# Part VII

# ECHONET Communications Equipment Specifications

Revision rHistory Note) Version numbers except Ver.3.20 indicate Japanese editions.

| | | | |
|---|---|---|---|
| •Version 1.00 | March 18th | 2000 | Released / Open to consortium members |
| | July | 2000 | Open to the public |
| •Version 1.01 | May 23rd | 2001 | Open to the public |
| | | Version 1.0 addendum & corrigendum | |
| •Version 2.00 | August 7th | 2001 | Open to consortium members |
| | | Version 1.0 addendum & corrigendum | |
| •Version 2.01 | December 19th | 2001 | Open to consortium members. |
| | | Errors in Version 2.00 were corrected. | |
| •Version 2.10 Preview | December 28th | 2001 | Open to consortium members |
| •Version 2.10 Draft | February 15th | 2002 | Open to consortium members |

The following Table-of-Contents entries were revised:

| | Revised entry | Revision/addition |
|---|---|---|
| 1 | 3.6.7, Table 3.6 | - The "protocol difference absorption processing block status acquisition" service was changed to an optional service.<br>- The "stop notice" service and its response "stop notice process response" were added. |
| 2 | 3.6.7 (3) | - The "(19) Stop notice service" was added.<br>- The "(20) Complete initialization request service" was added.<br>- The "(21) Communication stop request service" was added.<br>- The "(22) Stop request service" was added.<br>- The "(5) Reset request service" was renamed to "(5) Start request service".<br>- Explanations of all services were added. |
| 3 | 3.6.2 | - The name of Type 4 was changed to "RS-232C/UART type interface". |
| 4 | 3.6.3 (1) (2) (4) | - For Type 4, the requirement about an open-collector type was added. |
| 5 | 3.6.4 (1) | - For Type 4, the requirement about an open-collector type was added. |
| 6 | 3.6.4 (4) | - The Type 4 logic description was divided into "RS-232C level" and "open collector" logic descriptions. |
| 7 | 3.6.4 (5) | - The Type 4 signal level description was divided into "RS-232C level" and "open collector" signal level descriptions. |
| 8 | 3.6.5 (1) | - A note was added to the description of the Type 4 RTS/CTS control method. |
| 9 | 3.6.3 (5) | - For Type 4 (open collector), the connector shape description was added. |
| 10 | 3.6.6 | - The reference circuit diagram for Type 4 adapter communication interface was added. |

•Version 2.10 March 7th 2002 Open to consortium members

The following Table-of-Contents entries were revised:

| | Revised entry | Revision/addition |
|---|---|---|
| 1 | 3.6.8 | - The "(5) Reset request service" was renamed "(5) Warm start request service".<br>- "(19) Stop notice service" was added.<br>- "(20) Complete initialization request service" was added.<br>- "(21) Communication stop request service" was added.<br>- "(22) Stop request service" was added.<br>- Explanations of all services were added. |
| 2 | 3.6.7 | - An explanation of lower-layer communication software ID was added. |
| 3 | 3.6.7 | - The number of software programs offering the lower-layer communication software mounting information request service was fixed at 0x01. |
| 4 | 3.6.8 | - Section 3.6.8 was specially added to explain the adapter communication interface service. |

•Version 2.11 April 26th 2002 Open to consortium members

The following Table-of-Contents entries were revised:

| | Revised entry | Revision/addition |
|---|---|---|
| 1 | 3.6.2 | - The name of Type 4 was changed to “RS232C/UART interface.” |
| 2 | 3.6.5 | - Explanation of service request collision was added. |
| 3 | 3.6.7 (1) | - Explanation of adapter vendor requirement service was added. |
| 4 | 3.6.7 (1)<br>Fig. 3.25 | - b0: The logical 0 precedence in relation to the collision detection bit was deleted. |
| 5 | 3.6.7 (2) | - Typographical errors relating to range and number of types of the requirement service code were corrected. |
| 6 | 3.6.7 (2) | - The phrase “(processing at time of service reception is mandatory)” was added to the end of the sentence “Shaded portions of four high-order service code bits of 0 to 3 must be mounted.” |
| 7 | 3.6.7 (2) | - Explanation of service codes 0x11 through 0x14 was corrected. The phrase “reserved for future use” was corrected. |
| 8 | 3.6.8 (1) | - An incorrect fixed value for device_num in (Note) was corrected. |
| 9 | 3.6.8 (2) | - “Cold start” was corrected to “cold start (2).” |
| 10 | 3.6.8 (5) | - Explanation of SD (1) through SD (5) was added to the service request frame. An incorrect SC value was corrected. |
| 11 | 3.6.8 (9) | - Incorrect SD (3), SD (5) and SD (9) LF values were corrected. |
| 12 | 3.6.8 (19) | - “SD (1)” was deleted from the service request frame. |
| 13 | 3.6.8 (20) | - “Cold start” was corrected to “cold start (1).” |

•Version 3.00 Draft June 12th 2002 Open to consortium members

The following Table-of-Contents entries were revised:

| | Revised entry | Revision/addition |
|---|---|---|
| 1 | 3.6.7,<br>Table 3.7 | - Addition of the “request for the number of lower-layer communication software address table data pairs,” “request for a lower-layer communication software address table data pair,” ”master router notification” and ”hardware address request” services and the service codes for the respective processing responses. |
| 2 | 3.6.8 (23) | - Addition of the “request for the number of lower-layer communication software address table data pairs” service. |
| 3 | 3.6.8 (24) | - Addition of the “request for a lower-layer communication software address table data pair” service. |

| | | |
|---|---|---|
| 4 | 3.6.8 (25) | - Addition of the “master router notification” service. |
| 5 | 3.6.8 (26) | - Addition of the ”hardware address request” service. |

•Version 3.00 August 29th 2002 Open to consortium members

The following Table-of-Contents entries were revised:

| | Revised entry | Revision/addition |
|---|---|---|
| 1 | 3.6.8 | - Amendments to SD (3) LF, SD (5) LF and SD (9) LF. |
| 2 | 3.6.8 (19) | - Amendment to the processing result SHD. |

•Version 3.10 Draft November 8th Open to consortium members

The following Table-of-Contents entries were revised:

| | Revised entry | Revision/addition |
|---|---|---|
| 1 | 1.1 | - Addition of explanation of ECHONET MAC address server. |
| 2 | 1.6 | - Addition of a section for ECHONET MAC address server. |
| 3 | Chapter 7 | - Addition of a chapter for ECHONET MAC address server. |

•Version 3.10 December 18th 2002 Open to consortium members.

•Version 3.11 March 7th 2003 Open to consortium members

•Version 3.12 May 22nd 2003 Open to consortium members

The parts to which changes have been made are as follows:

| | Revised entry | Revision/addition |
|---|---|---|
| 1 | 3.6.5 | - Amendment to the title. |

•Version 3.21Draft October 17th 2003 Open to consortium members

The following Table-of-Contents entries were revised:

| | Revised entry | Revision/addition |
|---|---|---|
| 1 | 1.7 | - Addition of an overview of the NetID server communications equipment specifications. |
| 2 | 3.6.8 | - Correction of incorrect descriptions in (4), (5), (9), (24) and (25).<br>- Amendments to descriptions in (23) and (24).<br>- Addition of (27), (28) and (29). |
| 3 | Chapter 8 | - Addition of “NetID server.” |
| 4 | | |
| 5 | 3.6.7 | - The maximum SD sizes for the “request for a lower-layer communication software address table data pair” and “adapter vendor requirement” services were changed. |
| 6 | Chapter 8 | - Addition of ECHONET middleware adapter specifications. |

•Version 3.20 December 12th 2006 Open to the Public.

The following Table-of-Contents entries were revised:

| | Revised entry | Revision/addition |
|---|---|---|
| 1 | Chapter 8 | - Amendment to "ECHONET Middleware Adapter." |
| 2 | Chapter 9 | - Addition of "NetID Server." |
| 3 | 3.6.5<br>Fig. 3.18(a) | - data field → data section |
| 4 | 3.6.5<br>Fig. 3.18(a) | - data field → data section |
| 5 | 3.6.8 | - The unused areas for SD were changed to "Reserved for future use." |
| 6 | 3.6.8(2) | - Addition of explanation of "transmission buffer size," "receiving buffer size," "maximum retention time for transmission messages" and "maximum retention time for received messages." |
| 7 | 3.6.8(20) | - Addition of explanation of "transmission buffer size," "receiving buffer size," "maximum retention time for transmission messages" and "maximum retention time for received messages." |
| 8 | 3.6.8(27),(28) | - Correction of descriptions. |

•Version 3.30 December 2nd 2004 Open to consortium members.

The following Table-of-Contents entries were revised:

| | Revised entry | Revision/addition |
|---|---|---|
| 1 | 3.1 | • "9: IEEE802.11/11b" was added in Fig. 3.2. |
| 2 | 3.6.8 | • The values "0x91" to "0x9F" for IEEE802.11/11b were added to the explanation about device_id. |
| 3 | 7.3 | • IEEE802.11/11b-related explanations were added to the explanations about the MAC address server. |
| 4 | 7.3.1 | • IEEE802.11/11b-related explanations were added to the explanations about the mechanical and physical characteristics. |
| 5 | 7.8.8 | • Object generation type interface commands were added:<br>"Equipment enquiry request/response with object specified"<br>"Equipment status access request/response (all)"<br>"Equipment status access UP request/response (all)"<br>"Equipment status notification request/response (all)"<br>"Object access request/response (all)" |
| 6 | 7.8 | • Incorrect descriptions were corrected. |

•Version 3.40 Draft December 28th 2004 Open to consortium members.

The following Table-of-Contents entries were revised:

| | Revised entry | Revision/addition |
|---|---|---|
| 1 | 8.6.2 | - Explanations about 9P connectors for middleware adapters were added under the heading "(4) Connector shape" of "8.6.2 Mechanical and physical characteristics of ECHONET middleware adapter communication interfaces."<br>- Explanations about 9P connectors for middleware adapters and Fig. 8.8(b) were added under the heading "(5) Relationship between connector pins and signals" of "8.6.2 Mechanical and physical characteristics of ECHONET middleware adapter communication interfaces." |
| 2 | 3.1 | - Explanations about the Power Line Communication Protocol C and D Systems were added in Fig. 3.2. |
| 3 | 8.8.4.2 | - "0x11 － 0x1F Power Line Communication Protocol System" under the heading "FD (1) lower-layer communication software ID" was changed to "0x11 －0x1F Power Line Communication Protocol A and D Systems."<br>"0xA1 Power Line Communication Protocol C System" was added. |
| 4 | 3.6.8 | - "0x11 － 0x1F Power Line Communication Protocol System" in "3.6.8 Adapter communication interface service" was changed to "0x11 －0x1F Power Line Communication Protocol A and D Systems."<br>"0xA1 Power Line Communication Protocol C System" was added. |

• Version 3.40 February 3rd 2005 Open to consortium members.

The following Table-of-Contents entries were revised:

| | Revised entry | Revision/addition |
|---|---|---|
| 1 | 8.8.3.5, 8.8.5.3 | - ECHONET middleware adapter communication interface-related explanations were added in Subsection 8.8.3.5 ("Normal operation state") and to the explanations about the sequence in Subsection 8.8.5.3. |

• Version 3.41 May 11th 2005 Open to consortium members.

The following Table-of-Contents entries were revised:

| | Revised entry | Revision/addition |
|---|---|---|
| 1 | 8.6.2 | - Amendments were made with regard to unnecessary dimensional accuracy requirements in the figure (Fig. 8.7) showing the dimensional requirements for 9P connectors for middleware adapters. |

• Version 3.2 October 13th 2005 Open to the public.
• Version 3.42 October 27th 2005 Open to consortium members.
• Version 3.50 Draft August 3rd 2006 Open to consortium members.
• Version 3.50 September 20th 2006 Open to consortium members.

• Version 3.51 Draft February 2nd 2007 Open to consortium members.

The following Table-of-Contents entries were revised:

| | Revised entry | Revision/addition |
|---|---|---|
| 1 | 8.6.2 | The specifications for MA9 connectors (recommended connectors) were changed; The requirement was added that MA9 connectors be used for Power Feed Class 1 only, and the specifications for MA9B connectors were added. |
| 2 | 8.6.4 | "(4) Frame composition" was added to make the specifications clearer. As a result, the old heading numbers (4) to (9) were changed to (5) to (10). |
| 3 | 8.7.2 | The first byte of the device type code of FD(2) of the response command was changed from "class category" to "class group." The class group code was formally adopted in the ECHONET Specification. |
| 4 | 8.8.4.1 | Explanations about FD(2) of the request command were made clearer. |
| 5 | 8.8.4.4(1) | Incorrect descriptions about "Length" of the request command were corrected; the terms "referencing of the status" and "changing the status" were switched. |
| 6 | 8.8.4.4(3) | Explanation about "Length" of FD(0) of the request command was amended; the value "0x01" was changed to "0x03." |
| 7 | 8.9<br>8.9.1<br>8.9.2<br>8.9.3 | Explanations about the "Program Download Method" were added in "8.9 Communication Protocols for the Peer-to-Peer Type." |
| 8 | 8.10 | A new section entitled "(Recommended) Specifications for Interpreter Method-based Program Execution Environments for the 'Program Download Method' for the 'Peer-to-Peer Type' " was added. |
| 9 | Appendix 2 | A new section entitled "Example Interpreter Method Programs" was added. |

• Version 3.60 March 5th 2007 Open to consortium members.

The following Table-of-Contents entries were revised:

| | Revised entry | Revision/addition |
|---|---|---|
| 1 | 8.10.1 | Incorrect descriptions in the bottom part of "Fig. 8.34 Scope of Application" were corrected; the terms "ECHONET middleware adapter side" and "ECHONET-ready equipment side" were switched. |
| 2 | 8.10.2.1 | An incorrect description in line 3 of 8.10.2.1 was corrected; the phrase "introduced in the concept interpreter API of virtual objects called intermediate objects" was changed to "the concept of virtual objects called intermediate objects is introduced in the interpreter API." |

• Version 3.60 December 11th 2007 Open to the public.

The specifications published by the ECHONET Consortium are established without regard to industrial property rights (e.g., patent and utility model rights). In no event will the ECHONET Consortium be responsible for industrial property rights to the contents of its specifications.

The publisher of this specification is not authorized to license and/or exempt any third party from responsibility for JAVA, IrDA, Bluetooth or HBS.
A party who intends to use JAVA, IrDA, Bluetooth or HBS should take action in being licensed for above-mentioned specifications.

In no event will the publisher of this specification be liable to you for any damages arising out of use of this specification.

The original language of The ECHONET Specification is Japanese. The English version of the Specification was translated the Japanese version. Queries in the English version should be refereed to the Japanese version.

# Contents

# Chapter 1 Overview of ECHONET Communications Equipment Specifications

## 1.1 Basic Concept

Part 7 defines the specifications for ECHONET nodes, ECHONET equipment adapters, ECHONET gateways, ECHONET routers, ECHONET MAC address servers and ECHONET middleware adapters as communications equipment. Detailed requirements are also defined for the interfaces between ECHONET equipment and ECHONET middleware adapters and equipment adapters as well as their respective functions.

## 1.2 Overview of Communications Equipment Specifications for ECHONET Nodes

The term “ECHONET node” is a generic name representing a communication terminal that permits direct information exchange through the ECHONET network. It is used to indicate a communication terminal without identifying its functionality. The requirements for the ECHONET node are stated below:

- ECHONET lower-layer communication software
- ECHONET Communication Middleware

## 1.3 Overview of Communications Equipment Specifications for ECHONET Device Adapters

The ECHONET device adapter adds ECHONET node functionality to a device that cannot function as an ECHONET node by itself. In the ECHONET Standard, the ECHONET device adapter is defined as an adapter that connects a device without ECHONET lower-layer communication software and protocol difference absorption processing block to the ECHONET network. Therefore, the requirements for the ECHONET device adapter are as stated below:

- Single ECHONET lower-layer communication software
- Protocol difference absorption processing block
- Adapter communication software

The adapter communication software provides communications between an ECHONET device adapter and ECHONET device. Its specifications are stated in Chapter 3.

## 1.4 Overview of Communications Equipment Specifications for ECHONET Gateways

The term “ECHONET gateway” refers to an ECHONET node that is capable of connecting an ECHONET domain to an external network. More specifically, the ECHONET gateway is an ECHONET node that has essential functions of a gateway basic block as service middleware.

Note, however, that the ECHONET gateway does not have to be a dedicated ECHONET node having gateway functionality. An ECHONET node having various functions in addition to the gateway functionality can serve as an ECHONET gateway. When the ECHONET gateway is viewed as a communications device, it is not different from an ECHONET node. Therefore, no particular communications equipment specifications are established for the ECHONET gateway.

## 1.5 Overview of Communications Equipment Specifications for ECHONET Routers

The ECHONET router is an ECHONET node that (1) connects each subnet to be controlled as a range in which ECHONET lower-layer communication software guarantees the seamless unity of MAC addresses by the ECHONET Communication Middleware protocol and (2) performs data routing processing. Like the ECHONET gateway, the ECHONET router need not always be a dedicated ECHONET node; it may provide both this function and other functions.

Considering the ECHONET router as communications equipment, its requirements differ depending on the type of ECHONET lower-layer communication software.

In the case of power lines, low-power radios, extended HBS, and LonTalk®, requirement do not differ from those of ECHONET nodes. Accordingly, the communications equipment specifications of the ECHONET router are not provided separately, and the functional requirements for the ECHONET router are described in Chapter 5.

Considering the IrDA Control as ECHONET lower-layer communication software, the functional requirements native to IrDA Control must be satisfied for routing. Special functional requirements are described in Chapter 6.

## 1.6 Overview of Communications Equipment Specifications for ECHONET MAC Address Servers

“ECHONET MAC address server” is a generic name for communications terminals that centrally manage, using a “client-server” method, the MAC addresses of all nodes located in the ECHONET subnets to which they belong. An address management method is specified for each of the transmission media defined in Part 3. An ECHONET MAC address server does not have to be an ECHONET node (i.e. implementation of the ECHONET Communication Middleware is not required).

## 1.7 Overview of Communications Equipment Specifications for NetID Servers

A NetID server is an ECHONET node that distributes a NetID unique to the subnet. As in the case of ECHONET routers and ECHONET gateways, a NetID server does not have to be an ECHONET node dedicated to performing this function and may have other additional functions.

## 1.8 Overview of Communications Equipment Specifications for ECHONET Middleware Adapters

An ECHONET middleware adapter is a piece of equipment that cannot become an ECHONET node by itself but is supplemented with the ECHONET node functions.

For the purposes of this Specification, an “ECHONET middleware adapter” is defined as an adapter for pieces of equipment that do not have an ECHONET communications processing section, ECHONET lower-layer communication software program and protocol difference absorption processing section to connect them to the ECHONET network. Therefore, an ECHONET middleware adapter must meet the following requirements:

* ECHONET communications processing section
* ECHONET lower-layer communication software
* ECHONET protocol difference absorption processing section
* ECHONET middleware adapter communication software

ECHONET middleware adapter communication software is a software program for communication between an ECHONET middleware adapter and a piece of equipment (ECHONET-ready equipment). The specifications for this software are defined in Chapter 8.

# Chapter 2 ECHONET Nodes

## 2.1 Basic Concept

An ECHONET node is specified as a communication terminal that permits direct information exchange through the ECHONET network. For a device to be recognized as an ECHONET terminal, it must be an ECHONET node. Accordingly, the full ECHONET device, ECHONET gateway, and ECHONET router are each ECHONET nodes.

## 2.2 Function Definition

Mandatory functions of an ECHONET node are described below.

(1) Function to distinguish other ECHONET nodes from the self-node

This function distinguishes the self-node from other ECHONET nodes in the same domain and specifies the self-node on the ECHONET network. Therefore, it must be possible to specify the subnet to which the self-node belongs using a NetID, and to specify the self-node in the subnet to which it belongs using a NodeID.

(2) Input/output function with transmission media

This function inputs and outputs data via the transmission media. Therefore, one or more transceivers capable of handling the ECHONET lower-layer communication software are required.

(3) Data processing function

This function assembles and disassembles data in each ECHONET communications layer and inputs and outputs data between the layers.

Optional functions of an ECHONET node are described below.

(4) NetID server function

When the ECHONET network consists of multiple subnets, this function assigns a NetID to each subnet with any one ECHONET node as a master router and distributes this NetID to the ECHONET routers.

To implement this processing, the ECHONET node must satisfy the following two requirements.

- ECHONET lower-layer communication software
- ECHONET communication middleware

An overview is shown in Fig. 2.1. In the figure, the shaded portions are the requirements.

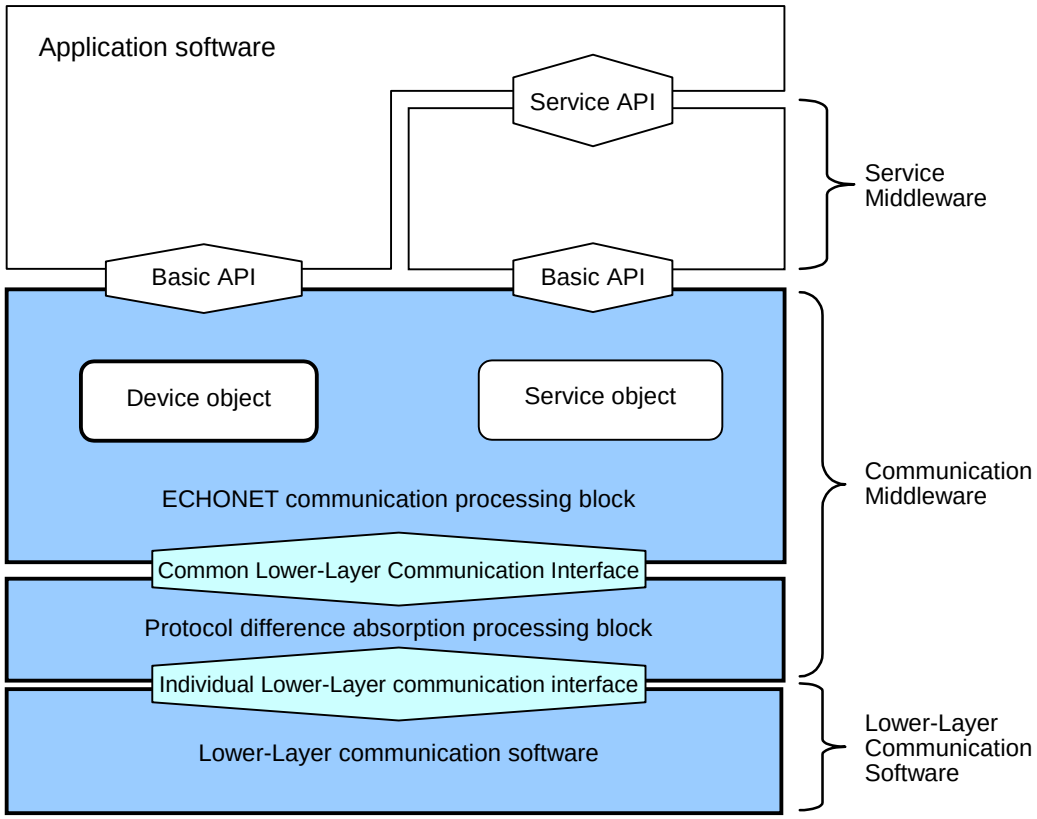



**Fig. 2.1 Requirements for ECHONET Nodes**

## 2.3 Mechanical and Physical Characteristics

Regarding connection to the transmission media, the specifications in Part 3 shall be observed in accordance with lower-layer communication protocols corresponding to the ECHONET node. Other mechanical and physical characteristics of an ECHONET node are specified below.

### 2.3.1 Shape

Regarding the shape, specifications shall be provided with the exception of the connection block to the transmission media of a radio system. The shape of this connection block shall conform to the specifications of the ECHONET lower-layer communication software to be used.

### 2.3.2 Display block

When an LED is equipped to display the operation status as ECHONET node communications equipment, the following minimum requirements must be satisfied. For display methods not specified here, the specifications native to the product shall be applicable.

- Number of LEDs

  1 (for operation status display)

  • LED color

  Green

  • Status display method

  Normal operation : ON

  Initial processing : Blink (long cycle)

  Error : Blink (short cycle)

  Non-operation : OFF

  * Long cycle ..... Repetition of ON for about 2 sec and OFF for about 0.5 sec

  * Short cycle ..... Repetition of ON for about 0.5 sec and OFF for about 0.5 sec

Note: Initial processing means a cold start (full reset start) and a warm start (hardware executes reset processing while retaining acquired addresses and initial setting information). The initial processing state includes the initialization process state, startup standby state, and suspension state.

## 2.4 NetID Server Function

When the employed configuration contains two or more subnets, the unique NetID server function exists within the domain. The ECHONET node having the NetID server function must have a switch that specifies whether or not to initiate the “Basic Sequence for Parent Router Startup” (Part 2, Section 5.4.1). However, this need not be a mechanical switch. Methods using keyboard instructions or icons will not be specified here.

## 2.5 ECHONET Nodes and Subnets

ECHONET nodes can belong to only one subnet. That is, a single ECHONET node can have only one ECHONET address. Accordingly, an ECHONET router consists of two or more ECHONET nodes.

## 2.6 ECHONET Nodes and Domains

ECHONET nodes cannot belong to multiple domains simultaneously. Accordingly, an ECHONET gateway for connecting two domains is defined as a device having two ECHONET nodes (see Fig. 2.2).

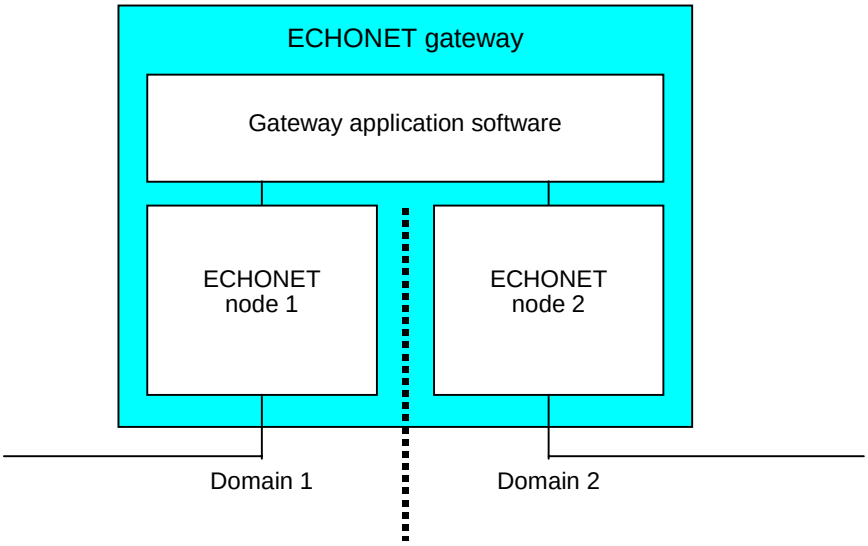



**Fig. 2.2 Relationship Between ECHONET Gateway and Nodes**

## 2.7 Limitation on Number of Connections

Assuming that an ECHONET node is considered as an address specification layer of Level 2 subsequent to the subnet, the following limitation is applied:

- The maximum number of ECHONET nodes that can exist in a single subnet is 256. Accordingly, the maximum value of ECHONET nodes existing within a domain is represented by:

  $n \times m$

  n : Maximum number of subnets in a domain 255

  m : Maximum number of ECHONET nodes in a subnet 256

  $255 \times 256 = 65{,}280$

However, this value is only a theoretical maximum and may be made smaller by limitations of the ECHONET lower-layer communication software, etc.

# Chapter 3 ECHONET Device Adapters

## 3.1 Basic Concept

In ECHONET, there are two ways to provide a device with ECHONET node functions:

(1) Accommodating the requirements for an ECHONET node in the device itself.
(2) Adding an adapter to a device that is not provided with ECHONET node functions, thereby satisfying the requirements for the ECHONET node.

Method (2) can be further divided into two methods based on the interface between the adapter and device:

(2-1) Using an adapter with an interface specified as the ECHONET standard.
(2-2) Using an adapter with a unique interface that is not specified in the ECHONET standard.

For the purposes of the ECHONET Specification, an “ECHONET equipment adapter” is defined as an adapter for a piece of equipment that falls under (2-1) above and does not have an ECHONET lower-layer communication software program and protocol difference absorption processing section to connect it to the ECHONET network. Only one ECHONET lower-layer communication software program can be implemented in an ECHONET equipment adapter.

An ECHONET equipment adapter is an adapter for a piece of equipment that does not have an ECHONET lower-layer communication software program and protocol difference absorption processing section (flex ECHONET equipment) to connect it to the ECHONET network, as shown in Fig. 3.1.

The specifications of the interface between a device and an ECHONET device adapter stipulate four types: a low-speed system (Type 1: 600 bps) and three high-speed systems (Type 2, Type 3, Type 4: 9600 bps). Furthermore, these types are identified by the ECHONET lower-layer communication software corresponding to the ECHONET device adapter. Accordingly, the ECHONET device adapter is indicated as shown in Fig. 3.2 by the corresponding ECHONET lower-layer communication software and transmission rate, and this shall be indicated on the ECHONET device adapter as the specification. The following are examples:

(Example 1) When the interface is of Type 2 and the ECHONET lower-layer communication software corresponds to LonTalk®: EAT6-2

(Example 2) When the interface is of Type 2 and the ECHONET lower-layer communication software is an extended HBS: EAT4-4

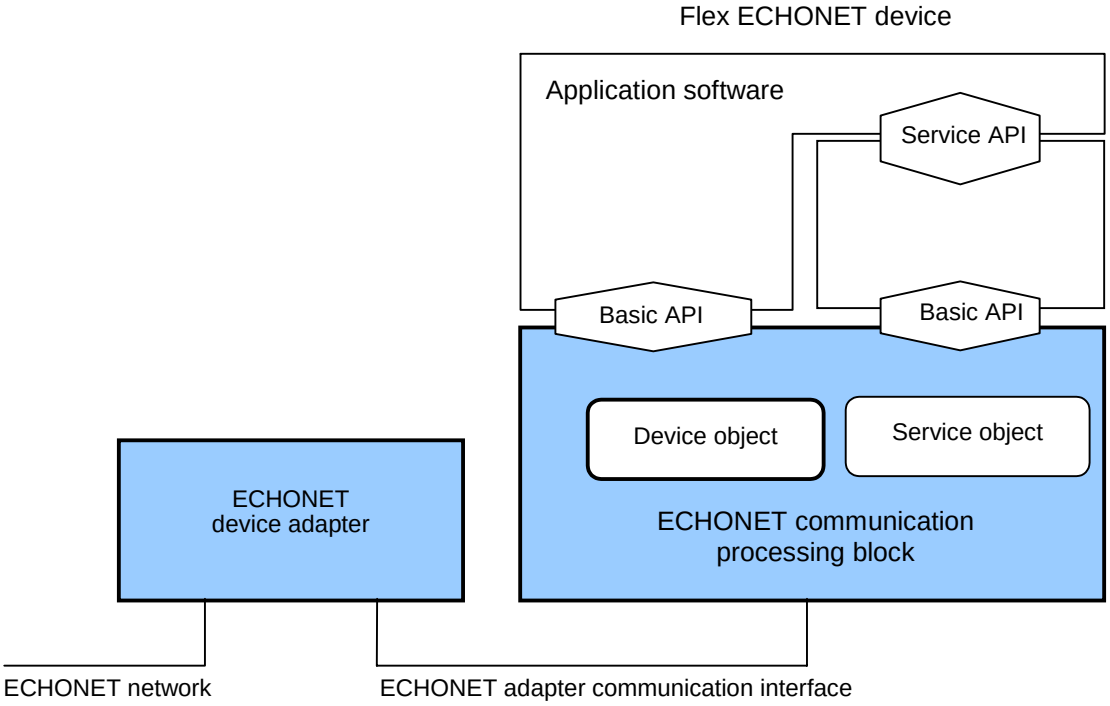



**Fig. 3.1 ECHONET Adapter and Flex ECHONET Device**

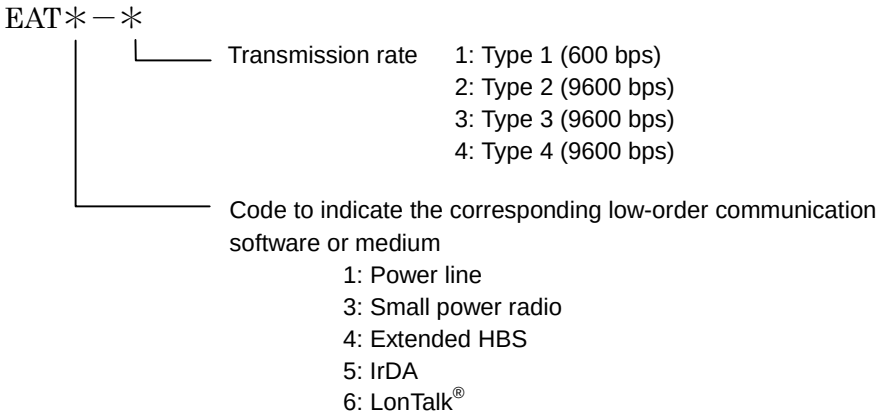



**Fig. 3.2 Type Indication of ECHONET Device Adapters**

## 3.2 Function Definition

The functions required for the ECHONET device adapter are specified as follows:

(1) Input/output function with transmission media

This function inputs/outputs data via the transmission media in accordance with the lower-layer communication protocol specifications provided in Part 3. This function is executed by the ECHONET lower-layer communication software. That is, a single transceiver that can handle the ECHONET lower-layer communication protocol is required as a function.

(2) Protocol difference absorption function

This function performs the processing specified in Part 2, Chapter 7 “Protocol Difference Absorption Processing Block Processing Specifications”, for carrying out mutual translation between the ECHONET lower-layer communication software and the ECHONET communications processing block protocol. This function is executed by the protocol difference absorption processing block.

(3) Adapter communication interface function

As specified in 3.6 “Adapter Communication Software” in this section, this function translates the ECHONET communications processing block protocol input from the protocol difference absorption processing block through the common lower-layer communication interface into an adapter communication interface protocol and then outputs it. In addition, this function outputs the input adapter communication interface protocol to an ECHONET communications processing block protocol and then outputs it to the common lower-layer communication interface. This function is processed by the adapter communication software.

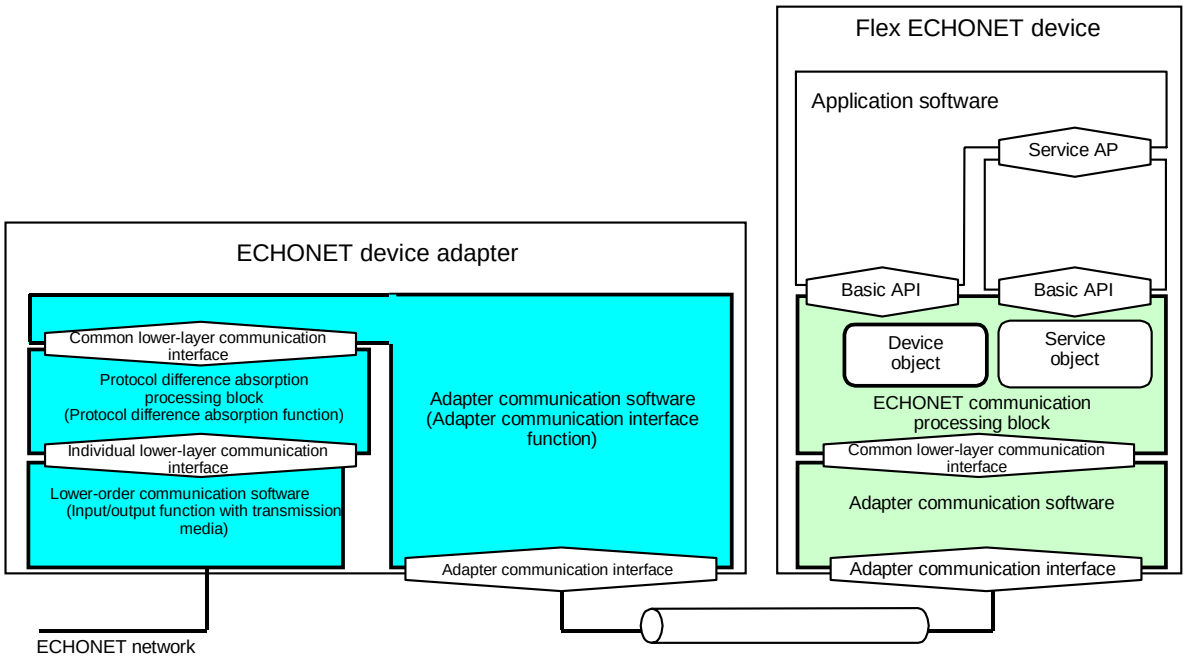



**Fig. 3.3 Functions of ECHONET Device Adapters**

## 3.3 Mechanical and Physical Characteristics

Regarding connection to the transmission media, the specifications in Part 3 shall be observed in accordance with the ECHONET lower-layer communication software corresponding to the ECHONET device adapter. Specifications for connection to ECHONET devices are provided in Chapter 6. Other mechanical and physical characteristics of ECHONET device adapters are specified below.

### 3.3.1 Shape

Shape shall not be specified with the exception of the connection block to the ECHONET device via the adapter communication interface. The shape of the connection block shall conform to the specifications of the ECHONET lower-layer communication software to be used.

### 3.3.2 Display block

When an LED is provided to display the operation status of the ECHONET device adapter, the following minimum specifications must be satisfied. For a display method not specified here, the specifications native to each product shall be applicable.

- Number of LEDs
  1 (for operation status display)
  - LED color

  Green
  - Status display method

  Normal operation : ON
  Initial processing : Blink (long cycle)
  Error : Blink (Short cycle)
  Non-operation : OFF
  * Long cycle ..... Repetition of ON for about 2 sec and OFF for about 0.5 sec
  * Short cycle ..... Repetition of ON for about 0.5 sec and OFF for about 0.5 sec

Note: Initial processing means a cold start (full reset start) and a warm start (hardware executes reset processing while retaining acquired addresses and initial setting information).

## 3.4 Electrical Characteristics

Regarding connection to the transmission media, the specifications in Part 3 shall be observed in accordance with the ECHONET lower-layer communication software corresponding to the ECHONET device adapter. Specifications for connection to ECHONET devices are provided in Chapter 6.

## 3.5 Logical Conditions

The logical conditions for adapter communication software are specified in Chapter 6. For the logical conditions related to the ECHONET lower-layer communication software and protocol difference absorption processing block, see Parts 3 and 2, respectively.

## 3.6 Adapter Communication Software

The adapter communication software is defined as follows:

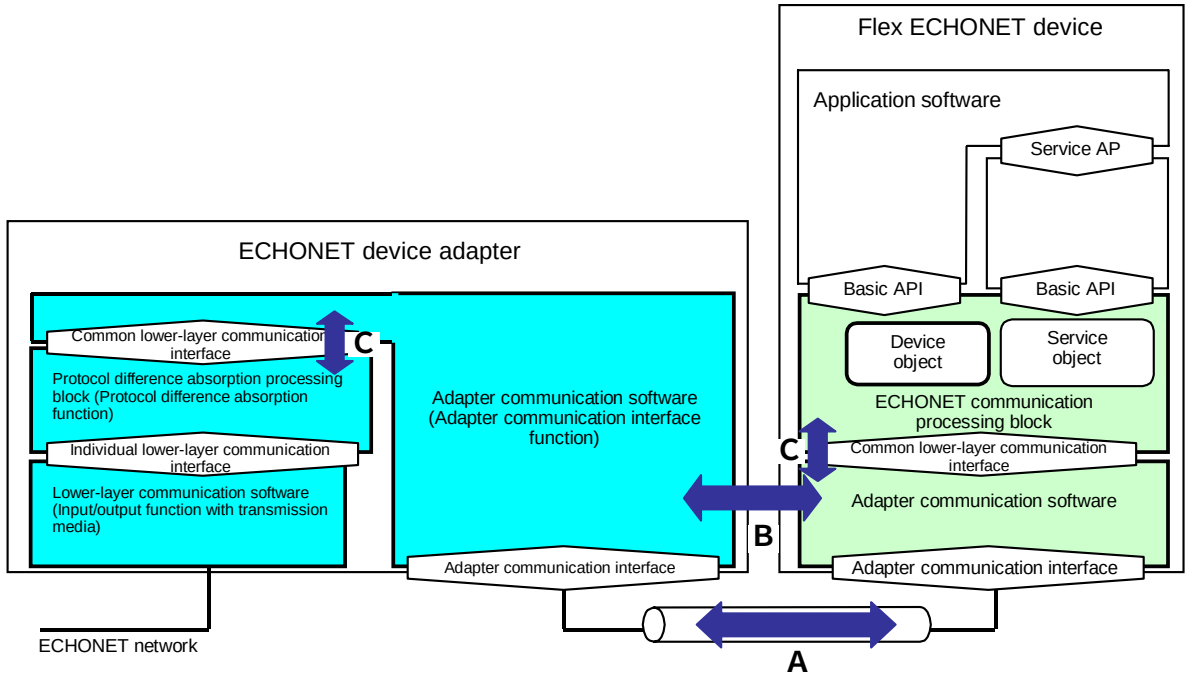
① This software runs on ECHONET device adapters and flex ECHONET devices.

② Differences in the common lower-layer communication interface implementation method between the ECHONET device adapter and the flex ECHONET device (e.g. APIs with different specifications) are absorbed.

The adapter communication software handles the adapter communication software protocol that is an intermediate step of the above translation. The adapter communication interface, adapter communication software protocol, and their handling are specified below.

### 3.6.1 Overview of adapter communication software

Figure 3.4 shows data exchange by the adapter communication software between the ECHONET device adapter and a device. Figure 3.5 shows the relationship between the common lower-layer interface service, adapter communication software protocol, and adapter communication interface protocol. In addition to services specified as common lower-layer communication interface services in this standard, the services originally specified by the adapter vendor shall also be handled. Distinctions between different services are made by the service header (SHD).




A: Adapter communication interface protocol
B: Adapter software protocol
C: Common low-order communication interface service

**Fig. 3.4 Data Exchange Between ECHONET Device Adapter and Device**

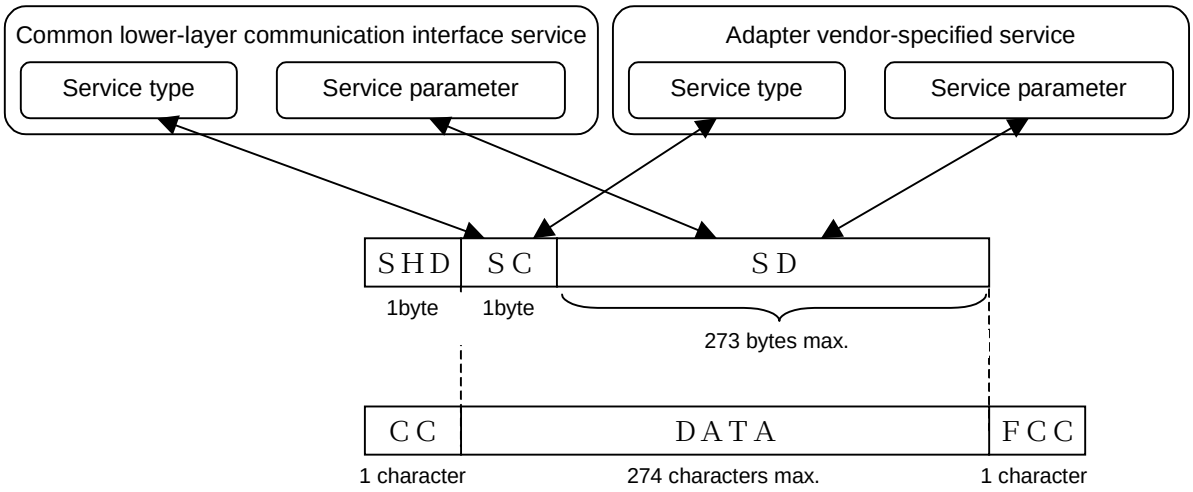



**Fig. 3.5 Relationship Between Services and Protocols**

## 3.6.2 Adapter communication interface

The adapter communication interface is positioned between the ECHONET device adapter and a device. The four items shown in the tables below are specified.

- Type 1 (Low-speed non-polarity type interface)
- Type 2 (High-speed interface)
- Type 3 (High-speed coaxial interface)
- Type 4 (RS-232C/UART interface)

Type 1 supports a transmission rate of 600 bps, and Types 2, 3, and 4 support a transmission rate of 9600 bps.

Type 2 permits a flex ECHONET device to supply power to an ECHONET adapter.

## 3.6.3 Mechanical and physical characteristics of adapter communication interface

The mechanical and physical characteristics of the adapter communication interface are stipulated below:

(1) Transmission media

Transmission media using the adapter communication interface are specified in Table 3.1.

**Table 3.1 Transmission Media of Adapter Communication Interface**

| | Medium | Number of cables | Specification |
|---|---|---|---|
| Type 1 | Twisted pair cable | 1 pair | Conductor diameter 0.65 mm |
| Type 2 | Twisted pair cable | 1 pair | Conductor diameter 0.65 mm |
| Type 3 | Coaxial cable | 1 cable | S-4C-FB, TVEFCX |
| Type 4 | Conforming to RS-232C | 5 cables | Conforming to RS-232C (RS-232C level) |
| | Multi-conductor cable | 6 cables | The conductor diameter is not stipulated (open collector). |

(2) Cable length

The maximum cable length that can be used for the transmission media is specified in Table 3.2.

**Table 3.2 Maximum Cable Length of Transmission Media**

| | Medium | Maximum cable length |
|---|---|---|
| Type 1 | Twisted pair cable | 30 m |
| Type 2 | Twisted pair cable | 200 m |
| Type 3 | Coaxial cable | 200 m |
| Type 4 | Medium conforming to RS-232C | 15 m |
| | Multi-conductor cable | 2 m |

(3) Connection form

For all types (1, 2, 3, and 4), one flex ECHONET device is used for one ECHONET device adapter per adapter communication interface. In other words, a one-to-one connection is used.

(4) Connector shape

① Connector on the flex ECHONET device side

The connector shape is stipulated in Table 3.3.

**Table 3.3 Connector on the Flex ECHONET Device Side**

| | Connector shape | |
|---|---|---|
| Type 1 | 2.5-mm 2-pin metric mutual connection system | |
| Type 2 | Modular type 6-split 2-pin connector | |
| Type 3 | Screw type RCA pin connector | |
| Type 4 | RS-232C level | 9-pin D-SUB male connector (recommended) |
| | Open collector | Original male connector (recommended) |

For Type 4 (RS-232C level), the use of a 9-pin D-SUB male connector is recommended, but not mandatory. The connector can also be switched to a 9-pin D-SUB male connector outside of a device. For Type 4 (open collector), the use of the original male connector shown in Fig. 3.11 is recommended. Connection examples are shown in Fig. 3.6 and Fig. 3.7.

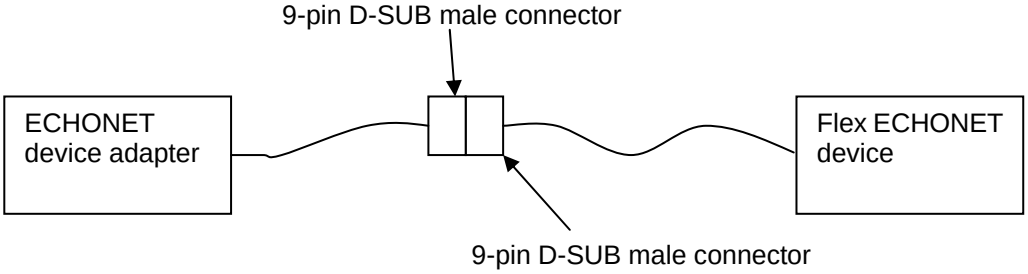
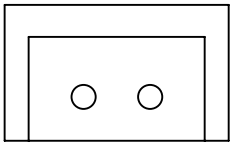



**Fig. 3.6 Device/Adapter Connection Example for Type 4 (RS-232C Level)**


**Fig. 3.7 Device/Adapter Connection Example for Type 4 (Open Collector)**

② Connector on the ECHONET device adapter side
No particular stipulations.

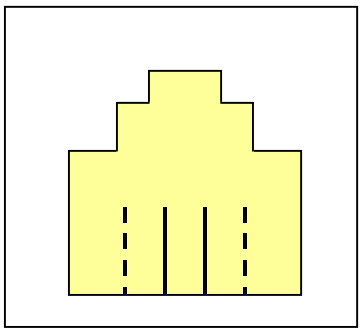
(5) Relationship between connectors and signals
Figures 3.8 and 3.9 show the modular connector's jack-end signal assignment and cable color scheme stipulated for the Type 1 interface and Type 2 interface, respectively. Figure 3.10 shows the Type 4 interface pin assignment for the RS-232C level. Figure 3.11 shows the Type 4 interface connector pin assignment for an open collector.



| Terminal name | D | D |
|---|---|---|
| Terminal No. | 2 | 1 |

D (Black): signal line

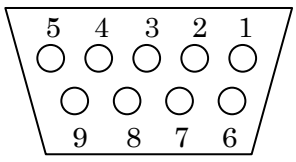
**Fig. 3.8 Type 1 Interface**



| Terminal name | D2 | D1 |
|---|---|---|
| Terminal No. | 2 | 1 |

D1 (Red) : signal and power supply (+)
D2 (Gray) : signal and power supply (–)

**Fig. 3.9 Type 2 Interface**




1: NC
2: RXD
3: TXD
4: NC
5: SG
6: NC
7: RTS
8: CTS
9: NC

**Fig. 3.10 Type 4 Interface (RS-232C Level)**

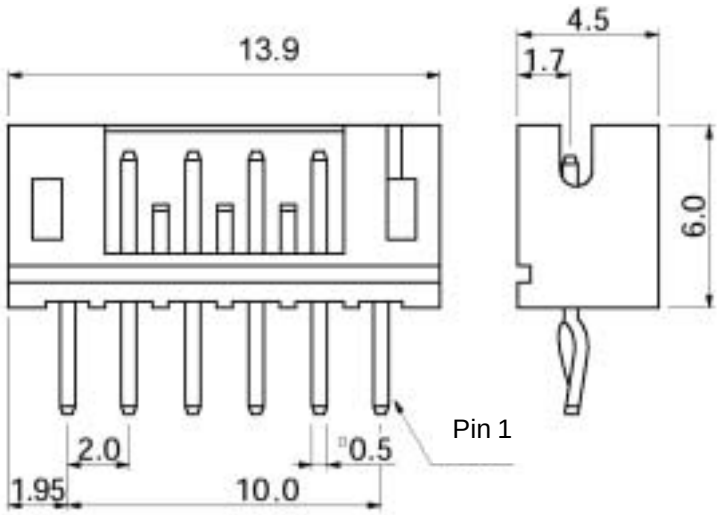



**Fig. 3.11 Type 4 Interface (Open Collector)** reference document (1)

For the Type 3 interface (screw-type RCA pin connector), the terminals are specified as follows:

D1 (+) : Inner conductor
D2 (–) : Outer conductor

(6) Cable used for Type 4 interface

A cross cable shall be used for Type 4. The signal correlations between a flex ECHONET device and ECHONET device adapter are indicated in Fig. 3.12.

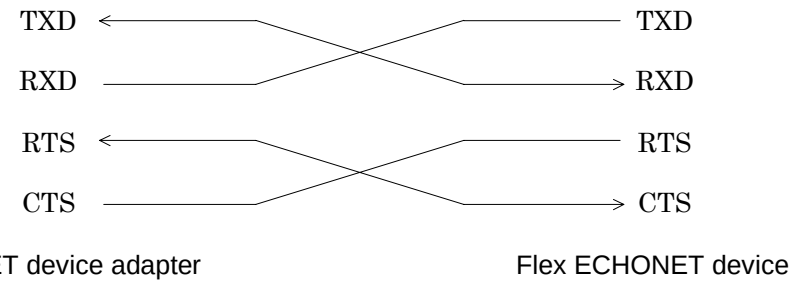



**Fig. 3.12 Type 4 Interface Signal Correlations**

## 3.6.4 Electrical characteristics of adapter communication interface

The electrical characteristics of the adapter communication interface are specified below.

