                    * */
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/* *     Port number, IP address                                                   * */
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
#define         MY_PORT         0x2000      /* soure port */
#define         MY_IP           0x11000     /* my IP address */
#define         DST_PORT        0x2200      /*destination port */
#define         DST_IP          0x12000     /*destination IP address */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/* *     The number of times A1SJ71E71 is accessed is set                          * */
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
#define         ACTLOOP         10          /* access count to ACPU */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/* *  If the communication destination ACPU is at the self station, FF             * */
/* *  is set at PC_NO; if it is another station, the relevant station num-         * */
/* *  ber is set at PC_NO.                                                         * */
/* *  The device type to be accessed is set at D_TYPEL and D_TYPEH.                * */
/* *  The extension file register block number is set at R_BLOCKL and              * */
/* *  D_TYPEH.                                                                     * */
/* *  The head address to be accessed is set at D_NO.                              * */
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
#define         PC_NO           0xff        /*PLC station No. */
#define         D_NO            100         /* device No. */
#define         D_TYPEL         0x20        /* device type (L) D */
#define         D_TYPEH         0x44        /* device type (H) D */
#define         R_TYPEL         0x20        /* device type (L) R */
#define         R_TYPEH         0x52        /* device type (H) R */
#define         R_BLOCKL        0x01        /* R device block (L) */
#define         R_BLOCKH        0x00        /* R device block (H) */
```

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/* *      Data for A1SJ71E71                                                        * */
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
#define         E71RD           0x01        /* sub_header "Word batch Read" */
#define         E71WR           0x03        /* sub_header "Word batch Write" */
#define         E71R_RD         0x17        /* sub_header "Word batch Read R" */
#define         E71R_WR         0x18        /* sub_header "Word batch Write R" */
#define         E71RM_RD        0x61        /* sub header random access read_out */
#define         E71RM_WR        0x62        /* sub header random access write_out*/

#define         ACCRLOOP        30000       /* data access count */
#define         A_TIMEL         0x02        /* ACPU supervising time (L) * 500ms */
#define         A_TIMEH         0x00        /* ACPU supervising time (H) * 500ms */
#define         DWORDW_MAX      253         /* dwordw takes number */
#define         ADDR_MAX        1792        /* devices takes number*/
```

(5) Sample program

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/* *      A1SJ71E71 sample program                                        * */
/* *                                                                      * */
/* *      This sample program is used to perform a connection test        * */
/* *      on the connection between the A1SJ71E71 and the PC/AT.          * */
/* *      This program is presented as an example; it accesses the        * */
/* *      data registers (D) and extension file registers (R) of the      * */
/* *      ACPU installed with the A1SJ71E71 and reads the random          * */
/* *      buffer of the A1SJ71E71 unit.                                   * */
/* *      It is possible to access other stations and other devices by    * */
/* *      modifying the E71INC.H.                                         * */
/* *                                                                      * */
/* *                                              Date  19/01/91          * */
/* *                                                                      * */
/* *      Copyright (C) 1991 Mitsubishi Electric Corporation              * */
/* *      All Rights Reserved                                             * */
/* *                                                                      * */
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/* *                                                                      * */
/* *    < Program modify list>                                            * */
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

# include < stdio.h>
# include < ctype.h>
# include < conio.h>
# include < io.h>
# include < fcntl.h>

# include "socket.h"
# include "e71inc.h"                          /* Definition for A1SJ71E71 */

char name_0[ ] = "ACPUAAA";

char sdata [2048], resp_data [4096];
int         s [1];
struct sckidtbl sk [1], *scktbl;
int         scktbl_len;
struct sockaddr sa [1], *saddr;
struct host_type h_typ;