(1) Characteristic cable impedance (nominal value)

The characteristic cable impedance is specified in Table 3.4.

**Table 3.4 Characteristic Cable Impedance**

| | Medium | Characteristic impedance (nominal value) |
|---|---|---|
| Type 1 | Twisted pair cable | 300Ω |
| Type 2 | Twisted pair cable | 300Ω |
| Type 3 | Coaxial cable | 75Ω |
| Type 4 | Conforming to RS-232C | Not specified |
| | Multi-conductor cable | Not specified |

(2) Load resistance (Type 2/Type 3 interface)

In Type 2 and Type 3 interfaces, a load resistance shall be provided on the ECHONET device adapter to suppress waveform distortion. In the Type 2 interface, a condenser must be connected in series with the load resistance to cut the direct current in consideration of the power feed (see Fig. 3.13). The load resistance value is shown below.

- Load resistance value R = 39Ω
- DC cut condenser C = 10 to 47 μF (Type 2 interface only)

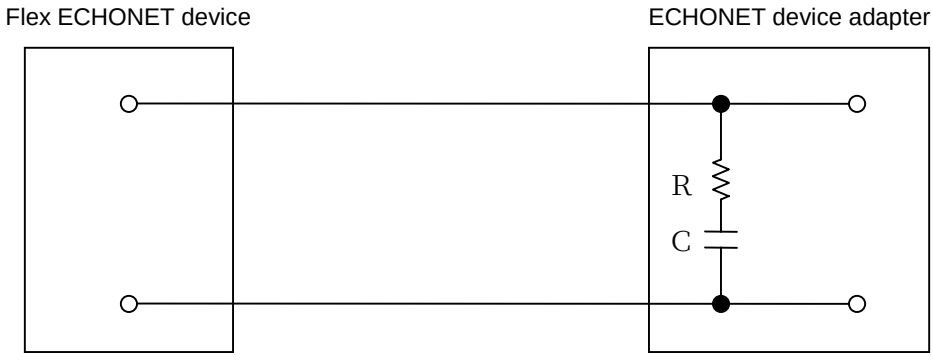



**Fig. 3.13 Load Resistance**

(3) Signal transmission rate

Signal transmission rates are specified as follows:

- Type 1 600 bps ± 1%
- Type 2 9600 bps ± 2.0%
- Type 3 9600 bps ± 2.0%
- Type 4 9600 bps ± 2.0%

(4) Signal transmission system and transmission waveform

The signal transmission system and transmission waveform are specified as follows:

◇ Type 1 and Type 4

① Transmission system : Base band transmission

② Transmission waveform : NRZ

③ Logic

Type 1 : Negative logic

Type 4 (RS-232C level)

Data signal line : RS-232C level negative logic, TTL level positive logic

Control line : RS-232C level positive logic, TTL level negative logic

Type 4 (open collector)

Data signal line : Negative logic, TTL level positive logic

Control line : Positive logic, TTL level negative logic

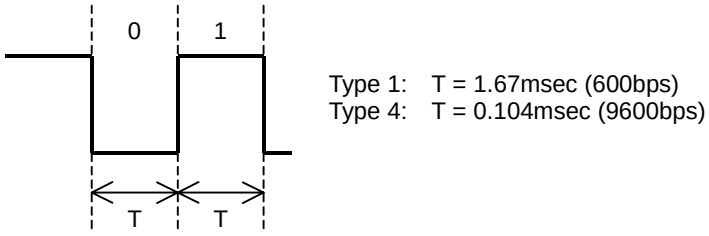



**Fig. 3.14 Signal Transmission System and Transmission Waveform of Type 1/Type 4 Adapter Communication Interface**

◇ Type 2 and Type 3

① Transmission system : Base band transmission

② Transmission waveform : AMI (Alternate Mark Inversion)
Duty ratio: 50% $^{+4\%}_{-2\%}$

③ Logic : Negative logic

④ The start bit is sent out from the positive (+) side.

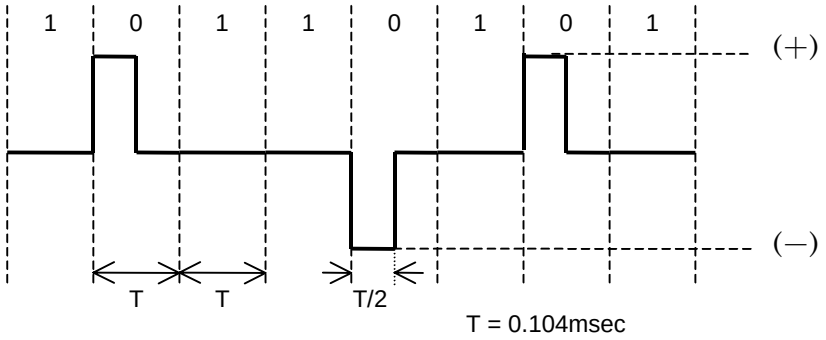



**Fig. 3.15 Signal Transmission System and Transmission Waveform of Type 2/Type 3 Adapter Communication Interface**

(5) Signal transmitting/receiving level

The electrical characteristics of the adapter communication interface are specified as follows:

◇ Type 1

① ECHONET device adapter receiving level

| | |
|---|---|
| Logic 0 transmission | : 4 to 5 V |
| Logic 1 transmission | : 2.5 to 3 V |
| Holding | : 2.5 to 3 V |

② Device receiving level

| | |
|---|---|
| Logic 0 transmission | : 1.5 V or less |
| Logic 1 transmission | : 2.5 to 3 V |
| Holding | : 2.5 to 3 V |

* Regarding the transmitting level, a value that satisfies the receiving level shall be specified in consideration of transmission loss, EMI, etc.

◇ Type 2 and Type 3

Receiving level

Logic 0 transmission : 1.4 V or less

Logic 1 transmission : 0.6 or less

Holding : 0.6 or less

(Common to both the ECHONET device adapter and device)

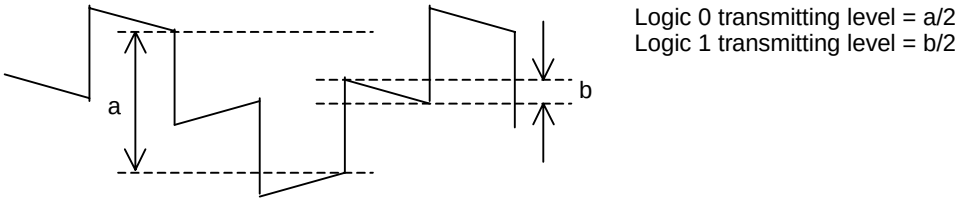



**Fig. 3.16 Signal Transmission System and Transmission Waveform of Type 2 Adapter Communication Interface**

* Regarding the transmitting level, a value that satisfies the receiving level shall be specified in consideration of transmission loss, EMI, etc.

◇ Type 4 (RS-232C level)

Data signal: RS-232C level negative logic, TTL level positive logic

Logic 0 (OFF) transmission : +5 V to +15 V

Logic 1 (ON) transmission : –5 V to –15 V

Control signal: RS-232C level positive logic, TTL level negative logic

Logic 0 (OFF) transmission : –5 V to –15 V

Logic 1 (ON) transmission : +5 V to +15 V

◇ Type 4 (open collector)

Open collector

Data signal: Negative logic, TTL level positive logic

Control signal: Positive logic, TTL level negative logic

(6) Power feed (Type 2/Type 4 (open collector) interface only)

For the Type 2 interface, a DC current fed to the signal line is permitted. The power feed method is specified in Table 3.5.

**Table 3.5 Power Feed Method for Type 2 Interface**

| | |
|---|---|
| Power feed voltage | 10 ±1 V |
| Receiving power voltage | 7 to 10 V |
| Receiving power capacity | 50 mA or less |

Furthermore, the Type 2 interface power feed method must satisfy the following requirements:

- The flex ECHONET device shall be a power feeder, and the ECHONET device adapter shall be a power receiver.
- A protective circuit shall be installed on the power feeder side.
- A reverse-current prevention device shall be installed in the power feeder.

The Type 4 (open collector) interface permits the feed of DC power. The power feed method is stipulated as indicated in Table 3.6.

**Table 3.6 Type 4 (Open Collector) Interface Power Feed Method**

| | |
|---|---|
| Power feed voltage | 5 V |

Furthermore, Type 4 (open collector) must satisfy the following requirements:

- A flex ECHONET device must be used as a power feeder. An ECHONET device adapter must be used as a power receiver.

## 3.6.5 Logical conditions for adapter communication interfaces

This section defines the logical conditions for adapter communication interfaces.

(1) Control method

◇ Type 1, Type 2 and Type 3

CSMA/CD (Carrier Sense Multiple Access with Collision Detection) method on the winner-first basis with priority given to the equipment side.

◇ Type 4

No control method is specified, but the following procedure shall be used when RTS/CTS is used.

① Prior to transmission, check that the CTS is OFF.

② When the CTS is OFF, turn ON the RTS, and notify the remote station of a request for transmission.
③ Output data to the TXD.
④ After completion of output, turn OFF the RTS.

Note: Note that the above procedure differs from the procedure for RS-232C RTS/CTS control.

Furthermore, provision must be made so that the device adapter's CTS input is normally OFF in consideration of situations where the flex device does not use RTS/CTS. The one that issued the service request first has precedence. However, in the event of a service request collision, the device adapter's service request has precedence.

(2) Synchronization method

The synchronization is specified as follows:

Synchronization method: Start-stop synchronization

① Character structure (common to Types 1 2, 3 and 4)
Start bit : 1 bit
Data : 8 bits
Parity : 1 bit
Stop bit : 1 bit (11 bits in total)

② Start bit transmitting polarity (Type 2 and Type 3 only)
The start bit transmitting polarity of the Type 2 adapter communication interface shall be the positive (+) side.

③ Data transmitting sequence (common to Types 1, 2, 3, and 4)
LSB first

④ Start bit (common to Types 1, 2, 3, and 4)
Logic 0

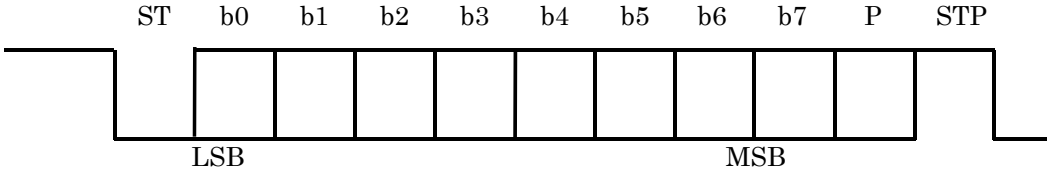
⑤ Stop bit (common to Types 1, 2, 3, and 4)
Logic 1

⑥ Parity (common to Types 1, 2, 3, and 4)
Even parity

⑦ Character spacing

- Common to Types 1, 2, and 3
  There must be no spacing interval between the stop bit and the next character.
- Type 4
  The spacing interval between the stop bit and the next character must not exceed 10 msec.

[Character structure of Type 1/Type 4 adapter communication interface]




[Character structure of Type 2/Type 3 adapter communication interface]

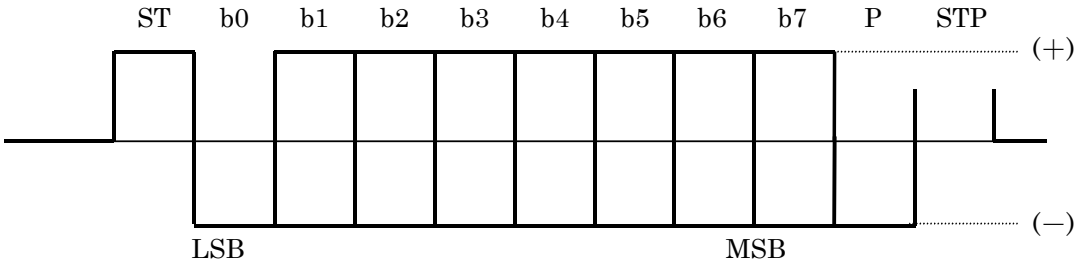
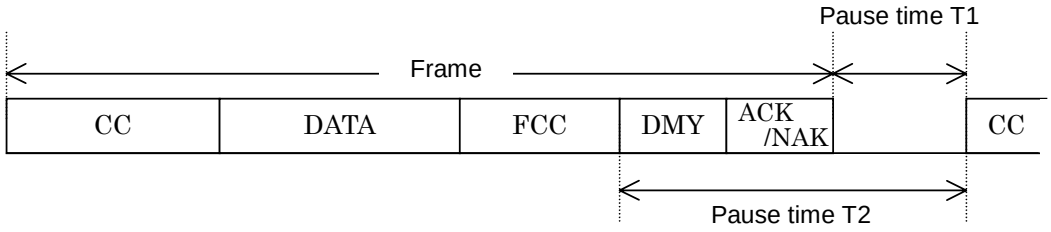



**Fig. 3.17 Character Structure of Adapter Communication Interface**

(3) Basic format of signal

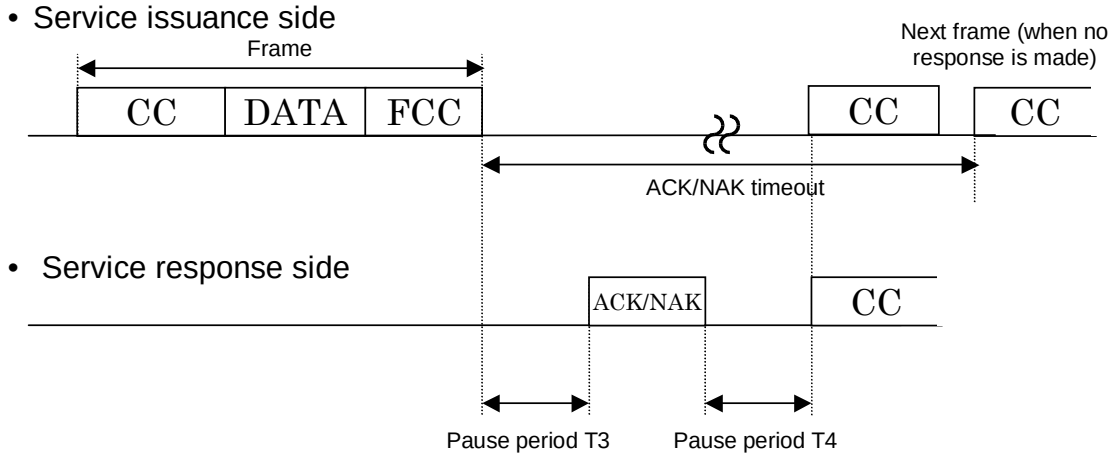
The basic signal format, which is common to Types 1, 2, and 3, is stipulated as indicated below:




CC : Control code
DATA : Data field (274 characters max.)
FCC : Check code
DMY : Dummy
ACK/NAK : ACK/NAK code

**Fig. 3.18 (a) Basic Signal Format**

For Type 4, the basic signal format is stipulated as indicated below:




CC : Control code
DATA : Data field (274 characters max.)
FCC : Check code
ACK/NAK : ACK/NAK code

**Fig. 3.18 (b) Basic Signal Format (Type 4)**

(4) Pause time and pause period

The pause time and pause periods for Types 1, 2, 3, and 4 are stipulated as indicated below:

① Pause time T1

◇ Type 1

Time equivalent to 11 bits from the end of the ACK/NAK stop bit

18.37 msec (600 bps)

◇ Type 2, Type 3, and Type 4

10 msec from the end of ACK/NAK stop bit

② Pause period T2

◇ Type 2 and Type 3

Time equivalent to 22 bits plus 10 msec from the end of the FCC stop bit

12.29 msec

③ Pause period T3

◇ Type 4

Longer than 10 msec and shorter than 200 msec from the end of the FCC stop bit

(10 msec < T3 < 200 msec)

④ Pause period T4

◇ Type 4

Longer than 10 msec from the end of the ACK/NAK stop bit

(10 msec < T4)

(5) Control code (CC)

Control code bit assignments, which are common to Types 1, 2, 3, and 4, are stipulated as indicated below:

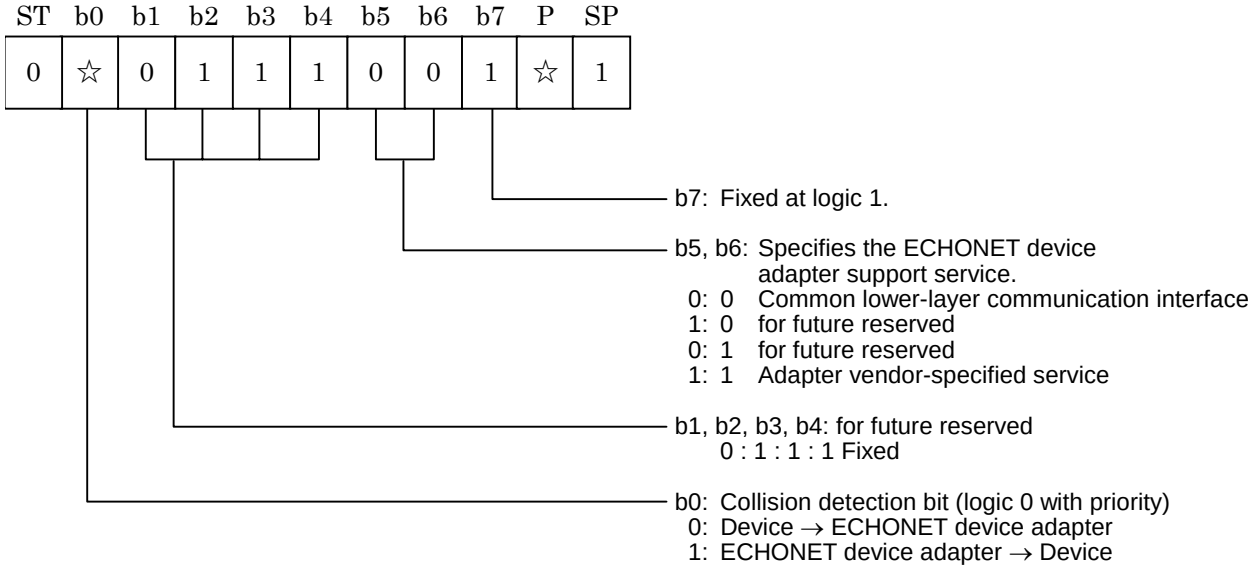



**Fig. 3.19 Control Code**

① Collision detection bit (b0)

For Types 1, 2, and 3, when packets are sent out from both an ECHONET device adapter and a device, the surviving packet is determined as follows:

- Collision detection bit value

  Device output packet : b0 = 0

  ECHONET device adapter output packet : b0 = 1

- Collision detection point

  Point at which a time of 1/4 bits has elapsed after transmission of the collision detection bit:

  0.418 msec later (600 bps)

  0.026 msec later (9600 bps)

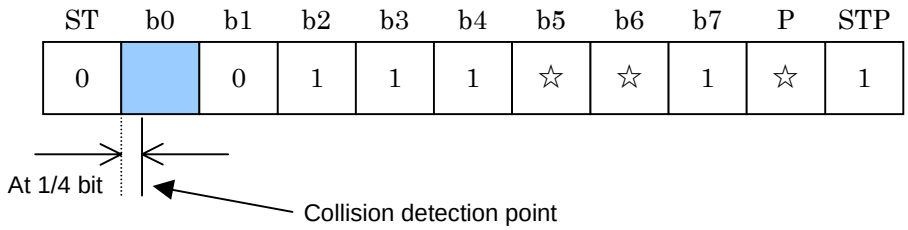



**Fig. 3.20 Collision Detection Point**

② ECHONET device adapter support specification bits (b5, b6)

These bits indicate the type of service to be carried by the packet. The following two services are specified:

- Common lower-layer interface service (b5 = 0, b6 = 0)

  Indicates that the packet is used to exchange the service specified in the common lower-layer communication interface between the flex ECHONET device and the ECHONET device adapter.

- Adapter vendor-specified service (b5 = 1, b6 = 1)

  Indicates that the packet is originally specified by the adapter vendor.

Regarding b5 = 0, b6 = 1 and b5 = 1, b6 = 0 “reserved for future use” is specified.

(6) Collision detection at contention (Type 1, Type 2, and Type 3)

When packets are sent out simultaneously from both the ECHONET device adapter and the device, resulting in contention, the surviving packet is determined by the following procedure for Types 1, 2, and 3.

① At the collision detection point, transmitted data and received data are collated.

② As a result of collation, when a mismatch is detected between the transmitted data and the received data, the transmission is stopped immediately and reception is started. Bit data of logic 0 has priority over bit data of logic 1. (The device side has priority.)

③ The loss-detected side (ECHONET device adapter side) stops transmission and starts reception. When transmission is enabled, this side starts the transmission.

(7) Data division (DATA)

The data division is a device adapter software protocol divided into byte units, to each of which are added a start bit, parity, and stop bit.

(8) Check code (FCC)

The check code shall be a 2's complement of the total character values existing in the data division for frame transmission error detection.

(9) Packet end detection

If a start bit is not detected after a stop bit is detected, it is considered to be the end of the packet.

(10) Dummy

A dummy is assigned as an error check calculation time, as shown below. During this period, neither packets nor characters exist.

◇ Type 1: Time equivalent to 2 bits
3.34 msec (600 bps)

◇ Type 2 and Type 3: Time equivalent to 11 bits
1.15 msec (9600 bps)

(11) ACK/NAK

When both the device and ECHONET device adapter receive a signal of 3 characters or more in normal operation mode, error detection is performed for the received signal frame. If the signal is correctly received, the ACK signal is returned. If the signal is not correctly received, the NAK signal is returned. For ACK/NAK, the following characters are assigned:

ACK : 0x06

NAK : 0x15

(12) Error detection and error control

Error detection and error control are provided as indicated below:

- Common to Types 1, 2, and 3
  - ① Error detection

    One bit is provided as the parity for each byte, and one byte is provided as the check code for the whole packet to increase the reliability of the received packet. Even parity shall be used.

  - ② Error control

    The error detection result is indicated by the ACK/NAK code subsequent to DMY. When there is no error, the ACK code is returned. When there is an error, the NAK code is returned.
    - Any code other than ACK/NAK is regarded as NAK. Non-response is regarded as NAK.
    - When the packet transmitting side receives NAK after DMY, it resends the packet after the pause time elapses. The maximum number of resend processing times shall be 3.

- Type 4
  - ① Error detection

    One bit is provided as the parity for each byte, and one byte is provided as the check code for the whole packet to increase the reliability of the received packet. Even parity shall be used.

② Error control

The error detection result is indicated by the ACK/NAK code. When there is no error, the ACK code is returned. When there is an error, the NAK code is returned.

- All codes other than ACK and NAK are regarded as NAK codes.
- When the packet transmitting side receives NAK, it retransmits the packet after the pause period T4 elapses. It also retransmits a packet if the no-response time exceeds the ACK/NAK timeout time as shown in Fig. 3.18 (b). The ACK/NAK timeout time shall be calculated from the timeout time stipulated in Section 3.6.15.
- The maximum permissible number of retransmissions shall be 3.

## 3.6.6 Adapter communication interface circuit (reference circuit)

Figures 3.21, 3.22 and 3.23 show examples of a reference circuit to implement the adapter communication interface.

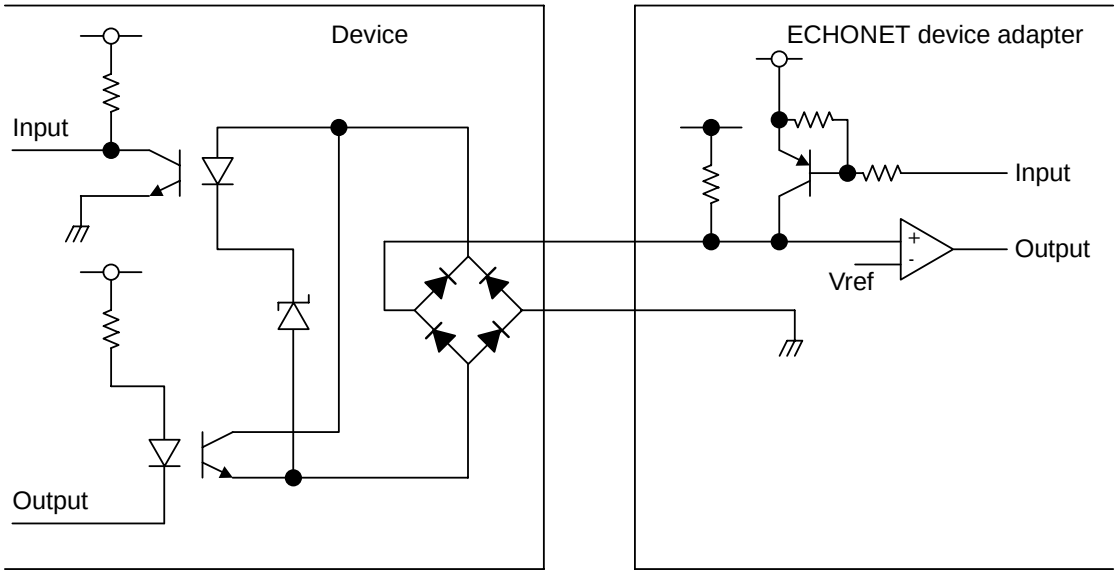
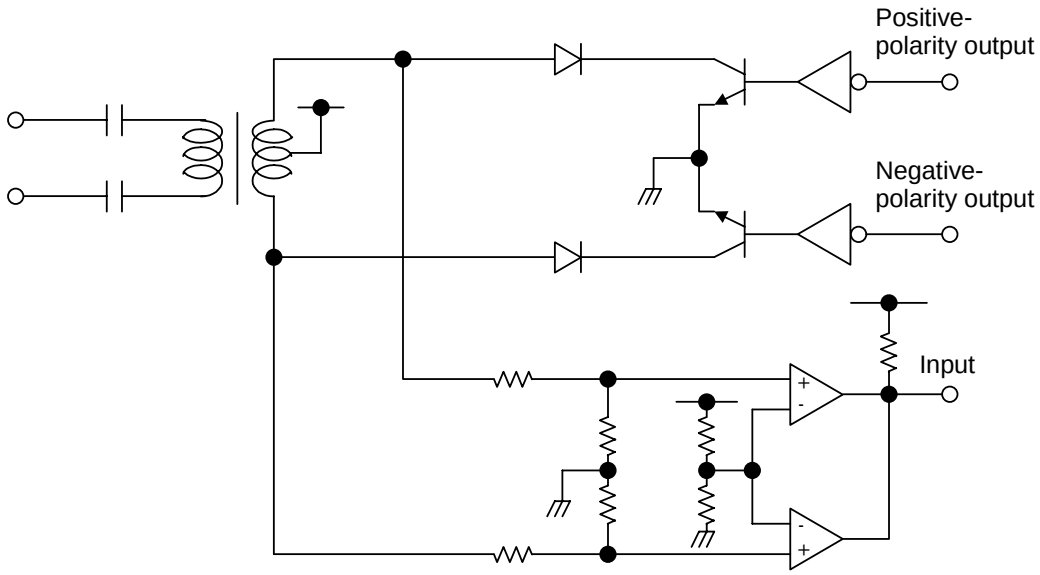



**Fig. 3.21 Example of Type 1 Adapter Communication Interface Circuit**




* Common circuit for both device and ECHONET device adapter

**Fig. 3.22 Type 2/Type 3 Adapter Communication Interface Reference Circuit**

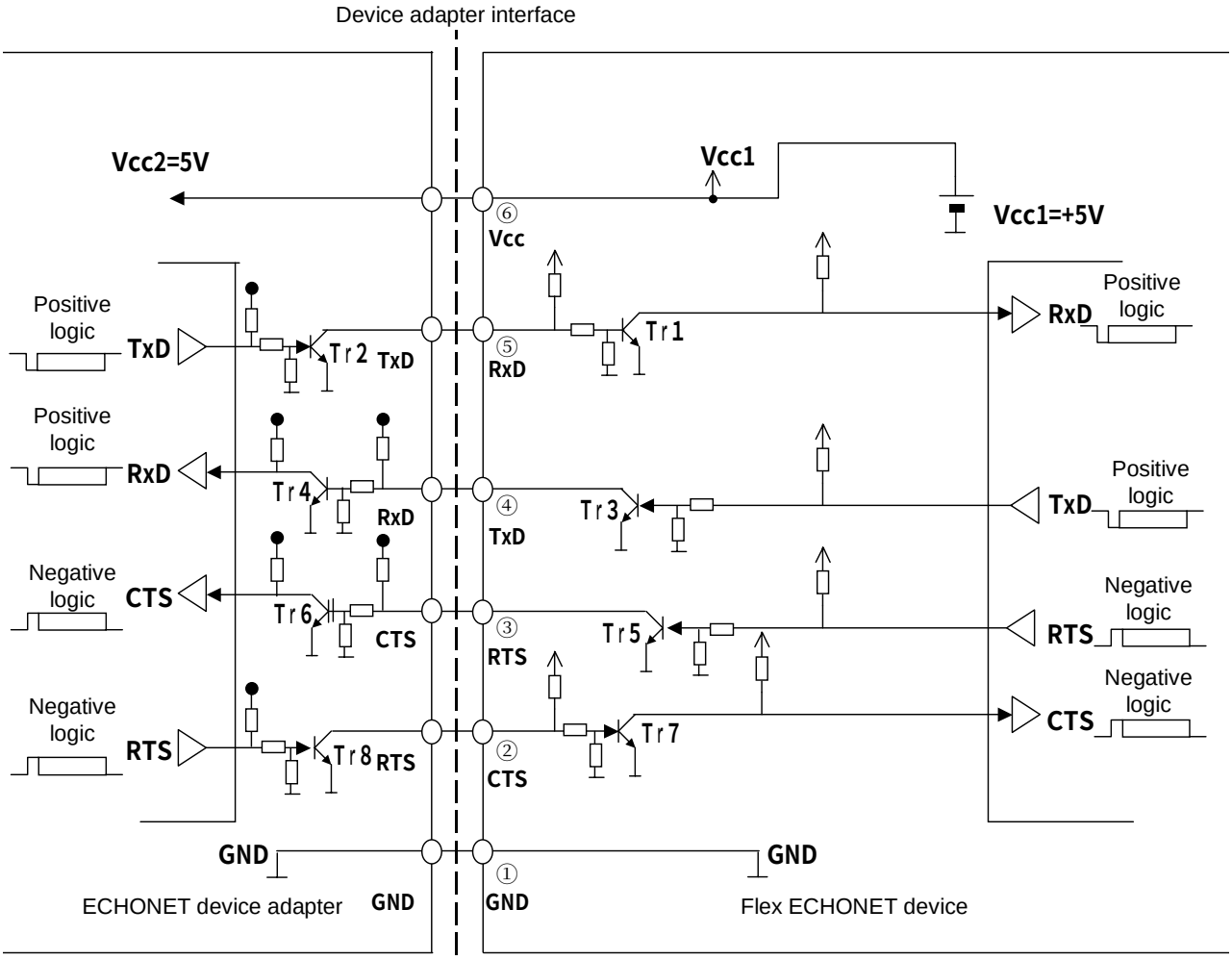



**Fig. 3.23 Reference Circuit for Type 4 Adapter (Open Collector) Interface**

## 3.6.7 Adapter communication software protocol

The adapter communication software protocol configuration is specified in Fig. 3.24. The service code (SC) and service data (SD) comprise DATA of the adapter communication interface protocol, and the service header (SHD) becomes CC in the adapter communication interface protocol.

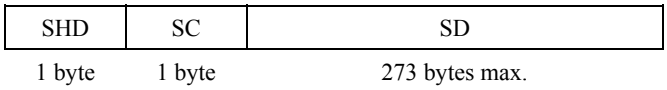



**Fig. 3.24 Adapter Communication Software Protocol**

(1) Service header (SHD)

The service header indicates whether or not the type of service code (SC) is based on the adapter communication interface service and shows the direction of communication. It is stipulated as indicated in Fig. 3.25. When both b5 and b6 are 1, the adapter vendor may specify its own service code.

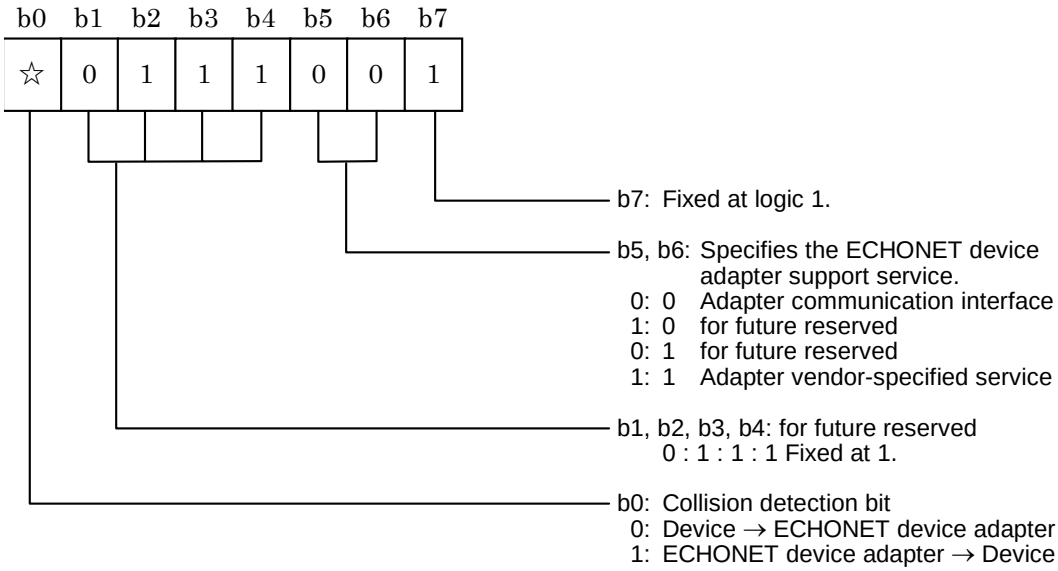



**Fig. 3.25 Service Header**

(2) Service code (SC)

Service code is a 1-byte code that specifies a service (prototype) defined for the adapter vendor or the adapter communication interface. In this Specification, 25 service codes (0x00 to 0x18) are defined for adapter communication interface services as shown in Table 3.7. Service codes 0x20 to 0x38 are used to return the result responses to the 0x00 to 0x18 services. Service code 0x3f is a response for a service that cannot be processed. An optional adapter communication interface service may not always be supported by the processing side even when it is supported by the service requesting source, whereupon it is notified by a non-processable service response (0x3f). Shaded portions of four high-order service code bits of 0 to 3 must be mounted (processing at time of service reception is mandatory). Four high-order bits of 0 to 3 having no service code assignments can be reserved for future use. Furthermore, four high-order bits of 4 to F are reserved for future use.

## Table 3.7 Service Codes for Adapter Communication Interface Service (1/2)

| 4 lower-order bits \ 4 high-order bits | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | Request for lower-layer communication software mounting information | Request for NodeID setting | "Lower-layer communication software mounting information results" processing response | "NodeID setting" processing response | | | | | | | | | | | | |
| 1 | Request for initialization | Stop notice | "Initialization" processing response | "Stop notice" processing response | | | | | | | | | | | | |
| 2 | Request for operation start | Request for complete initialization | "Operation start" processing response | "Complete initialization" processing response | | | | | | | | | | | | |
| 3 | Fault notice | Request for communication stop | "Fault notice" processing response | "Communication stop" processing response | | | | | | | | | | | | |
| 4 | Request for warm start | Request for complete stop | "Warm start" processing response | "Complete stop" processing response | | | | | | | | | | | | |
| 5 | Suspension request | Request for the number of lower-layer communication software address table data pairs | "Suspension" processing response | "Request for the number of lower-layer communication software address table data pairs" processing response | | | Reserved for future use | | | | | | | | | |
| 6 | Operation restart instruction | Request for a lower-layer communication software address table data pair | "Operation restart" processing response | "Request for a lower-layer communication software address table data pair" processing response | | | | | | | | | | | | |
| 7 | Protocol difference absorption processing section profile acquisition | Master router notification | "Protocol difference absorption processing section profile acquisition" processing response | "Master router notification" processing response | | | | | | | | | | | | |
| 8 | Lower-layer communication software profile acquisition | Hardware address data request | "Lower-layer communication software profile acquisition" processing response | "Hardware address data request" processing response | | | | | | | | | | | | |
| 9 | Protocol difference absorption processing section status acquisition | NodeID list acquisition request | "Protocol difference absorption processing section status acquisition" processing response | "NodeID list acquisition request" processing response | | | | | | | | | | | | |

**Table 3.7 Service Codes for Adapter Communication Interface Service (2/2)**

| | | 4 high-order bits | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
| | A | Lower-layer communication software status acquisition | Master router list acquisition request | "Lower-layer communication software status acquisition" processing response | "Master router list acquisition" processing response | | | | | | | | | | | | |
| | B | Message transmission request | Hardware address conversion request | "Message transmission" processing response | "Hardware address conversion" processing response | | | | | | | | | | | | |
| | C | Transmission result acquisition | | "Transmission result acquisition" processing response | | | | | | | | | | | | | |
| | D | Transmission cancellation request | | Transmission cancellation processing response | | | | | | | | | | | | | |
| | E | Request for received message | | Received message processing response | | | | | | | | | | | | | |
| | F | NodeID acquisition request | | "NodeID acquisition request" processing response | "Unsupported service" notification | | | | | | | | | | | | |

(3) Service data (SD)

Service data is for the adapter communication interface or for services originally specified by the adapter vendor. For the adapter communication interface, the data corresponds to the argument of API. Each unit of service data consists of two fields, namely, a data length field (LF) and a data field (DF). The data length field is 1 byte long and indicates the number of bytes comprising the data field. The data field is for input/output data specified in the common lower-layer communication interface. A data array of 2 bytes or more is regarded as big-endian. Figure 3.26 shows the service data structure.

Regarding service data that is optional and not supported, the data field length value shall be 0x00, and the next service data length field shall continue immediately after it.

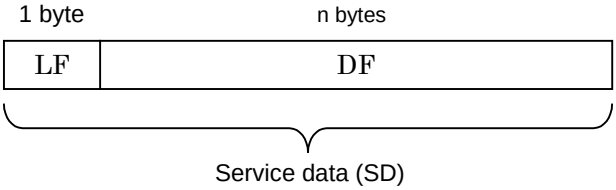



**Fig. 3.26 Service Data Format**

The service data corresponding to each service code is shown together with the adapter communication software protocol. Shaded service data shall be mandatory and must be mounted. In Fig. 3.26, the number of bytes is for cases in which service data is supported. Service data that is not supported has only 1 byte in the data length field and has no data field.

(4) Lower-layer communication software ID (device_id)

The lower-layer communication software ID (device_id) indicates the type of lower-layer communication software mounted in an ECHONET device adapter. Since the ECHONET device adapter can mount only one lower-layer communication software program, it has only one lower-layer communication software ID (device_id). A flex ECHONET device must use the “lower-layer communication software mounting information request service” to obtain a lower-layer communication software ID (device_id) from an ECHONET device adapter and use the obtained ID whenever the lower-layer communication software ID (device_id) is requested in relation to the subsequent use of various services.

## 3.6.8 Adapter communication interface services

This section details various adapter communication interface services.

(1) Lower-layer communication software mounting information request service (Required)

Acquires information about the lower-layer communication software mounted in an ECHONET device adapter (the number of mounted lower-layer communication software programs and their IDs).

◇ Service direction

Flex ECHONET device → ECHONET device adapter

◇ Service request

| 1 byte | 1 byte | 2 bytes |
|---|---|---|
| SHD | SC | SD(0) |

SHD ：0x9C
SC ：0x00
SD(0) ：LF 0x01 DF Dummy (0x88)

◇ Processing result

| 1 byte | 1 byte | 2 bytes | 2 bytes | n + 1 bytes |
|---|---|---|---|---|
| SHD | SC | SD(0) | SD(1) | SD(2) |

SHD ：0x9D
SC ：0x20
SD(0) ：LF 0x01 DF Processing result (0x00：TRUE, 0x01:FALSE)
SD(1) ：LF 0x01 DF Number of mounted Lower-layer communication software (device_num)
SD(2) ：LF n DF Lower-layer communication software ID (device_id)
※ n ＝device_num
device_id
0x11～0x1F Power line
0x31～0x3F Low power wireless
0x41～0x4F Extended HBS
0x51～0x5F IrDA_Control
0x61～0x6F LonTalk®

Note: Since the current version permits an ECHONET device adapter to mount only one lower-layer communication software program, device_num, which is the DF of SD(1), is fixed at 0x01.

(2) Initialization request service (Required)

Requests an ECHONET device adapter to cold start (2) the lower-layer communication software related to a specified lower-layer communication software ID, and place it in the communication stop state. In this case, the MAC address is acquired again.

◇ Service direction

Flex ECHONET device → ECHONET device adapter

◇ Service request

| 1 byte | 1 byte | 2 bytes | 3 bytes | 3 bytes | 3 bytes | 3 bytes | 2 bytes |
|---|---|---|---|---|---|---|---|
| SHD | SC | SD(0) | SD(1) | SD(2) | SD(3) | SD(4) | SD(5) |

SHD ：0x9C
SC ：0x01

| | | | | |
|---|---|---|---|---|
| SD(0) | ：LF | 0x01 | DF | Lower-layer communication software ID (device_id) |
| SD(1) | ：LF | 0x02 | DF | Transmitting buffer size (in bytes) |
| SD(2) | ：LF | 0x02 | DF | Receiving buffer size (in bytes) |
| SD(3) | ：LF | 0x02 | DF | Maximum retention time for transmission messages (msec.)<br>The maximum time for which a transmission message to be retained can be retained (ECHONET equipment adapter). Any transmission message that has been retained for longer than this time period may be discarded regardless of the result of the transmission processing. |
| SD(4) | ：LF | 0x02 | DF | Maximum retention time for received messages (msec.)<br>The maximum time for which a received message to be retained can be retained (ECHONET equipment adapter). Any received message that has been retained for longer than this time period may be discarded regardless of the result of the receiving processing. |
| SD(5) | ：LF | 0x01 | DF | Operation mode specification |

0x00: Normal operation
0x01: Test/maintenance mode

◇ Processing result

| 1 byte | 1 byte | 2 bytes |
|---|---|---|
| SHD | SC | SD(0) |

SHD ：0x9D
SC ：0x21

| | | | | |
|---|---|---|---|---|
| SD(0) | ：LF | 0x01 | DF | Processing result (0x00：TRUE, 0x01：FALSE) |

(3) Operation start request service (Required)

Requests an ECHONET device adapter to place the lower-layer communication software related to a specified lower-layer communication software ID in the normal operation state.

◇ Service direction

Flex ECHONET device → ECHONET device adapter

◇ Service request

| 1 byte | 1 byte | 2 bytes |
|---|---|---|
| SHD | SC | SD(0) |

SHD ：0x9C
SC ：0x02
SD(0) ：LF 0x01 DF Lower-layer communication software ID (device_id)

◇ Processing result

| 1 byte | 1 byte | 2 bytes |
|---|---|---|
| SHD | SC | SD(0) |

SHD ：0x9D
SC ：0x22
SD(0) ：LF 0x01 DF Processing result (0x00：TRUE, 0x01：FALSE)

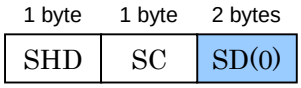
(4) Fault notice service (Optional)

Notifies a device adapter that a problem occurred or was cleared in a flex ECHONET device.

◇ Service direction

Flex ECHONET device → ECHONET device adapter

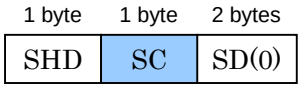
◇ Service request




SHD ：0x9C
SC ：0x03
SD(0) ：LF 0x01 DF Trouble No. (trouble_no)

0x01: An application software abnormality was found.
0x02: The ECHONET communications processing block was abnormal.
0xFF: A problem was cleared.

◇ Processing result




SHD ：0x9D
SC ：0x23
SD(0) ：LF 0x01 DF Processing result (0x00：TRUE, 0x01：FALSE)

(5) Warm start request service (Required)

Requests an ECHONET device adapter to warm start the lower-layer communication software related to a specified lower-layer communication software ID, and place it in the communication stop state. When this process is performed, the MAC address remains unchanged.

◇ Service direction

Flex ECHONET device → ECHONET device adapter

◇ Service request

| 1 byte | 1 byte | 2 bytes | 3 bytes | 3 bytes | 3 bytes | 3 bytes | 2 bytes |
|---|---|---|---|---|---|---|---|
| SHD | SC | SD(0) | SD(1) | SD(2) | SD(3) | SD(4) | SD(5) |

SHD ：0x9C
SC ：0x01

| | | | | |
|---|---|---|---|---|
| SD(0) | ：LF | 0x01 | DF | Lower-layer communication software ID (device_id) |
| SD(1) | ：LF | 0x02 | DF | Transmitting buffer size (in bytes) |
| SD(2) | ：LF | 0x02 | DF | Receiving buffer size (in bytes) |
| SD(3) | ：LF | 0x02 | DF | Maximum holding time for transmission data (msec) |
| SD(4) | ：LF | 0x02 | DF | Maximum holding time for received data (msec) |
| SD(5) | ：LF | 0x01 | DF | Operation mode specification |

0x00: Normal operation
0x01: Test/maintenance mode

◇ Processing result

| 1 byte | 1 byte | 2 bytes |
|---|---|---|
| SHD | SC | SD(0) |

SHD ：0x9D
SC ：0x24

| | | | | |
|---|---|---|---|---|
| SD(0) | ：LF | 0x01 | DF | Processing result (0x00：TRUE, 0x01：FALSE) |

(6) Suspension request service (Optional)

Requests a device adapter to place the lower-layer communication software related to a specified lower-layer communication software ID in the suspension state.

◇ Service direction

Flex ECHONET device → ECHONET device adapter

◇ Service request

| 1 byte | 1 byte | 2 bytes |
|---|---|---|
| SHD | SC | SD(0) |

SHD ：0x9C
SC ：0x05
SD(0) ：LF 0x01 DF Lower-layer communication software ID (device_id)

◇ Processing result

| 1 byte | 1 byte | 2 bytes |
|---|---|---|
| SHD | SC | SD(0) |

SHD ：0x9D
SC ：0x25
SD(0) ：LF 0x01 DF Processing result (0x00：TRUE, 0x01：FALSE)

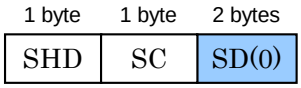
(7) Operation restart request service (Optional)

Requests an ECHONET device adapter to change the status of the lower-layer communication software related to a specified lower-layer communication software ID from the suspension state to the normal operation state.

◇ Service direction

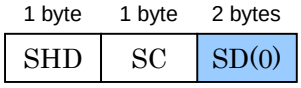
Flex ECHONET device → ECHONET device adapter

◇ Service request




SHD ：0x9C
SC ：0x06
SD(0) ：LF 0x01 DF Lower-layer communication software ID (device_id)

◇ Processing result




SHD ：0x9D
SC ：0x26
SD(0) ：LF 0x01 DF Processing result (0x00：TRUE, 0x01：FALSE)

(8) Protocol difference absorption processing block profile acquisition service (Required)

Requests an ECHONET device adapter to furnish the profile data on the protocol difference absorption processing block.

◇ Service direction

Flex ECHONET device → ECHONET device adapter

◇ Service request

| 1 byte | 1 byte | 2 bytes |
|---|---|---|
| SHD | SC | SD(0) |

SHD ：0x9C
SC ：0x07
SD(0) ：LF 0x01 DF Dummy (0x88)

◇ Processing result

| 1 byte | 1 byte | 2 bytes | 4 bytes | 4 bytes | 3 bytes | 3 bytes |
|---|---|---|---|---|---|---|
| SHD | SC | SD(0) | SD(1) | SD(2) | SD(3) | SD(4) |

SHD ：0x9D
SC ：0x27
SD(0) ：LF 0x01 DF Processing result (0x00：TRUE, 0x01：FALSE)
SD(1) ：LF 0x03 DF Version information about protocol difference absorption processing block
SD(2) ：LF 0x03 DF Manufacturer code
SD(3) ：LF 0x02 DF Number of transmittable data bytes
SD(4) ：LF 0x02 DF Number of receivable data bytes

(9) Lower-layer communication software profile acquisition service (Required)

Requests an ECHONET device adapter to furnish the profile data on the lower-layer communication software related to a specified lower-layer communication software ID.

◇ Service direction

Flex ECHONET device → ECHONET device adapter

◇ Service request

| 1 byte | 1 byte | 2 bytes |
|---|---|---|
| SHD | SC | SD(0) |

SHD ：0x9C
SC ：0x08
SD(0) ：LF 0x01 DF Lower-layer communication software ID (device_id)

◇ Processing result

| 1 byte | 1 byte | 2 bytes | 4 bytes | 4 bytes | 2 bytes | 8 bytes | 2 bytes |
|---|---|---|---|---|---|---|---|
| SHD | SC | SD(0) | SD(1) | SD(2) | SD(3) | SD(4) | SD(5) |

| 9 bytes | 3 bytes | 3 bytes | 2 bytes | 3 bytes |
|---|---|---|---|---|
| SD(6) | SD(7) | SD(8) | SD(9) | SD(10) |

SHD ：0x9D
SC ：0x28
SD(0) ：LF 0x01 DF Processing result (0x00：TRUE, 0x01：FALSE)
SD(1) ：LF 0x03 DF Lower-layer communication software version information
SD(2) ：LF 0x03 DF Manufacturer code
SD(3) ：LF 0x02 DF Mac address bit length
SD(4) ：LF 0x07 DF Mac address information (mac_address)
If the number of bits is less than 56, the data is entered in such a way that no space is left before the first bit and the remaining space after the last bit is padded with zeros
SD(5) ：LF 0x02 DF House code bit length
SD(6) ：LF 0x08 DF House code information
SD(7) ：LF 0x02 DF Number of transmittable data bytes
SD(8) ：LF 0x02 DF Number of receivable data bytes
SD(9) ：LF 0x02 DF Existence/non-existence of broadcast function
(0x00: Nonexistence, 0x01: Existence)
SD(10) ：LF 0x02 DF Transmission rate (bps)

(10) Protocol difference absorption processing block status acquisition service (Optional)

Requests an ECHONET device adapter to furnish the status information on the protocol difference absorption processing block.

◇ Service direction

Flex ECHONET device → ECHONET device adapter

◇ Service request

| 1 byte | 1 byte | 2 bytes |
|---|---|---|
| SHD | SC | SD(0) |

SHD ：0x9C
SC ：0x09
SD(0) ：LF 0x01 DF Dummy (0x88)

◇ Processing result

| 1 byte | 1 byte | 2 bytes | 2 bytes | 2 bytes | 2 bytes | 2 bytes |
|---|---|---|---|---|---|---|
| SHD | SC | SD(0) | SD(1) | SD(2) | SD(3) | SD(4) |

SHD ：0x9D
SC ：0x29
SD(0) ：LF 0x01 DF Processing result (0x00：TRUE, 0x01：FALSE)
SD(1) ：LF 0x01 DF Protocol difference absorption processing block transition state information (state)
0: Stop state
1: Initializing state
2: Normal operation state
3: Error stop state
SD(2) ：LF 0x01 DF Protocol high-order layer fault (upper_trouble)
SD(3) ：LF 0x01 DF Protocol difference absorption processing block fault (low_trouble)
SD(4) ：LF 0x01 DF Protocol difference absorption processing block operation mode (low_mode)
0: Normal operation mode
1: Test mode
2: Monitoring mode

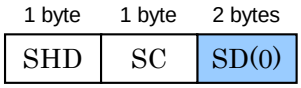
(11) Lower-layer communication software status acquisition service (Required)

Requests an ECHONET device adapter to furnish the status information on the lower-layer communication software related to a specified lower-layer communication software ID.

◇ Service direction

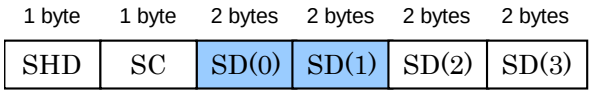
Flex ECHONET device → ECHONET device adapter

◇ Service request




SHD ：0x9C
SC ：0x0A
SD(0) ：LF 0x01 DF Lower-layer communication software ID (device_id)

◇ Processing result




SHD ：0x9D
SC ：0x2A
SD(0) ：LF 0x01 DF Processing result (0x00：TRUE, 0x01：FALSE)
SD(1) ：LF 0x01 DF Lower-layer communication software transition state information (state)
0: Stop state
1: Initializing state
2: Normal operation state
3: Error stop state
SD(2) ：LF 0x01 DF Lower-layer communication software fault (low_trouble)
SD(3) ：LF 0x01 DF Protocol difference absorption processing block operation mode (low_mode)
0: Normal operation mode
1: Test mode
2: Monitoring mode

(12) Data transmission request service (Required)

Requests an ECHONET device adapter to transmit data with the lower-layer communication software related to a specified lower-layer communication software ID.

◇ Service direction

Flex ECHONET device → ECHONET device adapter

◇ Service request

| 1 byte | 1 byte | 2 bytes | 2 bytes | 2 bytes | 3 bytes | |
|---|---|---|---|---|---|---|
| SHD | SC | SD(0) | SD(1) | SD(2) | SD(3) | SD(4) |

SHD ：0x9C
SC ：0x0B

| | | | | |
|---|---|---|---|---|
| SD(0) | ：LF | 0x01 | DF | Lower-layer communication software ID (device_id) |
| SD(1) | ：LF | 0x01 | DF | Transmitting destination NodeID information |
| SD(2) | ：LF | 0x01 | DF | Broadcast specification information<br>0x00: Does not specify a broadcast.<br>0xFF: Specifies a broadcast. |
| SD(3) | ：LF | 0x02 | DF | Number of transmittable data bytes (data_len) |
| SD(4) | ：LF | n | DF | Transmission data (send_data)<br>n = data_len (0xFF for 255 bytes or more) |

◇ Processing result

| 1 byte | 1 byte | 2 bytes |
|---|---|---|
| SHD | SC | SD(0) |

SHD ：0x9D
SC ：0x2B

| | | | | |
|---|---|---|---|---|
| SD(0) | ：LF | 0x01 | DF | Processing result<br>(0x00: Buffer full,<br>0x01: Transmission acceptable,<br>0x02: Buffer size error,<br>0x03: Low-order communication software error,<br>0x04: Processing failure.) |

(13) Transmission result acquisition service (Optional)

Requests an ECHONET device adapter to furnish the status information on a data transmission request process that was performed by the lower-layer communication software related to a lower-layer communication software ID specified immediately before this request.

◇ Service direction

Flex ECHONET device → ECHONET device adapter

◇ Service request

| 1 byte | 1 byte | 2 bytes |
|---|---|---|
| SHD | SC | SD(0) |

SHD ：0x9C
SC ：0x0D
SD(0) ：LF 0x01 DF Lower-layer communication software ID (device_id)

◇ Processing result

| 1 byte | 1 byte | 2 bytes | 2 bytes |
|---|---|---|---|
| SHD | SC | SD(0) | SD(1) |

SHD ：0x9D
SC ：0x2C
SD(0) ：LF 0x01 DF Processing result
(0x00: Transmission stop,
0x01: Normal,
0x02: Transmitting status,
0x03: Lower-layer communication software error)
SD(1) ：LF 0x01 DF Transmission result
(0x00: Success in transmission,
0x01: Failure in transmission,
0xFF: No response,
0x04: Processing failure.)

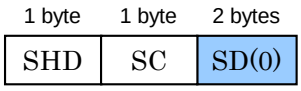
(14) Transmission stop request service (Optional)

Requests an ECHONET device adapter to stop the ongoing data transmission process performed by the lower-layer communication software related to a specified lower-layer communication software ID in compliance with a data transmission request that was issued immediately before this request.

◇ Service direction

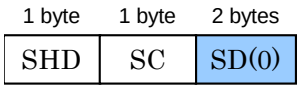
Flex ECHONET device → ECHONET device adapter

◇ Service request




SHD ：0x9C
SC ：0x0D
SD(0) ：LF 0x01 DF Lower-layer communication software ID (device_id)

◇ Processing result




SHD ：0x9D
SC ：0x2D
SD(0) ：LF 0x01 DF Processing result
(0x00: Termination of transmission,
0x01: Normal,
0x03: Lower-layer communication software error,
0x04: Processing failure.)

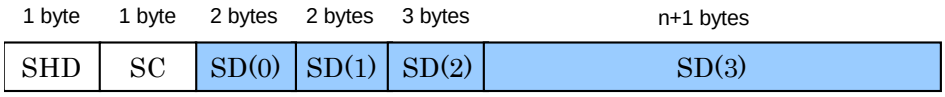
(15) Data reception request service (Required)

Notifies a flex ECHONET device of data received by the lower-layer communication software related to a specified lower-layer communication software ID.

◇ Service direction

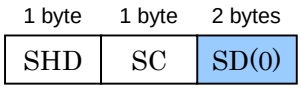
ECHONET device adapter→ Flex ECHONET device

◇ Service request




| | | | | |
|---|---|---|---|---|
| SHD | ：0x9D | | | |
| SC | ：0x0E | | | |
| SD(0) | ：LF | 0x01 | DF | Lower-layer communication software ID (device_id) |
| SD(1) | ：LF | 0x01 | DF | Transmitting source NodeID |
| SD(2) | ：LF | 0x02 | DF | Number of received data bytes (data_len) |
| SD(3) | ：LF | n | DF | Received data (receive_data)<br>n = data_len (0xFF for 255 bytes or more) |

◇ Processing result




| | | | | |
|---|---|---|---|---|
| SHD | ：0x9C | | | |
| SC | ：0x2E | | | |
| SD(0) | ：LF | 0x01 | DF | Processing result (0x00：TRUE, 0x01：FALSE) |

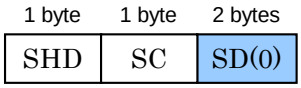
(16) NodeID acquisition service (Required)

Requests an ECHONET device adapter to furnish a node ID that is based on the MAC address currently retained by the lower-layer communication software related to a specified lower-layer communication software ID.

◇ Service direction

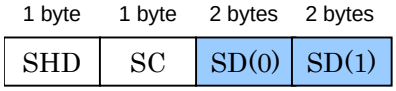
Flex ECHONET device → ECHONET device adapter

◇ Service request




SHD ：0x9C
SC ：0x0F
SD(0) ：LF 0x01 DF Lower-layer communication software ID (device_id)

◇ Processing result




SHD ：0x9C
SC ：0x2F
SD(0) ：LF 0x01 DF Processing result (0x00：TRUE, 0x01：FALSE)
SD(1) ：LF 0x01 DF NodeID

(17) Node ID setup request service (Optional)

Requests an ECHONET device adapter to set a node ID for the lower-layer communication software related to a specified lower-layer communication software ID.

◇ Service direction

Flex ECHONET device → ECHONET device adapter

◇ Service request

| 1 byte | 1 byte | 2 byte | 2 byte |
|---|---|---|---|
| SHD | SC | SD(0) | SD(1) |

SHD ：0x9C
SC ：0x10
SD(0) ：LF 0x01 DF Lower-layer communication software ID (device_id)
SD(1) ：LF 0x01 DF NodeID

◇ Processing result

| 1 byte | 1 byte | 2 bytes |
|---|---|---|
| SHD | SC | SD(0) |

SHD ：0x9D
SC ：0x30
SD(0) ：LF 0x01 DF Processing result (0x00：TRUE, 0x01：FALSE)

(18) Non-processable service notice (Required)

Sends notification of a non-processable service.

◇ Service direction

Flex ECHONET device ⇔ ECHONET device adapter

◇ Notice format

| 1 byte | 1 byte | 2 bytes |
|---|---|---|
| SHD | SC | SD(0) |

SHD ：0x9C or 0x9D
SC ：0x3F
SD(0) ：LF 0x01 DF Requested service code

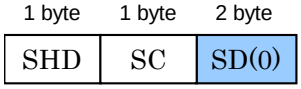
(19) Stop notice service (Required)

When the lower-layer communication software related to a specified lower-layer communication software ID is in the stop state, this service notifies a flex ECHONET device that the lower-layer communication software is in the stop state.

◇ Service direction

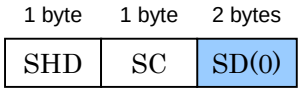
ECHONET device adapter → Flex ECHONET device

◇ Service request




SHD ：0x9D
SC ：0x11
SD(0) ：LF 0x01 DF Lower-layer communication software ID (device_id)

◇ Processing result




SHD ：0x9D
SC ：0x31
SD(0) ：LF 0x01 DF Processing result (0x00：TRUE, 0x01：FALSE)

(20) Complete initialization request service (Optional)

Requests an ECHONET device adapter to cold start (1) the lower-layer communication software related to a specified lower-layer communication software ID and place it in the communication stop state. In this case, the house code information and MAC address are acquired again.

◇ Service direction

Flex ECHONET device → ECHONET device adapter

◇ Service request

| 1 byte | 1 byte | 2 bytes | 3 bytes | 3 bytes | 3 bytes | 3 bytes | 2 bytes |
|---|---|---|---|---|---|---|---|
| SHD | SC | SD(0) | SD(1) | SD(2) | SD(3) | SD(3) | SD(5) |

SHD ：0x9C
SC ：0x12

| | | | | |
|---|---|---|---|---|
| SD(0) | ：LF | 0x01 | DF | Lower-layer communication software ID (device_id) |
| SD(1) | ：LF | 0x02 | DF | Transmitting buffer size (in bytes)<br>Indicates the maximum frame length which can be transmitted by a FLEX ECHONET piece of equipment. |
| SD(2) | ：LF | 0x02 | DF | Receiving buffer size (in bytes)<br>Indicates the maximum frame length for which a FLEX ECHONET piece of equipment can perform receiving processing |
| SD(3) | ：LF | 0x02 | DF | Maximum retention time for transmission messages (msec.)<br>The maximum time for which a transmission message to be retained can be retained (ECHONET equipment adapter). Any transmission message that has been retained for longer than this time period may be discarded regardless of the result of the transmission processing. |
| SD(4) | ：LF | 0x02 | DF | Maximum retention time for received messages (msec.)<br>The maximum time for which a received message to be retained can be retained (ECHONET equipment adapter). Any received message that has been retained for longer than this time period may be discarded regardless of the result of the receiving processing. |
| SD(5) | ：LF | 0x01 | DF | Operation mode<br>0x00: Normal operation<br>0x01: Test/maintenance mode |

◇ Processing result

| 1 byte | 1 byte | 2 bytes |
|---|---|---|
| SHD | SC | SD(0) |

SHD ：0x9D
SC ：0x32

| | | | | |
|---|---|---|---|---|
| SD(0) | ：LF | 0x01 | DF | Processing result (0x00：TRUE, 0x01：FALSE) |

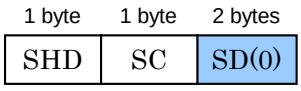
(21) Communication stop request service (Optional)

Requests an ECHONET device adapter to place the lower-layer communication software related to a specified lower-layer communication software ID in the communication stop state.

◇ Service direction

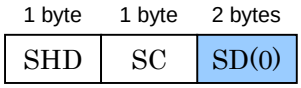
Flex ECHONET device → ECHONET device adapter

◇ Service request




SHD ：0x9C
SC ：0x13
SD(0) ：LF 0x01 DF Lower-layer communication software ID (device_id)

◇ Processing result




SHD ：0x9D
SC ：0x33
SD(0) ：LF 0x01 DF Processing result (0x00：TRUE, 0x01：FALSE)

(22) Complete stop request service (Optional)

Requests an ECHONET device adapter to place the lower-layer communication software related to a specified lower-layer communication software ID in the stop state.

◇ Service direction

Flex ECHONET device → ECHONET device adapter

◇ Service request

| 1 byte | 1 byte | 2 bytes |
|---|---|---|
| SHD | SC | SD(0) |

SHD ：0x9C
SC ：0x14
SD(0) ：LF 0x01 DF Lower-layer communication software ID (device_id)

◇ Processing result

| 1 byte | 1 byte | 2 bytes |
|---|---|---|
| SHD | SC | SD(0) |

SHD ：0x9D
SC ：0x34
SD(0) ：LF 0x01 DF Processing result (0x00：TRUE, 0x01：FALSE)

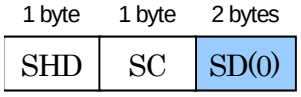
(23) "Request for the number of lower-layer communication software address table data pairs" service (Optional)

Requests the number of address table data pairs held by the lower-layer communication software program that corresponds to the specified lower-layer communication software ID (ECHONET equipment adapter).

◇ Service direction

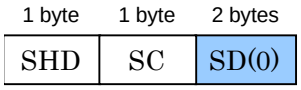
Flex ECHONET equipment → ECHONET equipment adapter

◇ Service request




SHD : 0x9C
SC : 0x15
SD(0) : LF 0x01 DF Lower-layer communication software ID (device id)

◇ Processing result




SHD : 0x9D
SC : 0x35
SD(0) : LF 0x01 DF Processing result (0x00: TRUE, 0x01: FALSE)
0x02 to 0xFF: Reserved for future use
SD(1) : LF 0x01 DF Number of table data pairs

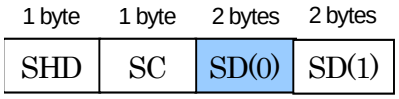
(24) "Request for a lower-layer communication software address table data pair" service (Optional)

Acquires an address table data pair held by the lower-layer communication software program that corresponds to the specified lower-layer communication software ID (ECHONET equipment adapter). The address table contains data pairs each of which comprises a master router flag indicating whether or not the node is a master router, the NodeID and the hardware address defined in Part 3, Sections 7.4 and 7.6.

The data pair number in the service request specifies the data pair to be acquired. The data pair number can take a value between 0 and "the number of data pairs – 1" inclusive. If 0xFF is specified as the data pair number, all data pairs in the address table will be returned (however, when the total service data size would exceed 273 bytes if all data pairs were to be returned, the data pairs returned are subject to the 273-byte limit).
The number of data pairs acquired is indicated in "number of data pairs acquired."

◇ Service direction
Flex ECHONET equipment → ECHONET equipment adapter

◇ Service request




SHD : 0x9C
SC : 0x15
SD(0) : LF 0x01 DF Lower-layer communication software ID (device_id)
SD(1) : LF 0x01 DF Table data pair number
(Specified within the range between 0 and "the number of data pairs - 1" inclusive.)

◇ Processing result

| 1 byte | 1 byte | 2 bytes | 2 bytes | n+1 byte | 2 bytes | 2 bytes |
|---|---|---|---|---|---|---|
| SHD | SC | SD(0) | SD(1) | SD(2) | SD(3) | SD(4) |

SHD : 0x9C
SC : 0x16
SD(0) : LF 0x01 DF Processing result (0x00: TRUE, 0x01: FALSE)
0x02 to 0xFF: Reserved for future use
SD(1) : LF 0x01 DF Number of data pairs ("1" when a data pair number is specified as the number of table data pairs. When 0xFF is specified as the number of table data pairs, the value is the number of data pairs in the table.)
SD(2) : LF n DF Hardware address
SD(3) : LF 0x01 DF NodeID
SD(4) : LF 0x01 DF Master router flag
0x00: Non-master router
0x01: Master router
0x02 to 0xFF: Reserved for future use.
SD (5) and the succeeding sections: Does not exist when SD (1) is 1. When SD (1) is larger than 1, the same data structure as SD (2) through SD (4) is repeated for "the number of SD (1) –1" times.

(25) Master router notification service (Optional)

Notifies the lower-layer communication software program that corresponds to the specified lower-layer communication software ID whether or not the home node is a master router (ECHONET equipment adapter).

◇ Service direction

Flex ECHONET equipment → ECHONET equipment adapter

| 1 byte | 1 byte | 2 bytes | 2 bytes |
|---|---|---|---|
| SHD | SC | SD(0) | SD(1) |

SHD : 0x9C
SC : 0x17
SD(0) : LF 0x01 DF Lower-layer communication software ID (device_id)
SD(1) : LF 0x01 DF Master router flag
0x00: Non-master router
0x01: Master router
0x02 to 0xFF: Reserved for future use.

◇ Processing result

| 1 byte | 1 byte | 2 bytes |
|---|---|---|
| SHD | SC | SD(0) |

SHD : 0x9D
SC : 0x35
SD(0) : LF 0x01 DF Processing result (0x00: TRUE, 0x01: FALSE)
0x02 to 0xFF: Reserved for future use

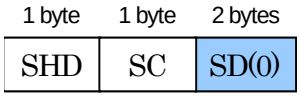
(26) Hardware address data request service (Optional)

Requests the hardware address data held by the lower-layer communication software program that corresponds to the specified lower-layer communication software ID (ECHONET equipment adapter).

◇ Service direction

Flex ECHONET equipment → ECHONET equipment adapter

◇ Service request




SHD : 0x9C
SC : 0x18
SD(0) : LF 0x01 DF Lower-layer communication software ID (device_id)

◇ Processing result


SHD : 0x9D
SC : 0x38
SD(0) : LF 0x01 DF Processing result (0x00: TRUE, 0x01: FALSE)
0x02 to 0xFF: Reserved for future use
SD(1) : LF n DF Hardware Address

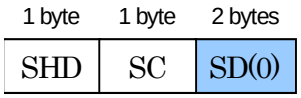
(27) NodeID list acquisition request service (Optional)

Requests the NodeID list held by the lower-layer communication software program that corresponds to the specified lower-layer communication software ID (ECHONET equipment adapter).

◇ Service direction

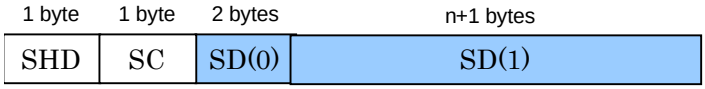
Flex ECHONET equipment → ECHONET equipment adapter

◇ Service request




SHD : 0x9C
SC : 0x19
SD(0) : LF 0x01 DF Lower-layer communication software ID (device_id)

◇ Processing result




SHD : 0x9D
SC : 0x39
SD(0) : LF 0x01 DF Processing result (0x00: TRUE, 0x01: FALSE)
0x02 to 0xFF: Reserved for future use
SD(1) : LF 0x20 DF NodeID List

The NodeID list sets "1" for the bits that correspond to the Node IDs present (hexadecimal notation), beginning with the first byte (see the 32-byte table below).

| | Bit 0 | Bit 1 | Bit 2 | Bit 3 | Bit 4 | Bit 5 | Bit 6 | Bit 7 |
|---|---|---|---|---|---|---|---|---|
| 1st byte | F8 | F9 | FA | FB | FC | FD | FE | FF |
| 2nd byte | F0 | F1 | F2 | F3 | F4 | F5 | F6 | F7 |
| 3rd byte | E8 | E9 | EA | EB | EC | ED | EE | EF |
| 4th byte | E0 | E1 | E2 | E3 | E4 | E5 | E6 | E7 |
| 5th byte | D8 | D9 | DA | DB | DC | DD | DE | DF |
| 6th byte | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 |
| 7th byte | C8 | C9 | CA | CB | CC | CD | CE | CF |
| 8th byte | C0 | C1 | C2 | C3 | C4 | C5 | C6 | C7 |
| 9th byte | B8 | B9 | BA | BB | BC | BD | BE | BF |
| 10th byte | B0 | B1 | B2 | B3 | B4 | B5 | B6 | B7 |
| 11th byte | A8 | A9 | AA | AB | AC | AD | AE | AF |
| 12th byte | A0 | A1 | A2 | A3 | A4 | A5 | A6 | A7 |
| 13th byte | 98 | 99 | 9A | 9B | 9C | 9D | 9E | 9F |
| 14th byte | 90 | 91 | 92 | 93 | 94 | 95 | 96 | 97 |
| 15th byte | 88 | 89 | 8A | 8B | 8C | 8D | 8E | 8F |
| 16th byte | 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 |
| 17th byte | 78 | 79 | 7A | 7B | 7C | 7D | 7E | 7F |
| 18th byte | 70 | 71 | 72 | 73 | 74 | 75 | 76 | 77 |
| 19th byte | 68 | 69 | 6A | 6B | 6C | 6D | 6E | 6F |
| 20th byte | 60 | 61 | 62 | 63 | 64 | 65 | 66 | 67 |
| 21st byte | 58 | 59 | 5A | 5B | 5C | 5D | 5E | 5F |
| 22nd byte | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 |
| 23rd byte | 48 | 49 | 4A | 4B | 4C | 4D | 4E | 4F |
| 24th byte | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |
| 25th byte | 38 | 39 | 3A | 3B | 3C | 3D | 3E | 3F |
| 26th byte | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 |
| 27th byte | 28 | 29 | 2A | 2B | 2C | 2D | 2E | 2F |
| 28th byte | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 |
| 29th byte | 18 | 19 | 1A | 1B | 1C | 1D | 1E | 1F |
| 30th byte | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
| 31st byte | 08 | 09 | 0A | 0B | 0C | 0D | 0E | 0F |
| 32nd byte | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 |

(28) Master router data acquisition request service (Optional)

Requests the master router list held by the lower-layer communication software program that corresponds to the specified lower-layer communication software ID (ECHONET equipment adapter).

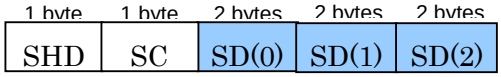
◇ Service direction

Flex ECHONET equipment → ECHONET equipment adapter

◇ Service request


SHD : 0x9C
SC : 0x1A
SD(0) : LF 0x01 DF Lower-layer communication software ID (device_id)

◇ Processing result




SHD : 0x9D
SC : 0x3A
SD(0) : LF 0x01 DF Processing result (0x00: TRUE, 0x01: FALSE)
0x02 to 0xFF: Reserved for future use
SD(1) : LF 0x20 DF Master router flag
0x00: Non-master router
0x01: Master router
SD(2) : LF 0x20 DF NodeID List

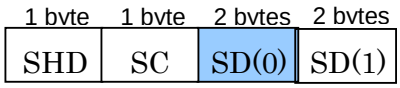
(29) Hardware address conversion request service (Optional)

Requests the hardware address corresponding to the NodeID delivered to the lower-layer communication software program that corresponds to the specified lower-layer communication software ID (ECHONET equipment adapter).

◇ Service direction

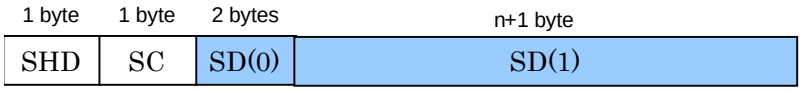
Flex ECHONET equipment → ECHONET equipment adapter

◇ Service request




SHD : 0x9C
SC : 0x1B
SD(0) : LF 0x01 DF Lower-layer communication software ID
SD(1) : LF 0x01 DF

◇ Processing result




SHD : 0x9D
SC : 0x3D
SD(0) : LF 0x01 DF Processing result (0x00: TRUE, 0x01: FALSE)
0x02 to 0xFF: Reserved for future use
SD(1) : LF n DF Hardware Address

## 3.6.9 Protocol translation processing

The translation processing to be performed between the adapter communication software protocol (ACSP) and the adapter communication interface protocol (ACIP) is specified as follows:

(1) Translation from ACSP to ACIP

Translation processing is shown in Fig. 3.27.

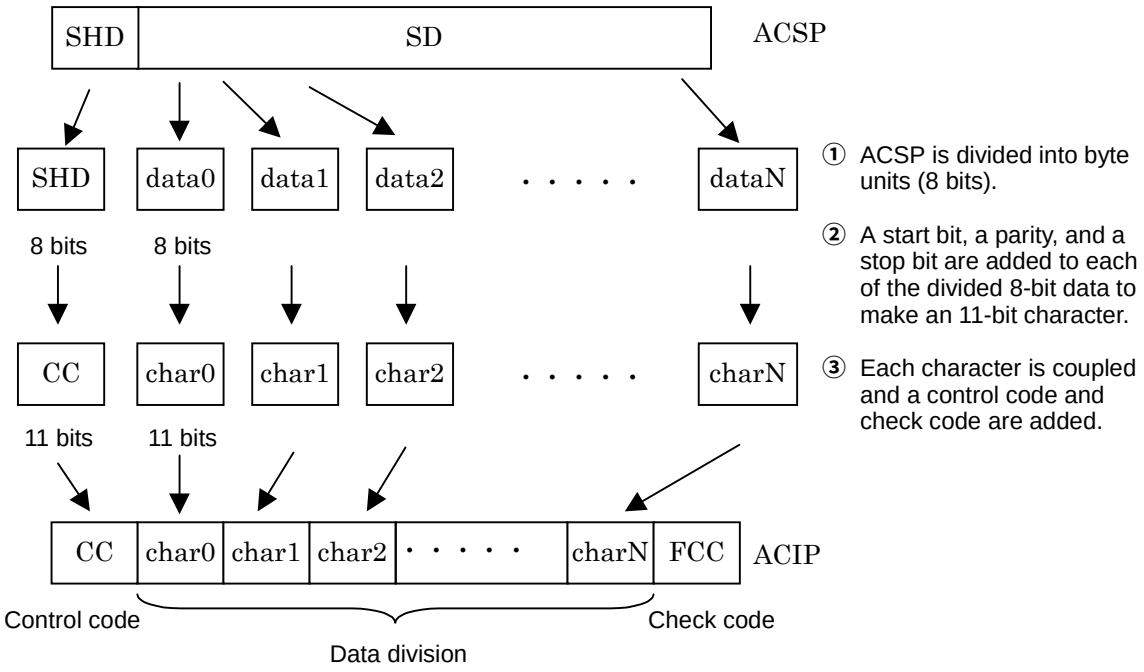



**Fig. 3.27 ACSP to ACIP Translation Processing**

(2) ACIP to ACSP translation processing

Figure 3.28 shows the translation processing.

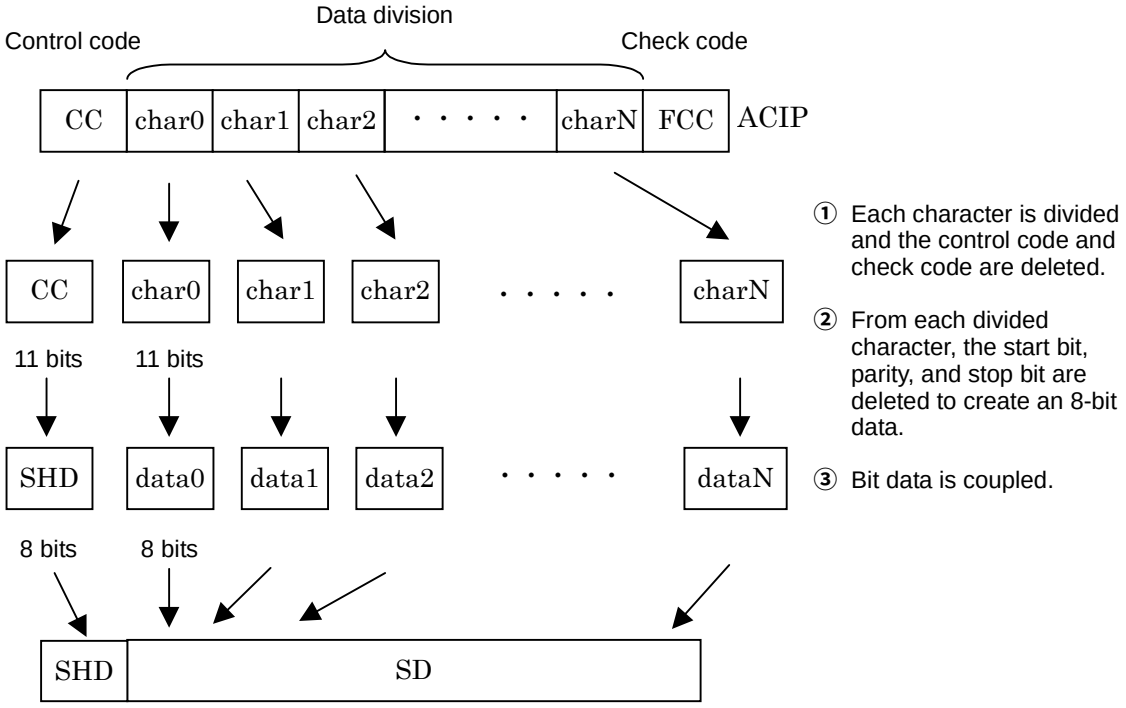



**Fig. 3.28 ACIP to ACSP Translation Processing**

## Operation sequence

The operation sequence of the adapter communication software depends on the common lower-layer communication interface processing mounted in the ECHONET communications processing block or the protocol difference absorption processing connected to the adapter communication software through the common lower-layer communication interface. In the case of a request for data reception, for example, the following processing methods can be considered.

① The ECHONET communications processing block issues a request for data reception to the protocol difference absorption processing block (polling processing).

② The protocol difference absorption processing block issues a trigger to notify the receipt of data when such data has been received (event processing).

Cases in which the ECHONET communications processing block and the protocol difference absorption processing block mounting the different methods above are connected. The adapter communication software must take this into consideration. Figure 3.29 shows an example of the operation sequence for the data reception request service when the ECHONET communications processing block performs event processing and the protocol difference absorption processing block performs polling processing.

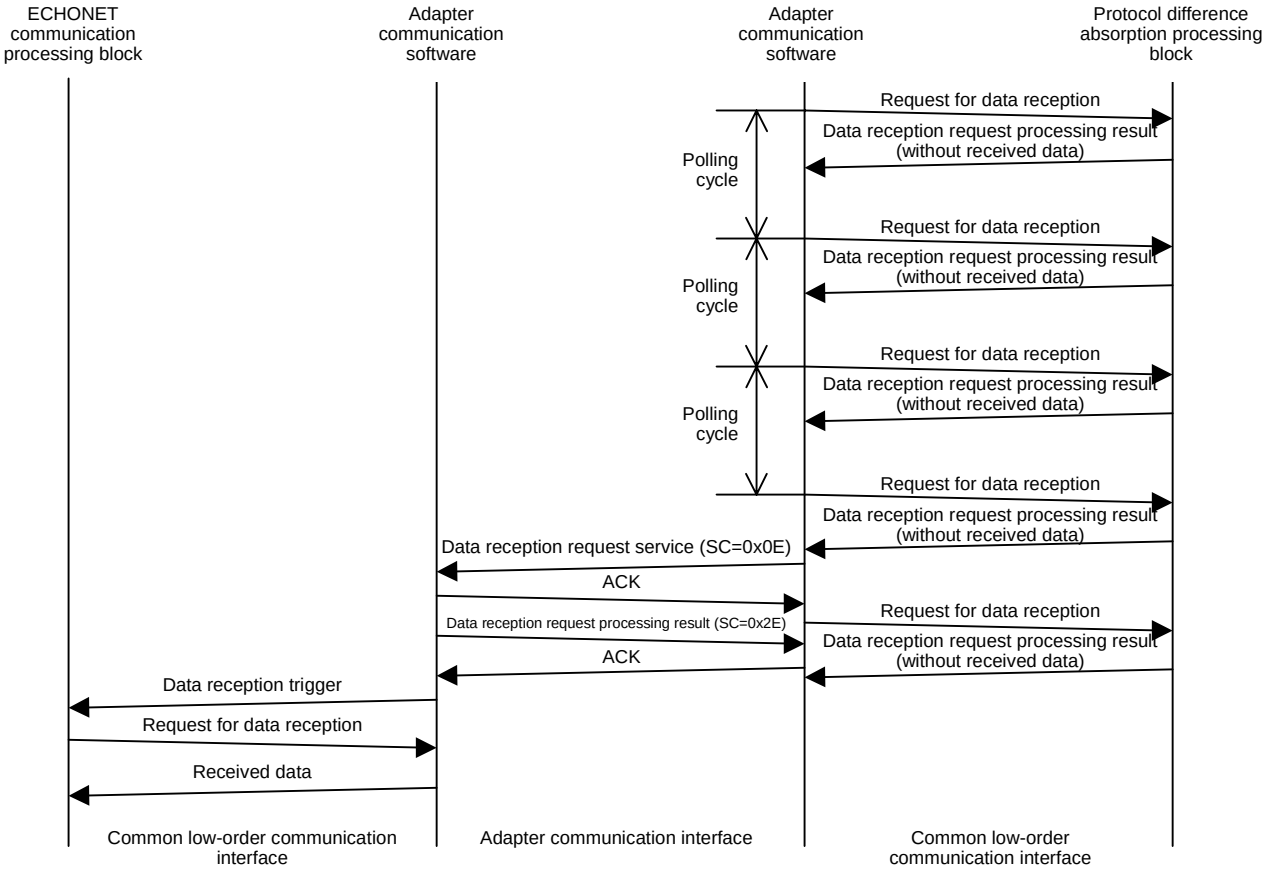



**Fig. 3.29 Adapter Communication Software Sequence**

## 3.6.10 Handling of optional services

Services supported by the common lower-layer communication interface include optional services. Therefore, a situation may occur in the adapter communication interface where the service response side is not ready for a service issued by the service issuer side. In such a case, the adapter communication interface software on the service response side must return an unsupported service notification (0x3F) as the processing result.

## 3.6.11 Handling of optional data

Data for services supported by the common lower-layer communication interface include optional data. Therefore, a situation may occur in the adapter communication interface where:

① The service response side is not ready to process service data of a service issued by the service issuer side; or

② A service issued by the service issuer side does not contain the service data required by the service response side.

The adapter communication software must be capable of operation taking into account the above possibilities. The following is an outline of the processing that must be performed for Case ① and for Case ②:

<Required processing for Case ①>
The service response side's adapter communication software shall perform the common lower-layer communication interface service ignoring the service data it cannot process.

<Required processing for Case ②>
The service response side's adapter communication software shall perform the common lower-layer communication interface service after compensating the deficient portion of the service data with default values.

## 3.6.12 Inhibition of simultaneous service issuance

Only one service at a time can be issued to the adapter communication interface. More specifically, when a service request is issued by a flex ECHONET device or ECHONET device adapter, neither the request receiving side nor the request issuing side can issue any subsequent service until the previously issued service is completed. If a flex ECHONET device and an ECHONET device adapter simultaneously issue a service, the service issued by the ECHONET device adapter takes precedence. The term “simultaneous service issue” means a situation where:

A service request is received before the response to the preceding service request.

Here, the term “service” refers to the information exchange between a flex ECHONET device and ECHONET device adapter during the time interval between the instant at which the service start conditions are established as stipulated in Section 3.6.14 and the instant at which the service ends. This time interval is referred to as the service processing period.

## 3.6.13 Service start/end conditions

In the adapter communication interface, a flex ECHONET device and an ECHONET device adapter shall conclude that a service is started or ended when the following conditions are established:

(1) Service start conditions

Service issuing side

- A service request is issued.

Service responding side

- A service request is received.

(2) Service end conditions

Service issuing side

- A service response is received.
- The timeout time has elapsed.
- The ACK/NAK timeout time for retransmission service request has elapsed.
- A new service request is received during service processing (flex ECHONET device only).

Service responding side

- The ACK signal for a service response is received.
- The ACK/NAK timeout time for a retransmitted service response has elapsed.
- A new service request is received during service processing.

## 3.6.14 Timeout

- Common to Types 1, 2, and 3

  If no processing response is returned after 100 msec (timeout period) has elapsed following the issue of a service request, the next service request can be issued. Figure 3.30 (a) shows the timeout period.

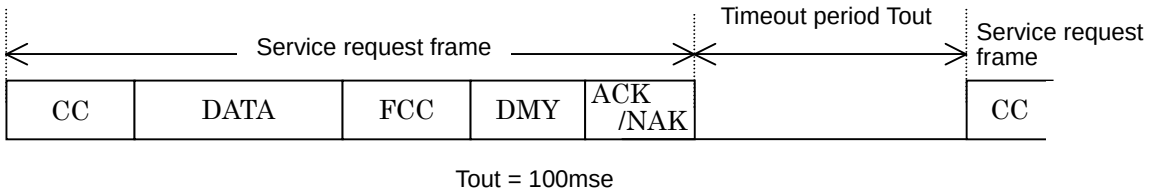



**Fig. 3.30 (a) Timeout Period (Common to Types 1, 2, and 3)**

- Types 4

  If no process response is received within 200 msec (timeout period) after ACK signal reception, a service frame can be issued. The timeout period is indicated in Fig. 3.30 (b).

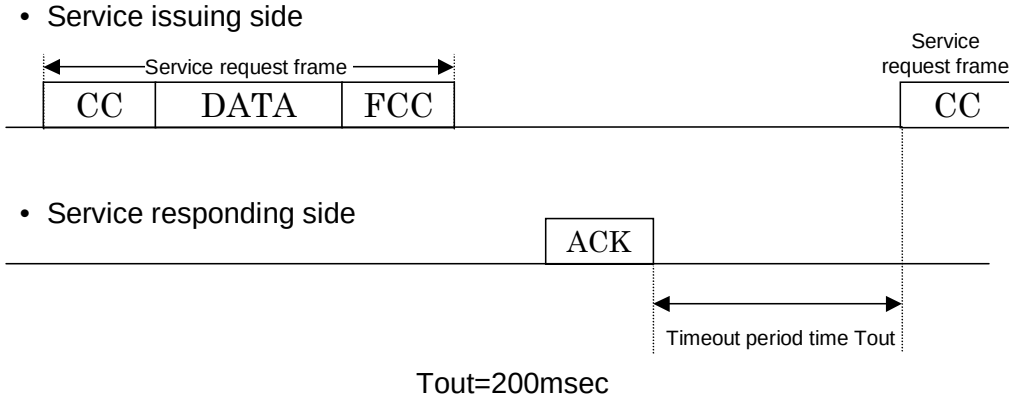



**Fig. 3.30 (b) Timeout Period (Type 4)**

# Chapter 4 ECHONET Gateway

## 4.1 Basic Concept

The application software for connecting an ECHONET domain to an external system using the ECHONET protocol is called a gateway. Devices mounting this gateway are called gateway equipment. In ECHONET, however, processing to be executed by the application is not specified at present. Accordingly, the connection between ECHONET domains and external systems depends on the application software functions.

When the system is installed in an ordinary residence, we recommend that users prepare a security function, including a verification function and access control function, for the gateway application to ensure the security of the ECHONET domain. The functional definitions in such a case are described in Part 9.

# Chapter 5 ECHONET Router

## 5.1 Basic Concept

ECHONET permits different types of networks to be connected as one operational system. The ECHONET router makes the connection between two networks. This is not a TCP/IP router but an ECHONET-dedicated device that can perform ECHONET communications processing. However, the ECHONET router need not always be special equipment that performs only routing; for example, a PC or controller provided with multiple ECHONET communication interfaces may be operated as a router with a routing function. (As another example, an air conditioner may be provided with a function for routing between infrared and a power line. Any device type may be operated as a router with a routing function.)

Accordingly, the ECHONET router becomes communications equipment having two or more ECHONET addresses and consists of two or more nodes as necessary.

The requirements for the use of IrDA Control as ECHONET lower-layer communication software are described in Chapter 6.

## 5.2 Function Definition

The ECHONET router shall be provided with the following minimum conditions and functions:

(1) The ECHONET router is physically connected to two or more subnets. These subnets can have either different lower-layer communication protocols or the same transmission media communication protocols. For each subnet to be connected, MAC addresses and ECHONET addresses must be retained and managed. (The ECHONET router consists of two or more nodes.)

(2) The ECHONET communications processing block is provided with a routing function, and the communications definition object is provided with a routing table. This routing function conforms to the routing specifications explained in Part 2.

## 5.3 Mechanical and physical characteristics

Regarding connection to the transmission media, the specifications in Part 3 shall be observed in accordance with each communication protocol corresponding to the ECHONET router. Other mechanical and physical characteristics for ECHONET routers are specified below.

### 5.3.1 Display block

To display the operation status of the ECHONET router, the following minimum specifications must be satisfied. For display methods not specified here, the specifications native to the product shall be applicable. Regarding the operation status, see Part 2, Chapter 5.

- Number of LEDs

  1 LED (for operation status display)

  - LED color

  Green

  - Status display method

| | |
|---|---|
| Normal operation (NetID acquires) | : ON |
| Normal operation (NetID not acquired) | : Blink (cycle 1) |
| Initial operation | : Blink (cycle 2) |
| Error | : Blink (cycle 3) |
| Non-operation | : OFF |

  * Cycle 1 ..... Repetition of ON for 2 sec and OFF for 2 sec
  * Cycle 2 ..... Repetition of ON for 2 sec and OFF for 0.5 sec
  * Cycle 3…… Repetition of ON for 0.5 sec and OFF for 0.5 sec

Note: The term “initial processing” means a cold start or warm start (which is a startup achieved by performing a hardware reset process while retaining the previously acquired address and initial setup information).

## 5.4 Electrical characteristics

For transmission media connections, the specifications in Part 3 shall be observed for each lower-layer communication protocol corresponding to the ECHONET router.

## 5.5 Logical specifications

Regarding the logical specifications for transmission media communication protocols, the specifications provided in Part 3 for individual transmission media communication protocols supported by the ECHONET router are observed. For the protocol difference absorption processing block, the logical specifications provided in Chapter 7 “Protocol Difference Absorption Processing Block Specifications” in Part 2 are observed. The routing specifications shall conform to those explained in Part 2.

# Chapter 6 IrDA Control Routers

## 6.1 Basic Concept

In ECHONET, routing processing with a subnet adjacent to subnets consisting of IrDA Controls must satisfy the requirements native to the IrDA Control in addition to the requirements for a general ECHONET router. That is, the functions as an ECHONET router must be implemented on the IrDA Control host. This is intended to absorb restrictions on IrDA Control communication functions when the IrDA Control host functions as a router.

In this section, only the contents of the specifications native to "IrDA Control Routers" are described. Accordingly, see Chapter 5 for contents common to "General Routers".

1) Restrictions*

The IrDA Control is designed based on the specifications provided for communication between a PC (host) and a peripheral device (peripheral) by infrared, and is not provided with the following functions:

| |
|---|
| ① Communication between peripheral devices (peripherals) (because communications between the mouse and the keyboard are not required) |
| ② Simultaneous broadcast communications |
| ③ Bind start request from the host side (because communications are started with input on a peripheral device) |

2) Restriction absorbing method

To compensate for the functions in Items ① to ③ above, ECHONET specifies the following functions:

| |
|---|
| • For Item ①: Communications between peripheral devices are implemented by the host's relay of data (as described in Section **6.2 "Communications Between Peripherals"**). |
| • For Item ②: For simultaneous broadcast data, the host transmits data individually (as described in Section **6.3 "Communications of Broadcast-specified Data"**). |
| • For Item ③: The peripheral side starts binding periodically to solve the problem of Item ③ above. It is specified that the host should be provided with a receiving buffer (as described in Section **6.4 "Communication to a Peripheral in Unbind Status"**). |

## 6.2 Communication Between Peripherals

As described in the previous section, the IrDA cannot perform direct communication between peripherals. Communication between peripherals (so-called N:M communication) can be performed by the host's relay of data. This section describes only individual specified data. Broadcast-specified data is described in the next section.

Figure 6.1 shows the procedure for transmitting data from peripheral A to peripheral B.

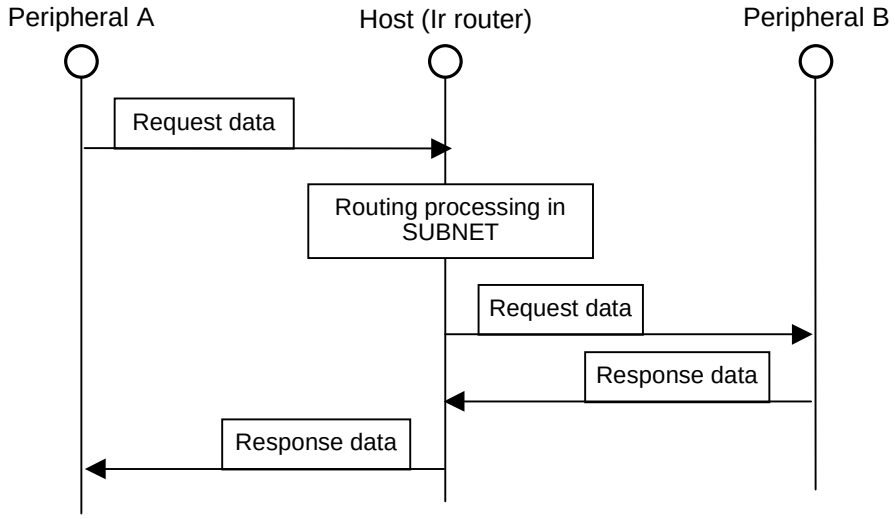



**Fig. 6.1 Communication Sequence Between Peripherals**

Thus, peripheral A transmits data to the host. The host that has received data from peripheral A performs routing processing in the subnet and transmits the received data as transmission data to peripheral B. In cases requiring response data as shown in Fig. 6.1, the host can relay data in the same way as it can request data.

Figure 6.2 shows an outline of processing between layers when peripheral-to-peripheral communication is performed. Characteristic processing for IrDA Control (host routing processing, virtual MAC address, address control table, etc.) is mainly explained as follows:

(Processing in peripheral A of the transmitting source)

① In the ECHONET Communication Middleware of peripheral A, transmission data is created. At this time, specify SEA = self NodeID and DEA = transmitting destination NodeID. (SEA = 0x02, DEA = 0x03 in the case shown in Fig. 6.2)

② The protocol difference absorption layer performs address translation processing. Specify host NodeID (= MAC address) instead of DEA NodeID as the MAC address. Next, the data is delivered to data divide processing and transmit processing. A request for data transmission is sent to the ECHONET lower-layer communication software.

(Relay processing by the host)

③ The peripheral address (PADD, 4 bits) of the transmitting source is translated into the "virtual MAC address" (8 bits) by referencing the address control table managed by the host.

④ In the protocol difference absorption processing block, the "virtual MAC address" of the transmitting source is translated as "NodeID" of the transmitting source.

⑤ In the received data judgment processing block, when both transmitting source NetID and transmitting destination NetID are specified as those of a self-subnet, the data is determined to be intra-subnet communication data, and intra-subnet routing processing is performed. In this case, the number of EHD hops is not added, and the received data is transferred to address translation processing as transmission data.

⑥ In the address translation processing block, DEA NodeID is extracted and address translation processing is performed. In this case, DEA NodeID and transmitting destination MAC address (virtual MAC address) have a 1:1 association, so no translation is required.

⑦ In the ECHONET lower-layer communication software, the transmitting destination MAC address (virtual MAC address) is translated into PADD (4 bits) by referencing the address control table. (If bind processing has already been performed, said virtual MAC address is translated into PADD by referencing the address control table. If bind processing has not been performed, the transmission data is held in the transmitting buffer.) At this time, because the PADD of the transmitting destination peripheral is not available, the virtual MAC address must be held.

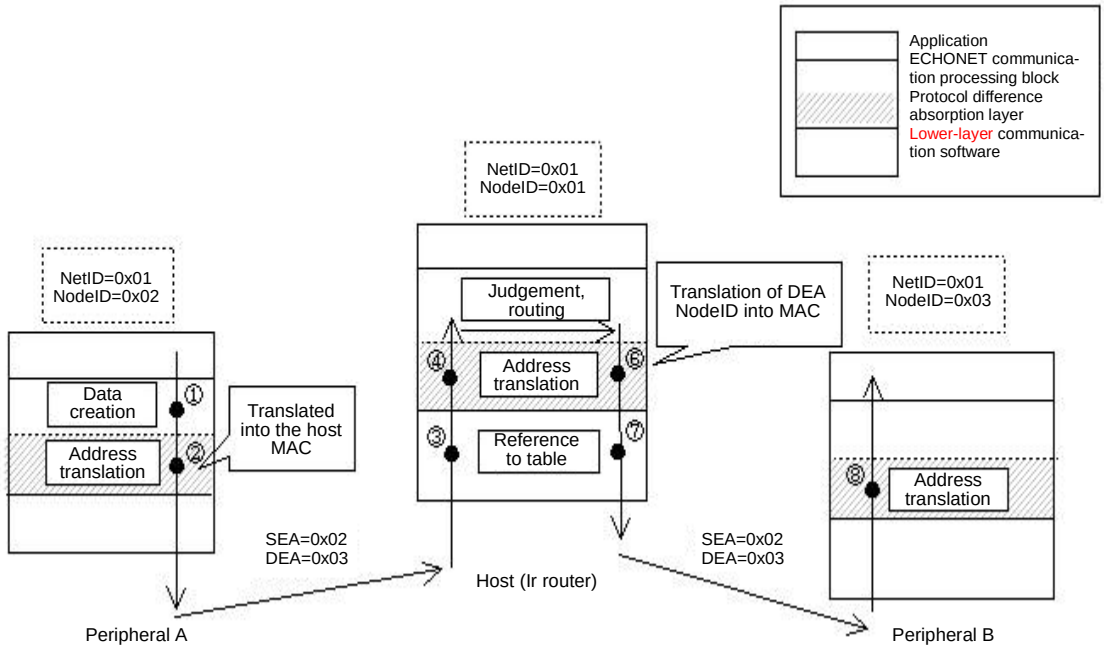



**Fig. 6.2 Outline of Within-Layer Processing of Transmission Sequence Between Peripherals**

## 6.3 Rules of Broadcast-specified Data Communication

### 6.3.1 Overview

The IrDA Control must not be provided with a broadcast-specified data function as described in the previous section.

When the host receives broadcast-specified data including the self-subnet, it must transmit data individually to peripherals in the subnet. This section describes the processing sequence for when the host receives broadcast data.

The self-subnet broadcast specifications are classified into the following two cases:

(1) When receiving broadcast-specified data from outside the IrDA subnet.
(2) When receiving broadcast-specified data from inside the IrDA subnet.

Each of these cases is described below.

### 6.3.2 When receiving broadcast-specified data from outside IrDA subnet

Figure 6.3 shows a processing sequence for when broadcast-specified data, including the IrDA Control subnet, is received from outside the IrDA Control subnet. This example explains the case in which broadcast-specified data is transmitted from node X of the adjacent subnet to the IrDA Control subnet. The sequence shown in Fig. 6.3, “Response data (Note) (from host)”, is only for cases in which the host must return a response.

When the host receives request data (broadcast) oriented to the IrDA Control subnet, internal processing is as follows:

- In data judgment processing, when the EHD of received data is the broadcast specification, and the destination of the broadcast data includes “self-subnet NetID” by referencing “Broadcast specification type code” and “Broadcast target specification code”, “routing processing” is performed in the same way as the general router.
- In “routing processing”, the EHD hop count is incremented by 1, and the received data is transferred as transmission data to the protocol difference absorption processing block through the common lower-layer communication interface.
- The protocol difference absorption layer transfers the request for transmission received from the predecessor to the ECHONET lower-layer communication software of the IrDA Control together with broadcast specification information through the individual lower-layer communication interface.
- The ECHONET lower-layer communication software transmits broadcast data individually to transmittable peripherals (see Note 1) other than the transmitting source peripheral. In this case, the MAC address of the peripheral is the information held by the host at bind processing and requires no translation.

Note 1: “Transmittable peripherals” are peripherals in a bind status for the host. Data transmission to peripherals in unbind status is described in Section 6.4.

When transmission to all peripherals is completed or the holding time has elapsed, data in the buffer is abandoned and processing is terminated.