int         init ( );
int         socket ( );
int         connect ( );
int         term ( );
int         shutdown ( );
int         send ( );
int         recv ( );
void        cursor ( );
void        cls ( );
```

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/* *      Main program (initial display)                                          * */
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
void main ( )
{
        int     sub_main ( );
        int     id, sts;

        cls ( );
        cursor (2,20);
        printf ("< < Ethernet test/General data processing> > \n");

        cursor (3,1);
        printf ("Board statuses\n");
        cursor (6,1);
        printf ("D_reg Access\n");
        curosr (11,1);
        printf ("R_reg access\n");
        cursor (16,1);
        printf ("Random access\n");
        cursor (22,1);
        printf ("Error message\n");
        curosr (5,20);
        printf ("returned at the main, %x\n", sub_main ( ));
        exit (0);
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/* *      Subprogram                                                              * */
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
int     sub_main ( )
{
        int     dreg_wr ( );
        int     rreg_wr ( );

        int     dwordw;
        int     addr= D_NO;                 /* Head address setting */
        int     accr =  0;
        int     err_p;
        int     shutf =  1;
        int     retv;

        sa [0] .sa_family     = SOCK_STREAM;
        sa [0] .sa_port       = MY_PORT;
        sa [0] .dst_port      = DST_PORT;
        sa [0] .dst_Ip        = DST_IP;
        sa [0] .e_addr [6]    = 0x00;
```

```
        h_typ.host_name [16] = 0x00;
        h_typ.Ip_addr      = MY_IP;
        h_typ.e_addr [6]   = 0x00;

        saddr = sa;

/*= = = = board intialization = = = = */
        if (init (&err_p) != 0)          /* Initialization of Ethernet board */
                {
                cursor (22,20);
                printf ("Initialization error %x\n",err_p);
                return (-1);
                }
        cursor (3,20);
        printf ("Init sucess.\n");

/*= = = = socket entry = = = = */
        s[0] = socket (saddr, name_0, sizeof (name_0), &err_p);
                        /* "s" is the registered socket ID number */
        if (s [0] = -1)
                {
                cursor (22,20);
                printf ("Socket entry err. s1_1, err code %x\n",err_p);
                return (-1);
                }

/*= = = = Connection = = = = */
        scktbl = sk; /* Make connection with A1SJ71E71 (passive state) */
                /* Data register access                                  */
        if (connect (s [0] ,scktbl,&scktbl_len,&err_p) != 0)
                return (err_p);
        cursor (3,20);
        printf ("Cnctd %x, %x, %s, %x, %x, %lx\n",
                 scktbl-> id,scktbl-> lcn,scktbl-> rmt_name,
                 scktbl-> local_prt, scktbl-> remote_prt,scktbl-> remote_ip);

/*= = = = communication start = = = = */
        dwordw = 1;
        for (accr= 0;accr< ACTLOOP;+ + accr)
                {
                if (accr = = ACCRLOOP)
                        accr = 0;
                cursor (8,20);
                printf ("                                                \n")
                cursor (9,20);
                printf ("                                                \n")

        /*= = = = Data_register access = = = = */
                if (dreg_wr (dwordw, accr, addr) = = -1)
                                                /*Data register access */
```

```
                    break;
                cursor (13,20);
                printf ("                                                    \n");
                cursor (14,20);
                printf ("                                                    \n");

            /*= = = = File_register access = = = = */
                if (rreg_wr (dwordw,accr, addr) = = -1)
                                                /* Extension file register access */
                    break;
                cursor (18,20);
                printf ("                                                    \n");

            /*= = = = A1SJ71E71 random_buffer access = = = = */
                if (randm_buf (512, addr) = = -1)     /* A1SJ71E71 random buffer */
                    break;

                + + dwordw;
                + + addr;
                if (dwordw = = DWORDW_MAX) /* Number of data items accessed */
                    dwordw = 1;
                if (addr = = ADDR_MAX)                /* Head device address */
                    addr = 0;
                }
            printf ("Accr %d\n",accr);

        /*= = = = communication end = = = = */
            while (shutf ! = 0)
                {
                shutf = shutdown (s [0], 1,&err_p);          /* Disconnection */
                if ( (shutf = = -1)& (err_p != 0x0a))
                    {
                    printf ("shutdown err code %x\n",err_p);
                    shutf = 0;
                    }
                }
            cursor (6,20);
            printf ("Shutdown %d ",s [0]);

            if (term (&err_p) = = -1              /* Termination of Ethernet board */
                {
                printf ("termination err %x\n",err_p);
                return (-1);
                }

            printf ("Terminated\n");
            cursor (22,1);
            return (0);
        }
```

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/* *     Data Register Access Program                                         * */
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/*= = = = Data register access = = = = */
int      dreg_wr (dwordw, accr, addr)
int      dwordw, accr, addr;
{
         extern char sdata [2048], resp_data [4096];
         extern int s [1];

         void    data_gen ( );
         void    devide ( );
         int     retv;
         char dvded_data [2];

         data_gen (dwordw, 18, accr); /* Creation of write data */
         sdata [0] = E71WR;             /* Subheader: device batch write */
         sdata[1] = PC_NO;              /* PC number */
         sdata [2] = A_TIMEL;           /* ACPU watchdog timer */
         sdata [3] = A_TIMEH;

         devide (addr, dvded_data);
         sdata [4] = dvded_data [0];  /* Head device number */
         sdata [5] = dvded_data [1];
         sdata [6] = 0x00;
         sdata [7] = 0x00;
         sdata [8] = D_TYPEL;           /* Device code */
         sdata [9] = D_TYPEH;

         devide (dwordw+ 3, dvded_data);
         sdata [10] = dvded_data [0]; /* Number of device points */
         sdata [11] = 0x00;
         sdata [12] = 0x03;             /* Write data 1 */
         sdata [13] = 0x00;
         sdata [14] = 0x02;             /* Write data 2 */
         sdata [15] = 0x00;
         devide (accr, dvded_data);
         sdata [16] = dvded_data [0]; /* Write data 3 */
         sdata [17] = dvded_data [1];

/*= = = = data writing into data reg.acpu = = = = */
         cusor (6,20);
         printf ("DATA REG. Counter = %6d, Address = %6d /", accr, addr);
         retv = data_send (s [0], dwordw+ 9); /* Device write */
         if (retv = = -1)
                 return (-1);
         cursor (8,20);
         printf ("%4d b WR com. sent\n",retv);
         rev = 0;
```

```
        retv = rcv_resp (s [0], 1);     /* A1SJ71E71 response receive */
        if (retv = = -1)
                return (-1);

        cursor (9,20);
        printf ("%4d b WR resp.rcvd\n",retv);

        sdata [0] = E71RD;                      /* Subheader: device batch read */
        retv = data_send (s [0], 6);  /* Read request */
        if (retv = = -1)
                return (-1);

        cursor (8,42);
        printf ("%4d b RD com.sent\n",rev);
        retv = rcv_resp (s [0],dwordw+ 4);                  /* Device read */
        if (retv = = -1)
                return (-1);
        cursor (9,42);
        printf ("%4d b RD resp.revd\n",retv);
        retv = data_cmp (& (sdata [18], & (resp_data [8]),dwordw);
                                                      /* Data comparison */
        if (retv != 0)
                {
                cursor (22,20);
                printf ("Data comp err D_reg head= %d, offset = %d, WR = %d, RD
= %d\n",addr+ 0x400, retv, sdata [18+ retv], resp_data [8+ retv]);
                return (-1);
                }
        cursor (9,64);
        printf ("Data compared\n");
        return (0);                     /* normal end */
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/* *     Extension File Register Access Program                           * */
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/*= = = = file register access = = = = */
rreg_wr (dwordw, accr, addr)
int     dwordw, accr, addr;
{
        int     retv;
        char dvded_data [2];

        extern char sdata [2048], resp_data [4096];
        extern int s [1];
        void    data_gen ( );
        void    devide ( );

        data_gen (dwordw, 20, accr+ 0x100);     /* Creation of write data */
        sdata [0] = E71R_WR; /* Subheader: extension file register batch write */
```