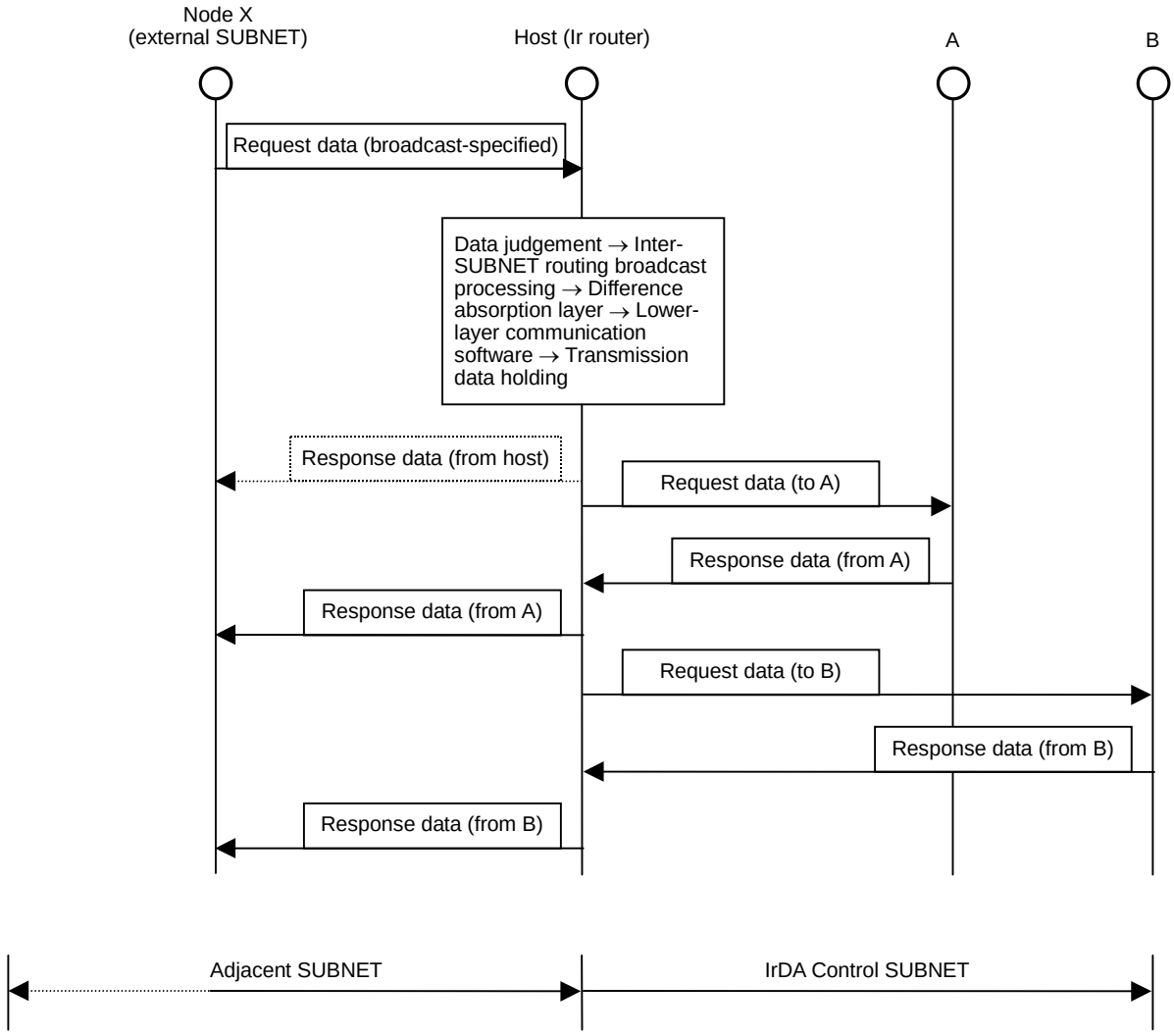



**Fig. 6.3 Broadcast-specified Data Communication Sequence from Outside IrDA Control Subnet**

### 6.3.3 When receiving broadcast-specified data from inside IrDA subnet

Figure 6.4 shows the processing sequence for when broadcast-specified data is received from the self-IrDA Control subnet.

When the host receives request data (broadcast) oriented to the IrDA Control subnet, internal processing in the host is performed as follows:

- In data judgment processing, when the EHD of the received data is the broadcast specification, the destination of the broadcast data includes “self-subnet NetID” by referencing “Broadcast specification code” and “Broadcast target specification code”, and the transmitting source NetID matches the self-subnet NetID, “Intra-subnet routing processing” native to the IrDA Control is performed. (However, routing processing to the data transmitting destination node is not performed.)
- In “Intra-subnet routine processing”, the EHD hop count is not incremented, and the received data is transferred as transmission data to the protocol difference absorption processing block through the common lower-layer communication interface.
- The protocol difference absorption layer transfers the request for transmission received from the predecessor to the ECHONET lower-layer communication software of the IrDA Control together with broadcast specification information through the individual lower-layer communication interface.
- The ECHONET lower-layer communication software transmits broadcast data individually to transmittable peripherals (see Note 1). In this case, the MAC address of the peripheral is the information held by the host at bind processing and requires no translation.

Note 1: “Transmittable peripherals” are peripherals in a bind status for the host. Data transmission to peripherals in the unbind status is described in Section 6.4. When transmission to all peripherals is completed or the holding time has elapsed, data in the buffer is abandoned and processing is terminated.

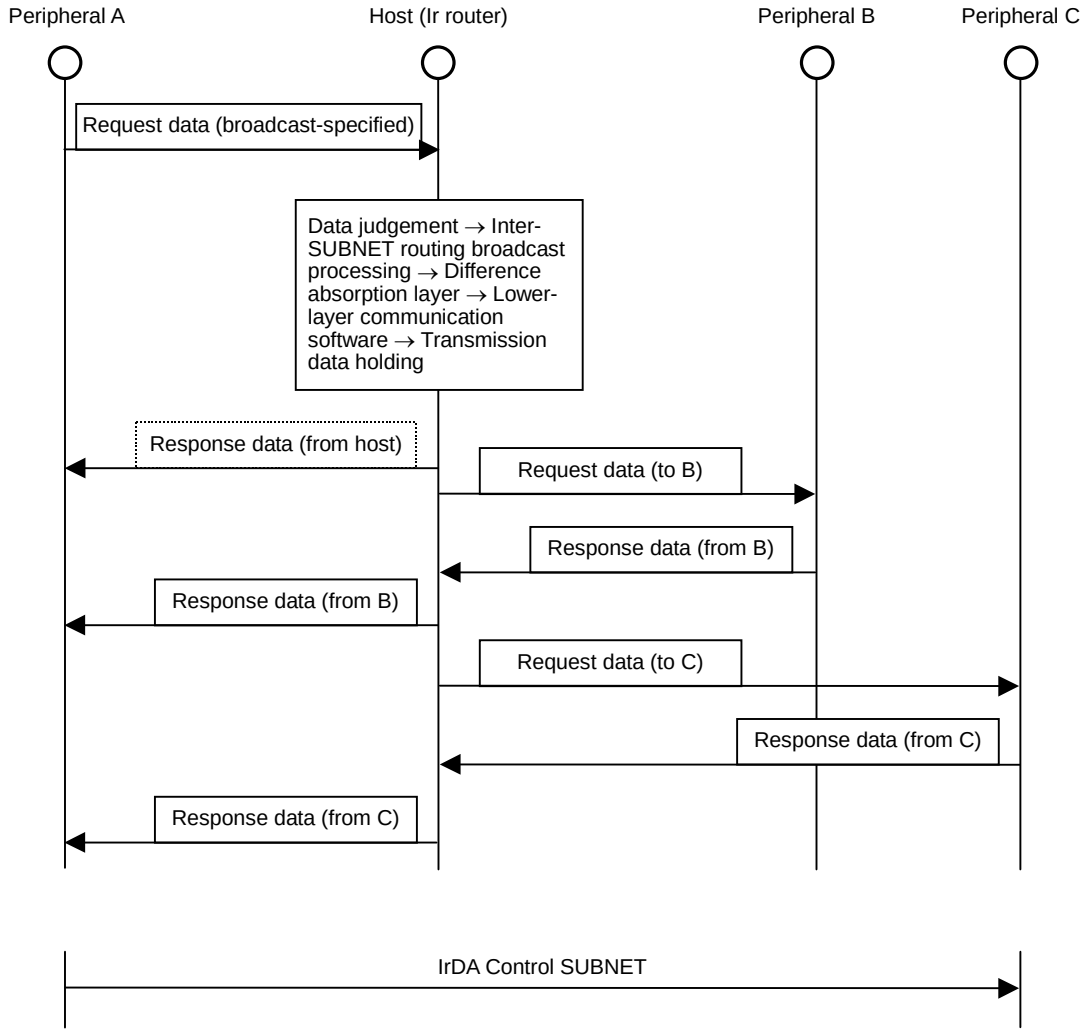



**Fig. 6.4 Broadcast-specified Data Communication Sequence from Inside IrDA Control Subnet**

## 6.4 Communication to a Peripheral in the Unbind Status

### 6.4.1 Basic concept

As described in 1) Restrictions in Section 6.1, the IrDA Control cannot perform a bind start request function from the host side. In this situation, data communication cannot be performed from the host in the unbind status (idle status) or another subnet. In ECHONET, a means for compensating for this problem has been adopted in the specifications to secure bi-directionality of communication start in pseudo form. Therefore, the ECHONET lower-layer communication software must be provided with the following functions:

Peripheral-dedicated lower-layer communication software specifications

- It is mandatory to mount a function to make a bind request to the host periodically in accordance with “Bind request interval (*1)” that can be set by an application. However, this interval can be set arbitrarily. An infinite interval shall also be allowed.

Host-dedicated communication software specifications

- The host can hold data oriented to the self-subnet received from an external subnet or the self-subnet during the “Data holding time (*2)” set for each peripheral. Mounting of this function shall also be mandatory.

### 6.4.2 Sequence

Figure 6.5 shows a processing sequence for data from another subnet to a peripheral in the unbind status. The sequence is explained below.

- The host receives data oriented to the self-subnet.
  (Includes both individual and broadcast; data from self-subnet is the same.)
- The host performs data judgment and routing processing as described in the previous section and transfers the request for transmission to the ECHONET lower-layer communication software.
- If the transmitting destination peripheral is in the bind status, data is transmitted immediately. If it is in the unbind status (including broadcast), data is held in the buffer.
- Each peripheral mounts a fixed-time communication function and makes a bind request to the host at the set interval.
- When receiving a bind request from the peripheral, the host starts bind processing for the peripheral.
- Data is transmitted to a peripheral in the bind status. If a response is required, the response is returned.

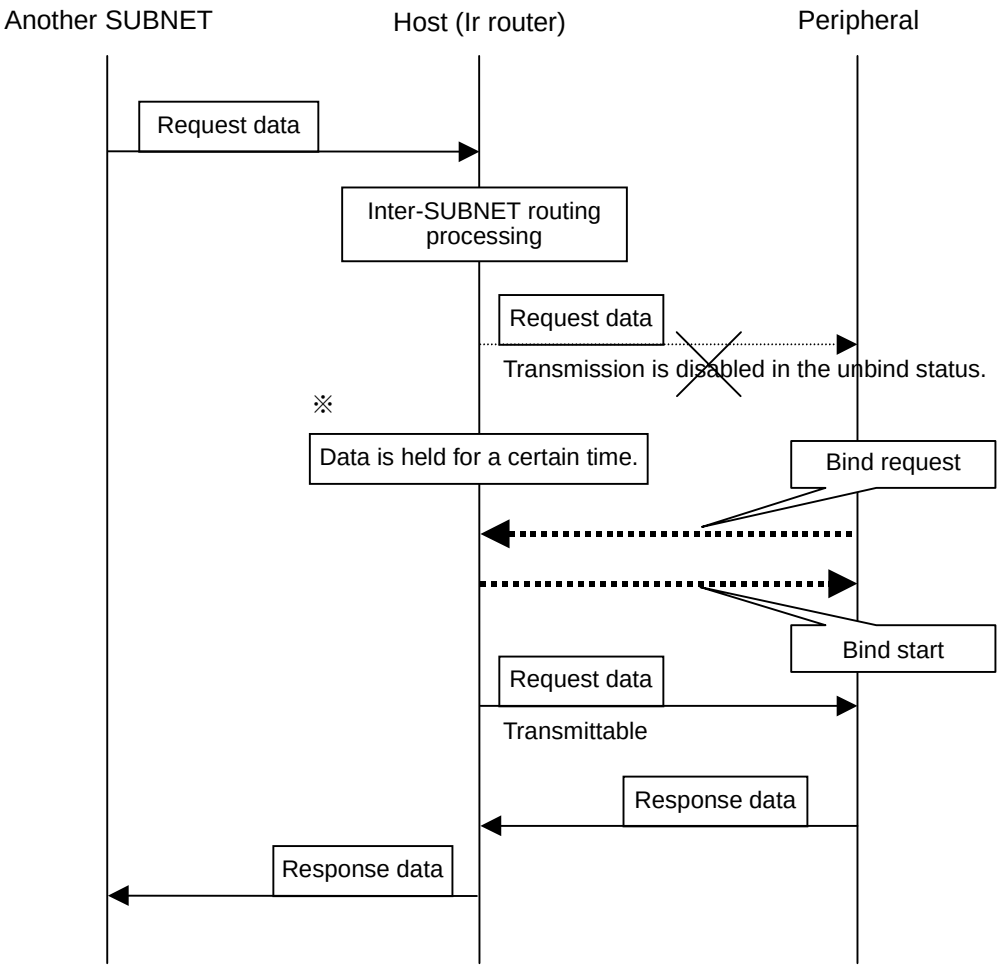



**Fig. 6.5 Communication Sequence to a Peripheral in the Unbind Status**

"Bind Interval (*1)"

The ECHONET standard does not specify any "Bind interval (*1)" setting. However, the following point must be considered in system design.

The IrDA Control Specifications specify that current status switches to unbind status if non-communication status continues for 5 or 30 seconds between the host and a peripheral. (This value can be set from five to thirty seconds.) Accordingly, to retain the bind status at all times in ECHONET, the transition time to the unbind status must be set to "30 sec" and "Bind interval (*1)" must be set to 30 sec or less. Usually, this setting is desirable when constructing a system.

When the main purpose of an application is to notify the central monitoring equipment of the occurrence of an event by using a body sensor when a peripheral is driven by a battery, "Bind interval (*1)" should be set to "Infinite", thereby minimizing battery power consumption and extending battery life.

When setting “Data holding time (*2)”, the following two issues should be considered from the viewpoint of system design and operation:

1. When “Bind interval (*1)” is set so as not to switch to the unbind status
   The host can transmit received data to the corresponding peripheral immediately, so the data holding time may be set to several seconds.
2. When “Bind interval (*1)” is set so as to switch to the unbind status
   “Data holding time (*2)” must be set to a longer value than “Bind interval (*1)”, or the host will not be able to transmit received data to the corresponding peripheral.

# Chapter 7 ECHONET MAC Address Servers

## 7.1 Basic Concept

Part 3 of the ECHONET Specification defines the specifications for centrally managing, using a client-server architecture, the ECHONET MAC addresses of ECHONET nodes that reside in the same subnet. This chapter provides the specifications for devices that act as servers in such a management system, which are generically referred to as ECHONET MAC address servers herein. An ECHONET MAC address server does not necessarily have to be an ECHONET node (i.e. it is not mandatory to have the ECHONET communication middleware functions).

## 7.2 Definitions of the Basic Functions

An ECHONET MAC address server centrally manages MAC addresses using a different method for each of the media specified in Part 3. This section specifies the basic functional requirements that all ECHONET MAC address servers must satisfy. These requirements are as follows:

* The extent to which an ECHONET MAC address server manages MAC addresses shall be within the single ECHONET subnet to which it belongs.

* An ECHONET MAC address server must be capable of assigning, in response to client ECHONET nodes' requests, ECHONET MAC addresses in such a manner that there is no overlapping with other ECHONET nodes.

Section 7.3 shows detailed definitions of the functions of ECHONET MAC address servers that use IP/Bluetooth or IP/Ethernet/IEEE802.3 as the transmission medium.

## 7.3 ECHONET MAC Address Servers for IP/Bluetooth or IP/Ethernet/802.3

ECHONET MAC address servers that use IP/Bluetooth as the transmission medium as well as those that use IP/Ethernet/802.3 must satisfy the minimum requirements listed below.

(1) The mechanical and physical characteristics specifications, electrical specifications, IP layer logical specifications and the logical specifications for the layers below the Bluetooth layer (which are provided in Part 3, Chapter 7 of the ECHONET Specifications) must be satisfied

(2) The UDP interface requirements specified in the “IP/Bluetooth Interface Layer Logical Specifications” must be satisfied.

(3) The packet types specified as compulsory for MAC address servers and the packet types specified as compulsory for all nodes must be supported.

(4) It must be possible to execute in accordance with the basic communication sequence requirements specified in the “IP/Bluetooth Interface Layer Logical Specifications” the communication sequences for the packet types supported.

(5) The server must be capable of cold starting. It must also be capable of determining its own MAC address during a cold start in accordance with the ECHONET MAC start sequence requirements specified in the “IP/Bluetooth Interface Layer Logical Specifications.”

(6) The server must be capable of warm starting. It must also be capable of determining its own MAC address during a warm start in accordance with the ECHONET MAC start sequence requirements specified in the “IP/Bluetooth Interface Layer Logical Specifications.”

(7) The MAC address server requirements specified in the “IP/Bluetooth Interface Layer Logical Specifications” must be satisfied.

### 7.3.1 Mechanical and physical characteristics

Regarding connection to the transmission medium, the specifications provided in Part 3 for the individual transmission medium communication protocols shall apply. This subsection provides the mechanical and physical specifications for the communication equipment (display section) used in addition to that described in Part 3.

Where LEDs are provided for indicating the operation status of an ECHONET MAC address server for IP/Bluetooth or IP/Ethernet/802.3, the minimum requirements listed below must be satisfied. For status indication using a method not described herein, the specifications for the individual products shall be used (For status-related matters, refer to Part 3).

* Number of LEDs
  One (used to indicate the operation status)
* LED color
  Yellow
* Status indication
  Plug and play: Lit
  Plug and play setting – abnormal completion: Blinking
  Other than above: Unlit

### 7.3.2 Electrical characteristics

Regarding connection to the transmission medium, the specifications provided in Part 3 for the individual transmission medium communication protocols shall apply.

### 7.3.3 Logical specifications

The logical specifications provided in Part 3 for the individual transmission medium communication protocols shall apply.

# Chapter 8 ECHONET Middleware Adapters

## 8.1 Basic Concept

Chapter 3 of Part 7 (this part) describes the methods for providing home appliances with ECHONET node functions in an ECHONET environment, as follows:

[Excerpt]

There are 2 methods for providing home appliances with the ECHONET node functions in an ECHONET environment:

(1) Designing home appliances in such a way that they satisfy the ECHONET node requirements by themselves; or

(2) Allowing home appliances that do not have the ECHONET node functions to meet the ECHONET node requirements by using adapters.

Method (2) can be further divided into 2 types according to the method used for interfacing the equipment with the adapter:

(2-1) Using an adapter that provides an interface that is specified in the ECHONET Specification; or

(2-2) Using an adapter that provides an interface that is not specified in the ECHONET Specification.

For the purposes of the ECHONET Specification, an “ECHONET equipment adapter” shall be defined as an adapter that falls under (2-1) above and is used for equipment that does not have the ECHONET lower-layer communication software and protocol difference absorption processing section to connect to the ECHONET network.

[End of Excerpt]

For the purposes of this chapter, an “ECHONET middleware adapter” shall be defined as an adapter that falls under (2-1) above and is used for equipment that does not have the ECHONET lower-layer communication software, protocol difference absorption processing section and ECHONET communications processing section to connect to the ECHONET

network (See Fig. 8.1).

As shown in Fig. 8.1, equipment to which an ECHONET middleware adapter can be connected shall be called “ECHONET-ready equipment.”

An ECHONET middleware adapter shall be configured in such a way that the burden placed on ECHONET-ready equipment in relation to network-related processing and the increase in ECHONET-ready equipment costs (software and hardware costs) are minimized.

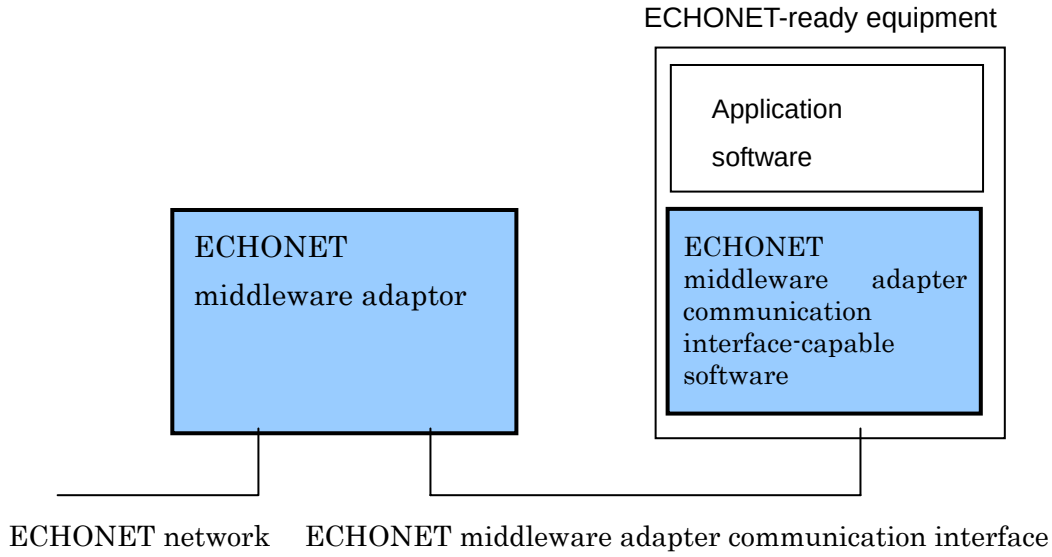



**Fig. 8.1 ECHONET Middleware Adapter and ECHONET-ready Equipment**

The device specifications and electrical and logical specifications for ECHONET middleware adapters are provided in Sections 8.1 through 8.5. The mechanical, physical and logical characteristics specifications for ECHONET middleware adapter communication interface software are provided in Section 8.6.

In this version of the ECHONET Specification, two types of protocol specifications for ECHONET middleware adapter communication interface software are provided to minimize the burden laced on ECHONET-ready equipment in relation to network-related processing:

(1) Object generation type
(2) Peer-to-peer type

Whether or not an ECHONET middleware adapter can be connected to a piece of

ECHONET-ready equipment is determined by the specifications implemented in the adapter and those implemented in the equipment. For this reason, an "equipment interface data recognition service" to identify the specifications implemented in the other party has been defined and is described in Section 8.7.

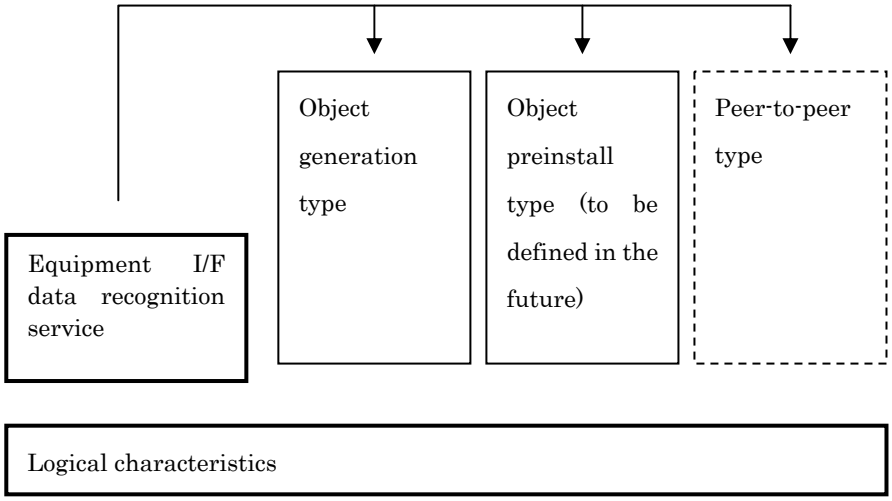



**Fig. 8.2 ECHONET Middleware Adapter Communication Software Hierarchy**

After identification by the equipment I/F data recognition service, the appropriate type for the ECHONET middleware adapter communication interface is selected.

Table 8.1 shows examples of acceptable combinations (i.e. combinations with which communication is possible) of types supported by ECHONET middleware adapters and types supported by ECHONET-ready equipment.

**Table 8.1 Acceptable Combinations of Types Supported by ECHONET Middleware Adapters and Types Supported by ECHONET-ready Equipment**

| ECHONET middleware adapter | ECHONET-ready equipment |
|---|---|
| Object generation type | Object generation type |
| Peer-to-peer type | Peer-to-peer type |

For object preinstall type-based connection, the ECHONET middleware adapter must have the appropriate basic object construction data for the ECHONET-ready equipment to be connected. For peer-to-peer type-based connection, the ECHONET middleware adapter must have the appropriate ECHONET middleware adapter communication software for the ECHONET-ready equipment to be connected.

## 8.1.1 Anticipated configurations for ECHONET middleware adapters (commentary)

To accommodate the implementation methods for ECHONET middleware adapters, several ECHONET middleware adapter communication software protocols have been defined. Figure 8.3 shows the anticipated ECHONET middleware adapter configurations.

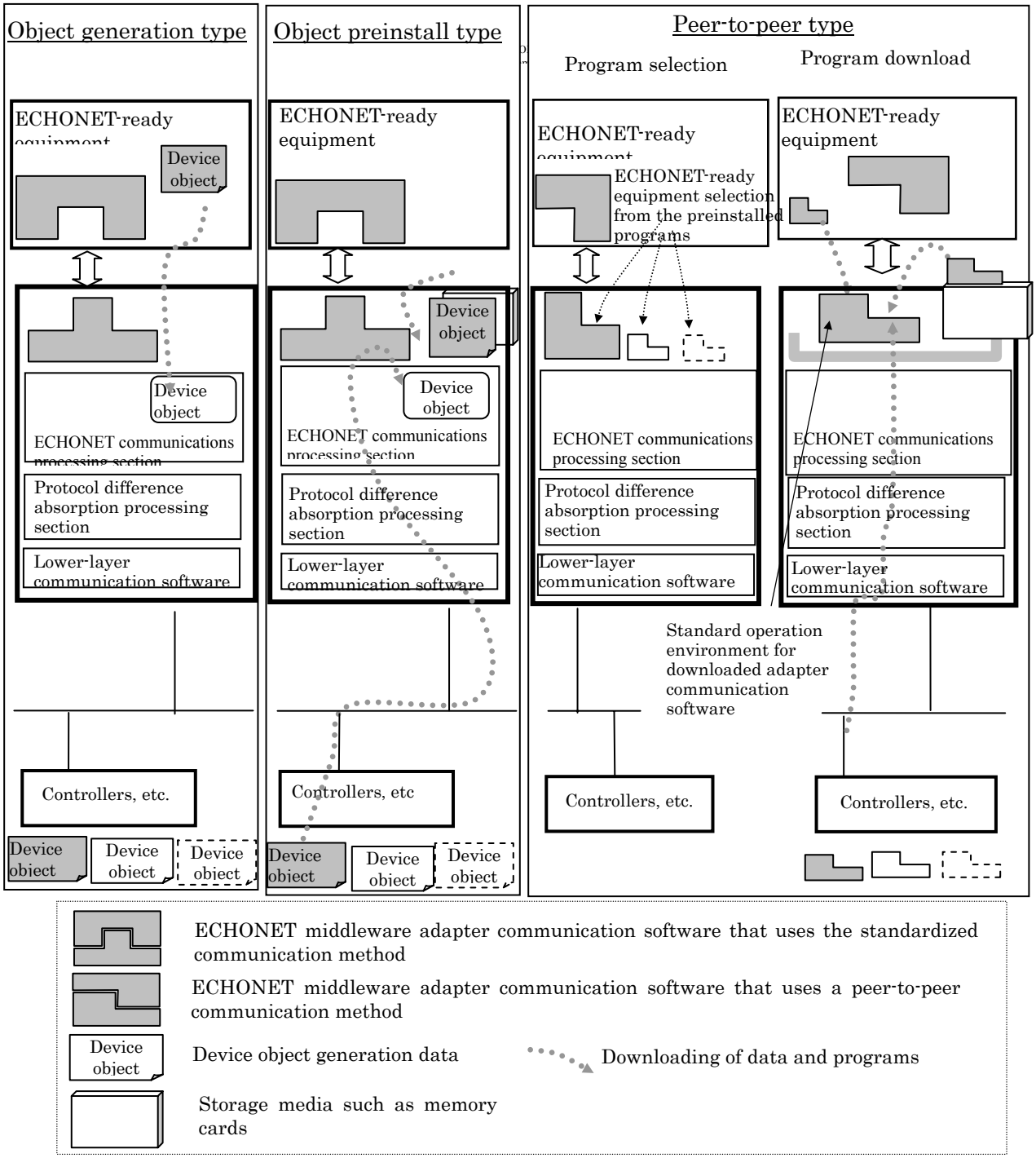



**Fig. 8.3 Anticipated Configurations for ECHONET Middleware Adapter**

As shown in Fig. 8.3, the following 3 implementation methods for ECHONET middleware adapters are anticipated:

(1) Object generation type

Data is exchanged between the ECHONET middleware adapter and ECHONET-ready equipment using a standardized communication method. Object generation data preinstalled in the ECHONET-ready equipment (at least one) is configured in the ECHONET middleware adapter using a standardized procedure.

(2) Object preinstall type

Data is exchanged between the ECHONET middleware adapter and ECHONET-ready equipment using a standardized communication method. The assumption is that the object generation data is either preinstalled in the ECHONET middleware adapter or downloaded from outside to generate objects. No requirement is defined herein with respect to object acquisition methods.

(3) Peer-to-peer type

Data is exchanged between the ECHONET middleware adapter and ECHONET-ready equipment using a user-defined communication method. The following two approaches are anticipated:

One is "Program selection" approach whereby the appropriate communication method (ECHONET middleware adapter communication software) for the ECHONET-ready equipment to be connected is preinstalled in the ECHONET middleware adapter.

The other is "Program download" approach based on downloading from outside.

These specifications anticipate the above-mentioned methods in consideration of the progress status with respect to support (home appliances) and defines the following so that when an ECHONET middleware adapter is connected to a piece of ECHONET-ready equipment in a valid combination, the proper identification, connection and operation will be achieved:

(1) Standard communication method to achieve object generation type communication

A standard communication method to achieve object generation type communication is defined as well as a method for generating objects in the ECHONET middleware adapter by acquiring object generation data from the ECHONET-ready equipment. This version of the ECHONET Specification does not define the requirements for cases in which object generation data is downloaded.

(2) Communication method for the "Program Download Method" for the "Peer-to-Peer Type"

This version of the ECHONET Specification specifies the communication method to download the ECHONET middleware adapter communication software from ECHONET-ready equipment. This version of the ECHONET Specification does not specify requirements for communication methods to download software from equipment other than ECHONET-ready equipment.

(3) Method to identify the communication method for the ECHONET-ready equipment (“equipment interface data recognition service”)

A service to acquire ECHONET-ready equipment interface data, identify the communication method (object generation method, peer-to-peer method or other new communication method) and interpret equipment data.

Using this service, an equipment interface data inquiry is made from the ECHONET middleware adapter to the ECHONET-ready equipment. After this inquiry, the ECHONET middleware adapter executes the ECHONET middleware adapter communication software based on the acquired data.

.

(4) Logical and physical characteristics of the communication interfaces mentioned above (common ones)

Figure 8.4 illustrates the process of executing the equipment interface data recognition service. For details, see Fig. 8.13.

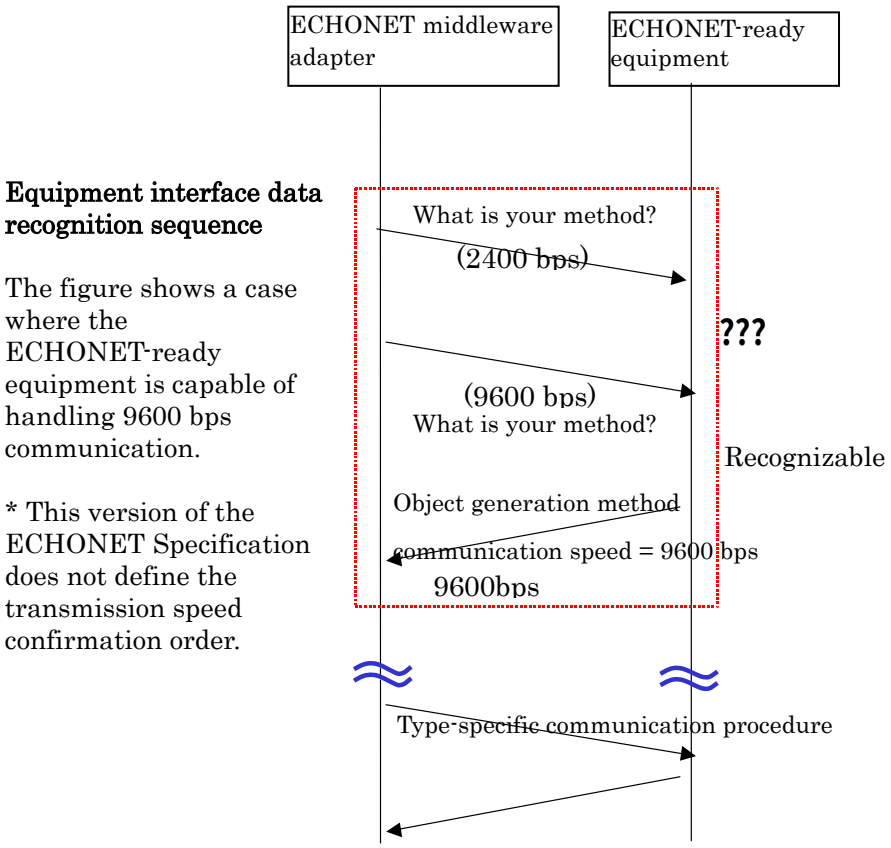



**Fig. 8.4 ECHONET-ready Equipment Recognition Process**

## 8.2 Definitions of Functions

The functions required for an ECHONET middleware adapter have been defined as follows:

(1) Message input and output function

A function to input and output electronic messages to and from the transmission medium in accordance with the lower-layer communication protocol specifications provided in Part 3. This function is performed by the ECHONET lower-layer communication software, which means that a transceiver capable of handling the ECHONET lower-layer communication protocol is required as a function.

(2) Protocol difference absorption function

A function to perform the processing specified in Part 2, Chapter 7: Specifications for Processing at the Protocol Difference Absorption Processing Section. This function is performed by the protocol difference absorption processing section and allows the necessary conversions to be made between the ECHONET lower-layer communication software and ECHONET communications processing section protocol.

(3) ECHONET communications processing function

A function to perform the processing specified in Part 2, Chapter 6: Specifications for Processing at the ECHONET Communications Processing Section. This function is performed by the ECHONET communications processing section.

(4) ECHONET middleware adapter communication interface function

This function, defined in “8.6 ECHONET Middleware Adapter Communication Software Specifications,” performs, between the ECHONET-ready equipment application software and ECHONET communications processing section, the necessary communications processing between the ECHONET middleware adapter and ECHONET-ready equipment, which is defined in the “ECHONET Middleware Adapter Communication Interface Specifications”. This function is performed by the ECHONET middleware adapter communication software.

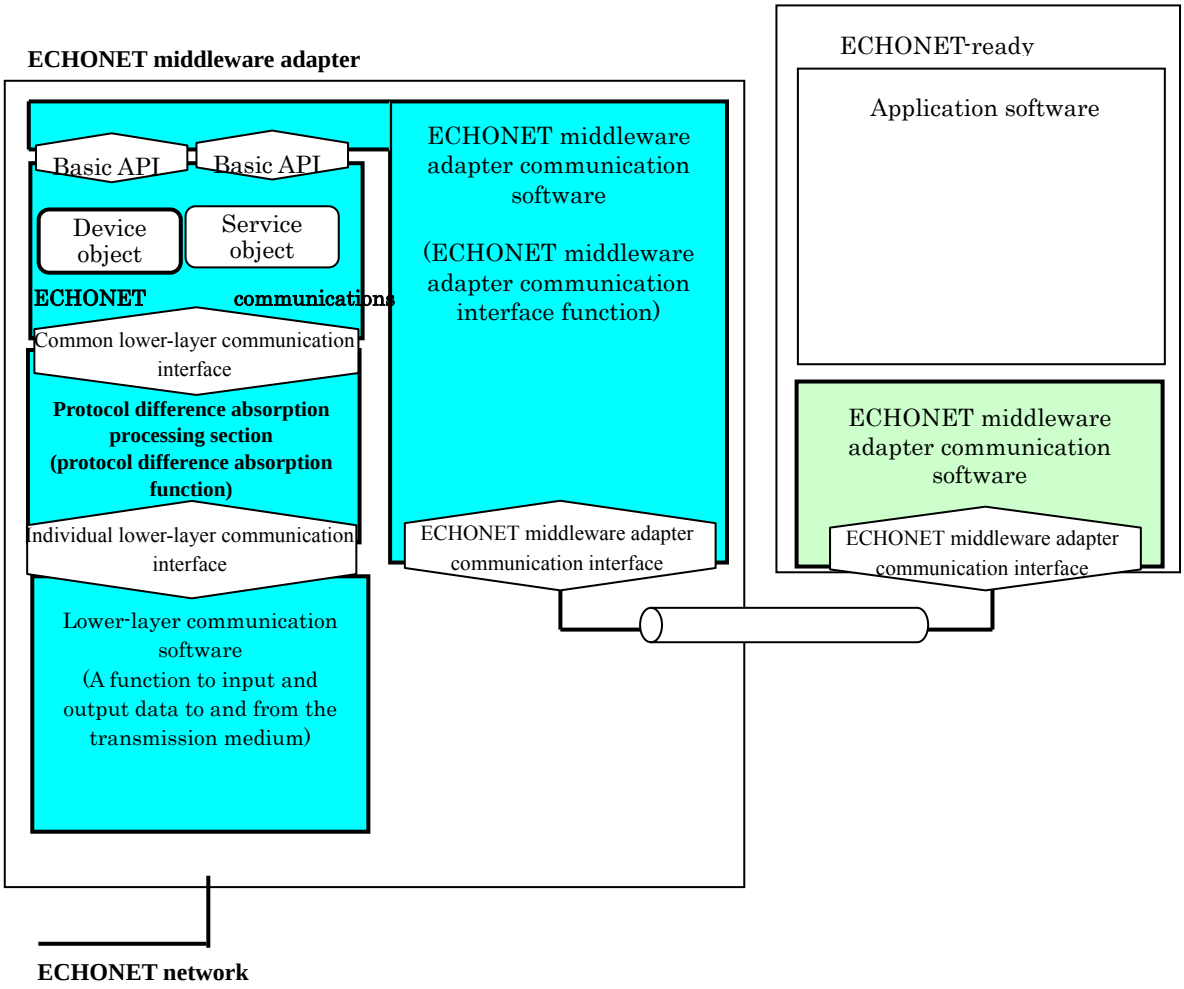



**Fig. 8.5 ECHONET Middleware Adapter Functions**

## 8.3 Mechanical and Physical Characteristics

The specifications for connection to the transmission medium provided in Part 3 for the individual ECHONET lower-layer communication software programs supported by ECHONET middleware adapters shall apply. The specifications for connection to the ECHONET-ready equipment shall be as defined in Section 6. This section defines the mechanical and physical characteristics specifications for ECHONET middleware adapters that are not defined in the above-mentioned part or section.

### 8.3.1 Shape

No specifications are defined for shape, with the exception of the connection block to the ECHONET-ready equipment (ECHONET middleware adapter communication interface). The shape of the connection block for connection to the transmission medium shall be as specified in the specifications for the individual ECHONET lower-layer communication software programs.

### 8.3.2 Display section

Where LEDs are provided as a means of indicating the operation status of an ECHONET middleware adapter, it is recommended that the minimum requirements listed below be satisfied. For status indication using a method not described herein, the specifications for the individual products shall be used.

① Number of LEDs: one (used to indicate the operation status)
② LED color: green
③ Status indication:
In operation: Lit
Initial processing: Slow blinking
Abnormal state: Rapid blinking
Not operating: Unlit

* Slow blinking – repeated lit-unlit sequence with approximately 2 seconds of lighting followed by approximately 0.5 second without lighting

* Rapid blinking - repeated lit-unlit sequence with approximately 0.5 second of lighting followed by approximately 0.5 second without lighting

Note) “Initial processing” means a cold start (i.e. a full-reset start) or a warm start (i.e. a start whereby hardware reset processing is performed with the previously acquired addresses and initial setting data retained).

## 8.4 Electrical Characteristics

The specifications for connection to the transmission medium provided in Part 3 for the individual ECHONET lower-layer communication software programs supported by ECHONET middleware adapters shall apply. The specifications for connection to the ECHONET-ready equipment shall be as defined in Section 6.

## 8.5 Logical Requirements

The logical requirements for ECHONET middleware adapter communication software shall be as defined in Section 6. For the logical requirements for ECHONET lower-layer communication software and the protocol difference absorption processing section, refer to Part 3 and Part 2, respectively.

## 8.6 ECHONET Middleware Adapter Communication Software Specifications

An ECHONET middleware adapter communication software program operates on an ECHONET middleware adapter and/or a piece of ECHONET-ready equipment and uses an ECHONET middleware adapter communication software protocol. This section defines the requirements for ECHONET middleware adapter communication interfaces and ECHONET middleware adapter communication software protocols and explains how they must be used.

### 8.6.1 ECHONET middleware adapter communication interfaces – overview

Figure 8.6 shows how the ECHONET middleware adapter communication software allows the ECHONET middleware adapter and the equipment to communicate with each other.

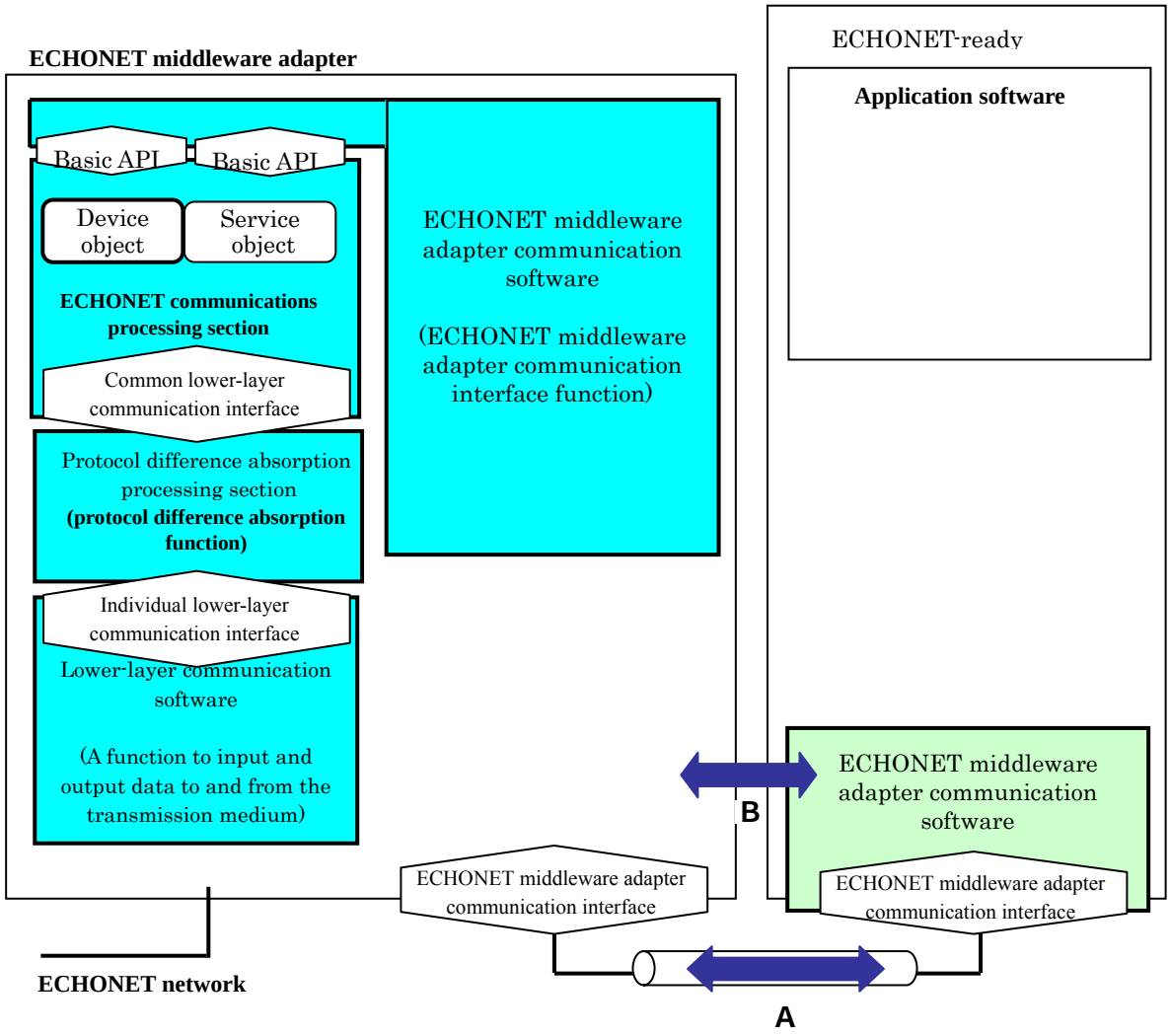



**Fig. 8.6 Data Communication Between the Equipment and ECHONET Middleware Adapter**

A: Mechanical, physical and electrical characteristics of the ECHONET middleware adapter communication interfaces

B: ECHONET middleware adapter communication software protocol

## 8.6.2 Mechanical and physical characteristics of ECHONET middleware adapter communication interfaces

This subsection defines the mechanical and physical characteristics requirements for ECHONET middleware adapter communication interfaces.

(1) Transmission medium

The recommended transmission medium for ECHONET middleware adapter communication interfaces is as follows:

Eight multi-conductor cables (conductor diameter is not specified).

(2) Cable length

In the case of an open collector, the length for which a guarantee must be provided shall be 2 m at the maximum.

(3) Connection style

One-to-one connection of an ECHONET middleware adapter and a corresponding piece of ECHONET-ready equipment.

(4) Connector shape

It is recommended that an 8-pin PH connector be used on the ECHONET-ready equipment side. These specifications do not define requirements for the connector on the ECHONET middleware adapter side (Reference: Fig. 3.11 in Part 7, Chapter 3).

In particular, in the case where connectors are provided based on the assumption that consumers will install ECHONET middleware adapters and replace them as necessary, it is necessary to clearly address the following considerations:

- Prevention of incorrect insertion
- The need to support live insertion and removal
- Power feed class

Therefore, this ECHONET Specification recommends that recommended 9P middleware adapter connectors (MA9/MA9B connectors) which are appropriate from the standpoint of the above-mentioned considerations be used as connectors for Power Feed Class 1 ECHONET-ready devices (MA9/MA9B sockets) and ECHONET middleware adapters (MA9/MA9B plugs).

* MA9/MA9B connectors must only be used for Power Feed Class 1 devices, because using MA9/MA9B connectors as common connectors for both Power Feed Class 1 devices and Power Feed Class 2/Power Feed Class 3 devices may result in problems

such as failures.

* To prevent a situation where a connector does not function because of a power feed capacity difference even though the Power Feed Class 1 range requirement is satisfied, this ECHONET Specification defines connectors for power feed capacities of less than 2000mVA as MA9 connectors and connectors for power feed capacities of 2000mVA or more as MA9B connectors.

There is no quantity requirement for multi-core cables used in conjunction with MA9/MA9B connectors.

It is recommended that manufacturers follow these MA9/MA9B connector specifications when adopting connectors, so that complete inter-compatibility is achieved.