```
        sdata [1] = PC_NO;              /* PC number */
        sdata [2] = A_TIMEL;            /* ACPU monitoring time */
        sdata [3] = A_TIMEH;

        devide (addr, dvded_data);
        sdata [4] = dvded_data [0]; /* Head device number */
        sdata [5] = dvded_data [1];
        sdata [6] = 0x00;
        sdata [7] = 0x00;
        sdata [8] = R_TYPEL;            /* Device code */
        sdata [9] = R_TYPEH;
        sdata [10] = R_BLOCKL;    /* Block No. */
        sdata [11] = R_BLOCKH;

        devide (dwordw+ 3, dvded_data);
        sdata [12] = dvded_data [0]; /* Number of device points */
        sdata [13] = 0x00;
        sdata [14] = 0x03;              /* Write data 1 */
        sdata [15] = 0x00;
        sdata [16] = 0x02;              /* Write data 2 */
        sdata [17] = 0x00;
        devide (accr, dvded_data);    /* Write data 3 */
        sdata [18] = dvded_data [0];
        sdata [19] = dvded_data [1];

/* = = = = data writing into file register in block 1 = = = = */
        cursor (11,20);
        printf ("FILE REG. Counter= %6d, Address = %6d / ",accr, addr);
        retv = data_send (s [0], dwordw+ 10); /* Extension file register write */
        if (retv = = -1)
                return (-1);
        cursor (13, 20);
        printf ("%4d b WR comm.sent\n",retv);
        retv = 0;
        retv = rcv_resp (s [0], 1);     /* A1SJ71E71 response receive */
        if (retv = = -1)
                return (-1);
        cursor (14,20);
        printf ("%4d b WR resp.rcdvd\n",retv);

        sdata [0] = E71R_RD; /* Subheader: extension file register batch read */
        sdata [1] = PC_NO;
        retv = data_send (s [0], 7);    /* Read request */
        if (retv = = -1)
                return (-1);
        cursor (13, 42);
        printf ("%4d b RD com.sent\n",retv);
        retv = rcv_resp (s [0],dwordw+ 4);/* Extension file register read */
        if (retv = = -1)
```

```
                return (-1);
        cursor (14,42);
        printf ("%4d b RD resp.rcvd\n",retv);
        retv= data_cmp(&(sdata[20]),&(resp_data[8]),dwordw);/* Data comparison */
        if (retv != 0)
                {
                cursor (22,20);
                printf ("Data comp ef, F_reg head = %d, offset= %d, WR = %d, RD =
%d\n",addr,retv, sdata [20+ retv],resp_data [8+ retv]);
                return (-1);
                }
        cursor (14,64);
        printf ("Data compared \n");
        return (0);
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/* *     A1SJ71E71 Random Buffer Access Program                    * */
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/* = = = = random buffer access = = = = */
int     randm_buf (bwordw, addr)
int     addr;
int     bwordw;
{
        extern char sdata [2048];
        extern char resp_data [4096];
        extern int s [1];

        int     retv;
        char    dvded_data [4];

        cursor (16,20);
        printf ("Read addr = %4d \n ", addr);
        sdata [0] = E71RM_RD;           /* Subheader: random buffer read */
        sdata [1] = 0x00;
        devide (addr, dvded_data);
        sdata [2] = dvded_data [0];     /* Head buffer address */
        sdata [3] = dvded_data [1];
        devide (bwordw, dvded_data);
        sdata [4] = dvded_data [0];     /* Number of words to read */
        sdata [5] = dvded_data [1];
        retv = data_send (s [0], 3);    /* Read request */
        if (retv = = -1)
                return (-1);
        cursor (18,20);
        printf ("%4d byte data sent\n",retv);

        retv = rcv_resp (s [0], bwordw + 1);        /* Random buffer read */
```

```
            if (retv = =  -1)
                    return (-1);
            if ( (resp_data [0] != 0xe1) | | (resp_data [1] != 0x00))
                    {
                    cursor (22,20);
                    printf ("Bad response %x, %x rcved\n", resp_data [0],
                    resp_data [1]);
                    return (-1);
                    }
            cursor (18,42);
            printf ("%4d byte resp.rcvd\n",retv);
            return (0);                         /* normal end */
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *\
/* *     Data Send Program                                                       * *\
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *\
/* = = = = data send function = = = = */
int         data_send (dst_sk,wordw)
int         dst_sk;                             /* socket ID */
int         wordw;                              /* data length in word */
{
            extern char sdata [2048];