MA9/MA9B Connector Specifications

● Physical Specifications

| Item | Requirements | Remarks |
|---|---|---|
| Number of poles | 9-pole | A home appliance device WakeUP function will be added in the future to the 8 poles which are specified, in the form of a recommendation, in the current ECHONET Middleware Adapter Specifications. (Details of the WakeUP function are to be determined.) |
| Connector pin spacing | 2mm | The spacing was determined based on considerations relating to the supply of power to home appliance devices and adapters and the exchanging of signals. |
| Rated current | 0.5ADC | The rated current is the current for the supply of power to home appliance devices in the ECHONET. |
| Rated voltage | 15VDC | The rated voltage is the voltage for the supply of power to home appliance devices in the ECHONET. |
| Operating temperature range | －20℃ to ＋85℃ | The operating temperature range is the ambient temperature range in which continuous use at the rated voltage and current is permitted. This operating temperature range was determined taking into consideration the possibility that the connector may be used outdoors. |
| Storage temperature range | －40℃ to ＋85℃ | The storage temperature range is the ambient temperature range in which storage in the unloaded condition is permitted. This storage temperature range was determined taking into consideration the possibility that the connector may be stored outdoors. |
| High temperature resistance | - Contact resistance<br>Up to two times the initial specification value<br>- Insulation resistance<br>Initial specification value. | As measured/observed after leaving the connector in an 85±2℃ environment for 500 hours and then in a normal-temperature, normal-humidity environment for 30 minutes. |

| | | |
|---|---|---|
| | - Withstand voltage<br>Initial specification value.<br>- External appearance<br>There shall be no crack, deformation or other abnormal condition. | |
| Low temperature resistance | - Contact resistance<br>Up to two times the initial specification value<br>- Insulation resistance<br>Initial specification value.<br>- Withstand voltage<br>Initial specification value.<br>- External appearance<br>There shall be no crack, deformation or other abnormal condition. | As measured/observed after leaving the connector in a 40±2℃ environment for 500 hours and then in a normal-temperature, normal-humidity environment for 30 minutes. |
| Humidity resistance | - Contact resistance<br>Up to two times the initial specification value<br>- Insulation resistance<br>Initial specification value.<br>- Withstand voltage<br>Initial specification value.<br>- External appearance<br>There shall be no crack, deformation or other abnormal condition. | As measured/observed after leaving the connector in a 60±2℃, 90 to 95%RH environment for 500 hours and then in a normal-temperature, normal-humidity environment for 30 minutes. |
| Contact resistance | 10mΩ or less | The inter-pin contact resistances (excluding the conductor resistances) measured with suitable connectors connected.<br>Measurement frequency: 1000Hz<br>Measurement current: 100mA or less |
| Insulation resistance | 1000MΩ or more | As measured after applying a voltage of 500VDC between the conduction points for 1 minute. |
| Withstand voltage | There shall be no arcing, dielectric breakdown or other abnormal condition. | As confirmed after applying a voltage of 500VAC between the conduction points for 1 minute.<br>Breaking current: 2mA |
| Insulation | Insulation distance: 2.5mm or more<br>See the *2 in Fig. 8.7(b). | There must be an insulation distance of 2.5mm or more between the MA9 plug pins (charging section) and the connector opening (outer wall)(supplementary insulation). |
| Material | RoHS directive-compliant housing<br>UL94 V-0 or higher<br>Must contain PBT glass<br>contact (conduction part)<br>Copper alloy | |
| Shape | Power Feed Class 1: Power feed capacities of less than 2000mVA:<br>MA9 socket (ECHONET-ready equipment side)<br>MA9 plug (ECHONET middleware adapter side)<br>Connection<br><br>Power Feed Class 1: Power feed capacities of 2000mVA or more:<br>MA9B socket (ECHONET-ready equipment side) | The shape must be as shown in Fig. 8.7 (a).<br><br>The shape must be as shown in Fig. 8.7 (b).<br><br>The shape must be as shown in Fig. 8.7 (c).<br><br>The shape must be as shown in Fig. 8.7 (d). |

| | | |
|---|---|---|
| | MA9B plug (ECHONET middleware adapter side)<br>Connection | The shape must be as shown in Fig. 8.7 (e).<br><br>The shape must be as shown in Fig. 8.7 (f). |
| Mechanism to prevent incorrect insertion | See *1 in Fig. 8.7(a). | A mechanism to prevent users from incorrectly inserting connectors must be provided. |
| Live insertion and removal function | See *1 in Fig. 8.7(b). | The safety of devices and adapters must be ensured through the use of a timing function that allows the No.1 pin to contact first during insertion and separates last during removal. |
| Number of times of insertion and removal | 500 times | |
| Locking mechanism | A locking mechanism(half-lock mechanism) shall be provided.<br>See *1 in Fig. 8.7(c). | The unlocking force must be 20 to 40N and it must be possible to confirm that the connector has been firmly inserted. |
| Waterproofing | Groove for O-ring<br>See *2 in Fig. 8.7(c). | A groove for installing an O-ring must be provided to make the connector waterproof and splash proof. |

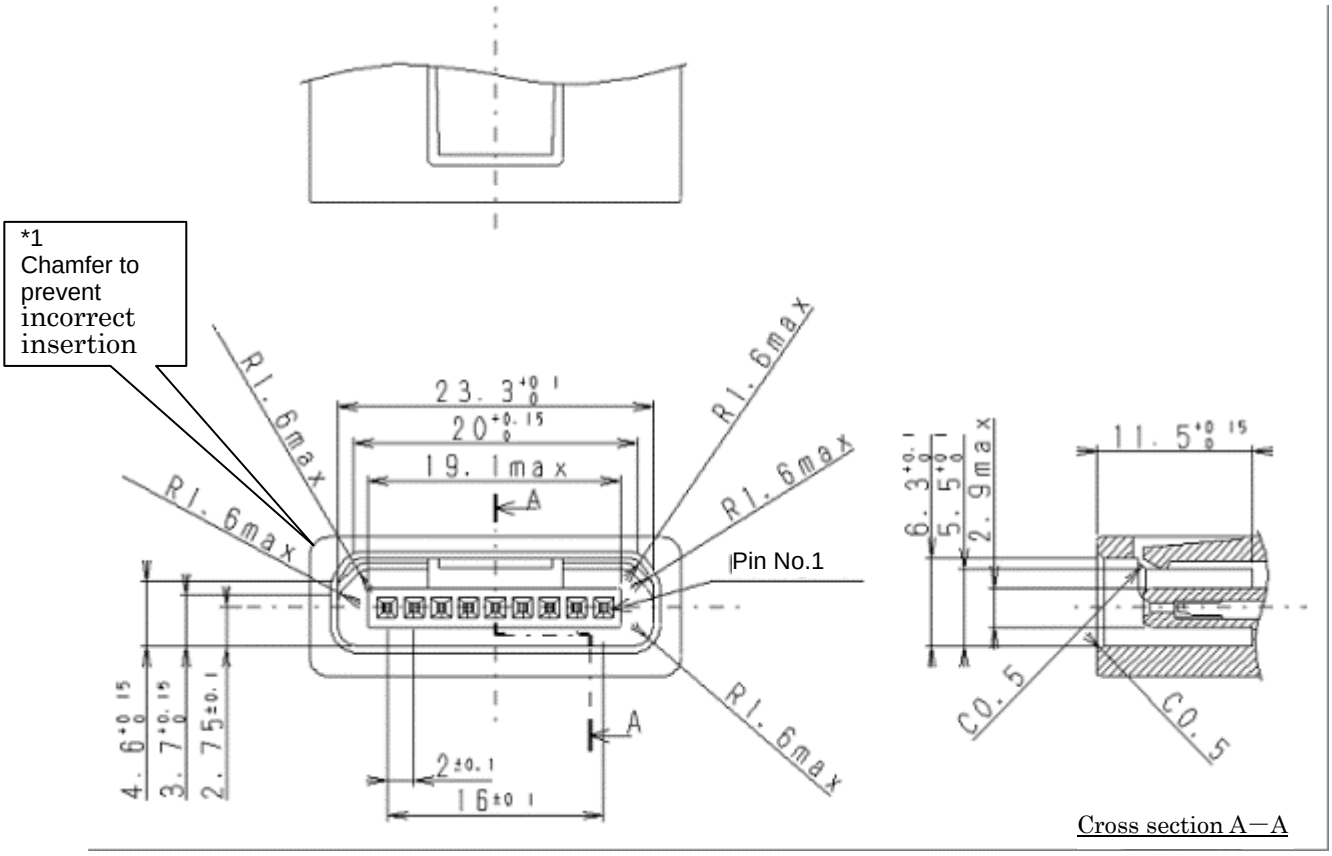



**Fig. 8.7(a) MA9 Socket (ECHONET-ready equipment side)**

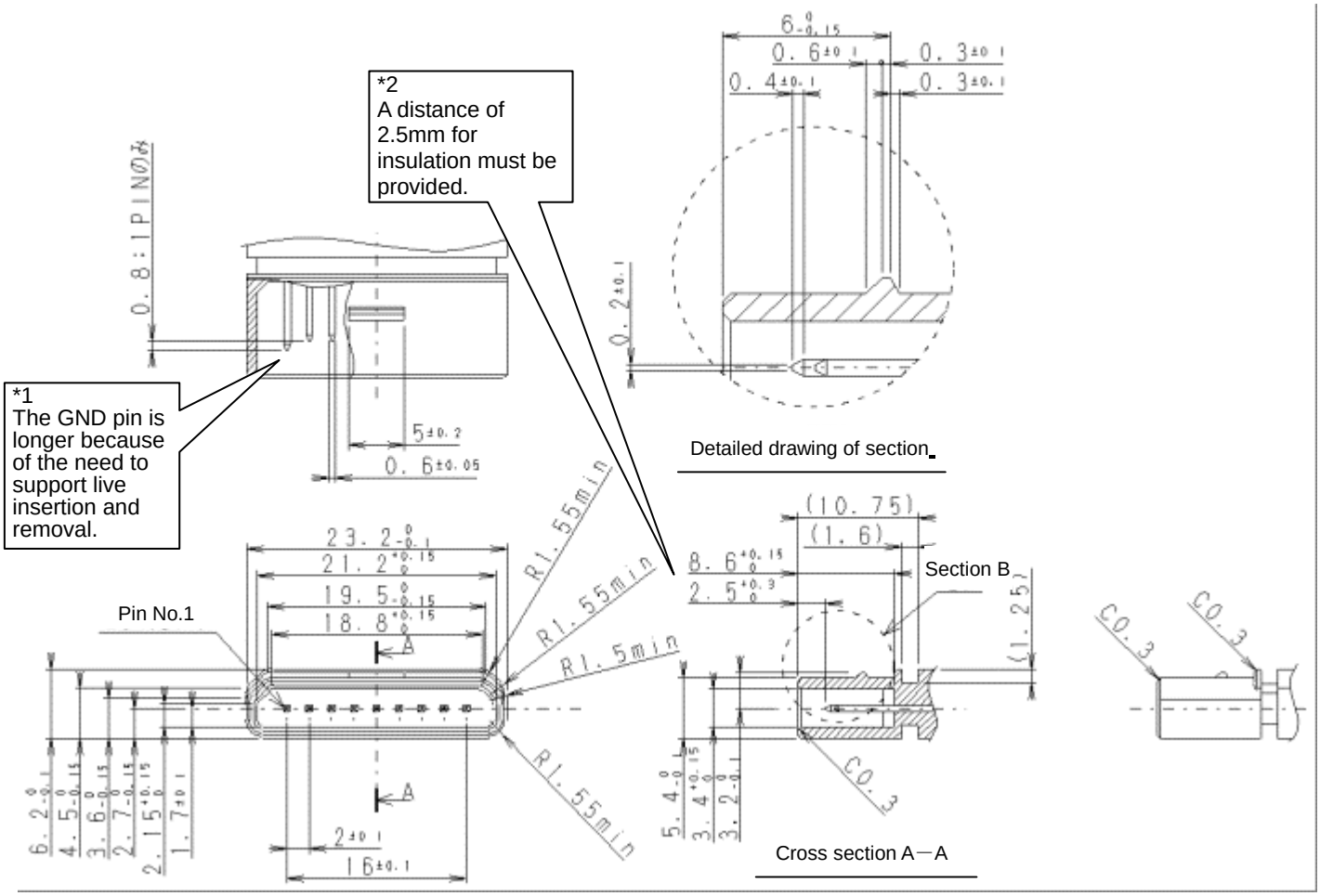



**Fig. 8.7(b) MA9 Plug (ECHONET middleware adapter side)**

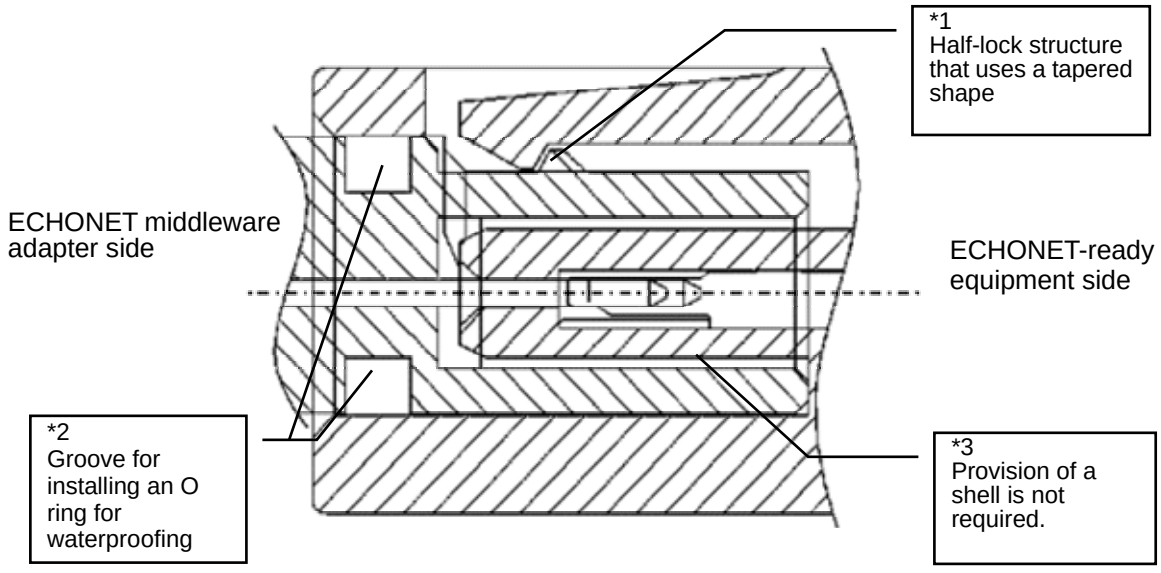
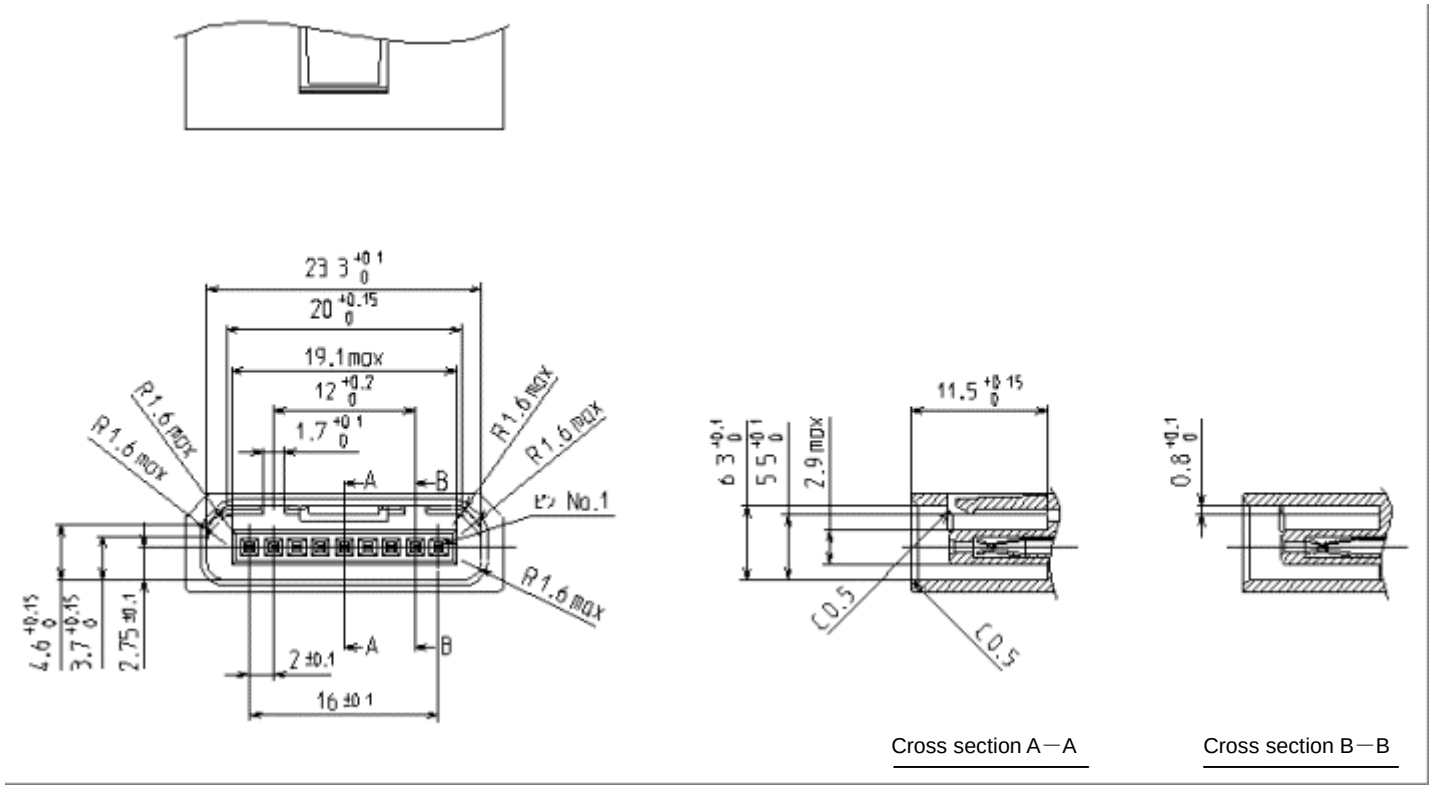


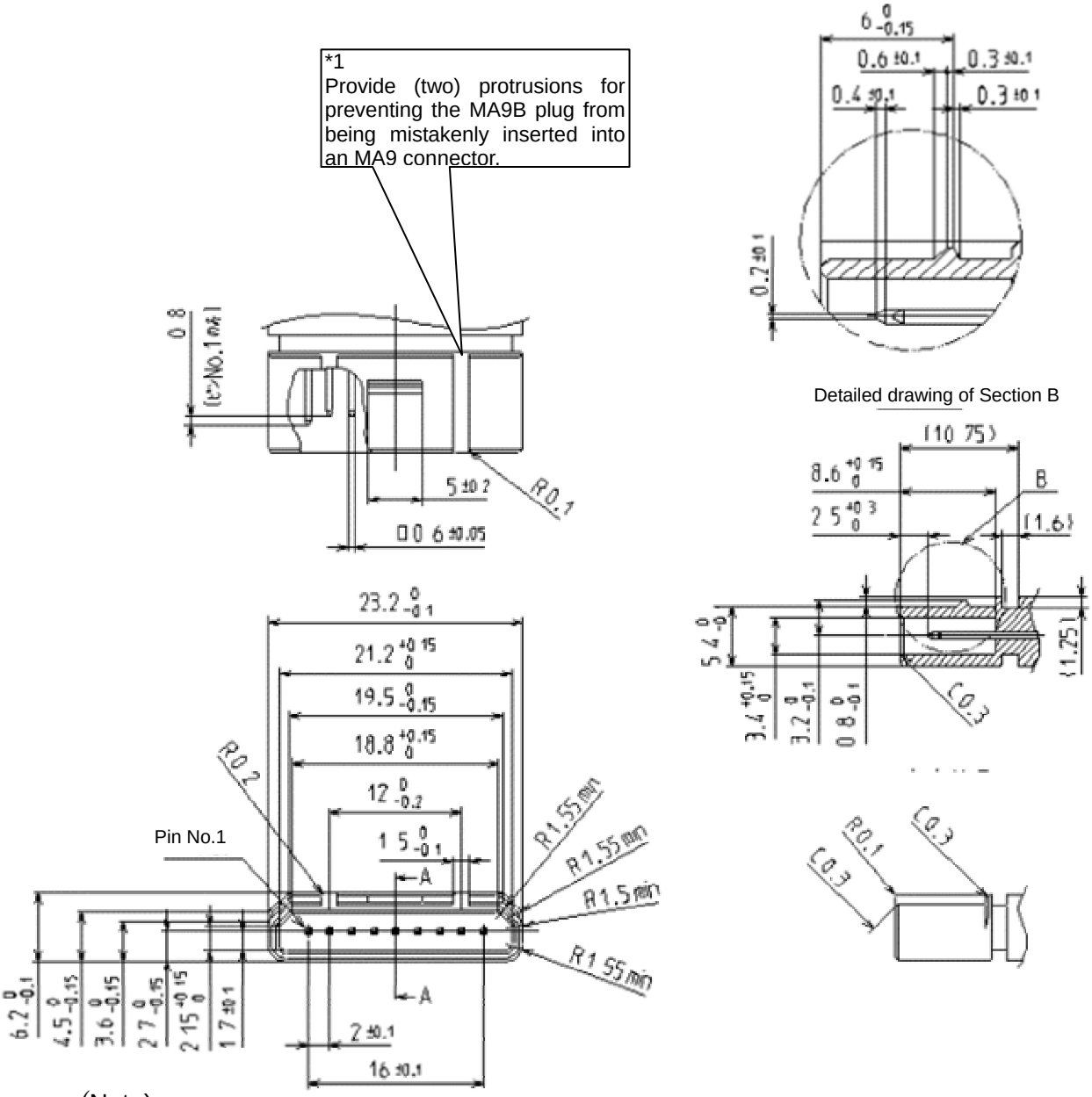

**Fig. 8.7(c) MA9 Connector (connection)**




(Note)

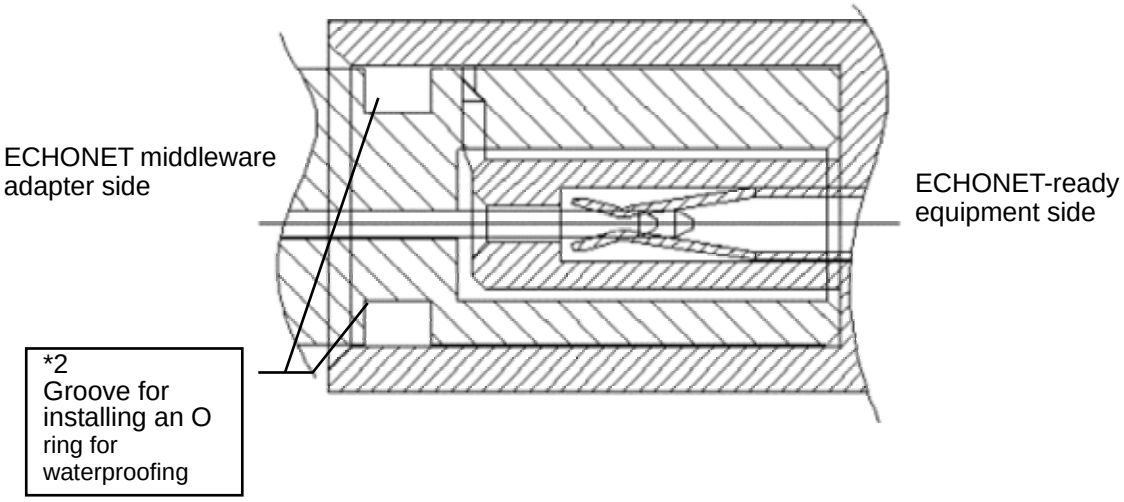
Because the shape of MA9B sockets is the same as that of MA9 sockets except for the (two) concave grooves shown in the “cross section B－B” diagram, it is possible to connect an MA9B socket to an MA9 plug instead of an MA9B plug.

**Fig. 8.7(d) MA9B Socket (ECHONET-ready equipment side)**




(Note)
The shape of MA9B plugs is the same as that of MA9 plugs except for the protrusions. MA9B plugs can only be connected to MA9B sockets equipped with concave grooves for accepting the protrusions. It is not possible to connect an MA9B plug to an MA9 socket.

**Fig. 8.7(e) MA9B Plug (ECHONET middleware adapter side)**




(Note)
The drawing of the connection for "cross section A－A" is not provided, because it is the same as Fig. 8.7(c).

**Fig. 8.7(f) MA9B Connector (connection, cross section B－B)**

(5) Relationship between the connector pins and signals

Fig. 8.8 shows the (recommended) PH connector pin assignment and the (recommended) pin assignment for the case where MA9/MA9B connectors are used.

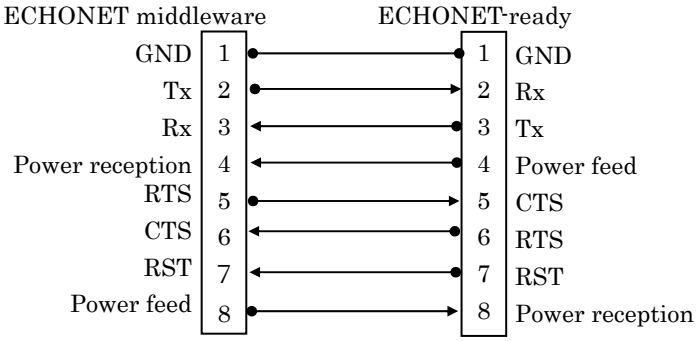



**Fig. 8.8(a) ECHONET Middleware Adapter Communication Interface (Recommended) Pin Assignment**

● Pin assignment for the case where MA9/MA9B connectors are used:

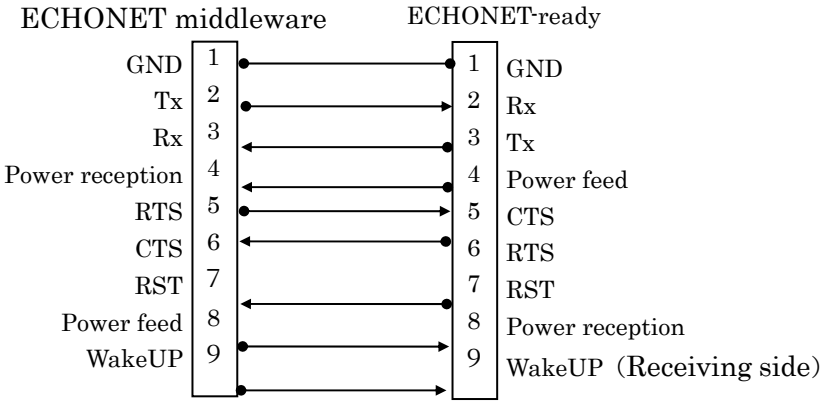



**Fig. 8.8(b) ECHONET Middleware Adapter Communication Interface (Recommended) Pin Assignment 2**

○ Receiving and Supplying Power (See "8.6.3 (5) Specifications for supplying power.")

ECHONET-ready equipment → ECHONET middleware adapter
ECHONET-ready equipment: optional
ECHONET middleware adapter: optional
ECHONET middleware adapter → ECHONET-ready equipment
ECHONET-ready equipment: optional
ECHONET middleware adapter: optional

○ RST (reset)

Reset output from the ECHONET-ready equipment to ECHONET middleware adapter
"Low": adapter's microcomputer stops
"Low→High": reset start

ECHONET-ready equipment: optional
ECHONET middleware adapter: compulsory

○ RTS/CTS (See "8.6.4 (1) Control method.")
ECHONET-ready equipment side: optional
ECHONET middleware adapter side: compulsory

## 8.6.3 Electrical characteristics

This subsection defines the electrical characteristics requirements for ECHONET middleware adapter communication interfaces.

(1) Cable characteristics impedance

Not specified.

(2) Signal transmission speed

For signals used by the equipment interface data recognition service of an ECHONET middleware adapter communication interface, the two transmission speeds specified below shall be implemented. The ECHONET-ready equipment shall be equipped with either of the two transmission speeds.

Transmission speeds: 2400 bps ± 2% / 9600 bps ± 2%

(3) Signal transmission method and waveform of transmitted signals

The signal transmission method and the waveform of transmitted signals shall be as follows (the interface points shall be the connector pins).

① Transmission method: base band transmission
② Waveform: single-current NRZ method
③ Logic

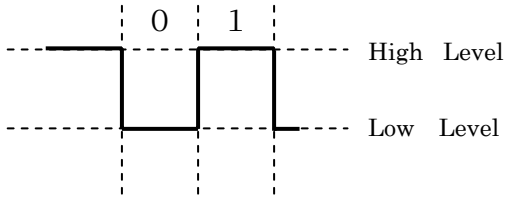



**Fig. 8.9(1) Logic Level**

(4) Output pin specifications

Type 1: open collector
Type 2: 3.3V CMOS

(5) Specifications for supplying power

**Table 8.2 Specifications for Supplying Power – ECHONET-ready Equipment (Class 1)**

| | |
|---|---|
| Supply voltage | 4.5 to 15 V |
| Supply capacity | 1200 mVA or more |

**Table 8.3 Specifications for Supplying Power – ECHONET-ready Equipment (Class 2)**

| | |
|---|---|
| Supply voltage | 4.5 to 5.5 V |
| Supply capacity | 300 mVA or more |

**Table 8.4 Specifications for Supplying Power – ECHONET-ready Equipment (Class 3)**

| | |
|---|---|
| Supply voltage | 3 to 4.5 V |
| Supply capacity | 300 mVA or more |

**Table 8.5 Specifications for Supplying Power – ECHONET Middleware Adapter**

| | |
|---|---|
| Supply voltage | 3.0 to 5.5 V |
| Supply capacity | 100 mVA or more |

(6) Reset

A reset pin (RST) to electrically reset the ECHONET middleware adapter from the ECHONET-ready equipment shall be implemented in the ECHONET middleware adapter. Implementation in the ECHONET-ready equipment shall be optional. The “High,” “Low” and “Low → High” states of the RST pin shall correspond to “normal operation,” “deactivation of the ECHONET middleware adapter” and “reset start,” respectively.

### 8.6.4 Logical requirements

This subsection defines the logical requirements for ECHONET middleware adapter communication interfaces.

(1) Control method

The control method shall be an RTS/CTS-based method. RTS shall correspond to signals to notify the other side that transmission will be started or that no message can be received and CTS shall correspond to signals that the other side sends either to indicate its status as to whether or not messages can be received or to notify that it will start communicating. The RTS/CTS control procedure for transmitting an electronic message shall be as follows:

① If no message can be received, RTS is set to “High Level.” If messages can be received, RTS is set to “Low Level.”

② A check is made before transmission to confirm that CTS is “Low Level.”
(No message is sent if CTS is “High Level.”)

③ Data is output to TXD.

Service requests shall be handled in the order they are transmitted. In the event of a service request collision, the service request from the ECHONET middleware adapter shall be handled first. A CTS status change during a frame transmission shall not require the sending side to abort transmission of the frame. If the sending side aborts transmission of the frame, that frame shall be considered invalid and the prescribed subsequent processing shall be performed.

(2) Synchronization method

Synchronization shall be achieved using a character-by-character start-stop synchronization method. The following requirements shall be satisfied:

① Character composition (see Fig. 8.9(2))　A total of 11 bits

Start bit (ST): 1 bit
Data: 8 bits
Parity: 1 bit
Stop bit (STP): 1 bit

② Data transmission order: LSB first

③ Start bit: logical 0

④ Stop bit: logical 1

⑤ Parity: even number parity

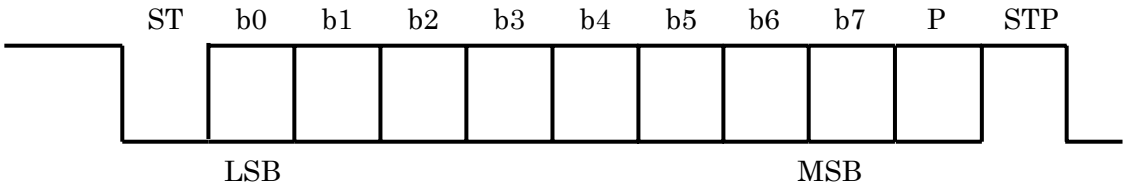



**Fig. 8.9(2) Character Composition**

(3) Timing requirements

The timing requirements for execution of the equipment I/F data recognition service shall be as shown in Fig. 8.10 and Table 8.6. The adapter shall send a request frame and the ECHONET-ready equipment shall receive it and send back a response frame.

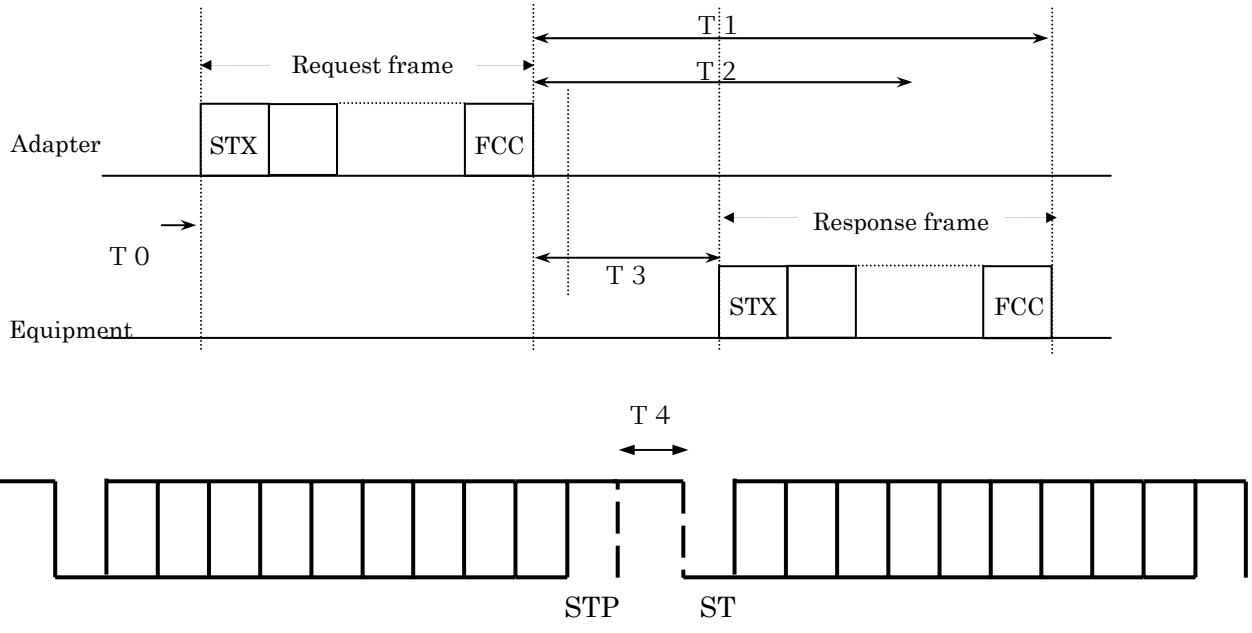



**Fig. 8.10 Timing Requirements**

**Table 8.6 Timing Requirements**

| Symbol | Applicable to: | Name | Requirement |
|---|---|---|---|
| T0 | ECHONET middleware adapter / ECHONET-ready equipment | Receiving side frame synchronization confirmation (frame completion / start position detection) | 9600 bps or less: No message is received for 10 msec or more. More than 9600 bps: No message is received for a period equal to 3-character composition length or more. |
| T1 | ECHONET middleware adapter | Response waiting timeout | 300 msec from the end of the frame sent by itself |
| T2 | ECHONET middleware adapter | Retransmission inhibition time | 300 msec from the end of the frame sent by itself |
| T3 | ECHONET-ready equipment | Response transmission inhibition time (T4 confirmation time) | 9600 bps or less: 10 msec or more More than 9600 bps: a period equal to 3-character composition length or more |
| T4 | ECHONET middleware adapter / ECHONET-ready equipment | Sending side character composition interval | 9600 bps or less: less than 10 msec More than 9600 bps: less than a period equal to 3-character composition length |

(4) Frame composition

Each ECHONET middleware adapter communication interface frame shall consist of a control code (STX) located at the head end, a check code (FCC) located at the tail end and data inserted between them. The content of the data differs depending on which of the

protocols described in “8.6.5 ECHONET middleware adapter communication software protocols” is used.

(5) Control code (STX)

Header code. This shall be fixed at 0x02.

(6) Data part (DATA)

The data part shall be the part sandwiched between the ECHONET middleware adapter communication interface STX and the FCC. The specific content varies depending on what type is selected for the ECHONET middleware adapter communication interface (for details, refer to “8.6.6 ECHONET middleware adapter communication software protocol”).

(7) Check code (FCC)

The check code, which is used to detect frame transmission errors, is the 2’s complement of the total value of the characters contained in the data part.

(8) Frame completion detection

If, after a stop bit is detected, the start bit of the next character is not detected for a period that is shorter than 3 field lengths (this period shall be 10 msec in the case of a transmission speed that is less than 9600 bps), it shall be assumed that the frame has completed.

(9) Error detection

The receiving terminal shall detect the following types of errors:

○ Byte reception errors

Each byte shall have a parity bit. The parity shall be an even number parity. If a parity error is detected while receiving a byte during a frame reception session, the receiving terminal shall consider the frame being received as an invalid frame and discard it (the frame shall be discarded upon completion of reception or upon reception timeout).

○ FCC errors

An FCC check shall be made every time the reception of a frame is completed. FCC shall be the 2’s complement of the sum of the content of the data part.

If the FCC code is valid, the frame shall be considered as a valid frame.

If the FCC code is invalid, the frame shall be considered as invalid and discarded.

(10) Error control

No ACK/NAK error control shall be provided.

## 8.6.5 ECHONET middleware adapter communication software protocols

The ECHONET middleware adapter communication software protocol differs depending on the type of ECHONET middleware adapter. The requirements for each of the following 4 protocols are defined in the following sections, with one section dedicated to each:

① Equipment interface data recognition service software protocol
② Communication software protocol for object generation type
③ Communication software protocol for peer-to-peer type

## 8.7 Equipment Interface Data Recognition Service

This section provides the service specifications for recognizing the equipment interface data of ECHONET-ready equipment connected to an ECHONET middleware adapter.

### 8.7.1 Frame composition for the equipment interface data recognition service

Figure 8.12 shows the frame composition for executing the “equipment interface data recognition service.” The frame type code (FT), command number code (CN), frame number code (FN), data length code (DL) and frame data (FD) sections comprise the ECHONET middleware adapter communication interface protocol data.

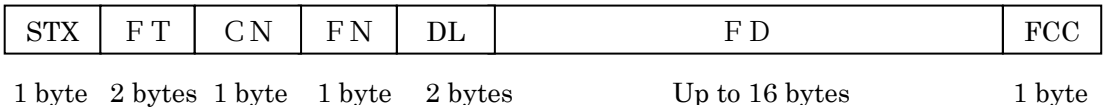



**Fig. 8.12 Equipment Interface Data Recognition Service Software Protocol**

(1) STX (control code)
Header code. For equipment interface data recognition service software protocols, this shall be fixed at 0x02.

(2) FT (frame type)
Indicates the frame type. Frames for equipment interface data recognition shall be fixed at 0xFFFF.

(3) CN (command number code)
The command number code shall be a 1-byte code that specifies a defined service (equipment interface data recognition service software protocols for ECHONET middleware adapter communication interfaces).

**Table 8.7 Equipment Interface Data Recognition Service Command Codes**

| 4 lowest-order bits \ 4 highest-order bits | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | Equipment interface data request | | | | | | | | Equipment interface data response | | | | | | | |
| 1 | Equipment interface data recognition notification | | | | | | | | Equipment interface data recognition notification acceptance response | | | | | | | |
| 2 | - | | | | | | | | - | | | | | | | |
| 3 | - | | | | | | | | - | | | | | | | |
| 4 | - | | | | | | | | - | | | | | | | |
| 5 | - | | | | | | | | - | | | | | | | |
| 6 | - | | | | | | | | - | | | | | | | |
| 7 | - | | | | | | | | - | | | | | | | |
| 8 | - | | | | | | | | - | | | | | | | |
| 9 | - | | | | | | | | - | | | | | | | |
| A | - | | | | | | | | - | | | | | | | |
| B | - | | | | | | | | - | | | | | | | |
| C | - | | | | | | | | - | | | | | | | |
| D | - | | | | | | | | - | | | | | | | |
| E | - | | | | | | | | - | | | | | | | |
| F | - | | | | | | | | - | | | | | | | |
| | | Reserved for future use | | | | | | | | Reserved for future use | | | | | | |

Note: "-" Equipment interface data recognition service command (reserved for future use)

(4) FN (frame number)

A number assigned by the requesting side (0x01 to 0xFF). A response frame must have the same number as the corresponding request frame.

For ECHONET-ready equipment that is not designed to use frame numbers, this shall be fixed at 0x00.

(5) DL (data length code)

The data length code shall be a 2-byte code that indicates the size of the frame data (FD) section that follows. The size shall be measured in bytes and be expressed using the hexadecimal notation. For this service, the maximum value of a DL shall be 0x0010. The data order shall be big-endian.

(6) FD (frame data)

The frame data section is a field of data that is defined by the frame type (FT) and command number code (CN). The data order for data having 2 bytes or more shall be big-endian. The specific composition shall be defined on a CN-by-CN basis.

(7) FCC (frame check code)
A 1-byte check code shall be used as the frame check code.

## 8.7.2 Commands for the equipment interface data recognition service

This subsection describes the commands used for the equipment interface data recognition service for each of the ECHONET middleware adapter communication interfaces.

(1) Equipment interface data request and response commands (Required)
These commands are used to allow the ECHONET middleware adapter to acquire information on the type of ECHONET middleware adapter communication interface implemented in the ECHONET-ready equipment.

◇ Direction of request commands
ECHONET middleware adapter → ECHONET-ready equipment

◇ Format for request commands

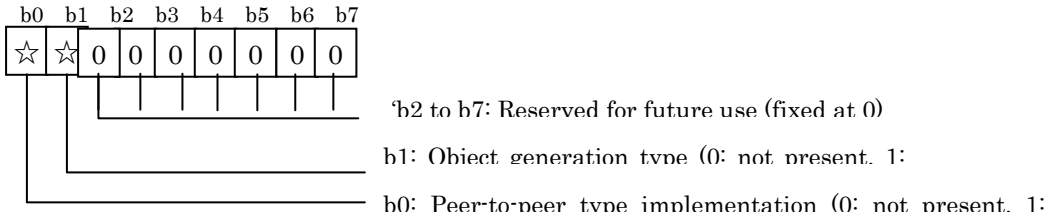
| 1 byte | 2 bytes | 1 byte | 1 byte | 2 bytes | 1 byte |
|---|---|---|---|---|---|
| STX | FT | CN | FN | DL | FCC |

STX : 0x02
FT : 0xFFFF
CN : 0x00
FN : 0x＊＊
DL : 0x0000
FCC : 0x＊＊

◇ Format for response commands

| 1 byte | 2 bytes | 1 byte | 1 byte | 2 bytes | 1 byte | 1 byte | n bytes | 1 byte |
|---|---|---|---|---|---|---|---|---|
| S T X | F T | CN | F N | DL | FD(0) | FD(1) | FD(2) | FCC |

S T X ：0x02
F T ：0xFFFF
C N ：0x80
F N ：Value at the time of the request (0x00 if the function is not implemented)
D L ：n + 2
F D(0) ：Middleware adaptor type of ECHONET-ready equipment
(0x00 if the function is not implemented)




FD (1): transmission speed

| | |
|---|---|
| 0x00 | 2400 bps |
| 0x01 | 4800 bps |
| 0x02 | 9600 bps |
| 0x03 | 19.2 Kbps |
| 0x04 | 38.4 Kbps |
| 0x05 | 57.6 Kbps |
| 0x06 | 115 Kbps |
| 0x07– 0xFF | Not defined (reserved for future use) |

F D(2) ：Method-specific definition area (n bytes)
－Object generation type: Not present (bytes)
－Peer to peer type

| FD(2) | | | |
|---|---|---|---|
| Interface data | ECHONET manufacturer | Equipment type code | Type code |

Interface data (1 byte)

| b0 | b1 | b2 | b3 | b4 | b5 | b6 | b7 |
|---|---|---|---|---|---|---|---|
| * | * | * | * | * | * | * | * |

b7, b6: Communication sequence
b7, b6 = 0,1: Adapter polling
1,1: Bidirectional
1,0 0,0: Not defined
b5: Flow control 0: without 1: with
b4: ACK/NAK response 0: without 1: with
b3: Downloading 0: without 1: with
b2 to b0: Not defined

ECHONET manufacturer code (3 bytes)
as per the ECHONET Specification
Equipment type code (2 bytes)
First byte: class group, second byte: class code
Type code (2 bytes)
Manufacturer code * independently defined for each equipment code

FCC : 0x**

(2) Equipment interface data recognition notification command and equipment interface data recognition notification acceptance response command (Required)

The ECHONET middleware adapter shall notify the ECHONET-ready equipment as to whether or not it can handle the communication method specified by the ECHONET-ready equipment using an equipment interface data recognition notification. This notification shall be made within 500 msec (Ti) after the receipt of the equipment interface data. If the ECHONET middleware adapter can handle the communication method specified by the ECHONET-ready equipment, it shall so notify and wait for an equipment interface data recognition notification acceptance response. If the ECHONET middleware adapter cannot handle the communication method specified by the ECHONET-ready equipment, it shall so notify and enter the “connection not possible” state.

Upon receipt of an equipment interface data recognition notification, the ECHONET-ready equipment shall check the content of the notification, and if the value indicates that the ECHONET middleware adapter can handle the communication method specified by the ECHONET-ready equipment, the ECHONET-ready equipment shall send within 500 msec (Ti) an equipment interface data recognition notification acceptance response to the ECHONET middleware adapter and enter the “unconfirmed” state. If the value indicates

that the ECHONET middleware adapter cannot handle the communication method specified by the ECHONET-ready equipment, the ECHONET-ready equipment shall enter the state to wait for an equipment interface data request.

If the ECHONET middleware adapter receives an equipment interface data recognition notification acceptance response within 500 msec after the transmission of an equipment interface data recognition notification, it shall enter the “unconfirmed” state.

If the ECHONET middleware adapter does not receive an equipment interface data recognition notification acceptance response within 500 msec after the transmission of an equipment interface data recognition notification, it shall send an equipment interface data request again.

◇ Direction of request commands

ECHONET middleware adapter → ECHONET-ready equipment

◇ Format for request commands

| 1 byte | 2bytes | 1byte | 1 byte | 2bytes | 1 byte | 1 byte |
|---|---|---|---|---|---|---|
| STX | FT | CN | FN | DL | FD(0) | FCC |

STX ：0x02
FT ：0xFFFF
CN ：0x01
FN ：0x＊＊
DL ：0x0001
FD(0) ：Result
0x01 ：Not supported
0x00 ：Supported
0x02 ：Present speed supported (Specified speed not supported)
0x11 ：Peer-to-peer type supported in case multiple specified
0x12 ：Object generation type supported incase multiple specified.
FCC ：0x＊＊

◇ Format for response commands

| 1 byte | 2bytes | 1 byte | 1 byte | 2 | 1 byte |
|---|---|---|---|---|---|
| STX | FT | CN | FN | DL | FCC |

STX ：0x02
FT ：0xFFFF
CN ：0x81

ＦＮ　：Value at the time of the request (0x00 if the function is not implemented)
ＤＬ　：0x0000
ＦＣＣ　：0x＊＊

## 8.7.3 Equipment interface data recognition service sequence

Figure 8.13 shows the operation sequence of the service. Figure 8.14 shows how the ECHONET middleware adapter and ECHONET-ready equipment change their states. As shown in these figures, both the ECHONET middleware adapter and ECHONET-ready equipment stay in the “unrecognized” state until the equipment interface data recognition service has been successfully executed and shall attempt to execute the equipment interface data recognition service using one (available) transmission speed after another. The ECHONET-ready equipment’s communication method is confirmed (recognized) by the equipment interface data recognition service and that method is used thereafter to perform communication.

To provide for cases in which both the ECHONET-ready equipment and ECHONET middleware adapter are reset after a start (or cases in which either one is reset), it is necessary to provide a system, for each communication method, that performs an appropriate recovery action (such as repeating attempts) when an “unsuccessful communication” state (such as that due to imperfect synchronization) is detected, and if the intended result is not achieved after the action, makes a shift to the “unrecognized” state as shown in Fig. 8.14.

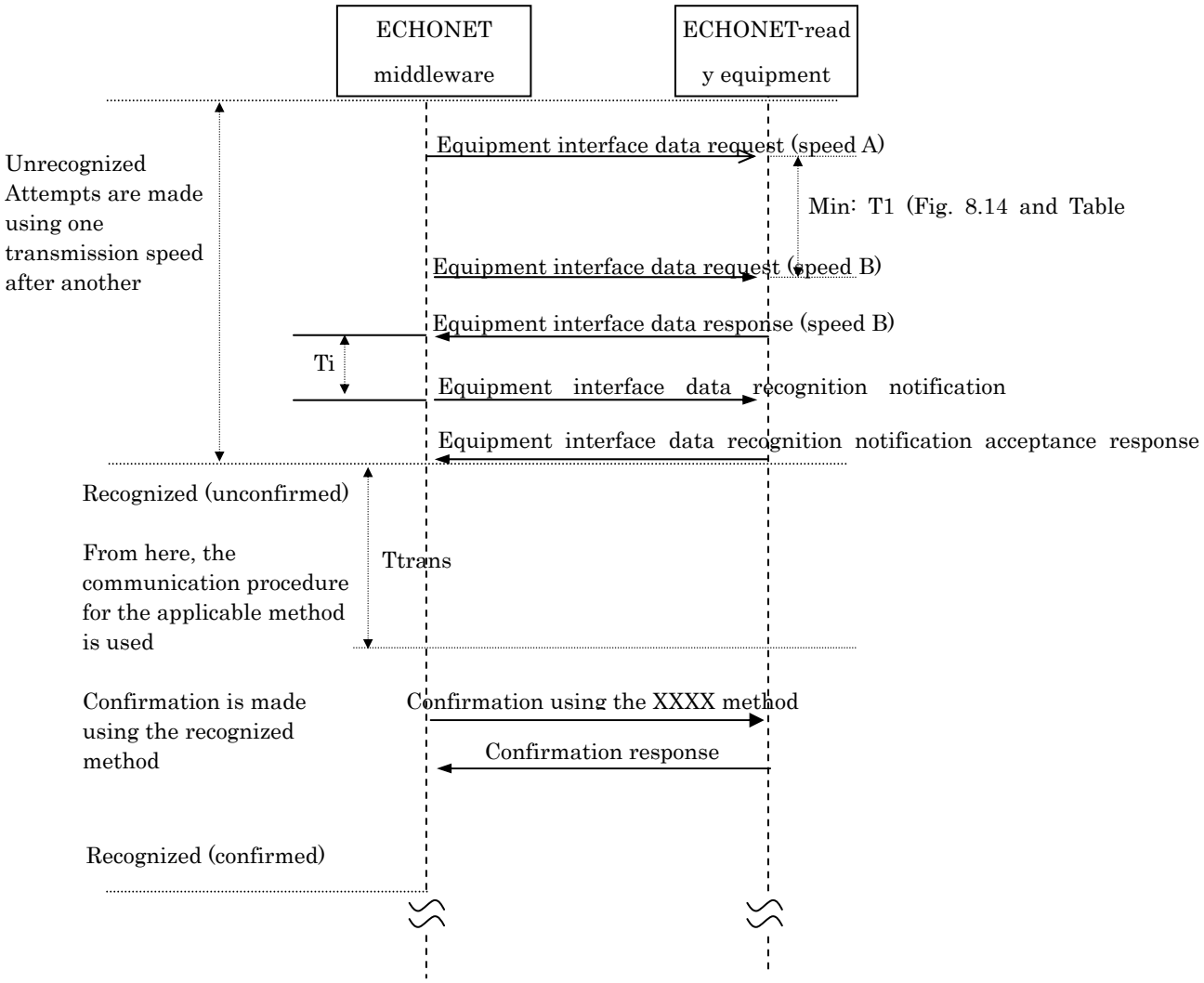



**Fig. 8.13 Operation Sequence**

Ttrans: The transition time to shift to the respective method (specifications). This shall be 500 msec or more.

Ti: Max. 500 msec.

If the adapter cannot properly recognize the response, it shall send the recognition service again after the T1 period (Fig. 8.10) has elapsed.

## 8.7.4 Status change diagram for all types

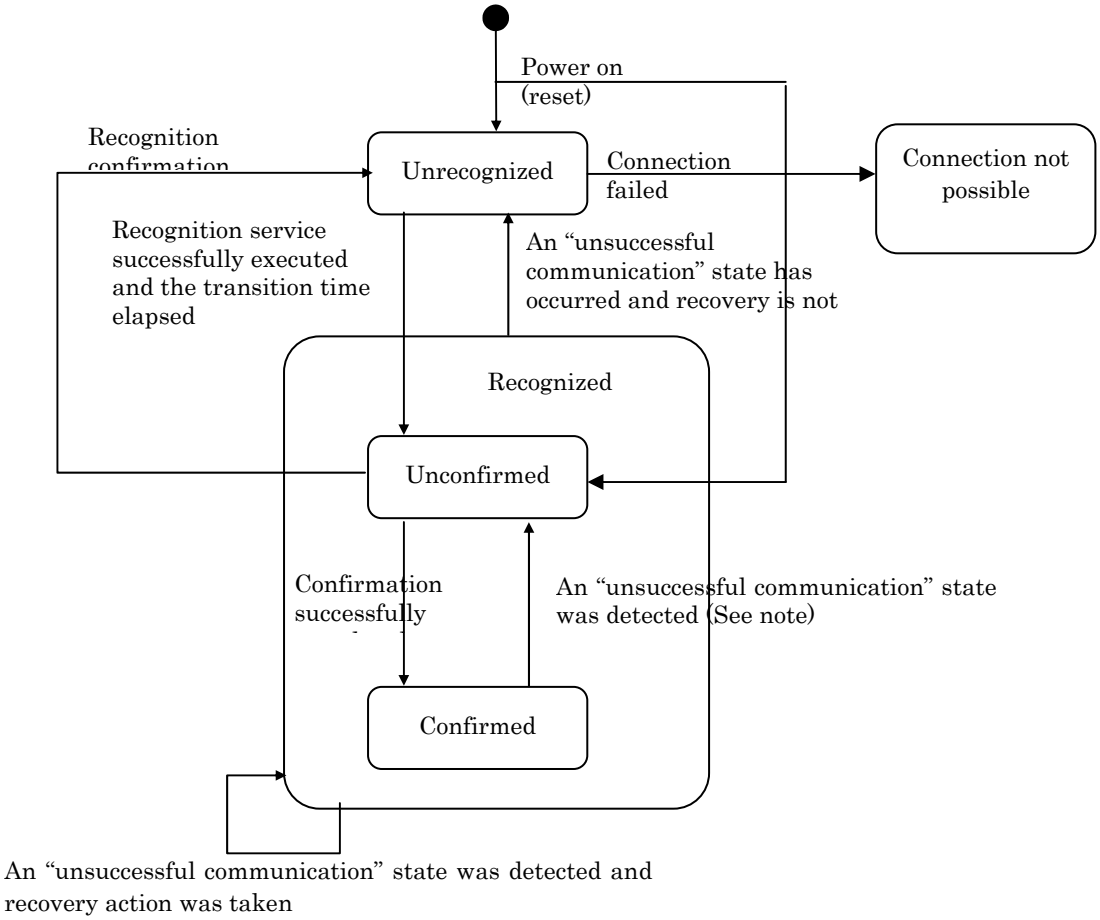



**Fig. 8.14 Status Change Diagram**

“Unsuccessful communication” state: as per the communication procedure for the respective method.

Note) The ECHONET middleware adapter may perform the confirmation processing again upon detection of an “unsuccessful communication” state.

Whether to start from the “unrecognized” state or from the “recognized” state upon power ON shall be appropriately determined based on the implementation.

**Table 8.8 Definition of States**

| State | Definition | | Notes |
|---|---|---|---|
| Unrecognized | The process of recognizing that the communication method has not been completed.<br><br>After the detection of an "unsuccessful communication" state. | ECHONET-ready equipment side:<br><br>Waiting for a recognition service request from the adapter. If the equipment interface data recognition notification from the ECHONET middleware adapter indicates that the adapter cannot handle the communication method specified by the ECHONET-ready equipment, the ECHONET-ready equipment will enter the state to wait for an equipment interface data request. If the adapter can handle the communication method specified by the ECHONET-ready equipment, the ECHONET-ready equipment will send an equipment interface data recognition notification acceptance response and enter the "unconfirmed" state. | |

| | | | |
|---|---|---|---|
| | | ECHONET middleware adapter side:<br><br>Attempting to make a recognition service request to the ECHONET-ready equipment.<br>Continues to make attempts on an as-available basis at certain intervals until the recognition process is completed. Repetitive attempts using two or more speeds (A, B, ...) .<br>If an equipment interface data response is received and communication in accordance with that response is possible, an equipment interface data recognition notification is sent to the ECHONET-ready equipment to so notify, and after receiving the corresponding acceptance response, the “unconfirmed” state is entered.<br>If communication using the received information is not possible, an equipment interface data recognition notification is sent to the ECHONET-ready equipment to so notify and the “connection not possible” state is entered. | The order in which the available speeds are repeated, the number of attempts and the interval (Min. T1 Fig. 8.15) are beyond the scope of these specifications. The optimal method for the characteristics of the adapter shall be used. |

| | | | | |
|---|---|---|---|---|
| Connection not possible | | The state to which a shift is made if there is no communication method that is acceptable to both the ECHONET middleware adapter and the ECHONET-ready equipment. | The "communication not possible" abort processing is performed. | |
| Recognized | Unconfirmed | The communication method recognition process has been completed and communication is being performed using the communication procedure for the applicable method. | Whether or not it is possible to communicate successfully using the communication procedure for the applicable method has not been confirmed.<br><br>ECHONET middleware adapter side:<br><br>Confirmation is made using the confirmation method for the applicable method. | If, after detecting an "unsuccessful communication" state and performing an appropriate recovery processing, recovery is not possible, a shift to the "unrecognized" state shall be made. |
| | confirmed | | It has been confirmed that communication is possible using the communication procedure for the applicable method. | |

## 8.7.5 Error processing

If a state should occur that falls under any one of Items (a) to (c) below, the ECHONET middleware adapter shall give the node profile object or device object a setting value indicating the occurrence of such an abnormal state and make an error announcement to the other nodes. In cases where an LED is provided as an indication means (display section) as described in Subsection 8.3.2, the LED indication shall change to the "abnormal state" indication.

(a) Communication not possible

The equipment interface data recognition service has determined that communication with the ECHONET-ready equipment is not possible. After setting the error property (EPC = 0x89) of the node profile object to 0x03E9 (middleware adapter recognition error), the

error status property (EPC = 0x88) is set to 0x41.

(b) Setting error

The setting of a parameter during the “confirmed” state has failed and it has been determined that continuing the setting operation is not possible. The error property (EPC = 0x89) of the node profile object is set to one of the 3 values listed below depending on the error-causing factor, and the error status property (EPC = 0x88) is set to 0x41. The factors that are considered as error-causing factors differ among communication methods.

Error code:
0x03EA: Object error
0xE3EB: Adapter initialization error
0x03EC: Other setting error

(c) Unsuccessful communication

Cases in which communication with the ECHONET-ready equipment is not possible during the “confirmed” state. After setting the device object error property (EPC = 0x89) for the equipment in question to 0x03E9 (unsuccessful communication), the error status property (EPC = 0x88) is set to 0x41. The definition of “unsuccessful communication” state differs among communication methods.

## 8.8 Object Generation Type

This section defines the communication software protocol for cases in which the communication interface between the ECHONET middleware adapter and ECHONET-ready equipment is an object generation type interface.

Middleware adapters are divided into the following 2 types:

Basic middleware adapter
Advanced middleware adapter

An advanced middleware adapter is capable of handling, on the middleware adapter interface, address information on other nodes located in the ECHONET network. A basic middleware adapter is not capable of handling, on the middleware adapter interface, address information on other nodes located in the ECHONET network. An advanced middleware adapter is also capable of distinguishing whether or not there is a vacancy in relation to elements before handling array properties. In addition to the above-mentioned advanced middleware adapter functions, an advanced middleware adapter shall have all the functions required for a basic middleware adapter.
With regard to communication definition objects, an advanced middleware adapter may implement everything (optional functions).

In cases where a piece of ECHONET-ready equipment that demands a level of processing that corresponds to an advanced middleware adapter is connected to a basic middleware adapter, the equipment shall satisfy the following requirements:

* The equipment shall indicate that the status of other nodes cannot be referenced or altered:
* Notifications to be sent with the addressees specified shall be sent as simultaneous broadcasting notifications.
* Responses to requests from other nodes shall be sent using basic middleware adapter communication frames.

This version of the ECHONET Specification defines the requirements for basic middleware adapters taking into consideration the necessary support in relation to advanced middleware adapters.

The basic middleware adapter must be capable of internally generating at least 3 device objects and managing them. In the case where the number of device objects the basic middleware adapter can manage is 3, it is necessary to set aside at least 1KB of memory space to store property values.

To allow the basic middleware adapter to generate and manage 4 or more device objects, it is necessary to use optional commands and set aside at least 2KB of memory space to store property values.

A basic middleware adapter must be capable of internally managing up to 3 device objects. In addition, it is necessary to provide at least 1 KB as the area to store property values.

### 8.8.1 Frame composition for object generation type interfaces

Figure 8.16 specifies the composition for object generation type software protocols (frame). The frame type code (FT), command number code (CN), frame number code (FN), data length code (DL) and frame data (FD) sections comprise the ECHONET middleware adapter communication interface protocol data. The composition shown in Fig. 8.15 is the same as that for the equipment interface data recognition service.

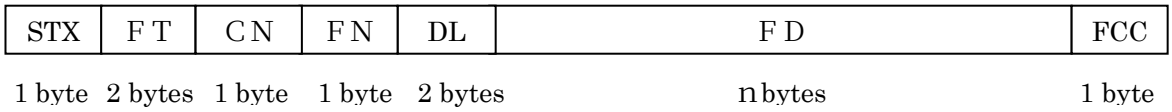



**Fig. 8.15 Object Generation Type ECHONET Middleware Adapter Communication Software Protocol**

(1) STX (control code)
Control code. This shall be fixed at 0x02.

(2) FT (frame type)
Indicates the frame type.

b15 to b12: version
Indicates the frame version number. This shall be 0000.
b11 to b8: Reserved for future use (0000)
b7 to b0: type
Indicates the frame type for various commands.
0x00: Equipment interface data confirmation frame
0x01: Adapter initialization frame
0x02: object construction frame
0x03: Basic regular ECHONET frame
0x04: Advanced regular ECHONET frame
0x05 to 0xDF: Reserved for future use
0xE0 to 0xFE: User defined area

0xFF: Error notification frame

It shall not take the same value as the equipment interface data recognition frame specified below.

0xFFFF: Equipment interface data recognition frame

(3) CN (command number code)

The command number code shall be a 1-byte code that specifies a defined service (object generation type software protocols for ECHONET middleware adapter communication interfaces). This version of the ECHONET Specification defines the commands shown in Table 8.9. Codes with no specific function assigned shall be reserved for future use.

**Table 8.9 Object Generation Type Interface Command Codes**

| Frame Type (FT) | Command Number Code (CN) | Command Name | |
|---|---|---|---|
| Equipment interface confirmation mode<br>Equipment interface data confirmation frame (0x0000) is used | 0x00 | Equipment interface data confirmation request | Required |
| | 0x80 | Equipment interface data confirmation response | Required |
| Adapter initialization mode<br>Adapter initialization frame (0x0001) is used | 0x01 | Adapter initialization setting request | Required |
| | 0x81 | Adapter initialization setting response | Required |
| | 0x02 | Adapter initialization completion notification | Required |
| | 0x82 | Adapter initialization completion notification acceptance response | Required |
| Object construction mode<br>Object construction frame (0x0002) is used | 0x00 | Equipment enquiry request | Required |
| | 0x80 | Equipment enquiry response | Required |
| | 0x01 | Equipment enquiry completion notification | Required |
| | 0x81 | Equipment enquiry completion notification acceptance response | Required |
| | 0x02 | Adapter startup notification | Required |
| | 0x82 | Adapter startup notification acceptance response | Required |
| | 0x03 | Equipment enquiry request with object specified | Opptional*1 |
| | 0x83 | Equipment enquiry response with object specified | Opptional*1 |
| ECHONET communication mode<br>Basic regular ECHONET frame (0x0003) is used | 0x10 | Equipment status access request | Required |
| | 0x90 | Equipment status access response | Required |
| | 0x11 | Equipment status notification request | Required*2 |
| | 0x91 | Equipment status notification response | Required |
| | 0x12 | Element designation equipment status access request | Required |

| | | | |
|---|---|---|---|
| | 0x92 | Element designation equipment status access response | Required |
| | 0x13 | Element designation equipment status notification request | Required *2 |
| | 0x93 | Element designation equipment status notification response | Required |
| | 0x14 | Object access request | Required *2 |
| | 0x94 | Object access response | Required |
| | 0x20 | Equipment status access request (all) | Optional |
| | 0xA0 | Equipment status access response (all) | Optional |
| | 0x21 | Equipment status access UP request (all) | Optional |
| | 0xA1 | Equipment status access UP response (all) | Optional |
| | 0x22 | Equipment status notification request (all) | Optional |
| | 0xA2 | Equipment status notification response (all) | Optional |
| | 0x23 | Object access request (all) | Optional |
| | 0xA3 | Object access response (all) | Optional |

*1: Required in the case where the maximum number of device objects that can be generated is 4 or more.

*2: Optional for the device side.

(4) FN (frame number)

A number assigned by the requesting side (0x01 to 0xFF), according to the order. A response frame must have the same number as the corresponding request frame.

For ECHONET-ready equipment that is not designed to use frame numbers, this shall be fixed at 0x00.

(5) DL (data length code)

The data length code shall be a 2-byte code that indicates the size of the frame data (FD) section that follows. The size shall be measured in bytes and be expressed using hexadecimal notation. For example, if the FD section has 20 bytes, the DL is 0x0014, which indicates 20 bytes. The data order shall be big-endian.

(6) FD (frame data)

The frame data section is a field of data that is defined by the frame type (FT) and command number code (CN). The data order for data having 2 bytes or more shall be big-endian. The specific composition shall be defined on a CN-by-CN basis.

(7) FCC (frame check code)

A 1-byte check code shall be used as the frame check code.

## 8.8.2 Internal services of adapters

For middleware adapters, settings on a property-by-property basis can be made as to whether the values are to be stored in the adapter (home node properties) or are to go through the adapter without being stored. Therefore, this subsection defines internal services for adapters that instruct the adapter on how to handle services received from other nodes such as Set/SetM (equipment status alteration) and Get/GetM (equipment status confirmation). These internal services are IASet, IASetup, IASetM, IASetMup, IAGet, IAGetup, IAGetM and IAGetMup. When a Set is received, an IASet or IASetup shall be performed and when a SetM is received, an IASetM or IASetMup shall be performed. When a Get is received, an IAGet or IAGetup shall be performed and when a GetM is received, an IAGetM or IAGetMup shall be performed.

| | |
|---|---|
| A) Services that store values in the adapter<br>To reduce the communication load on the ECHONET-ready equipment, responses are made by the adapter. | IASet, IASetM, IAGet, IAGetM |
| B) Services that do not store values in the adapter<br>Properties for which real-time handling is required are passed through the adapter and responses are made by the ECHONET-ready equipment. | IASetup, IASetMup, IAGetup, IAGetMup |

Whether to use an A) or B) service can be specified on a property-by-property basis (Acquired as equipment enquiry data from the ECHONET-ready equipment by means of an object construction command).

Requirements for IASetM and IAGetM will be specified in future versions of the ECHONET Specification. IASetM and IAGetM shall not be used in communications based on this version of the ECHONET Specification.

### 8.8.2.1 IASet/IASetM

IASet/IASetM is an internal service for middleware adapters whereby a middleware adapter that receives a request for a Set/SetM from another node writes the specified values into the corresponding areas of the device object(s) contained in the middleware adapter.

The reception (acceptance) response to the other node is made upon completion of the writing process. If no property exists, a rejection response is sent back to the other node.

ECHONET-ready equipment references the device object(s) of the middleware adapter at regular intervals to confirm the contents of status alteration requests from other nodes.

Figure 8.16 shows how this works.

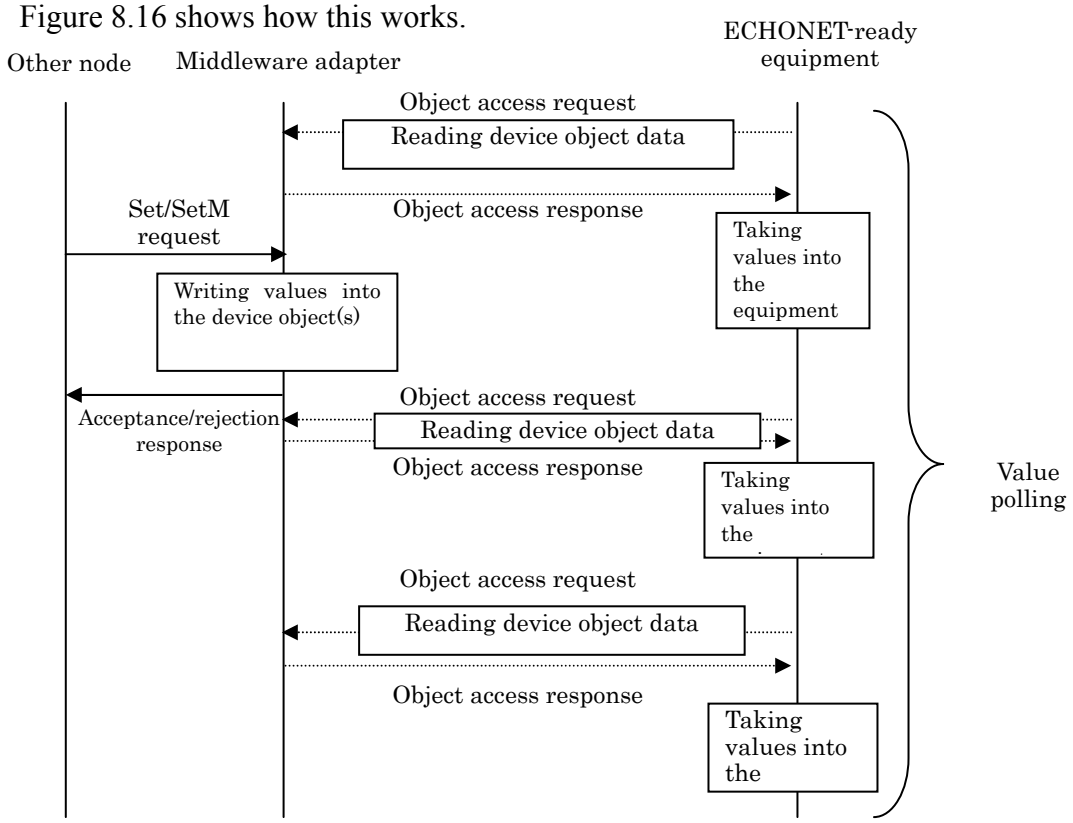



**Fig. 8.16 Operation of IASet (IASetM)**

### 8.8.2.2 IASetup/IASetMup

IASetup/IASetMup is an internal service for middleware adapters whereby a middleware adapter that receives a request for a Set/SetM from another node informs the contents of the request (status acquisition request) to the ECHONET-ready equipment.

The reception (acceptance) response to the other node is made upon reception of a status alteration response sent by the ECHONET-ready equipment in response to the IASetup. Figure 8.17 shows how this works. If no equipment status access response (element designation equipment status access response in the case of an IASetMup) is returned, a timeout occurs and a rejection response is sent back to the other node.

If no property exists, a rejection response is sent back to the other node without making an equipment status access request.

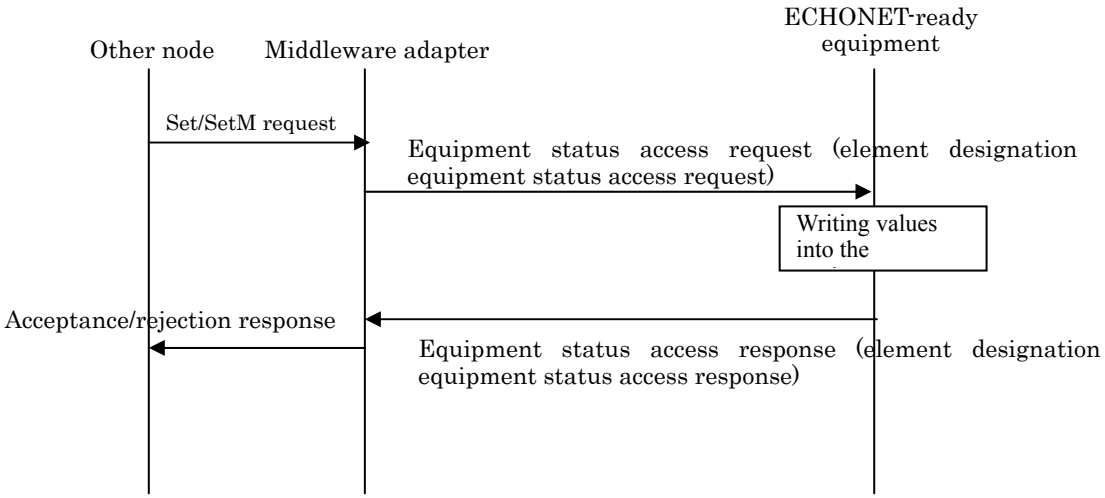



**Fig. 8.17 Operation of IASetup (IASetMup)**

### 8.8.2.3 IAGet/IAGetM

IAGet/IAGetM is an internal service for middleware adapters whereby a middleware adapter that receives a request for a Get/GetM from another node responds with the specified values in the corresponding areas of the device object(s) contained in the middleware adapter.

The ECHONET-ready equipment must change the corresponding property values of the device object(s) of the middleware adapter every time its own status changes.

If no property exists, a rejection response is sent back to the other node.

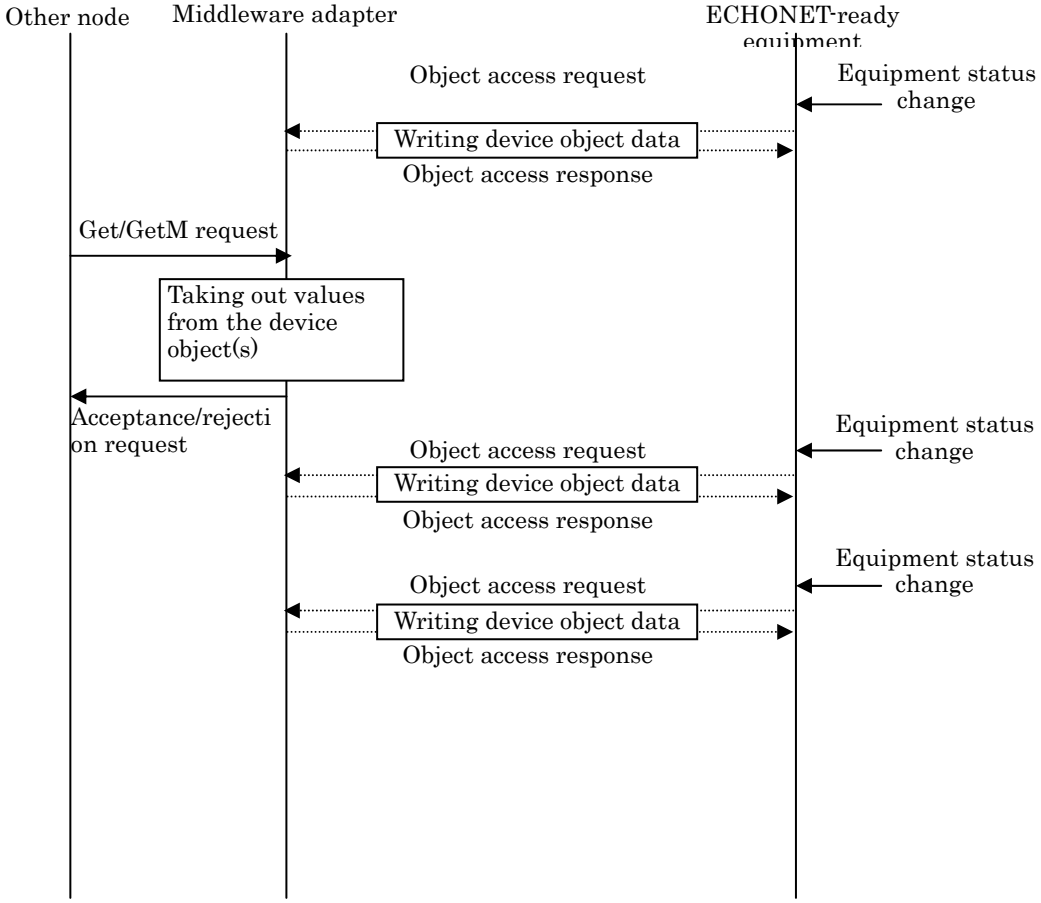



**Fig. 8.18 Operation of IAGet (IAGetM)**

### 8.8.2.4 IAGetup/IAGetMup

IAGetup/IAGetMup is an internal service for middleware adapters whereby a middleware adapter that receives a request for a Get/GetM from another node informs the contents of the request (status alteration request) to the ECHONET-ready equipment.

The response providing values to the other node is made upon reception of the response with the corresponding property values from the ECHONET-ready equipment.

If no property exists, a rejection response is sent back to the other node without making an equipment status access request.

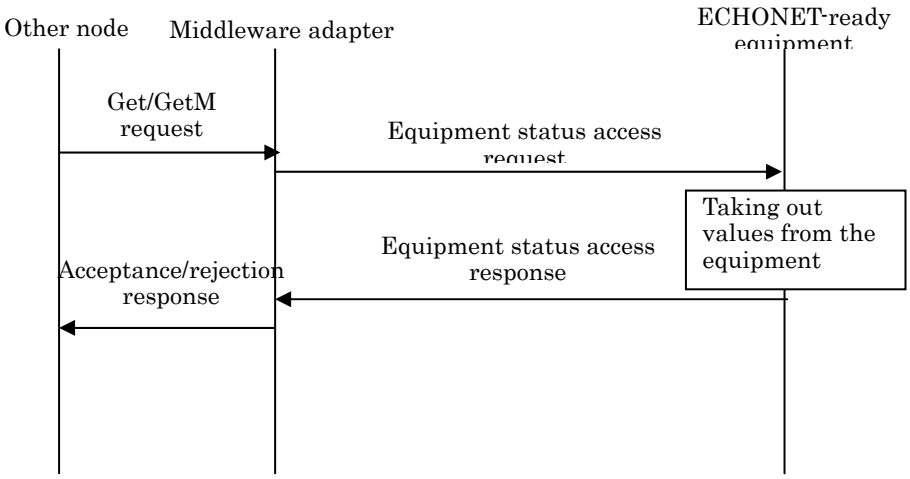
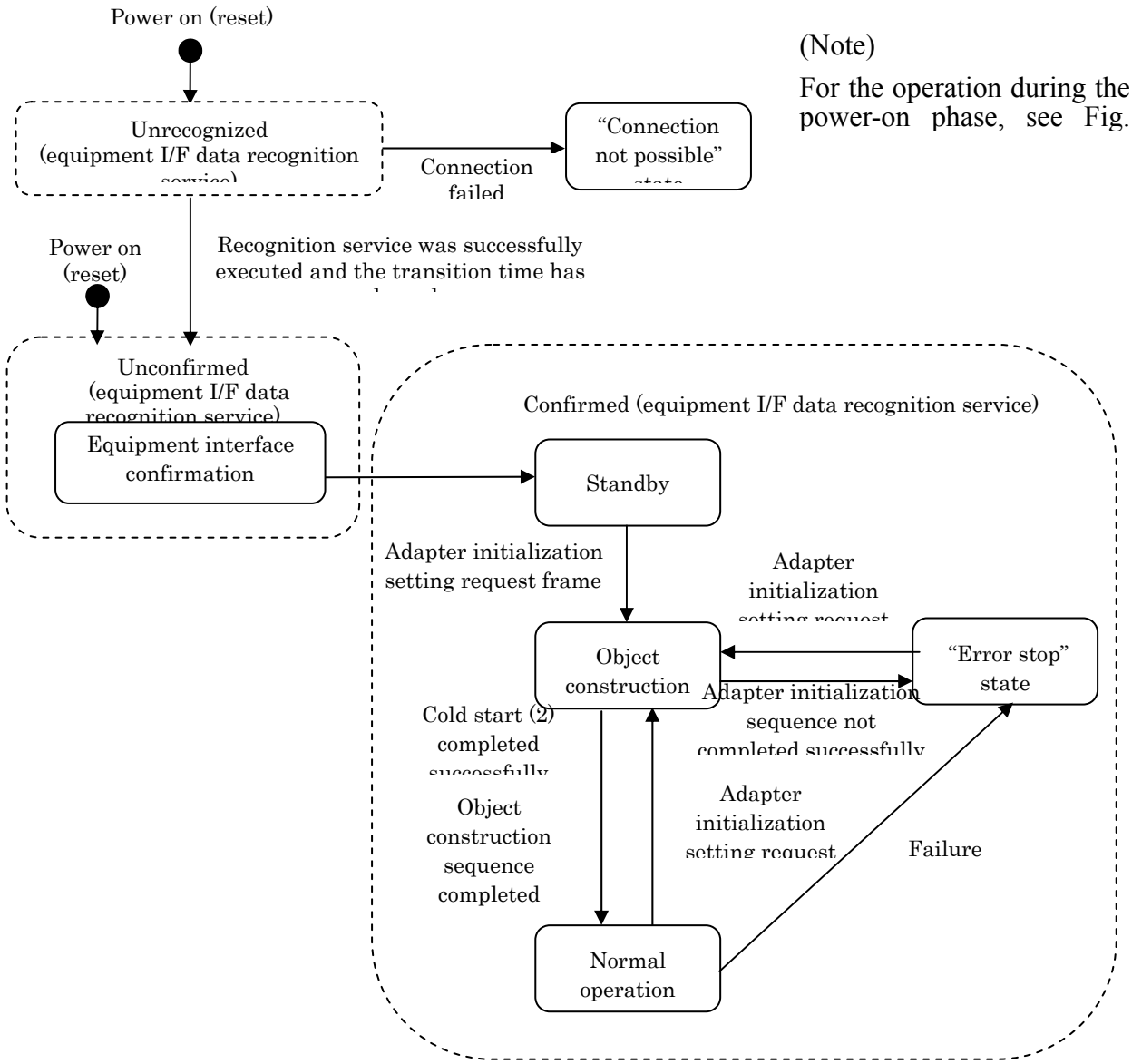



**Fig. 8.19 Operation of IAGetup (IAGetMup)**

## 8.8.3 ECHONET middleware adapter status changes for object generation type interfaces

This subsection defines the ECHONET middleware adapter status changes for cases in which an object generation type method is implemented for the ECHONET middleware adapter communication interface. The ECHONET middleware adapter shall not perform the processing to join the ECHONET until the process of internal object generation is completed.




(Note)

For the operation during the power-on phase, see Fig.

**Fig. 8.20 Middleware Adapter Status Changes**

#### 8.8.3.1 "Equipment Interface Confirmation" state

The state to which a shift is made immediately after power ON or a reset input from outside or immediately after completion of the recognition process of the equipment interface data recognition service and that lasts until the recognition confirmation processing by the equipment interface data recognition service is completed. No frames except for those relating to the equipment interface data confirmation processing are accepted and all inputs from the ECHONET network are discarded. After completion of the equipment interface data confirmation processing, a shift to the "standby" state is made.

If the equipment interface data confirmation processing ends abnormally for the specified number of times, a shift is made to the "unrecognized" state (equipment interface data).

#### 8.8.3.2 "Standby" state

The state for waiting for an initialization request frame after completion of the confirmation of the interface state with the ECHONET-ready equipment or after successful completion of the process of constructing device objects. No frames except for adapter initialization setting request frames are accepted and all inputs from the ECHONET network are discarded.

#### 8.8.3.3 "Object Construction" state

The state to which a shift is made from the "standby," "error stop" or "normal operation" state after an adapter initialization setting request frame input and during which the adapter side's initialization and object construction sequence are performed. No frames except for those relating to the adapter initialization mode and object construction mode are accepted during this state.

The middleware adapter performs a communication middleware cold start (acquisition of an ECHONET address) during this state.

If no internal object exists, the object construction sequence is performed.

If the adapter side's initialization sequence is completed successfully and the series of processing in the object construction sequence is completed successfully, a shift is made to the "normal operation" state. If either of these ends abnormally, a shift to the "error stop" state is made.

#### 8.8.3.4 "Error Stop" state

If the ECHONET middleware adapter initialization sequence fails or a failure occurs during the "normal operation" state, a shift to the "error stop" state is made. Upon a shift to this state, the "abnormal setting" processing described in Subsection 8.8.3 is performed.

The value to be set for the error property depends on the factor that caused the shift to the "error stop" state. The relationship between the shift-causing factors and error codes are as follows:

| | |
|---|---|
| Failed object construction | : 0x03EA (object error) |
| Failed adapter initialization sequence | : 0x03EB (adapter initialization error) |
| Failure during normal operation | : 0x03EC (other setting error) |

#### 8.8.3.5 "Normal Operation" state

A shift to this state occurs upon successful completion of the object construction sequence in the "object construction" state. In the case of a property that needs an initial value, the middleware adapter shall acquire an initial value by sending an equipment status access request to the ECHONET-ready equipment. Normal ECHONET communication is only possible in this state. During this time, object construction mode-related frames are not accepted.

## 8.8.4 Commands for object generation type interfaces

This subsection defines the requirements for the commands used for object generation type methods for ECHONET middleware adapter communication interfaces.

### 8.8.4.1 Equipment interface confirmation mode

(1) Equipment interface data confirmation request and response commands (Required)

The ECHONET middleware adapter makes an enquiry to the ECHONET-ready equipment to confirm the objects and communication method for the equipment adapter interface. If no object exists, an enquiry using 0 for the number of objects is made.

The ECHONET-ready equipment responds after confirming a match between the middleware adapter type and object(s).

◇ Direction of request commands

ECHONET middleware adapter → ECHONET-ready equipment

◇ Format for request commands

| 1 byte | 2 bytes | 1 byte | 1 byte | 2 bytes | 1 byte | 1 byte | n+1 bytes | 1 byte |
|---|---|---|---|---|---|---|---|---|
| STX | FT | CN | FN | DL | FD(0) | FD(1) | FD(2) | FCC |

STX ：0x02

FT ：0x0000

CN ：0x00

FN ：0x＊＊

DL ：0x＊＊＊＊

FD (0) : Middleware adapter type

0x01 Peer-to-peer method

0x02 Object generation method

0x03 Built-in object method

0x04 to 0xFF Not defined (reserved for future use)

FD (1) : Transmission speed

0x00 ：2400bps, 0x01：4800bps
0x02 ：9600bps, 0x03：19.2Kbps
0x04 ：38.4Kbps, 0x05：57.6Kbps
0x06 ：115Kbps, 0x07～0xFF：for future reserved

FD (2) : Area for defining adaptor maintenance object (0 to n-+1bytes)

n is the values of 18x < object > number.

If there is no object, this shall be 0 bytes.

<number of objects><EOJ><manufacturer code><product code>

These are listed for each of the objects (up to 3).

EOJ: Number of object(s) held by the adapter (up to 3)

Manufacturer code: manufacturer code(s) held by the adapter (3 bytes)

Product code: Product code(s) held by the adapter (12 bytes)

FCC : 0x**

◇ Format for response commands

| STX | FT | CN | FN | DL | FD(0) | FCC |
|---|---|---|---|---|---|---|

STX : 0x02

FT : 0x0000

CN : 0x80

FN : Value at the time of the request (If the function is not implemented, this shall be 0x00.)

DL : 0x0001

FD(0) : Processing result

| Value of FD (0) | Meaning |
|---|---|
| 0x0000 | Normal completion |
| 0x0011 | Adapter type mismatch error |
| 0x0012 | Object mismatch error |
| 0x0021 | Equipment interface data discarded |
| 0xFFFF | Other error |

FCC : 0x**

### 8.8.4.2 Adapter initialization mode

(1) Adapter initialization setting request and response commands (Required)

The ECHONET-ready equipment requests the basic middleware adapter to initialize the ECHONET communication middleware and the levels below the middleware.

Initialization shall be either a cold start (1), cold start (2) or cold start (3).

In cases where there already is an device object held, a selection must be possible between retaining it (equipment data retention cold start (1), equipment data retention cold start (2), equipment data retantion cold start (3)) and discarding it (equipment data disposal cold start (1), equipment data disposal cold start (2), equipment data disposal cold start (3)).

The basic middleware adapter makes a shift from the "standby" state to the "object construction" state upon reception of an adapter initialization setting request.

◇ Direction of request commands

ECHONET-ready equipment → ECHONET middleware adapter

◇ Format for request commands

| 1 byte | 2 bytes | 1 byte | 1 byte | 2 bytes | 2 bytes | 1 byte |
|---|---|---|---|---|---|---|
| STX | FT | CN | FN | DL | FD(0) | FCC |

STX : 0x02
FT : 0x0001
CN : 0x01
FN : 0x** (0x00 if the function is not implemented)
DL : 0x0002
FD(0) : Initialization method

| Value of FD(0) | Meaning |
|---|---|
| 0x0001 | Equipment data retention cold start (2)<br>If the basic middleware adapter already contains an device object, the adapter performs a cold start (2) with the object retained (No subsequent equipment enquiry request is made. Upon completion of the cold start (2), the basic middleware adapter makes a shift to the "communication" state). If no device object exists, an equipment enquiry request is made, and after completion of the device object generation process, a cold start (2) is performed. |
| 0x0002 | Equipment data disposal cold start (2)<br>If the basic middleware adapter already contains an device object, the adapter performs a cold start (2) with the object discarded. Then, an equipment enquiry request is made, and after completion of the device object generation process, a cold start (2) is performed. |
| 0x0003 | Equipment data retention cold start (1)<br>If the basic middleware adapter already contains an device object, the adapter performs a cold start (1) with the object retained (No subsequent equipment enquiry request is made. The basic middleware adapter immediately makes a shift to the "communication" state). If no device object exists, an equipment enquiry request is made, and after completion of the device object generation process, a cold start (1) is performed. |

| | |
|---|---|
| 0x0004 | Equipment data disposal cold start (1)<br>If the basic middleware adapter already contains an device object, the adapter performs a cold start (1) with the object discarded. Then, an equipment enquiry request is made, and after completion of the device object generation process, a cold start (1) is performed. |

FCC : 0x**

◇ Format for response commands

| 1 byte | 2 bytes | 1 byte | 1 byte | 2 bytes | 2 bytes | 2 bytes | 8 bytes | 1 byte |
|---|---|---|---|---|---|---|---|---|
| STX | FT | CN | FN | DL | FD(0) | FD(1) | FD(2) | FCC |

STX : 0x02
FT : 0x0001
CN : 0x81
FN : Value at the time of the request
DL : 0x0002 / 0x000B
FD(0) : Response result

1) 1 byte
2) 2 bytes

| Value of FD (0) | Meaning |
|---|---|
| 0x0000 | Initialization accepted |
| 0x0011 | Initialization rejected |
| 0x0101 | Status discrepancy error (“equipment interface confirmation” state (“unconfirmed”)) |
| 0xFFFF | Other error |

FD(1) : Lower-layer communication software ID

* Refer to the explanations on the “Node Identification Number” property in Part 2, “9.11.1 Node Profile Class: Detailed Specifications.” The middleware adapter also sets the node identification number information setting using the lower-layer communication software ID.

0x00 Node identification number has not been set

FD(2) : Unique number field for the node identification number

* See Part 2, Subsection 9.11.1.

FCC : 0x＊＊

(2) Adapter initialization completion notification command and adapter initialization completion notification acceptance response command (Required)

The basic middleware adapter must send an adapter initialization completion notification to the ECHONET-ready equipment upon completion of an initialization.

If the initialization is completed successfully, “initialization completed successfully” is sent.
If the initialization ends abnormally, “initialization ended abnormally” is given.
The ECHONET-ready equipment must return an adapter initialization completion notification acceptance response when it receives an adapter initialization completion notification.

◇ Direction of request commands
ECHONET middleware adapter → ECHONET-ready equipment

◇ Format for request commands

| 1 byte | 2 bytes | 1 byte | 1 byte | 2 bytes | 2 bytes | 1 byte |
|---|---|---|---|---|---|---|
| STX | FT | CN | FN | DL | FD(0) | FCC |

STX : 0x02
FT : 0x0001
CN : 0x02
FN : 0x＊＊
DL : 0x＊＊＊＊
FD(0) : Response result

| Value of FD (0) | Meaning |
|---|---|
| 0x0000 | Initialization completed successfully |
| 0x0011 | Initialization ended abnormally |

FCC : 0x＊＊

◇ Format for response commands

| 1 byte | 2 bytes | 1 byte | 1 byte | 2 bytes | 2 bytes | 1 byte |
|---|---|---|---|---|---|---|
| STX | FT | CN | FN | DL | FD(0) | FCC |

STX : 0x02
FT : 0x0001
CN : 0x82
FN : Value at the time of the request (0x00 if the function is not implemented)
DL : 0x0002
FD(0) : Response result

| FD(0)の値 | Meaning |
|---|---|
| 0x0000 | Notification acceptance |
| 0xFFFF | Other error |

FCC : 0x＊＊

### 8.8.4.3 Object construction mode

(1) Equipment enquiry request and response commands (Required)

When a basic middleware adapter does not contain an device object, it makes an equipment enquiry request to the ECHONET-ready equipment.

Upon reception of this request, the ECHONET-ready equipment returns the equipment data in the form of an equipment enquiry response.

The ECHONET-ready equipment that receives an equipment enquiry request returns, in the form of an equipment enquiry response to the basic middleware adapter, the data necessary to construct device objects in the basic middleware adapter. The maximum number of device objects that can be constructed is 3. Information on 3 pieces of equipment can be sent in a single response or divided into 2 or 3 pieces and sent in 2 or 3 responses, respectively.

◇ Direction of request commands

ECHONET middleware adapter → ECHONET-ready equipment

Format for request commands

| 1 byte | 2 bytes | 1 byte | 1 byte | 2 bytes | 1 byte |
|---|---|---|---|---|---|
| STX | FT | CN | FN | DL | FCC |

STX : 0x02
FT : 0x0002
CN : 0x00
FN : 0x＊＊
DL : 0x0000
FCC : 0x＊＊

◇ Format for response commands

| 1 byte | 2 bytes | 1 byte | 1 byte | 2 bytes | 2 bytes | 1 byte | ＊bytes | 1 byte |
|---|---|---|---|---|---|---|---|---|
| STX | FT | CN | FN | DL | FD(0) | FD(1) | FD(2) | FCC |

STX : 0x02
FT : 0x0002
CN : 0x80
FN : Value at the time of the request (0x00 if the function is not implemented)
DL : 0x＊＊＊＊

ＦＤ(0) ：Response result
0x0000 Normal
0xFFFF Other error

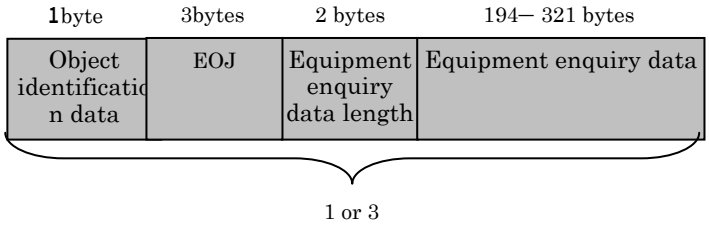
ＦＤ(1) ：Number of objects sent in the frame
number of objects included in the equipment enquiry response frame

ＦＤ(2) ：Object data
Refer to Object Data Format.

ＦＣＣ ：0x＊＊

<Object Data Format>

The following format is implemented (1 or 3):




| Name | Size (in bytes) | Explanation |
|---|---|---|
| Object identification data | 1 | Total number of objects to be managed and identification number(s) assigned to manage the object(s).<br>b7 to b4: Total number of objects<br>(1: 0001, 2: 0010, 3: 0011)<br>b3 to b0: Object identification numbers<br>(1: 0001, 2: 0010, 3: 0011) |
| EOJ | 3 | ECHONET object code for generated construction method A device object. |
| Equipment enquiry data length | 2 | Indicates the size of the equipment enquiry data |
| Equipment enquiry data | 194–321 | Refer to Equipment Enquiry Data Format. |

< Equipment Enquiry Data Format>

| 2 bytes | 17 bytes | 17 bytes | 17 bytes | 17 bytes | 17 bytes | 17 bytes |
|---|---|---|---|---|---|---|
| Effective/inef fective bit | SetM property | Set property map | GetM property map | Get property map | Status change | IASetup property |

| 17 bytes | 17 bytes | 17 bytes | 4 bytes | 3 bytes | 3 bytes | 12 bytes |
|---|---|---|---|---|---|---|
| IAGetup property | IASetMup property | IAGetMup property | Version information | Manufacture r code | Factory code | Product code |

| 12 bytes | 4 bytes | 1—128 bytes |
|---|---|---|
| Production number | Date of production | Property size map |

| Name | Size (in bytes) | Explanation |
|---|---|---|
| Effective/ineffectiv e bit map | 2 | Indicates the meaningful part in the data that follows:<br>b15: SetM property map<br>b14: Set property map<br>b13: GetM property map<br>b12: Get property map<br>b11: Status change announcement property map<br>b10: IASetup property operation map<br>b9: IAGetup property operation map<br>b8: IASetMup property operation map<br>b7: IAGetMup property operation map<br>b6: Version information<br>b5: Manufacturer code<br>b4: Factory code<br>b3: Product code<br>b2: Production number<br>b1: Date of production<br>b0: Property size map<br>The items set to 1 are effective. |
| SetM property map | 17 | Indicates the property that accepts SetM. The format shall be as per the ECHONET Specification. |
| Set property map | 17 | Indicates the property that accepts Set. The format shall be as per the ECHONET Specification. |
| GetM property map | 17 | Indicates the property that accepts GetM. The format shall be as per the ECHONET Specification. |
| Get property map | 17 | Indicates the property that accepts Get. The format shall be as per the ECHONET Specification. |
| Status change announcement property map | 17 | Indicates the property that performs status change announcements. The format shall be as per the ECHONET Specification. |
| IASetup property operation map | 17 | Indicates the internal service of the adapter for the property that accepts Set. The Set for the property that was specified as an effective one shall become IASetup. The process of reflecting it in the object is performed after the status alteration on the ECHONET-ready equipment side. Then the response is returned. The format shall be subject to the same rules as the other |

| | | |
|---|---|---|
| | | property maps. |
| IAGetup property operation map | 17 | Indicates the internal service of the adapter for the property that accepts Get. The Get for the property that was specified as an effective one shall become IAGetup. The process of reflecting it in the object is performed after the status alteration on the ECHONET-ready equipment side. Then the response is returned. The format shall be subject to the same rules as the other property maps. |
| IASetMup property operation map | 17 | Indicates the internal service of the adapter for the property that accepts SetM. The SetM for the property that was specified as an effective one shall become IASetMup. The process of reflecting it in the object is performed after the status alteration on the ECHONET-ready equipment side. Then the response is returned. The format shall be subject to the same rules as the other property maps. |
| IAGetMup property operation map | 17 | Indicates the internal service of the adapter for the property that accepts GetM. The GetM for the property that was specified as an effective one shall become IAGetMup. The process of reflecting it in the object is performed after the status alteration on the ECHONET-ready equipment side. Then the response is returned. The format shall be subject to the same rules as the other property maps. |
| Version information | 4 | Indicates the specification version used. The format shall be as per the ECHONET Specification. |
| Manufacturer code | 3 | Indicates the manufacturer of the ECHONET-ready equipment. The format shall be as per the ECHONET Specification. |
| Factory code | 3 | Indicates the vendor-dependent code that indicates the factory that manufactured the ECHONET-ready equipment. The format shall be as per the ECHONET Specification. |
| Product code | 12 | Indicates the vendor-dependent ECHONET-ready equipment product code. The format shall be as per the ECHONET Specification. |
| Production number | 12 | Indicates the vendor-dependent ECHONET-ready equipment production number. The format shall be as per the ECHONET Specification. |
| Date of production | 4 | Indicates the date on which the ECHONET-ready equipment was manufactured. The format shall be as per the ECHONET Specification. |
| Property size map | 1 to 128 | Indicates the size of each property in bytes. In the case of an array property, this shall be the product of the size of one array element and the maximum number of array elements. For the properties that are present, the sizes are listed starting from the one with the smallest EPC. |

(Note)

*1: The format shall be the format specified in Part 2, Appendix 2, “Property map description format (2)”(The size shall be fixed at 17 bytes.)

(2) Equipment enquiry completion notification command and equipment enquiry completion notification acceptance response command (Required)

The basic middleware adapter sends an equipment enquiry completion notification to the ECHONET-ready equipment when it receives the information to construct all the device objects.

If there is something wrong in the data received through the equipment enquiry response, “invalid” is returned and a shift is made to the “error stop” state.

Upon reception of an equipment enquiry completion notification, the ECHONET-ready equipment returns an equipment enquiry completion notification acceptance response to the basic middleware adapter.

If an equipment enquiry completion notification containing “invalid” is received, the initialization sequence must be redone.

◇ Direction of request commands

ECHONET middleware adapter → ECHONET-ready equipment

◇ Format for request commands

| 1 byte | 2 bytes | 1 byte | 1 byte | 2 bytes | 2 bytes | 1 byte |
|---|---|---|---|---|---|---|
| STX | FT | CN | FN | DL | FD(0) | FCC |

STX ：0x02

FT ：0x0002

CN ：0x01

FN ：0x＊＊

DL ：0x0002

FD(0) ：Equipment enquiry processing result

| Value of FD (0) | Meaning |
|---|---|
| 0x0000 | OK |
| 0x0011 | Invalid |

FCC ：0x＊＊

◇ Format for response commands

| 1 byte | 2 bytes | 1 byte | 1 byte | 2 bytes | 2 bytes | 1 byte |
|---|---|---|---|---|---|---|
| STX | FT | CN | FN | DL | FD(0) | FCC |

STX ：0x02

FT ：0x0002

CN ：0x81

FN ：Value at the time of the request (0x00 if the function is not implemented)

DL ：0x0002

FD(0) ：Response result

| Value of FD (0) | Meaning |
|---|---|
| 0x0000 | Accepted successfully |

| | |
|---|---|
| 0xFFFF | Other error |

F C C : 0x＊＊

(3) Adapter startup notification command and adapter startup notification acceptance response command (Required)

If both the initialization sequence and object construction sequence have been successfully completed and the basic middleware adapter has an device object or the process of constructing an device object has been successfully completed, the basic middleware adapter sends an adapter startup notification to the ECHONET-ready equipment.

Before sending this adapter startup notification, the node profile object specified in Part 2 must have been constructed.

If the construction of an device object fails, an adapter startup notification containing “startup failed” is sent and a shift is made to the “error stop” state.

If an adapter startup notification response containing “accepted successfully” is received, a shift is made to the “normal operation” state.

Upon reception of an adapter startup notification, the ECHONET-ready equipment returns an adapter startup notification acceptance response to the basic middleware adapter.

If an adapter startup notification containing “startup failed” is received, the initialization sequence must be redone.

◇ Direction of request commands

ECHONET middleware adapter → ECHONET-ready equipment

◇ Format for request commands

| 1 byte | 2bytes | 1 byte | 1 byte | 2 bytes | 2 bytes | 1 byte |
|---|---|---|---|---|---|---|
| S T X | F T | C N | F N | D L | FD(0) | F C C |

S T X : 0x02

F T : 0x0002

C N : 0x02

F N : 0x＊＊

D L : 0x0002

F D(0) : Initialization processing result

| Value of FD (0) | Meaning |
|---|---|
| 0x0000 | Startup completed successfully |
| 0x0011 | Startup failed |

F C C : 0x＊＊

◇ Format for response commands

| 1 byte | 2 bytes | 1 byte | 1 byte | 2 bytes | 2 bytes | 1 byte |
|---|---|---|---|---|---|---|
| STX | FT | CN | FN | DL | FD(0) | FCC |

STX : 0x02
FT : 0x0002
CN : 0x82
FN : Value at the time of the request (0x00 if the function is not implemented)
DL : 0x0002
FD(0) : Response result

| Value of FD (0) | Meaning |
|---|---|
| 0x0000 | Accepted successfully |
| 0xFFFF | Other error |

FCC : 0x＊＊

(4) “Equipment enquiry request with object specified” and “equipment enquiry response with object specified” commands (optional)

If no device object is present inside, the basic middleware adapter will send an “equipment enquiry request with object specified” to the ECHONET-ready equipment. In response to this request, the ECHONET-ready equipment will return device information in the form of an “equipment enquiry response with object specified.”

When the ECHONET-ready equipment receives an “equipment enquiry request with object specified,” it will send the information necessary to construct a device object in the basic middleware adapter to the basic middleware adapter in the form of an “equipment enquiry response with object specified.” The task of acquiring and managing object information is the responsibility of the ECHONET middleware adapter. The ECHONET-ready equipment assigns, for management purposes, such 1-byte object identification numbers (starting with “0x01”) that the numbers of objects to be generated become largest to object code (EOJ) values, and notifies the appropriate object identification numbers to the ECHONET middleware adapter through responses to requests from the ECHONET middleware adapter. The ECHONET middleware adapter identifies and manages objects using these object identification numbers.

(Note) *: Required in the case of an adapter capable of generating 4 or more device objects or a piece of ECHONET-ready equipment capable of requesting the generation of 4 or more device objects.