            int     sendf;
            int     err_p;
            int     j= 0;
            int     cmpltf =  0;
            int     sendw =  0;
            int     ptr= 0;
            int     bytew;

            bytew =  wordw +  wordw;
            while (compltf = =  0)
                    {
                    sendf= send (dst_sk,& (sdata [ptr]),bytew,&err_p);
                                                /* TCP data send */
                    sendw =  sendw +  sendf;
                    if (sendf <  0)
                          {
                          cursor (22,20);
                          printf ("Data send err to %d, error code %x\n",dst_sk,err_p);
                          cmpltf =  1;
                          return (-1);
                          }
                    else if (sendf <  bytew)
                          {
                          ptr =  sendw;
                          bytew =  bytew - sendf;
```

```
                    if (ptr > 526)
                            return (-1);
                    if (+ + j > 10)
                            {
                            cursor (22,20);
                            printf ("TIME OVER\n");
                            return (-1);
                            }
                    }
            else if (sendf = = bytew)
                    {
                    cmpltf = 1;
                    return (sendw);
                    }
            else
                    {
                    cmpltf = 1;
                    reutrn (-1);
                    }
            }
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/* *    A1SJ71E71 Response Receive Program                                   * */
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/*= = = = response receive function = = = = */
int     rcv_resp (dst_sk, wordw)
int     dst_sk;                                 /* socket ID, data length */
int     wordw;
{
        extern char resp_data [4096];

        int     err_p, rcvf, temp;
        int     l = 0;
        int     j;
        int     bytew;
        int     cmpltf = 0;
        int     ptr = 0;
        int     rcvdw = 0;

        bytew = wordw + wordw;
        rcvf = 0;
        while (cmpltf = = 0)
                {
                rcvf= recv (dst_sk,& (resp_data [ptr]), bytew+ 1026, &err_p);
                                                        /* TCP data receive */
                rcvdw = rcvdw + rcvf;
                if (rcvf < 0)
```

```
                {
                cursor (22,20);
                printf ("Response recv err frm %d, err code %x\n",dst_sk,err_p);
                cmpltf= 1;
                return (-1);
                }
        else if (rcvf < bytew)
                {
                ptr = rcdvdw;
                bytew = bytew - rcvf;
                if (ptr > 526)
                        return (-1);
                if (+ + i > 32000)
                        {
                        cursor (22,20);
                        printf ("TIME OUT\n");
                        return (-1);
                        }
                }
        else if (rcvf = = bytew)
                {
                cmpltf = 1;
                return (rcvdw);
                }
        else
                {
                cursor (22, 20);
                printf ("&&&&&Too many resp. rcvd, exp = %d, actual = %d,
rcvf = %d&&&&&\n", bytew, rcvdw, rcvf);
                temp = rcvf - bytew;
                if ((resp_data [0] = = resp_data [temp])&& (resp_data [1] = =
resp_data [temp + 1]))
                        {
                        for (i= 0;i< bytew;+ + i)
                                sresp_data [i] = resp_data [i+ temp];
                        rcvdw= rcvdw - temp;
                        return (rcvdw);
                        }
                cmpltf = 1;
                return (-1);
                }
        }
}
```

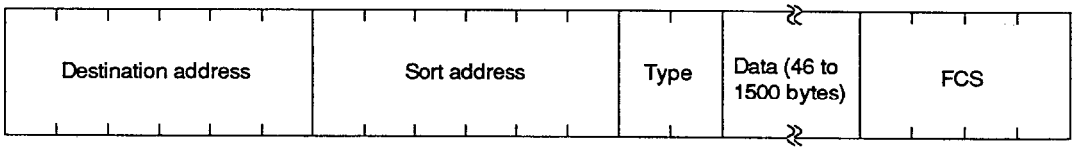
```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/* *     Send Data/Receive Data Comparison Program                    * */
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/*= = = = data comparison function = = = = */
int     data_cmp (sdata_ptr,rdata_ptr,wordw)
int     wordw;
char    *sdata_ptr;
char    *rdata_ptr;
{