◇ Direction of request commands

ECHONET middleware adapter → ECHONET-ready equipment

◇ Format for request commands

| 1 byte | 2 bytes | 1 byte | 1 byte | 2 bytes | 1 bytes | 1 byte |
|---|---|---|---|---|---|---|
| S T X | F T | C N | F N | D L | FD(0) | F C C |

S T X ：0x02
F T ：0x0002
C N ：0x03
F N ：0x＊＊
D L ：0x0001
F D(0) ：Object identification number (used in a sequential order starting with “0x01”)
F C C ：0x＊＊

| Value of FD (0) | Meaning |
|---|---|
| 0x0000 | Startup completed successfully |
| 0x0011 | Startup failed |

◇ Format for response commands

| 1 byte | 2 bytes | 2 byte | 1 byte | 2 bytes | 2 bytes | 1 byte | n bytes | 1 byte |
|---|---|---|---|---|---|---|---|---|
| S T X | F T | C N | F N | DL | FD(0) | FD(1) | FD(2) | F C C |

S T X ：0x02
F T ：0x0002
C N ：0x83
F N ：Value at the time of the request (0x00 if the function is not implemented)
D L ：0x＊＊＊＊
F D(0) ：Response result

| Name | Size (bytes) | Explanation |
|---|---|---|
| Result | 2 | Response result<br>0x0000: Acceptance response<br>0x0001: Acceptance response (network out of operation)<br>0x0011: Rejection response<br>0x0012: The specified device object does not exist.<br>0x0101: Status discrepancy error (“device interface confirmation” state (“unconfirmed”))<br>0x0103: Status discrepancy error (“standby” state)<br>0x0105: Status discrepancy error (“error stop” state)<br>0xFFFF: Other error |

F D(1) ：Number of objects sent in the frame
Fixed at “0x01”
F D(2) ：Object data
Refer to “<Object data format>.”
F C C ：0x＊＊

<Object data format>

| 2byte | 3 bytes | 2 byte | 194-289 byte |
|---|---|---|---|
| Object identification information | EOJ | Equipment enquiry data length | Equipment enquiry data |

| Name | Size (bytes) | Explanation |
|---|---|---|
| Object identification information | 2 | First byte: object identification number<br>Second byte: total number of objects |
| EOJ | 3 | ECHONET object code value for the device object to be constructed |
| Equipment enquiry data length | 2 | Indicates the size of the equipment enquiry data. |
| Equipment enquiry data | 194 - 289 | Refer to "<Equipment enquiry data format>." |

< Equipment enquiry data format >

| 2 byte | 17 bytes | 17 byte | 17 byte | 17bytes | 17bytes | 17 byte |
|---|---|---|---|---|---|---|
| Effective/ineffective bit map | SetM property map | Set property map | GetM property map | Get property map | Status change announcement property map | IASetup property operation map |

| 17 byte | 17 bytes | 17 byte | 4 byte | 3bytes | 3bytes | 12 byte |
|---|---|---|---|---|---|---|
| IAGetup property operation map | IASetMup property operation map | IAGetMup property operation map | Version information | Manufacturer code | Factory code | Product code |

| 12bytes | 4bytes | 1-128 byte |
|---|---|---|
| Production number | Date of production | Property size map |

| Name | Size (bytes) | Explanation |
|---|---|---|
| Effective/ineffective bit map | 2 | Indicates the meaningful ones of the pieces of data that follow:<br>b15: SetM property map<br>b14: Set property map<br>b13: GetM property map<br>b12: Get property map<br>b11: Status change announcement property map<br>b10: IASetup property operation map<br>b9: IAGetup property operation map<br>b8: IASetMup property operation map<br>b7: IAGetMup property operation map |

| | | |
|---|---|---|
| | | b6: Version information<br>b5: Manufacturer code<br>b4: Factory code<br>b3: Product code<br>b2: Production number<br>b1: Date of manufacture<br>b0: Property size map<br>The items for which "1" is specified are effective. |
| SetM property map | 17 | Indicates the properties that accept SetM.*1 |
| Set property map | 17 | Indicates the properties that accept Set.*1 |
| GetM property map | 17 | Indicates the properties that accept GetM.*1 |
| Get property map | 17 | Indicates the properties that accept Get.*1 |
| Status change announcement property map | 17 | Indicates the properties for which a status change announcement is to be made.*1 |
| IASetup property operation map | 17 | Indicates the adapter's internal service for the properties that accept Set. The processing for the properties is IASetup service processing.*1 |
| IAGetup property operation map | 17 | Indicates the adapter's internal service for the properties that accept Get. The processing for the properties is IAGetup service processing.*1 |
| IASetMup property operation map | 17 | Indicates the adapter's internal service for the properties that accept SetM. The processing for the properties is IASetMup service processing.*1 |
| IAGetMup property operation map | 17 | Indicates the adapter's internal service for the properties that accept GetM. The processing for the properties is IAGetMup service processing.*1 |
| Version information | 4 | Indicates the specification version used. The format shall be as per the ECHONET Specification. |
| Manufacturer code | 3 | Indicates the manufacturer of the ECHONET-ready equipment. The format shall be as per the ECHONET Specification. |
| Factory code | 3 | A vendor-dependent code indicating the factory that manufactured the ECHONET-ready equipment. The format shall be as per the ECHONET Specification. |
| Product code | 12 | A vendor-dependent ECHONET-ready equipment product code. The format shall be as per the ECHONET Specification. |
| Production number | 12 | A vendor-dependent ECHONET-ready equipment production number. The format shall be as per the ECHONET Specification. |
| Date of production | 4 | Indicates the date on which the ECHONET-ready equipment was manufactured. The format shall be as per the ECHONET Specification. |
| Property size map | 1 to 128 | Indicates, in bytes, the size of each property of the specified device objects. In the case of an array property, this shall be the product of the size of one array element and the maximum number of array elements. For the properties that are present, the sizes are listed starting with the one with the smallest EPC value. |

(Note)

*1: The format shall be the format specified in Part 2, Appendix 2, "Property map description format (2)"(The size shall be fixed at 17 bytes.)

### 8.8.4.4 ECHONET communication mode

(1) Equipment status access request and response commands (Required)

If a Set is performed for a property whose internal service is IASetup/IASetMup, or if a Get is performed for a property whose internal service is IAGetup/IAGetMup, the basic middleware adapter reports the Set/SetM or Get/GetM information to the ECHONET-ready equipment by means of an equipment status access request.

Upon reception of the equipment status access request, the ECHONET-ready equipment returns to the basic middleware adapter a corresponding response (when it is IASetup/IASetMup) or the status value that corresponds to the specified property (when it is IAGetup/IAGetMup)(equipment status access response).

◇ Direction of request commands

ECHONET middleware adapter → ECHONET-ready equipment

◇ Format for request commands

ＳＴＸ　：0x02
ＦＴ　：0x0003
ＣＮ　：0x10
ＦＮ　：0x＊＊
ＤＬ　：6+n
ＦＤ(0)　：Object information

| Name | Size (in bytes) | Explanation |
|---|---|---|
| EOJ | 3 | EOJ for the equipment (referenced or altered) |
| Length | 2 | The number of bytes obtained by adding up the EPC and EDT. The value 0x01 corresponds to IASetup (status reference) and the other values correspond to IAGetup (status alteration). |
| EPC | 1 | EPC for the property (referenced or altered) |
| EDT | n | The presence of this value indicates an alteration/property. In that case, the status that corresponds to the EPC (ECHONET-ready equipment) is altered (max. 245 bytes). If this value is not present, it indicates referencing with respect to status. |

ＦＣＣ　：0x＊＊

◇ Format for response commands

| 1 byte | 2 bytes | 1 byte | 1 byte | 2 bytes | 8+n bytes | 1 byte |
|---|---|---|---|---|---|---|
| ＳＴＸ | ＦＴ | ＣＮ | ＦＮ | ＤＬ | FD(0) | ＦＣＣ |

ＳＴＸ　：0x02
ＦＴ　：0x0003
ＣＮ　：0x90
ＦＮ　：Value at the time of the request (0x00 if the function is not implemented)
ＤＬ　：8+n
ＦＤ(0)　：Object information

| Name | Size (in bytes) | Explanation |
|---|---|---|
| EOJ | 3 | EOJ for the ECHONET-ready equipment (referenced or altered) |
| Result | 2 | Response result<br>0x0000: Acceptance response<br>0x0011: Rejection response<br>0xFFFF: Other error |
| Length | 2 | The number of bytes obtained by adding up the EPC and EDT. The value 0x01 corresponds to an alteration response and the other values correspond to a reference response. |
| EPC | 1 | EPC for the property (referenced or altered) |
| EDT | n | If this value is present, it becomes the reference response value (max. 245 bytes). |

ＦＣＣ　：0x＊＊

(2) Equipment status notification request and response commands (Required)

When the basic middleware adapter receives an equipment status notification, it shall return an equipment status notification response to the ECHONET-ready equipment, and, if the adapter's internal service for the target property for notification is IAGet or IAGetM, it shall write the value to be notified to the target property of the device object it contains and announce that value in the domain. If the adapter's internal service for the target property for notification is IAGetup or IAGetMup, it shall return an equipment status notification response to the ECHONET-ready equipment and announce the value to be notified in the domain.

◇ Direction of request commands

ECHONET-ready equipment → ECHONET middleware adapter

◇ Format for request commands

| 1 byte | 2 bytes | 1 byte | 1 byte | 2 bytes | 6+n bytes | 1 byte |
|---|---|---|---|---|---|---|
| ＳＴＸ | ＦＴ | ＴＸＮ | ＦＮ | ＤＬ | FD(0) | ＦＣＣ |

ＦＴ　：0x0003

ＣＮ　：0x11

ＦＮ　：0x** (0x00 if the function is not implemented)

ＤＬ　：6+n

ＦＤ(0)　：Object information

| Name | Size (in bytes) | Explanation |
|---|---|---|
| EOJ | 3 | EOJ for the ECHONET-ready equipment (referenced) |
| Length | 2 | The number of bytes obtained by adding up the EPC and EDT. |
| EPC | 1 | EPC for the property (reported) |
| EDT | n | Reported data (max. 245 bytes) |

ＦＣＣ　：0x＊＊

◇ Format for response commands

| 1 byte | 2 bytes | 1 byte | 1 byte | 2 bytes | 2 bytes | 3 bytes | 1 byte |
|---|---|---|---|---|---|---|---|
| ＳＴＸ | ＦＴ | ＣＮ | ＦＮ | DL | FD(0) | FD(1) | ＦＣＣ |

ＳＴＸ　：0x02

ＦＴ　：0x0003

ＣＮ　：0x91

ＦＮ　：Value at the time of the request

ＤＬ　：0x0005

ＦＤ(0)　：Processing result

| Name | Size (in bytes) | Explanation |
|---|---|---|
| Result | 2 | Response result<br>0x0000: Acceptance response<br>0x0011: Rejection response (network not operating)<br>0x0012: Rejection response (other)<br>0x0101: Status discrepancy error (“equipment interface confirmation” state (“unconfirmed”))<br><br>0x0102: status discrepancy error (“external equipment data confirmation (reference)” state)<br>0x0103: status discrepancy error (“standby” state)<br>0x0104: status discrepancy error (“object construction” state)<br>0x0105: status discrepancy error (“error stop” state)<br>0xFFFF: other error |

ＦＤ(1)　：FD (1) EOJ for the ECHONET-ready equipment (reported)

ＦＣＣ　：0x＊＊

(3) Element designation equipment status access request and response commands

(Required)

If a SetM is performed for a property whose internal service is IASetMup, or if a GetM is performed for a property whose internal service is IAGetMup, the basic middleware adapter reports the SetM or GetM information to the equipment by means of an element designation equipment status access request.

Upon reception of the element designation equipment status access request, the ECHONET-ready equipment returns to the basic middleware adapter a response to that request (when it is IASetMup) or the status value that corresponds to the element number of the property specified (when it is IAGetMup)(element designation equipment status access response).

◇ Direction of request commands

ECHONET middleware adapter →-> ECHONET-ready equipment

◇ Format for request commands

| 1 byte | 2 bytes | 1 byte | 1 byte | 2 bytes | 8+n bytes | 1 byte |
|---|---|---|---|---|---|---|
| STX | FT | CN | FN | DL | FD(0) | FCC |

STX ：0x02

FT ：0x0003

CN ：0x12

FN ：0x＊＊

DL ：8+n

FD(0) ：Middleware adapter type

| Name | Size (in bytes) | Explanation |
|---|---|---|
| EOJ | 3 | EOJ for the ECHONET-ready equipment (referenced or altered) |
| Length | 2 | The number of bytes obtained by adding up the element number, EPC and EDT. 0x01 indicates referencing, and the other values indicate an alteration. |
| EPC | 1 | EPC for the property (referenced or altered) |
| Element number | 2 | Element number for the array (referenced or altered) |
| EDT | n | The presence of this value indicates an alteration/property. In that case, the status that corresponds to the EPC (ECHONET-ready equipment) is altered (max. 245 bytes). If this value is not present, it indicates referencing with respect to status. |

FCC ：0x＊＊

◇ Format for response commands

| 1 byte | 2 bytes | 1 byte | 1 byte | 2 bytes | 10 +n bytes | 1 byte |
|---|---|---|---|---|---|---|
| S T X | F T | C N | F N | DL | FD(0) | F C C |

S T X ：0x02
F T ：0x0003
C N ：0x92
F N ：Value at the time of the request (0x00 if the function is not implemented)
D L ：10+n
F D(0) ：Object information

| Name | Size (in bytes) | Explanation |
|---|---|---|
| EOJ | 3 | EOJ for the ECHONET-ready equipment (referenced or altered) |
| Result | 2 | Response result<br>0x0000: Acceptance response<br>0x0011: Rejection response<br>0xFFFF: Other error |
| Length | 2 | The number of bytes obtained by adding up the element number, EPC and EDT. The value 0x01 corresponds to an alteration response and the other values correspond to a reference response. |
| EPC | 1 | EPC for the property (referenced or altered) |
| Element number | 2 | Element number for the array (referenced or altered) |
| EDT | N | If this value is present, it becomes the reference response value. |

F C C ：0x＊＊

(4) Element designation equipment status notification request and response commands (Required)

When the basic middleware adapter receives an element designation equipment status notification, it shall return an element designation equipment status notification response to the ECHONET-ready equipment, and, if the adapter's internal service for the target property for notification is IAGetM, it shall write the value to be notified to the element number of the target property of the device object it contains and announce that value in the domain. If the adapter's internal service for the target property for notification is IAGetMup, it shall return an element designation equipment status notification response to the ECHONET-ready equipment and announce in the domain the value specified by the property element number.

◇ Direction of request commands

ECHONET-ready equipment → ECHONET middleware adapter

◇ Format for request commands

| 1 byte | 2 bytes | 1 byte | 1 byte | 2 bytes | 8+n bytes | 1 byte |
|---|---|---|---|---|---|---|
| STX | FT | CN | FN | DL | FD(0) | FCC |

STX : 0x02

FT : 0x0003

CN : 0x13

FN : 0x** (0x00 if the function is not implemented)

DL : 8+n

FD(0) : object information

| Name | Size (in bytes) | Explanation |
|---|---|---|
| EOJ | 3 | EOJ for the ECHONET-ready equipment (reported) |
| Length | 2 | The number of bytes obtained by adding up the element number data, EPC and EDT. |
| EPC | 1 | EPC for the property (reported) |
| Element number | 2 | Element number (reported) |
| EDT | n | Reported data (max. 245 bytes) |

FCC : 0x＊＊

◇ Format for response commands

| 1 byte | 2 bytes | 1 byte | 1 byte | 2 bytes | 2 bytes | 3 bytes | 1 byte |
|---|---|---|---|---|---|---|---|
| STX | FT | CN | FN | DL | FD(0) | FD(1) | FCC |

STX : 0x02

FT : 0x0003

CN : 0x93

FN : Value at the time of the request

DL : 0x0005

FD(0) : Response result

| Name | Size (in bytes) | Explanation |
|---|---|---|
| EOJ | 3 | EOJ for the ECHONET-ready equipment (reported) |
| Result | 2 | Response result<br>0x0000: Acceptance response<br>0x0011: Rejection response (network not operating)<br>0x0012: Rejection response (other)<br>0x0101: Status discrepancy error (“equipment interface confirmation” state (“unconfirmed”))<br>0x0102: Status discrepancy error (“external equipment data confirmation (reference)” state)<br>0x0103: Status discrepancy error (“standby” state)<br>0x0104: Status discrepancy error (“object construction” state) |

| | | |
|---|---|---|
| | | 0x0105: Status discrepancy error (“error stop” state)<br>0xFFFF: Other error |

FD(1) : EOJ for the ECHONET-ready equipment (reported)

FCC : 0x＊＊

(5) Object access request and response commands (required)

If an object access request is made with writing specified for a property for which the internal service of the adapter is IASet or IAGet, the basic middleware adapter shall write the value of the property specified to the corresponding property. If an object access request is made with reading specified for a property for which the internal service of the adapter is IAGet or IASet, the basic middleware adapter shall return the value of the property specified to the ECHONET-ready equipment in the form of an object access response. If an object access request is made for a property for which IASetup and IAGetup are specified or a property for which IASetMup and IAGetMup are specified, an object access response shall be returned to the ECHONET-ready equipment with the “Result” set to “rejection response.”

◇ Direction of request commands

ECHONET-ready equipment → ECHONET middleware adapter

◇ Format for request commands

| 1 byte | 2 bytes | 1 byte | 1 byte | 2 bytes | 6+n bytes | 1 byte |
|---|---|---|---|---|---|---|
| STX | FT | CN | FN | DL | FD(0) | FCC |

STX : 0x02

FT : 0x0003

CN : 0x14

FN : 0x** (0x00 if the function is not implemented)

DL : 6+n

FD(0) : Object information

| Name | Size (in bytes) | Explanation |
|---|---|---|
| EOJ | 3 | EOJ for the ECHONET-ready equipment (referenced or altered) |
| Length | 2 | The number of bytes obtained by adding up the EPC and EPC data. The value 0x01 corresponds to referencing, and the other values correspond to an alteration. |
| EPC | 1 | EPC for the property (referenced or altered) |
| EDT | n | The presence of this value indicates an alteration/property. In that case, the status that corresponds to the EPC (ECHONET-ready equipment) is altered (max. 245 bytes). If this value is not present, it indicates referencing with respect to status. |

ＦＣＣ　：0x＊＊

◇ Format for response commands

| 1 byte | 2 bytes | 1 byte | 1 byte | 2 bytes | 2 bytes | 6+n bytes | 1 byte |
|---|---|---|---|---|---|---|---|
| ＳＴＸ | ＦＴ | ＣＮ | ＦＮ | DL | FD(0) | FD(0) | ＦＣＣ |

ＳＴＸ　：0x02

ＦＴ　：0x0003

ＣＮ　：0x94

ＦＮ　：Value at the time of the request

ＤＬ　：8+n

ＦＤ(0)　：Response result

| Name | Size (in bytes) | Explanation |
|---|---|---|
| Result | 2 | Response result<br>0x0000: Acceptance response<br>0x0001: Acceptance response (network not operating)<br>0x0011: Rejection response<br>0x0101: Status discrepancy error (“equipment interface confirmation” state (“unconfirmed”))<br>0x0102: Status discrepancy error (“external equipment data confirmation (reference)” state)<br>0x0103: Status discrepancy error (“standby” state)<br>0x0104: Status discrepancy error (“object construction” state)<br>0x0105: Status discrepancy error (“error stop” state)<br>0xFFFF: Other error |

ＦＤ(1)　：Object information

| Name | Size (in bytes) | Explanation |
|---|---|---|
| EOJ | 3 | EOJ for the ECHONET-ready equipment (referenced or altered) |
| Length | 2 | The number of bytes obtained by adding up the EPC and EPC data. The value 0x01 indicates an alteration response, and the other values indicate a reference response. |
| EPC | 1 | EPC for the property (referenced or altered) |
| EDT | n | If this value is present, it becomes the reference response value (max. 245 bytes). |

ＦＣＣ　：0x＊＊

(6) “Equipment status access request (all)” and “equipment status access response (all)” commands (optional)

These commands allow the ECHONET middleware adapter to read values from and set values in the ECHONET-ready equipment for all properties for which specific services are specified. Specifically, these commands allow the ECHONET middleware adapter to:

- Read the values of properties stored in the ECHONET-ready equipment for all properties for which the IAGet and IAGetM services are specified.
- Read the values of properties stored in the ECHONET middleware adapter for all properties for which the IASet and IASetM services are specified.
- Notify the equipment of the values of properties stored in the ECHONET middleware adapter for all properties for which the IAGet and IAGetM services are specified.
- Notify the equipment of the values of properties stored in the ECHONET middleware adapter for all properties for which the IASet and IASetM services are specified.

When the ECHONET-ready equipment receives an “equipment status access request (all),” it shall confirm whether the request is a request for reading or a request for notification, compose an appropriate response message and send it to the basic middleware adapter in the form of an equipment status access response.

◇ Direction of request commands

ECHONET-ready equipment → ECHONET middleware adapter

◇ Format for request commands

| 1 byte | 2 bytes | 1 byte | 1 byte | 2 bytes | 1 bytes | 3byte | n bytes | 1 byte |
|---|---|---|---|---|---|---|---|---|
| STX | FT | CN | FN | DL | FD(0) | FD(1) | FD(2) | FCC |

STX : 0x02

FT : 0x0003

CN : 0x20

FN : 0x00 – 0xFF

DL : Total size (in hexadecimal) of FD(0)－FD(2)

FD(0) : Indicates the content of the access request.

0x00: Read request for the properties for which the IAGet and IAGetM services are specified.

0x01: Notification request for the properties for which the IAGet and AGetM services are specified.

0x02: Read request for the properties for which the IASet and IASetM services are specified.

0x03: Notification request for the properties for which the IASet and IASetM services are specified.

0x02 to 0xFF: for future reserved.

ＦＤ(1) ：Object code (EOJ)

ＦＤ(2) ：This is present when FD(0) is 0x01 or 0x03.

When FD(0) is 0x01:

Lists the property values of the properties held by the middleware adapter as the IAGet/IAGetM properties excluding the property map properties and the “Version information,” “Manufacturer code,” “Factory code,” “Product code,” “Production number” and “Date of production” properties, in descending order of property code value.

When FD(0) is 0x03:

Lists the property values of the properties held by the middleware adapter as the IASet/IASetM properties excluding the property map properties and the “Version information,” “Manufacturer code,” “Factory code,” “Product code,” “Production number” and “Date of production” properties, in descending order of property code value.

ＦＣＣ ：0x＊＊

◇ Format for response commands

| 1 byte | 2 bytes | 1 byte | 1 byte | 2 bytes | 2 bytes | n bytes | 1 byte |
|---|---|---|---|---|---|---|---|
| ＳＴＸ | ＦＴ | ＣＮ | ＦＮ | ＤＬ | FD(0) | FD(1) | ＦＣＣ |

ＳＴＸ ：0x02

ＦＴ ：0x0003

ＣＮ ：0xA0

ＦＮ ：Value at the time of the request (0x00 if the function is not implemented)

ＤＬ ：Total size (in hexadecimal) of FD(0)－FD(1)

ＦＤ(0) ：Processing result.

| FD(0) value | Content | Remark |
|---|---|---|
| 0x0000 | Request accepted (normal) | |
| 0x0011 | Request rejected (The object specified in the request does not exist.) | |
| 0xFFFF | Request rejected (Reasons other than above) | |
| Other than above | for future reserved | |

ＦＤ(1) ：This is present when the FD(0) of the request message is “0x00” or “0x02” (i.e. a read request) and the FD(0) of the response message is “0x0000” (“request accepted”).

When the FD(0) of the request message is 0x00:

Lists the property values of the properties held by the ECHONET-ready equipment as the IAGet/IAGetM properties excluding the property map properties and the “Version information,” “Manufacturer code,”

"Factory code," "Product code," "Production number" and "Date of production" properties, in descending order of property code value.

When the FD(0) of the request message is 0x02:

Lists the property values of the properties held by the ECHONET-ready equipment as the IASet/IASetM properties excluding the property map properties and the "Version information," "Manufacturer code," "Factory code," "Product code," "Production number" and "Date of production" properties, in descending order of property code value.

ＦＣＣ　：0x＊＊

(7) "Equipment status access UP request (all)" and "equipment status access UP response (all)" commands (optional)

When the ECHONET middleware adapter is equipped with a function that enables it to handle composite messages, these commands allow the ECHONET middleware adapter to read values from and set values in the ECHONET-ready equipment for all properties specified in a composite message. Specifically, these commands allow the ECHONET middleware adapter to:

- Read the values of the target properties stored in the ECHONET-ready equipment in response to a read request from another node for the properties for which the IAGetup and IAGetMup services are specified.
- Notify the ECHONET-ready equipment of the requested values in response to a write request from another node for the properties for which the IASetup and IASetMup services are specified.

When the ECHONET-ready equipment receives an "equipment status access UP request (all)," it shall confirm whether the request is a request for reading or a request for writing, compose an appropriate response message and send it to the basic middleware adapter in the form of an "equipment status access UP response (all)."

If the middleware adapter receives, as a response to this command, a communication error command indicating a "command error," the middleware adapter shall make requests individually for the properties specified in the composite message using the equipment status access request command.

◇ Direction of request commands

ECHONET-ready equipment → ECHONET middleware adapter

◇ Format for request commands

| 1 byte | 2 bytes | 1 byte | 1 byte | 2 bytes | 1 bytes | 3byte | 2 bytes | 1-256 byte | 1 byte |
|---|---|---|---|---|---|---|---|---|---|
| STX | FT | CN | FN | DL | FD(0) | FD(1) | FD(2) | FD(3) | FCC |

ＳＴＸ　：0x02

ＦＴ　：0x0003

CN ：0x21

FN ：000-0xFF

DL ：Total size (in hexadecimal) of FD(0)－FD(3)

FD(0) ：Indicates the content of the access request.

0x00: Read request for the values of the properties for which the IAGetup and IAGetMup services are specified.

0x01: Write request for the values of the properties for which the IASetup and IASetMup services are specified.

0x02 to 0xFF: for future reserved.

FD(1) ：Object code (EOJ)

FD(2) ：Two bytes. Indicates the “number of EPC code values” or the “EPC code value and the number of array elements.”

When the specified property is a non-array type property:

・The first byte shall be fixed at “0x00” and the second byte shall indicate the number of EPC code values.

When the specified property is an array type property:

・The first byte shall indicate the EPC code value (0x80 to 0xFF) and the second byte shall indicate the number of array elements.

FD(3) ：The composition differs between the case where FD(0) is 0x00 and the case where FD(0) is 0x01:

When FD(0) is 0x00 and the first byte of FD(2) is 0x00:

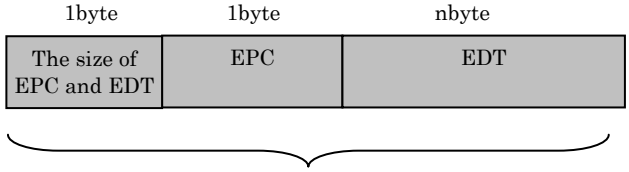
Lists the EPC code values for which an IAGetup request has been made. The number of EPC code values to list is specified by the second byte of FD(2).

When FD(0) is 0x00 and the first byte of FD(2) is other than 0x00 (i.e. the specified property is an array type property):

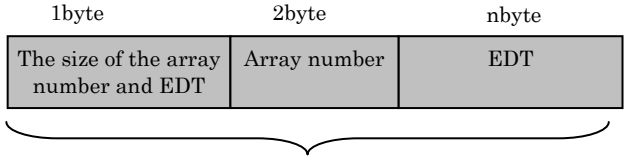
Lists the array element numbers for which an IAGetup request has been made. The number of array elements to list is specified by the second byte of FD(2).

When FD(0) is 0x01, the content of FD(3) differs depending on whether the specified property is a non-array type property or an array type property:

If FD(0) is 0x01 and the specified property is a non-array type property, the following shall be indicated for each of the EPC code values, whose total number is specified by FD(2):




If FD(0) is 0x01 and the specified property is an array type property, the following shall be indicated for each of the array elements, whose total number is specified by FD(2):




ＦＣＣ　：0x＊＊

◇ Format for response commands

| 1byte | 2byte | 1byte | 1byte | 2byte | 2byte | 2byte | nbyte | 1byte |
|---|---|---|---|---|---|---|---|---|
| STX | FT | CN | FN | DL | FD(0) | FD(1) | FD(2) | FCC |

STX　：0x02

FT　：0x0003

CN　：0xA1

FN　：Value at the time of the request (0x00 if the function is not implemented)

DL　：Total size (in hexadecimal) of FD(0)－FD(2)

FD(0)　：Processing result

| FD(0) value | Content | Remark |
|---|---|---|
| 0x0000 | Request accepted (normal) | |
| 0x0011 | Request rejected (The object specified in the request does not exist.) | |
| 0xFFFF | Request rejected (Reasons other than above) | |
| Other than above | for future reserved | |

ＦＤ(1)　：This is present when the FD(0) of the response message is “0x0000” (“request accepted”).

Two bytes. Indicates the “number of EPC code values” or the “EPC code value and the number of array elements.”

When the specified property is a non-array type property:

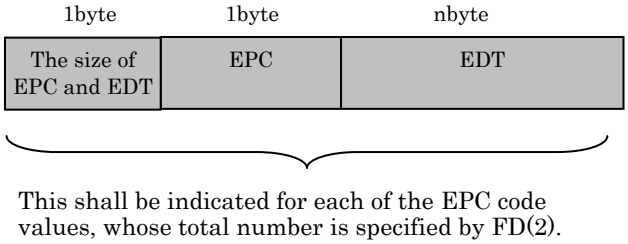
・The first byte shall be fixed at “0x00” and the second byte shall indicate the number of EPC code values.

When the specified property is an array type property:

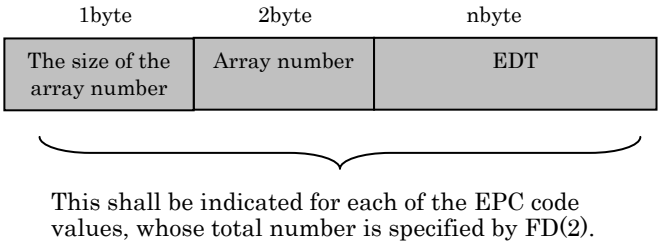
・The first byte shall indicate the EPC code value (other than 0x00) and the second byte shall indicate the number of array elements.

FD(2)　：This is present when the FD(0) of the response message is “0x0000” (“request accepted”) and the FD(0) of the request message is “0x00.”

If the specified property is a non-array type property, the following shall be indicated for each of the EPC code values, whose total number is specified by FD(2):




If the specified property is an array type property, the following shall be indicated for each of the array elements, whose total number is specified by




ＦＣＣ　：0x＊＊

(8) “Equipment status notification request (all)” and “equipment status notification response (all)” commands (optional)

These commands allow the ECHONET-ready equipment to notify values to the ECHONET middleware adapter for all properties for which specific services are specified. Specifically, these commands allow the ECHONET-ready equipment to:

- Request the notification, throughout the ECHONET, of the values of all properties for which the IAGet and IAGetM services are specified.

- Request the notification, throughout the ECHONET, of the values of all properties for which the IASet and IASetM services are specified.
- Request the notification, throughout the ECHONET, of the values of all properties for which the IAGetup and IAGetMup services are specified.
- Request the notification, throughout the ECHONET, of the values of all properties for which the IASetup and IASetMup services are specified.

When the basic middleware adapter receives an “equipment status notification request (all)” from the ECHONET-ready equipment, it shall return an “equipment status notification response (all)” to the ECHONET-ready equipment and announce in the domain the target properties for notification. In the case where the ECHONET middleware adapter is equipped with a function that enables it to handle composite messages, it shall compose and send a composite message.

For the properties for which IAGet, IAGetM, IASet and IASetM are specified, the ECHONET middleware adapter shall overwrite the values it contains with the received values.

◇ Direction of request command

ECHONET-ready equipment → ECHONET middleware adapter

◇ Format for request commands

| 1byte | 2byte | 1byte | 1byte | 2byte | 1byte | 3byte | nbyte | 1byte | mbyte | 1byte |
|---|---|---|---|---|---|---|---|---|---|---|
| STX | FT | CN | FN | DL | FD(0) | FD(1) | FD(2) | FD(3) | FD(4) | FCC |

STX ：0x02

FT ：0x0003

CN ：0x22

FN ：0x＊＊ (0x00 if the function is not implemented)

DL ：Total size (in hexadecimal) of FD(0)－FD(4)

FD(0) ：Indicates the content of the access request

0x00：Notification request for the properties for which the IAGet and IAGetM services are specified.

0x01：Notification request for the properties for which the IASet and IASetM services are specified.

0x02：Notification request for the properties for which the IAGetup and IAGetMup services are specified.

0x03：Notification request for the properties for which the IASetup and IASetMup services are specified.

0x04～0xFF：for future reserved

FD(1) ： Object code (EOJ)

FD(2) ：Lists the values of the target properties specified by FD(0) excluding the property map properties and the “Version information,”

"Manufacturer code," "Factory code," "Product code," "Production number" and "Date of production" properties, in descending order of property code value, regardless of whether a notification is made.

FD(3) ：Indicates the number of the properties to be notified to outside (i.e. the properties listed in FD(4)) out of the properties specified by FD(2).

FD(4) ：Lists the property code values to be notified to outside out of the properties specified by FD(2).

FCC ：0x＊＊

◇ Format for response commands

| 1byte | 2byte | 1byte | 1byte | 2byte | 2byte | 1byte |
|---|---|---|---|---|---|---|
| STX | FT | CN | FN | DL | FD(0) | FCC |

ＳＴＸ　：0x02
ＦＴ　：0x0003
ＣＮ　：0xA2
ＦＮ　：The value at the time of the request
ＤＬ　：0x0002
ＦＤ(0)　：Processing result (“Result”)

| Name | Size (bytes) | Explanation |
|---|---|---|
| Result | 2 | Response result:<br>0x0000: Acceptance response<br>0x0011: Rejection response (“network out of operation”)<br>0x0012: Rejection response (others)<br>0x0101: Status discrepancy error (“device interface confirmation” state (“unconfirmed”))<br>0x0103: Status discrepancy error (“standby” state)<br>0x0104: Status discrepancy error (“object construction” state)<br>0x0105: Status discrepancy error (“error top |

FCC　：0x＊＊

(9) “Object access request (all)” and “object access response (all)” commands (optional)

These commands allow the ECHONET-ready equipment to read values from and set values in the ECHONET middleware adapter for all properties for which specific services are specified. Specifically, these commands allow the ECHONET-ready equipment to:

- Read the values of the properties stored in the ECHONET middleware adapter for which the IASet and IASetM services are specified.
- Write the values of properties in the ECHONET middleware adapter for all properties for which the IAGet and IAGetM services are specified.

When the ECHONET middleware adapter receives an “object access request (all),” it shall confirm whether the request is a request for reading or a request for writing, compose an appropriate response message and send it to the ECHONET-ready equipment in the form of an “object access response (all)”.

◇ Direction of request command

ECHONET-ready equipment → ECHONET middleware adapter

◇ Format for request commands

| 1byte | 2byte | 1byte | 1byte | 2byte | 1byte | 3byte | nbyte | 1byte |
|---|---|---|---|---|---|---|---|---|
| STX | FT | CN | FN | DL | FD(0) | FD(1) | FD(2) | FCC |

STX ：0x02

FT ：0x0003

CN ：0x23

FN ：0x＊＊(0x00 if the function is not implemented)

DL ：Total size (in hexadecimal) of FD(0)－FD(2)

FD(0) ：Indicates the content of the access request.

0x00：Property value read request for the properties for which the IASet and IASetM services are specified.

0x01：Property value write request for the properties for which the IAGet and IAGetM services are specified.

0x02～0xFF：for future reserved

ＦＤ(1) ：Object code (EOJ)

ＦＤ(2) ：This is present when FD(0) is “0x01.”

Lists the property values of the properties held by the ECHONET-ready equipment as the IAGet/IAGetM properties excluding the property map properties and the “Version information,” “Manufacturer code,” “Factory code,” “Product code,” “Production number” and “Date of production” properties, in descending order of property code value.

ＦＣＣ ：0x＊＊

◇ Format for response commands

| 1byte | 2byte | 1byte | 1byte | 2byte | 2byte | nbyte | 1byte | mbyte | 1byte |
|---|---|---|---|---|---|---|---|---|---|
| STX | FT | CN | FN | DL | FD(0) | FD(1) | FD(2) | FD(3) | FCC |

STX ：0x02

FT ：0x0003

CN ：0xA3

FN ：Value at the time of the request

DL ：Total size (in hexadecimal) of FD(0)－FD(3)

FD(0) ：Processing result

| Name | Size (byte) | Explanation |
|---|---|---|
| Result | 2 | Response result:<br>0x0000: Acceptance response<br>0x0001: Acceptance response (“network out of operation”)<br>0x0011: Rejection response<br>0x0101: Status discrepancy error (“device |

| | | |
|---|---|---|
| | | interface confirmation" state ("unconfirmed"))<br>0x0103: Status discrepancy error ("standby" state)<br>0x0104: Status discrepancy error ("object construction" state)<br>0x0105: Status discrepancy error ("error stop" state)<br>0xFFFF: Other error |

FD(1) : This is present when the FD(0) of the response message is "0x0000" or "0x0001" (i.e. an acceptance response) and the FD(0) of the request message is "0x00."
Lists the property values of the properties held by the ECHONET-ready equipment as the IASet/IASetM properties excluding the property map properties and the "Version information," "Manufacturer code," "Factory code," "Product code," "Production number" and "Date of production" properties, in descending order of property code value.

FD(2) : Indicates the number of the properties to list in FD(3) out of the properties specified by FD(1). This is present when FD(1) is present in the response message.

FD(3) : Lists the property code values of the properties which are specified by FD(1) and to which a data write has been performed from outside after the last access request. This is present when FD(1) is present in the response message.

FCC : 0x**

### 8.8.4.5 Communication error notification command (Required)

If, during the process of receiving a frame, a communication error falling under any of the definitions given in the table below occurs on the middleware adapter / ECHONET-ready equipment, a communication error notification frame must be sent as response data.

◇ Direction of request commands

ECHONET middleware adapter → ECHONET-ready equipment

◇ Format for request commands

| 1 byte | 2 bytes | 1 byte | 1 byte | 2 bytes | 1 byte |
|---|---|---|---|---|---|
| STX | FT | CN | FN | DL | FCC |

STX : 0x02

FT : 0x00FF

CN : Error number

| Error name | Command number | Description of error |
|---|---|---|
| FCC error | 0x00 | The frame was received successfully but the FCC is abnormal. |
| Command error | 0x01 | There is no corresponding command number. |
| Result error | 0x02 | The result of the received response frame is an undefined value. |
| Intra-frame error | 0x03 | The frame was received successfully, but the format is abnormal. |
| Other error | 0xFF | Frame reception errors other than above. |

## 8.8.5 Communication sequences (Object generation type)

This subsection defines the communication sequences for object generation type methods used for ECHONET middleware adapter communication interfaces. Detailed sequence-by-sequence explanation is given below.

**Table 8.10 Communication Sequences (Object Generation Type)**

| No. | Sequence Name | Description | Note |
|---|---|---|---|
| 1 | Data confirmation sequence | Sequence allowing the ECHONET middleware adapter to confirm the interface method and objects with the ECHONET-ready equipment | |
| 2 | Initialization sequence | Sequence allowing the ECHONET-ready equipment to perform an initialization for the ECHONET middleware adapter | |
| 3 | Object construction mode operation sequence | Sequence allowing the ECHONET middleware adapter to acquire object information from the ECHONET-ready equipment and generate internal objects | |
| 4 | Equipment status access request sequence | Sequence allowing the ECHONET middleware adapter to report a Set/Get request (where there is a property in the ECHONET-ready equipment) | |
| 5 | Equipment status notification request sequence | Sequence allowing the ECHONET-ready equipment to write values and send notifications to the outside (where there is a property in the ECHONET middleware adapter) | |
| 6 | Element designation equipment status access request sequence | Sequence allowing the ECHONET middleware adapter to report a Set/Get request (where there is a property in the ECHONET-ready equipment) (element designation) | |
| 7 | Element designation equipment status notification request sequence | Sequence allowing the ECHONET-ready equipment to write values and send notifications to the outside (where there is a property in the ECHONET middleware adapter) (element designation) | |
| 8 | Object access request sequence | Sequence allowing the ECHONET-ready equipment to access properties held by the ECHONET middleware adapter | |

### 8.8.5.1 Equipment interface confirmation mode

(1) Data confirmation sequences (Required)

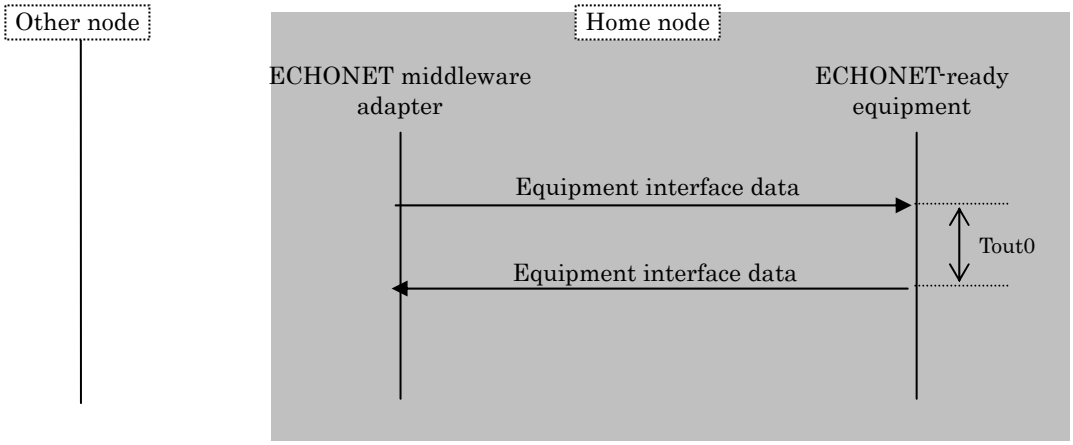



**Fig. 8.21 Data Confirmation Sequence**

【Equipment Interface Data Confirmation Sequence for the Equipment Side】

The ECHONET-ready equipment does not accept frames other than equipment interface data confirmation frames while it is in the "equipment interface confirmation" state. Upon reception of an equipment interface data confirmation request, the ECHONET-ready equipment confirms the adapter type and objects and returns an equipment interface data confirmation response within the Tout30 period. In the case of an adapter type mismatch or object data mismatch, an error is returned in the "result" field.

In the case of an adapter type mismatch, the ECHONET-ready equipment outputs a reset signal to the adapter after the elapse of the Tout40 period and waits for an equipment interface confirmation request. If the error state persists after repeating the attempt 3 times, the ECHONET-ready equipment sets the "result" field to "equipment interface data disposal" and returns an equipment interface data confirmation response. After the elapse of the Tout50 period, the ECHONET-ready equipment discards the equipment interface data, outputs a reset signal and makes a shift to the "unrecognized" state.

If, with all communication errors (such as frame errors) ignored by the ECHONET-ready equipment, no equipment interface data confirmation request comes in during the Tout50 period as measured from the transmission of a reset signal to the adapter or from a power ON reset, the ECHONET-ready equipment shall output a reset signal to the adapter and wait for an equipment interface confirmation request.

If the error state persists after repeating the attempt 3 times, the ECHONET-ready equipment discards the equipment interface data, outputs a reset signal to the adapter and makes a shift to the "unrecognized" state.

The operation on the ECHONET-ready equipment side during the "unrecognized" state shall be as per the equipment interface data recognition service specifications.

【Equipment Interface Data Confirmation Sequence for the Adapter Side】

After power ON, the basic middleware adapter sends an equipment interface data confirmation request to the ECHONET-ready equipment to confirm the objects and communication method for the equipment adapter interface maintained. If no object exists, an enquiry is made using 0 for the number of objects.

Based on the "result" of the equipment interface data confirmation response, the following processing is performed:

* If the "result" is "normal completion," a shift is made to the "standby" state.
* If the "result" is "adapter type mismatch," a shift is made to the "standby" state.
* If the "result" is "object mismatch," a shift is made to the "standby" state after discarding the device object data.
* If the "result" is "equipment interface data disposal," the device object and equipment interface data is discarded and a shift is made to the "unrecognized" state within the Tout50 period.

If, with all communication errors (such as frame errors) ignored by the adapter, no response comes in during the Tout61 period, an equipment interface data confirmation request is sent again.

If no response comes in after this, the device object and equipment interface data is discarded and a shift is made to the "unrecognized" state.

### 8.8.5.2 Adapter initialization mode

(1) Initialization sequence (required)

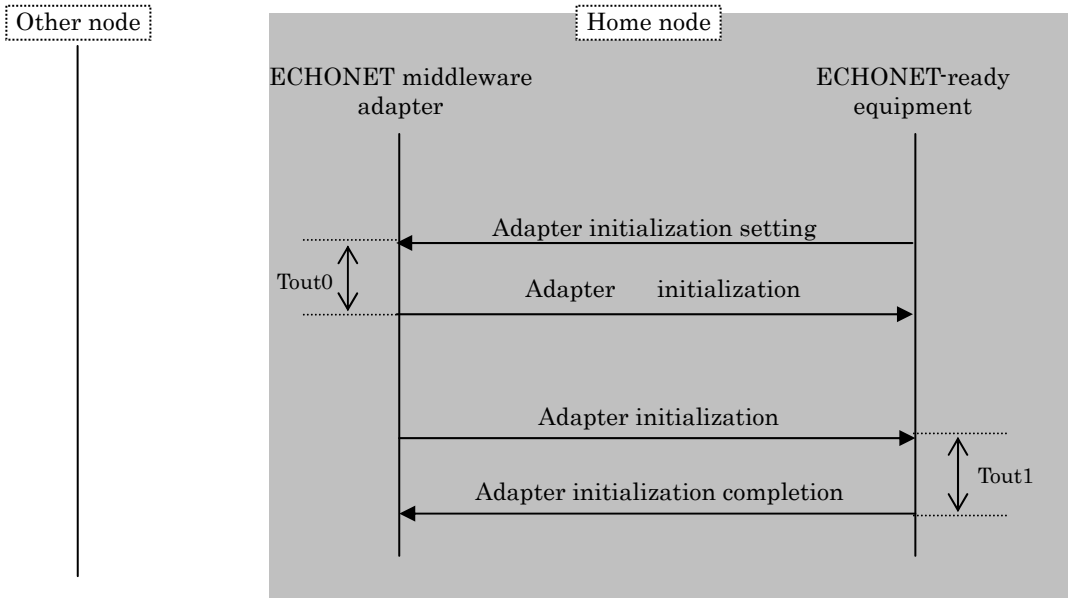



**Fig. 8.22 Initialization Sequence**

【Initialization Sequence for the Equipment Side】

After a shift to the "object construction" state, the ECHONET-ready equipment sends to the basic middleware adapter an adapter initialization setting request for an initialization.

If no initialization acceptance response comes in during the Tout0 period, the request is resent, but only once. If no initialization acceptance response comes in after this retransmission, the ECHONET-ready equipment starts operating in the stand-alone mode.

After receiving an initialization acceptance response, the ECHONET-ready equipment waits for an initialization setting completion notification at least for the Tout11 period. If no initialization setting completion notification is received during this period, the ECHONET-ready equipment terminates the initialization sequence and starts operating in the stand-alone mode.

If an initialization setting completion notification comes in, the ECHONET-ready equipment returns to the basic middleware adapter an adapter initialization setting completion notification acceptance response within the Tout1 period.

【Initialization Sequence for the Adapter Side】

If the basic middleware adapter receives an adapter initialization setting request for an initialization with equipment data retained, it will send back an adapter initialization setting response within the Tout0 period and start an initialization of the inside (i.e. the specified startup processing of the lower-layer communication software).

If the basic middleware adapter receives an adapter initialization setting request for an initialization with equipment data discarded, the device object data will be discarded and then the above-mentioned processing will be performed.

Upon completion of the initialization, an adapter initialization completion notification shall be sent to the ECHONET-ready equipment.

If no "initialization successfully completed" adapter initialization completion notification acceptance response is received within the Tout1 period after the transmission of the above-mentioned notification, the notification shall be resent, but only once. If no response is received after the retransmission, the basic middleware adapter shall stop the processing and start waiting for an adapter initialization setting request.

#### 8.8.5.3 Object construction mode

Fig. 8.23 shows the sequence for the object construction mode for the case where the device objects are retained, and Fig. 8.24 shows the sequence for the object construction mode for the case where the device objects are not retained.

(1) Operation sequence for the object construction mode (Required)

a) When there is an device object

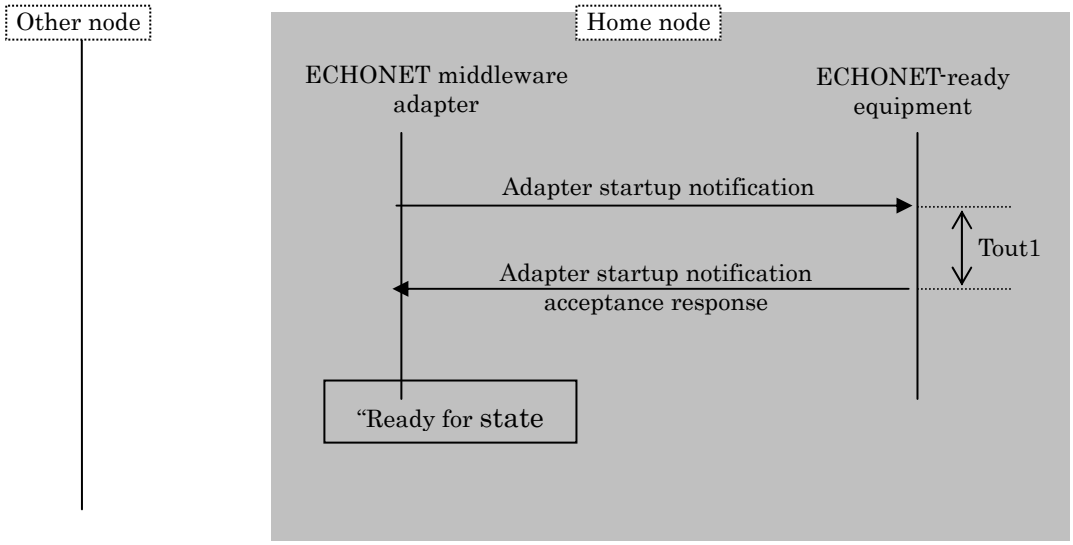



**Fig. 8.23 Object Operation Sequence (1)**

【Object construction sequence for the adapter side】

The adapter shall first send an adapter startup notification to the ECHONET-ready equipment. The adapter shall then confirm that an adapter startup notification acceptance response from the ECHONET-ready equipment has been received within the Tout1 period after the transmission of the notification. Then, a state change shall be made to the "normal operation" state.

【Object construction sequence for the equipment side】

If the equipment receives an adapter startup notification, it shall send an adapter startup notification acceptance response within the Tout1 period.

b) When there is no device object

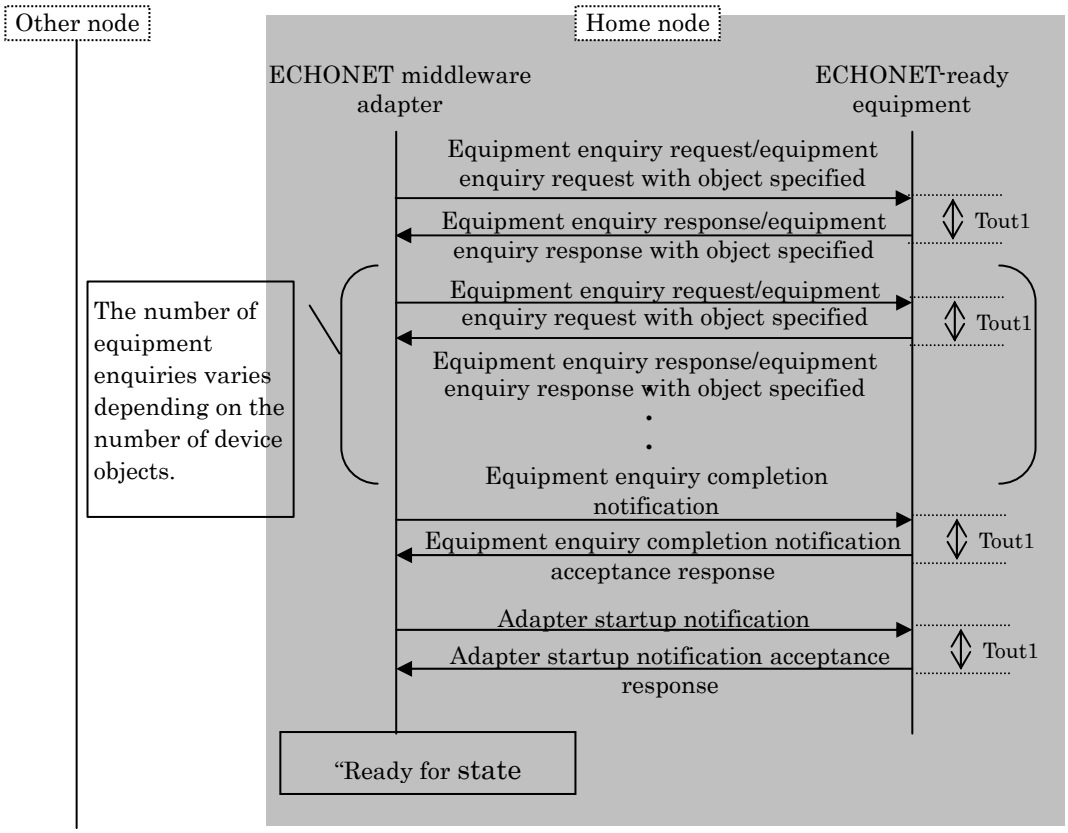



**Fig. 8.24 Object Operation Sequence (2)**

【Object construction sequence for the adapter side】

The adapter shall first send an equipment enquiry request (equipment enquiry request with object specified) to the ECHONET-ready equipment. The adapter shall then confirm that an equipment enquiry response (equipment enquiry response with object specified) from the ECHONET-ready equipment has been received within the Tout1 period after the transmission of the request. If there are 2 or more objects, this processing shall be repeated until all object data is acquired.