        int i= 0;
        char data;
        int bytew;

        bytew = wordw + wordw;
        data = *sdata_ptr;
        for (i;i< bytew;+ + i)
                {
                if (data != *rdata_ptr)
                        break;
                + + data;
                + + rdata_ptr;
                }
        if (i != bytew)
                {
                if (i = = 0)
                        return (-1);
                else
                        return (i);
                }
        else
                return (0);
}

/*= = = = data generator = = = = */
void    data_gen(wordw, ptr, f_dat)
int     wordw, f_dat;
int     ptr;
{
        int     i= 0;
        char    j = 0;
        char    data;
        int     bytew;

        data = (char)f_dat;
        bytew = wordw + wordw;
        if (wordw < = 253)
                {
```

```
                    for (i= 0;i< bytew;+ + i)
                            {
                            sdata [ptr+ i] = data + j;
                            + + j;
                            }
                    }
}

void    devide (in_data,out_data_ptr)
int     in_data;
char    *out_data_ptr;
{
        int temp;

        temp = in_data & 0xff;
        *out_data_ptr = (char) temp;
        + + out_data_ptr;
        temp = in_data > > 8 & 0xff;
        *out_data_ptr = (char) temp;
}

void    cls ( )
{
        printf ("\x1b [2J");
{

void    cursor (pl,pc)
char    pl,pc;
{
        printf ("\x1b [%d;%dH",pl,pc);
}
```

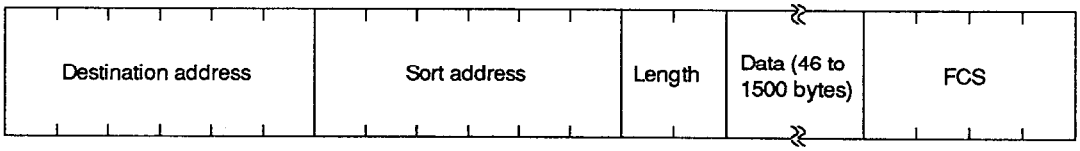
APPENDIX 6 DIFFERENCES BETWEEN ETHERNET AND IEEE802.3

The A1SJ71E71 conforms to Ethernet specifications.

(1) Ethernet



(2) IEEE802.3 (data link layer)



IMPORTANT

(1) Design the configuration of a system to provide an external protective or safety inter locking circuit for the PCs.

(2) The components on the printed circuit boards will be damaged by static electricity, so avoid handling them directly. If it is necessary to handle them take the following precautions.

(a) Ground your body and the work bench.

(b) Do not touch the conductive areas of the printed circuit board and its electrical parts with non-grounded tools, etc.

Under no circumstances will Mitsubishi Electric be liable or responsible for any consequential damage that may arise as a result of the installation or use of this equipment.

All examples and diagrams shown in this manual are intended only as an aid to understanding the text, not to guarantee operation. Mitsubishi Electric will accept no responsibility for actual use of the product based on these illustrative examples.

Owing to the very great variety in possible applications of this equipment, you must satisfy yourself as to its suitability for your specific application.

Ethernet Interface Module type A1SJ71E71-B2/B5

User's Manual

MODEL	A1SJ71E71-U-E
MODEL CODE	13JE87
IB(NA)66547-A(9508)MEE	

MITSUBISHI ELECTRIC CORPORATION

HEAD OFFICE : MITSUBISHI DENKI BLDG MARUNOUCHI TOKYO 100-0005 TELEX : J24532 CABLE MELCO TOKYO
NAGOYA WORKS : 1-14 , YADA-MINAMI 5 , HIGASHI-KU, NAGOYA , JAPAN