After acquiring the object data, the adapter shall send an equipment enquiry completion notification to the ECHONET-ready equipment. The adapter shall then confirm that an equipment enquiry completion notification acceptance response from the ECHONET-ready equipment has been received within the Tout1 period after the transmission of the notification.

Then, the adapter shall send an adapter startup notification to the ECHONET-ready equipment. The adapter shall then confirm that an adapter startup notification acceptance response from the ECHONET-ready equipment has been received within the Tout1 period after the transmission of the notification. After the confirmation, a state change to the "normal operation" state shall be made.

【Object construction sequence for the equipment side】

If the equipment receives an equipment enquiry request (equipment enquiry request with object specified), it shall send an equipment enquiry response (equipment enquiry response with object specified) within the Tout1 period. If the number of implemented pieces of object data is 2 or more, the processing shall be repeated. If the equipment receives an equipment enquiry completion notification, it shall send an equipment enquiry completion notification acceptance response within the Tout1 period.

If the equipment receives an adapter startup notification, it shall send an adapter startup notification acceptance response within the Tout1 period.

### 8.8.5.4 ECHONET communication mode

There are 5 command sequences for the ECHONET communication mode; “equipment status access request” (Fig. 8.25), “equipment status notification request” (Fig. 8.26), “element designation equipment status access request” (Fig. 8.27), “element designation equipment status notification request” (Fig. 8.28) and “object access request” (Fig. 8.29).

(1) Equipment status access request sequence (Required)

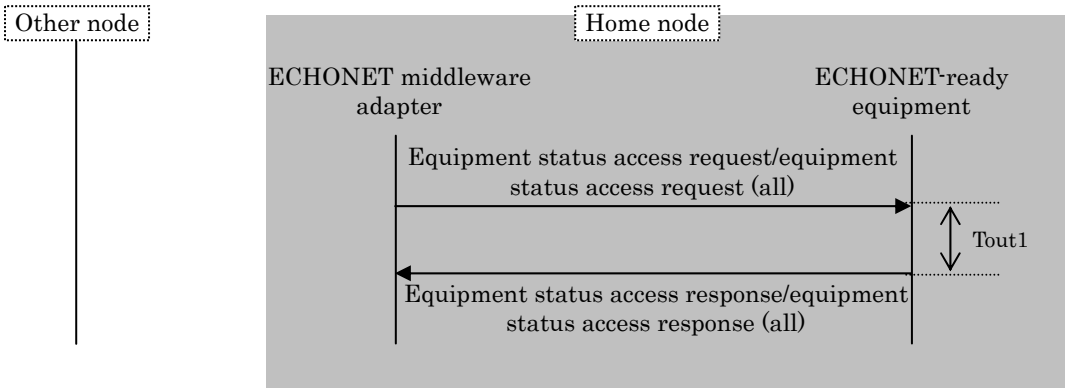



**Fig. 8.25 Equipment Status Access Request Sequence**

【Equipment status access request sequence for the adapter side】

The adapter shall first send an equipment status access request (equipment status access request (all)) to the ECHONET-ready equipment. The adapter shall then confirm that an equipment status access response (equipment status access response (all)) from the ECHONET-ready equipment has been received within the Tout1 period after the transmission of the request.

【Equipment status access request sequence for the equipment side】

If the equipment receives an equipment status access request (equipment status access request (all), it shall send an equipment status access response (equipment status access

response (all) within the Tout1 period.

(2) Equipment status notification request sequence (Required)

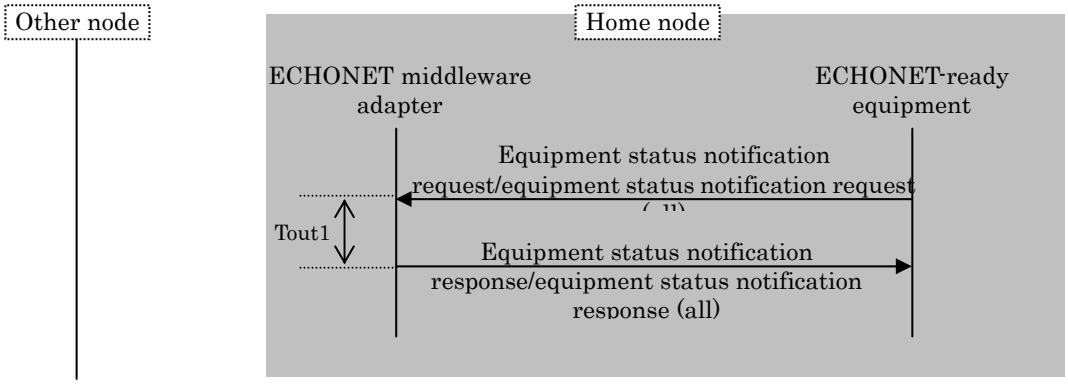



**Fig. 8.26 Equipment Status Notification Request Sequence**

【Equipment status notification request sequence for the adapter side】
If the adapter receives an equipment status notification request (equipment status notification request (all)) from the ECHONET-ready equipment, it shall send an equipment status notification response (equipment status notification response (all)) to the ECHONET-ready equipment within the Tout1 period.

【Equipment status notification request sequence for the equipment side】
The equipment shall first send an equipment status notification request to the adapter. The equipment shall then confirm that an equipment status notification response (equipment status notification response (all)) has been received within the Tout1 period after the transmission of the request.

(3) Element designation equipment status access request sequence (Required)

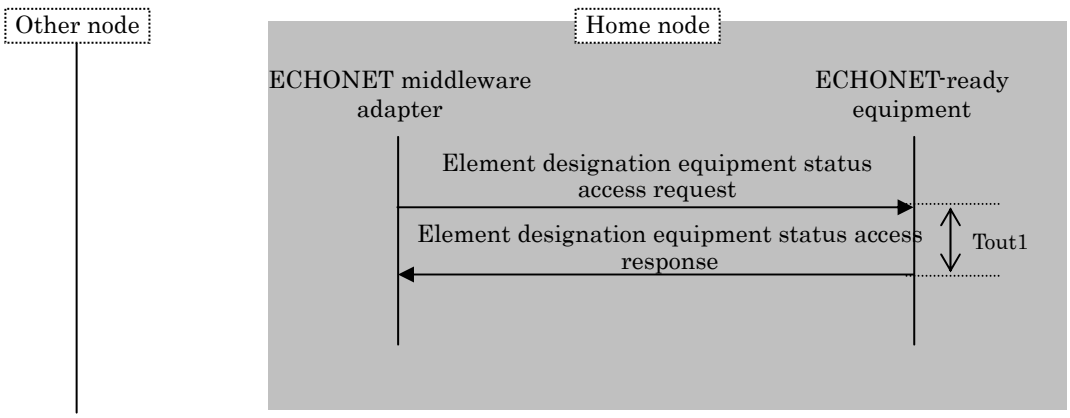



**Fig. 8.27 Element Designation Equipment Status Access Request Sequence**

【Element designation equipment status access request sequence for the adapter side】
The adapter shall first send an element designation equipment status access request to the ECHONET-ready equipment. The adapter shall then confirm that an element designation equipment status access response from the ECHONET-ready equipment has been received within the Tout1 period after the transmission of the request.

【Element designation equipment status access request sequence for the equipment side】
If the equipment receives an element designation equipment status access request, it shall send an element designation equipment status access response within the Tout1 period.

(4) Element designation equipment status notification request sequence (Required)

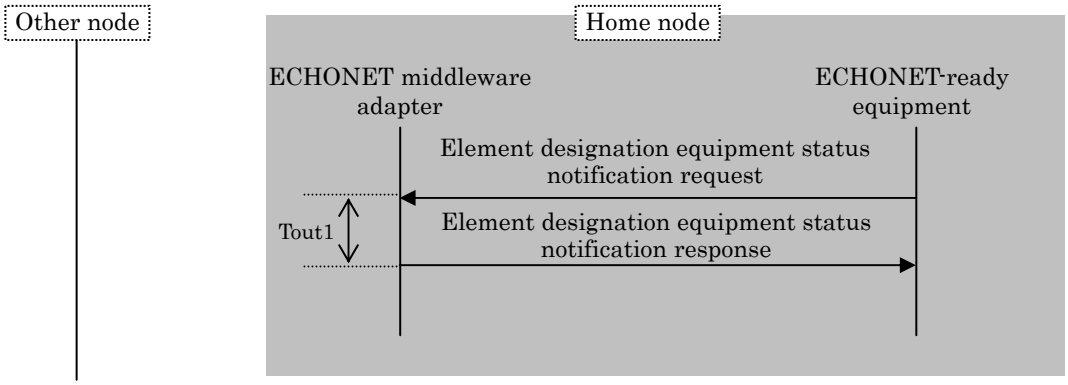



**Fig. 8.28 Basic Sequence for Object Generation**

【Element designation equipment status notification request sequence for the adapter side】
If the adapter receives an element designation equipment status notification request from the ECHONET-ready equipment, it shall send an element designation equipment status notification response to the ECHONET-ready equipment within the Tout1 period.

【Element designation equipment status notification request sequence for the equipment side】
The equipment shall first send an element designation equipment status notification request. The equipment shall then confirm that an element designation equipment status notification response has been received within the Tout1 period after the transmission of the request.

(5) Object access request sequence (Required)

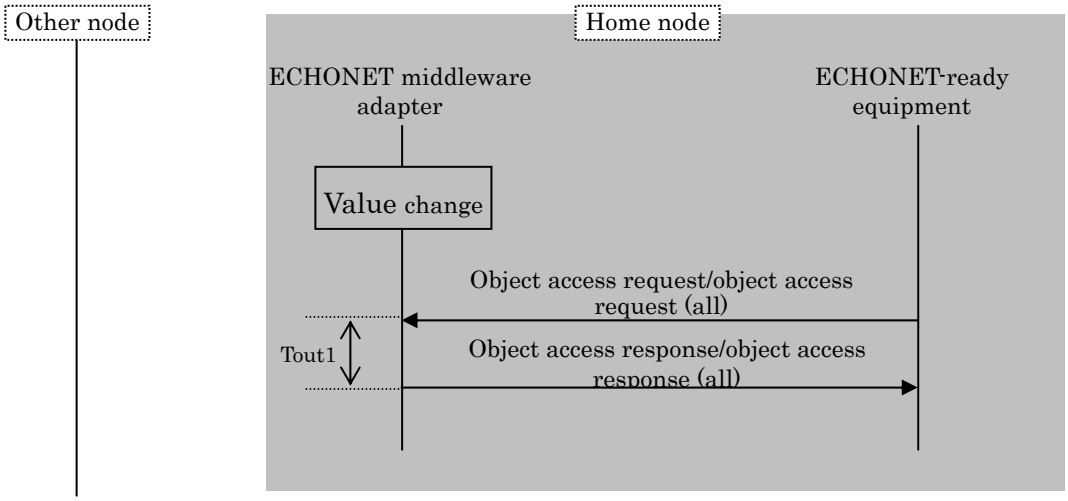



**Fig. 8.29 Object Access Request Sequence**

【Object access request sequence for the adapter side】

If the adapter receives an object access request (object access request (all)) from the ECHONET-ready equipment, it shall send an object access response (object access response (all)) to the ECHONET-ready equipment within the Tout1 period.

【Object access request sequence for the equipment side】

The equipment shall first send an object access request (object access request (all)). The equipment shall then confirm that an object access response (object access response (all)) has been received within the Tout1 period after the transmission of the request.

### 8.8.5.5 Communication error processing sequence for the equipment side

If a communication error occurs while receiving a frame from the middleware adapter, the ECHONET-ready equipment discards that frame and sends a communication error notification frame.

If the ECHONET-ready equipment receives a communication error frame from the middleware adapter as a response to a command that the ECHONET-ready equipment sent, the ECHONET-ready equipment decides that the command was not successfully communicated and resends the command, as in the case of a timeout.

It is recommended that a means be provided to indicate errors when communication errors occur frequently.

#### 8.8.5.6 Communication error processing sequence for the adapter side

If a communication error occurs while receiving a frame from the ECHONET-ready equipment, the middleware adapter discards that frame and sends a communication error notification frame.

If the middleware adapter receives a communication error frame from the ECHONET-ready equipment as a response to a command that the middleware adapter sent, the middleware adapter decides that the command was not successfully communicated and resends the command, as in the case of a timeout.

When communication errors occur frequently, the middleware adapter sets the error property to the corresponding error code value, sets the error status property to the value that corresponds to error occurrence and performs the communication error processing specified in Subsection 8.3.3.

#### 8.8.5.7 Number of frames that can be sent simultaneously

In the case of a basic middleware adapter, the ECHONET middleware adapter and ECHONET-ready equipment shall not send a second request or notification frame until they receive a response frame for the first request or notification frame. Any new request or notification frame received before sending back a response frame must be discarded.

#### 8.8.5.8 Handling of composite messages

When a basic middleware adapter equipped with a function that enables it to handle composite messages receives a composite Get request message from another node in the ECHONET, the following processing shall be performed:

(1) If the properties specified in the composite message do not include a property for which the IAGetup service is specified, the basic middleware adapter shall compose and send a response message.

(2) If the properties specified in the composite message include a property for which the IAGetup service is specified, the values shall be acquired from the equipment using the “equipment status access request” command or the “equipment status access UP request (all)” command. After the acquisition of the values, a response message shall be composed and transmitted.

When a basic middleware adapter equipped with a function that enables it to handle composite messages receives a composite Set request message from another node in the ECHONET, the following processing shall be performed:

(1) If the properties specified in the composite message do not include a property for which the IASetup service is specified, the basic middleware adapter shall compose and send a response message.

(2) If the properties specified in the composite message include a property for which the

IASetup service is specified, the Set request shall be notified using the "equipment status access request" command or the "equipment status access UP request (all)" command. After this, a response message shall be composed and transmitted.

### 8.8.5.9 Timeout

The timeout values shall be as follows:

Table 8.11 Timeout Values

| Symbol | Name | Timeout Value | Explanation |
|---|---|---|---|
| Tout0, Tout1 | Maximum response time 1 | 3 sec. | The maximum time the middleware adapter or ECHONET-ready equipment can take to send back a response after receiving a request from the ECHONET-ready equipment or middleware adapter, respectively. |
| Tout10 | Maximum initialization processing time | 5 sec. | The maximum time that can be spent to perform initialization processing. |
| Tout11 | Maximum initialization processing waiting time | 6 sec. | The maximum time that can be spent waiting for the completion of initialization processing. |
| Tout2 | Maximum response time 2 | 5 sec. | The maximum time a node can take to send back a response after receiving a request from another node. |
| Tout30 | Maximum equipment interface data confirmation processing waiting time (ECHONET-ready equipment) | 3 sec. | The maximum time the ECHONET-ready equipment can take to send back an equipment interface data confirmation response to the middleware adapter. |
| Tout40 | Minimum object data disposal waiting time | 6 sec. | The maximum time the ECHONET-ready equipment can wait for the middleware adapter to discard the equipment interface data. |
| Tout50 | Maximum equipment interface data confirmation request issuance time | 3 sec. | The maximum time that can elapse before the ECHONET-ready equipment sends back an equipment interface data confirmation response to the middleware adapter. |
| Tout61 | Maximum equipment interface data confirmation processing waiting time (middleware adapter) | 5 sec. | The maximum time the middleware adapter can wait for an equipment interface data confirmation response. |

* Note: Tout61 > Tout30, Tout61 > Tout50

## 8.8.6 Mechanical and physical characteristics – Object generation method

The mechanical and physical characteristics requirements for communication interfaces shall basically be the same as those for the equipment interface data recognition service, with the exception of the following:

○ RST (reset)

Reset output from the ECHONET-ready equipment to the adapter:

Low = “operation stop” state

Low → High = reset start

ECHONET-ready equipment: Compulsory

Middleware adapter: Compulsory

○ Transmission speeds: 2400 bps / 9600 bps / 19200 bps / 28800 bps / 38400 bps / 57600 bps / 115200 bps

## 8.9 Communication Protocols for the “Peer-to-Peer Type”

Communication protocols for the “Peer-to-Peer Type” allow an ECHONET middleware adapter and a piece of ECHONET-ready equipment to exchange data using a user-defined communication method. The “Peer-to-Peer Type” can be achieved by either of the following:

Program Selection Method
Program Download Method

### 8.9.1 Program Selection Method

The Program Selection Method is a method in which a peer-to-peer type communication method that supports ECHONET-ready equipment (ECHONET middleware adapter communication software) is implemented in advance in ECHONET middleware adapters. This ECHONET Specification does not specify requirements for the Program Selection Method.

## 8.9.2 Program Download Method

The Program Download Method is a method in which a peer-to-peer type communication method that supports ECHONET-ready equipment (ECHONET middleware adapter communication software (hereinafter referred to as "the program")) is obtained from outside by means of downloading. This version of the ECHONET Specification specifies the requirements for protocols to download the program from ECHONET-ready equipment. This version of the ECHONET Specification also defines the (recommended) specifications for program execution environments to execute the downloaded program in the ECHONET middleware adapter (See Fig. 8.30).

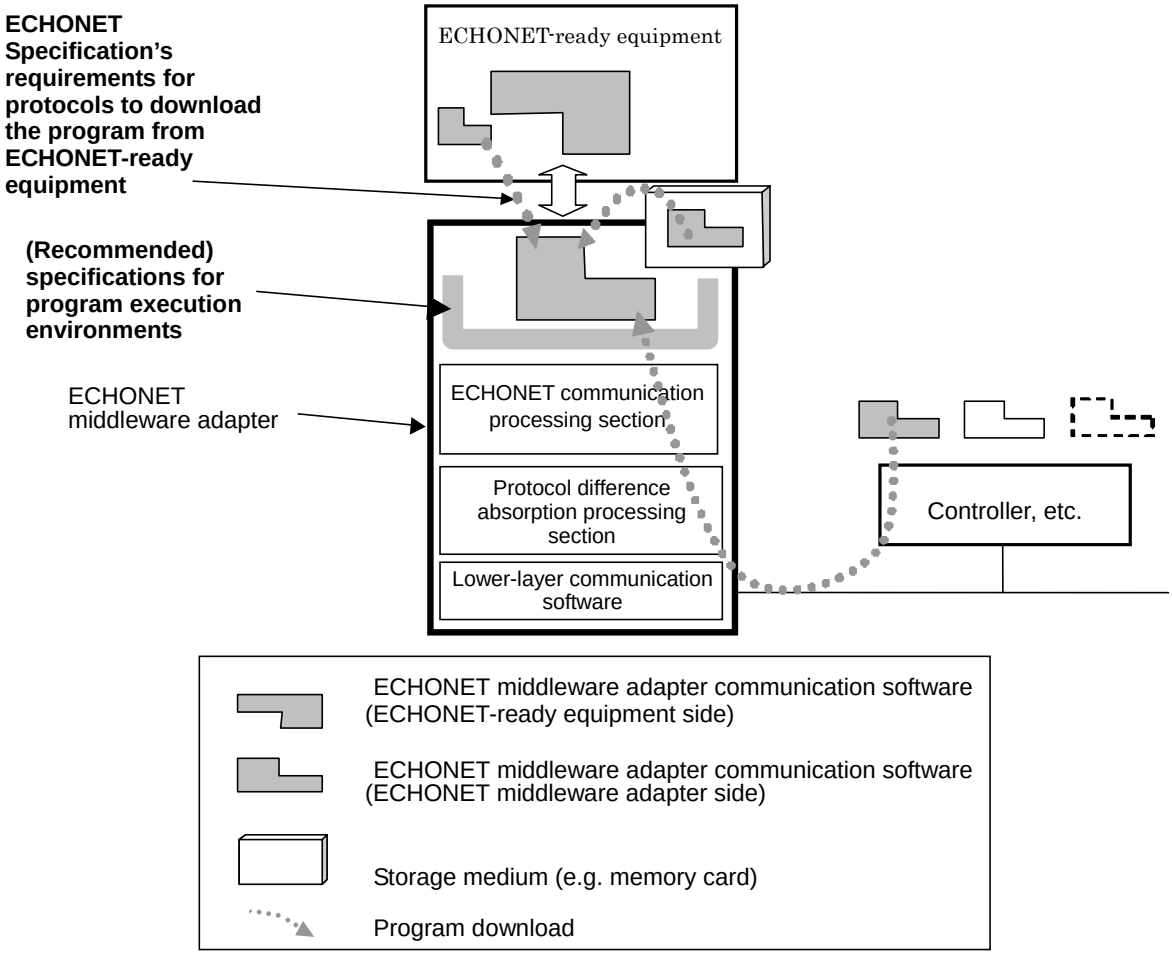



**Fig. 8.30 Scope of the ECHONET Specification with Regard to the Program Download Method**

## 8.9.3 Protocols to download the program from ECHONET-ready equipment

This section specifies the requirements for protocols to download the program from ECHONET-ready equipment via the ECHONET middleware adapter communication interface.

### 8.9.3.1 Composition of the frame for downloading the program

The composition of the frame used for protocols for the Program Download Method is as shown in Fig. 8.31. The "frame type code (FT)," "command number code (CN)," "frame number code (FN)," "data length code (DL)" and "frame data (FD)" fields are the DATA of the ECHONET middleware adapter transmission interface protocol.

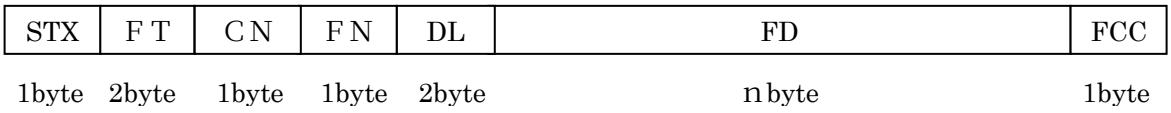



**Fig. 8.31 Composition of the Frame for the Program Download Method**

(1) STX (Control code)

Control code. The value is fixed at 0x02.

(2) FT (Frame type)

Indicates the frame type. For protocols to download the program via the ECHONET middleware adapter communication interface, the value is fixed at 0x0100.

(3) CN (Command number code)

A 1-byte code that specifies the command for the protocol to download the program via the ECHONET middleware adapter communication interface. In communications based on this version of the ECHONET Specification, the commands listed in Table 8.12 shall be used. The code values to which no command is assigned are reserved for future use.

**Table 8.12 Command Code Values for the Program Download Interface**

| Command name | Command number code (CN) | Implementation |
|---|---|---|
| Program download request | 0x00 | Required |
| Program download response | 0x80 | Required |
| Download completion notification | 0x01 | Required |
| Download completion response | 0x81 | Required |

(4) FN (Frame number)

This is a number (0x01 to 0xFF) assigned by the requesting side. The requesting side must

assign numbers incrementally. Each response frame shall be assigned with the same number as the corresponding request frame.

In the case of ECHONET-ready equipment that cannot assign response frames with the same numbers as the corresponding request frames, the fixed frame number value “0x00” shall be used.

(5) DL (Data length code)

The data length code is a 2-byte-long code indicating the size of the frame data (FD) section that follows. The size shall be in bytes and shall be indicated in hexadecimal. For example, if the size of the FD section is 20 bytes, the DL value is 0x0014, which represents 20 bytes. The data shall be arranged in the big endian order.

(6) FD (Frame data)

The frame data section is the field for the data, which is defined by the frame type (FT) value and the command number code (CN) value. When there are 2 or more bytes of data, the data shall be arranged in the big endian order. The specific composition is defined for each command number code (CN) value.

(7) FCC (Frame check code)

The frame check code is a 1-byte check code. The FCC value shall be the 2’s complement of the sum of the data from “frame type” to “frame data.”

### 8.9.3.2 Program download commands

This section specifies the requirements for the commands for the program download protocol for the ECHONET middleware adapter communication interface.

(1) Program download request / program download response commands (Required)

◇ Direction of request commands

ECHONET middleware adapter → ECHONET-ready equipment

◇ Format for request commands

| 1byte | 2byte | 1byte | 1byte | 2byte | 2byte | 7byte | 9byte | 1byte |
|---|---|---|---|---|---|---|---|---|
| STX | FT | CN | FN | DL | FD(0) | FD(1) | FD(2) | FCC |

STX : 0x02

FT : 0x0100

CN : 0x00

FN : 0x＊＊

DL : 0x0012

FD(0) ：0x**** Transmission request sequence number (2 bytes):

0x0000：Request for acquisition of the data to transmit

0x0001-：Split data sequence number

FD(1) ：Information on the target device (7 bytes):

ECHONET manufacturer code (3 bytes)

Model code (2 bytes)

Type code (2 bytes)

* The above shall be filled with the information acquired using the equipment interface data recognition service.

FD(2) ：Software information (9 bytes):

ECHONET manufacturer code (3 bytes)

Program identifier (6 bytes)

* If a download has already been performed and the ECHONET middleware adapter already contains the program, the above shall be filled with that data.

* If no download has been performed and the ECHONET middleware adapter does not contain the program, all of the above shall be filled with “0xff.”

FCC: 0x**

◇ Response command

a) When the FD(0) of the request command is “0x00” (Request for acquisition of the data to transmit)

| 1byte | 2byte | 1byte | 1byte | 2byte | 1byte | 2byte | 9byte | 2byte | 1byte |
|---|---|---|---|---|---|---|---|---|---|
| STX | FT | CN | FN | DL | FD(0) | FD(1) | FD(2) | FD(3) | FCC |

STX ：0x02

FT ：0x0100

CN ：0x80

FN ：0x＊＊

DL ：0x000e

FD(0) ：Data type (Program 0x00 / Acquisition source 0x01)

* “Acquisition source” provides such information as the URL, download data path, etc. This ECHONET Specification does not specify requirements regarding the description method or acquisition source-based processing.

FD(1) ：Number of split messages (2 bytes)

FD(2) ：Software information (9 bytes):

ECHONET manufacturer code (3 bytes)

Program identifier (6 bytes)

* “Program identifier” is an identifier that allows the program to be uniquely identified at the ECHONET-ready equipment manufacturer. The program identifier

shall be defined independently by the manufacturer.

* If FD(0) is "0x01" (i.e. the response data is "Acquisition source"), all of the above shall be filled with "0xff."

FD(3) : Download data size (2 bytes)

FCC : 0x＊＊

b) When the FD(0) of the request command is other than "0x00" (Split data sequence number)

| 1byte | 2byte | 1byte | 1byte | 2byte | 2byte | Up to 128 | 1byte |
|---|---|---|---|---|---|---|---|
| STX | FT | CN | FN | DL | FD(0) | FD(1) | FCC |

STX : 0x02

FT : 0x0100

CN : 0x80

FN : 0x＊＊

DL : 0x＊＊＊＊

FD(0) : Split data sequence number

FD(1) : Download data (n bytes, up to 128 bytes)

FCC : 0x＊＊

(2) "Download completion notification" and "download completion response" commands (Required)

◇ Direction of notification command

ECHONET middleware adapter → ECHONET-ready equipment

◇ Notification command

| 1byte | 2byte | 1byte | 1byte | 2byte | 1byte | 1byte |
|---|---|---|---|---|---|---|
| STX | FT | CN | FN | DL | FD(0) | FCC |

ＳＴＸ : 0x02

ＦＴ : 0x0100

ＣＮ : 0x01

ＦＮ : 0x＊＊

ＤＬ : 0x0001

ＦＤ(0) : Download result

0x00：Failure

0x01：Success

ＦＣＣ : 0x＊＊

◇ Response command

| 1byte | 2byte | 1byte | 1byte | 2byte | 1byte |
|---|---|---|---|---|---|
| STX | FT | CN | FN | DL | FCC |

ＳＴＸ　：0x02
ＦＴ　：0x0100
ＣＮ　：0x81
ＦＮ　：0x＊＊
ＤＬ　：0x0000
ＦＣＣ　：0x＊＊

### 8.9.3.3 Download data format specifications

Fig. 8.32 shows the download data format. The download data shall consist of a 1-byte program execution environment identifier and the program itself (the main body of the program). The program execution environment identifier shall satisfy the requirements specified below. The format of the main body of the program is defined for each program execution environment identifier value.

| 1byte | Max 65534 byte(=65535-1) |
|---|---|
| Program execution environment identifier value | Main body of the program |

**Fig. 8.32 Download Data Format**

(1) Program execution environment identifier
This is a 1-byte code that indicates the program execution environment in which the main body of the program can be executed. The middleware adapter shall make a comparison with the implemented program execution environment to determine whether it is possible to execute the main body of the downloaded program. In communications based on this version of the ECHONET Specification, the program execution environment identifier value specified in Table 8.13 shall be used. For new execution environments, additional program execution environment identifier values will be defined. The code values to which no execution environment name is assigned are reserved for future use.

**Table 8.13 Program Execution Environment Identifier Code**

| Execution environment name | Program execution environment identifier value |
|---|---|
| Interpreter Method (as defined in Section 8.10) | 0x00 |

#### 8.9.3.4 Program download communication sequence

This section specifies the requirements for the communication sequence for the program download protocol for the ECHONET middleware adapter communication interface.

Fig. 8.33 shows the communication sequence for the program download protocol. If the communication method is “Peer-to-Peer” (i.e. the b0 of the FD(0) of the equipment interface data response command is 1) and the ECHONET-ready equipment is to retain the downloaded data (i.e. the b3 (interface information) of the FD(2) of the equipment interface data response command is 1), the ECHONET middleware adapter shall wait until the transition period (Ttrans) determined by the equipment interface data recognition service elapses and then transmit the program download request command.

Transmissions of program download requests from adapters shall be at the transmission speed that has been detected with the equipment interface data recognition service. If a transmission at the transmission speed specified by the equipment interface data response command is not possible, “Transmission at current speed possible (transmission at specified speed not possible)” shall be sent in an equipment interface data validation notification and program downloading shall be performed at the current speed.

If the software information contained in the program download response command (request sequence number = 0) is different from the currently held software information and the value of the program execution environment identifier of the program download response command (request sequence number = 1) indicates the implemented program execution environment, the adapter shall perform downloading starting with “request sequence number = 2.” If the software information contained in the program download response command (request sequence number = 0) is the same as the currently held software information and the value of the program execution environment identifier of the program download response command (request sequence number = 1) indicates the implemented program execution environment, no downloading shall be performed and a program download completion notification (FD(0) = 0x01 (Success)) shall be transmitted.

The period of time (T1) to wait for a response shall be 300ms.

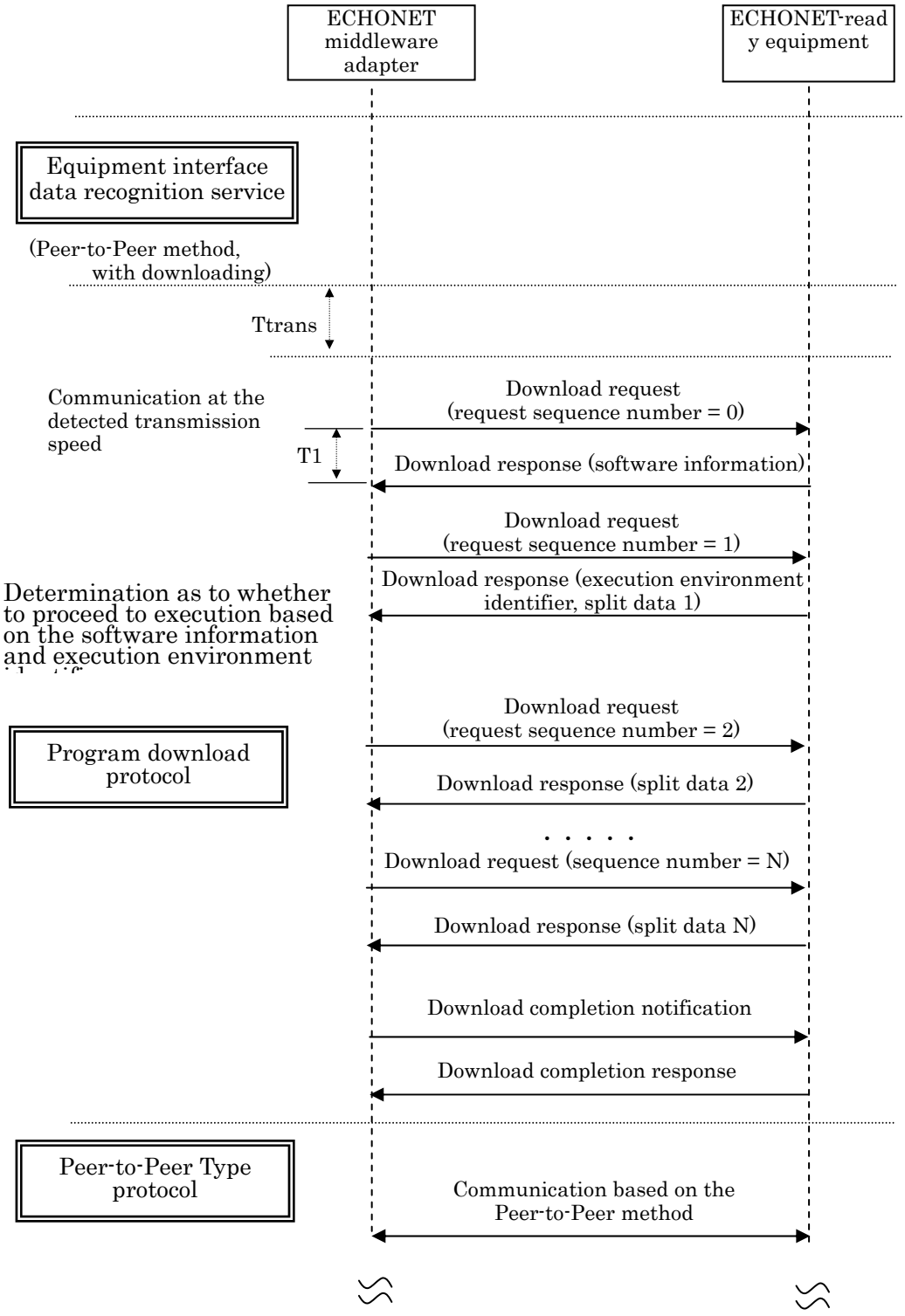



**Fig. 8.33 Program Download Communication Sequence**

Ttrans ：The period of time for the transition to the relevant method. This period must be 500msec. or longer.

Ti ：The period of time to wait for a response. This period must be 300msec. or shorter.

### 8.9.3.5 Handling of errors

This section specifies the requirements for the handling of errors that may occur during the program download communication sequence.

(1) Display section

The middleware adapter shall be equipped with a display section which indicates the errors that have occurred during downloading. This ECHONET Specification does not specify requirements for the method of indicating errors, but, in the case where LED's are used as the means of indicating errors, it is recommended that the requirements specified in Part 7, "8.3.2 Display section" be followed.

(2) Failure of download data reception

If the reception of the download data fails during a program download (expiration of the reception period, FCC error, "Frame number incorrect" error[*1]), a retransmission shall be made. If the reception of the download data fails after a third retransmission, it shall be deemed that a "program download error" has occurred and the display section shall indicate that an error has occurred.

*1: "Frame number incorrect" error: The frame number indicated by the received response command is other than 0 and is different from the frame number indicated by the request command.

(3) The size of the download data exceeds the reception limit ("Download response failure")

If, while receiving a download response, the size of the data contained exceeds the maximum download data size permitted by the adapter, the download data shall be discarded and the display section shall indicate that an error has occurred.

(4) Download data error

If the ECHONET manufacturer code value contained in the program download response command is different from the requested manufacturer code value, or if the program execution environment identifier value of the program download response command (request sequence number = 1) is different from the home execution environment, the download data shall be discarded and the display section shall indicate that an error has occurred.

# 8.10 (Recommended) Specifications for Interpreter Method-based Program Execution Environments for the "Program Download Method" for the "Peer-to-Peer Type"

## 8.10.1 Scope of the Recommended Specifications

These recommended specifications cover the program execution environment for the "Program Download Method" for the "Peer-to-Peer Type," which is implemented in the ECHONET middleware adapter and is based on the "Interpreter Method" whereby the main body of the program contained in the download data is interpreted and executed. These recommended specifications are classified into the following 4 categories:

(1) Program main body format specifications
(2) Specifications for the language of the download program
(3) Interpreter API specifications
(4) Program compression and uncompression specifications

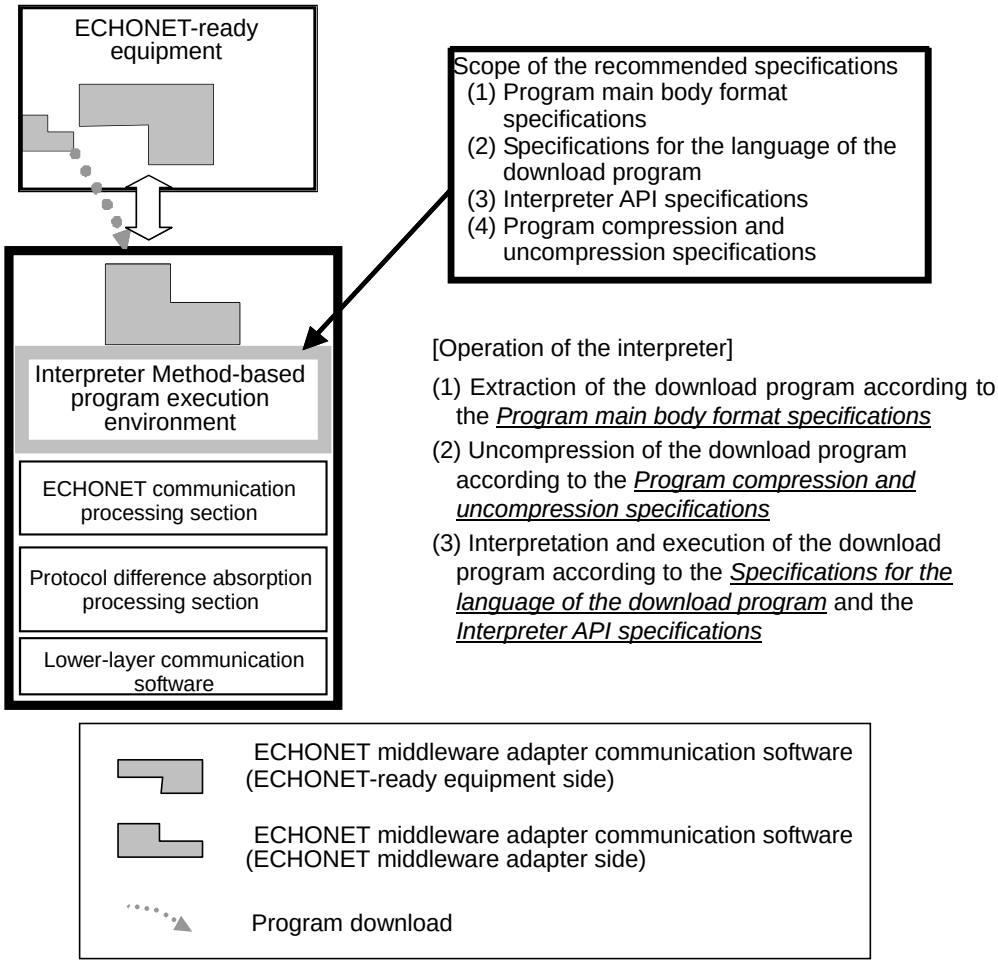



**Fig. 8.34 Scope of the Recommended Specifications**

## 8.10.2 Overview of Interpreter Method-based program execution environments

### 8.10.2.1 Roles of the download program and the interpreter

Fig. 8.35 shows the roles of the download program and the interpreter. The concept of virtual objects called intermediate objects is introduced in the interpreter API so that commands of the communication method defined by the user whose format differs depending on the method can be handled in the same manner.

The download program shall convert commands of the communication method defined by the user (for the ECHONET middleware adapter communication interface) into commands that use intermediate objects and access the interpreter API.

The interpreter shall perform conversions from the intermediate objects to ECHONET objects using the conversion table specified in advance by the download program and access the ECHONET communication processing section API.

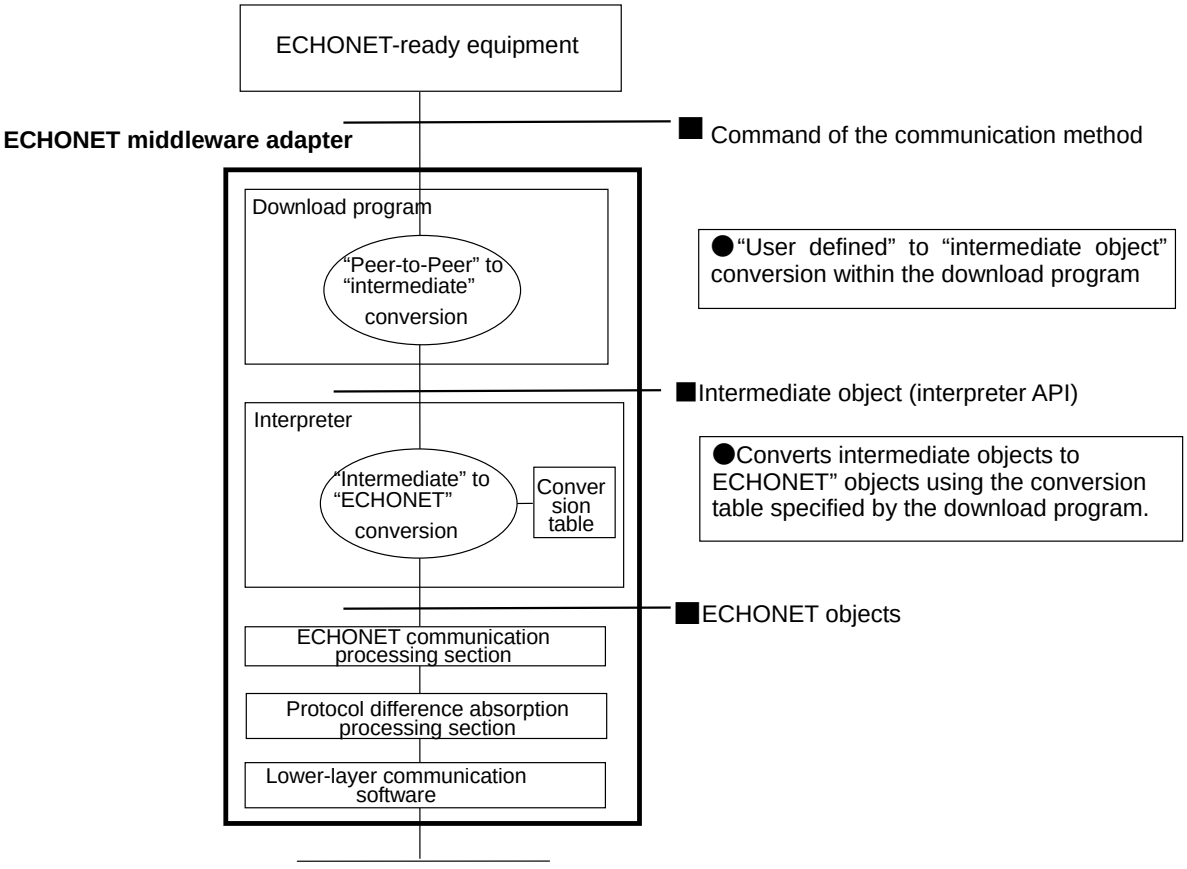



**Fig. 8.35 Roles of the Download Program and the Interpreter**

### 8.10.2.2 Operation procedure

Fig. 8.36 shows the operation procedure for an ECHONET middleware adapter equipped with an Interpreter Method-based program execution environment.

(1) Program download

Downloading of the program from the ECHONET-ready equipment

(2) Generation of ECHONET objects

The interpreter reads the download program, starts the execution and generates ECHONET objects from the download program.

The home node objects and node profile objects shall be generated by the interpreter.

(3) Initialization of the conversion table

The conversion table for performing conversions between intermediate objects and ECHONET objects is initialized using the download program.

(4) ECHONET initialization

The ECHONET initialization API is called from the download program.

(5) Transmission and reception of ECHONET commands and Peer-to-Peer commands

All communications that use the ECHONET shall be performed using the conversion table.

Communications with ECHONET-ready equipment and the interpretation of the data shall be performed directly by the download program.

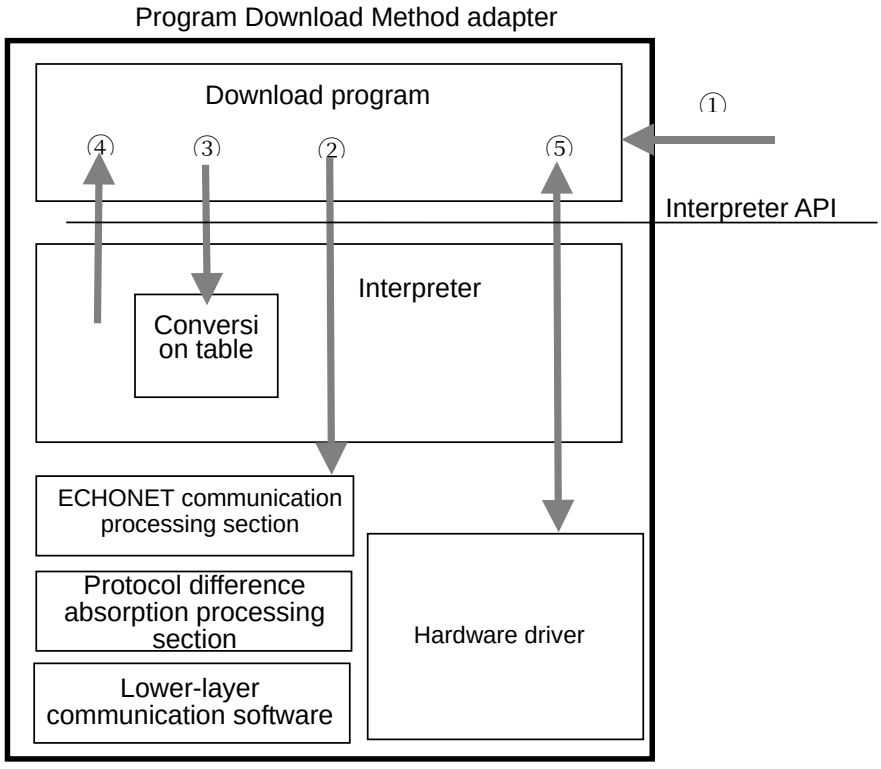



**Fig. 8.36 Operation Procedure**

### 8.10.2.3 Conversion table model

(1) Structure of the conversion table model

Fig. 8.37 shows the model of the conversion table used in this method.

- The ”ECHONET node” has two or more (n) intermediate objects.
- The intermediate objects are identified by ID and have ECHONET class group, class and instance information.
- Each intermediate object has two or more (N) intermediate object properties and two or more (M) ECHONET properties.
- Intermediate object properties and ECHONET properties are associated with each other through the conversion table.
- There are 3 types of associations between properties; the identical value type, mapping type and function type. A suitable conversion table is generated according to the type.
- An identical value type conversion table indicates one-to-one associations between property codes. A mapping type conversion table indicates N-to-M associations between property codes and IDT/EDT values. A function type conversion table indicates N-to-M associations between property codes.
- Each adapter has table sets, each consisting of the above-mentioned elements, in a number equal to the number of nodes.

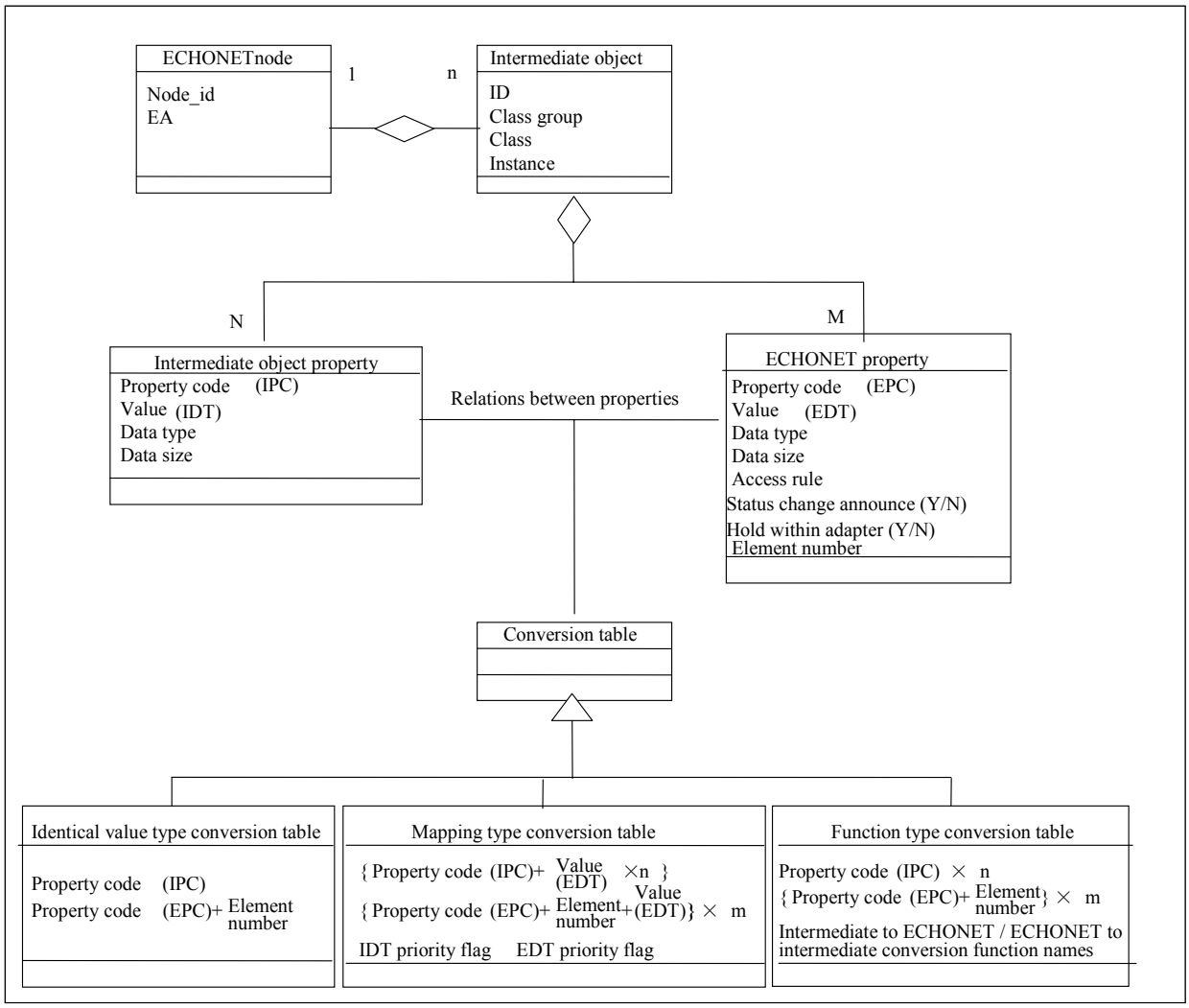



**Fig. 8.37 Overview of the Conversion Table Model**

(2) ECHONET node

◇ Overview

Definition information for the ECHONET nodes possessed by the adapter.

◇ Record format

| Node ID | EA (ECHONET address) | |
|---|---|---|
| | NetID | NodeId |

* "Node ID" is different from the NodeID used in the ECHONET. "Node ID" is an identifier that uniquely identifies the node within the adapter.

◇ Example of use

When the EA of the home node is 0x0001 and the EA of the controller node is 0x0002:

| Node ID | EA (ECHONET address) | |
|---|---|---|
| | NetID | NodeId |

| | | |
|---|---|---|
| 0 | 0x00 | 0x01 |
| 1 | 0x00 | 0x02 |

* The home node is registered with Node ID set to 0.

(3) Intermediate object model

◇ Overview

Definition information for the intermediate nodes possessed by the adapter.

◇ Record format

| Intermediate object ID | ECHONET | | |
|---|---|---|---|
| | Class group | Class | Instance |

◇ Example of use

When a home air conditioner, a temperature sensor and a humidity sensor are possessed:

| Intermediate object ID | ECHONET | | |
|---|---|---|---|
| | Class group | Class | Instance |
| 1 | 0x01 | 0x30 | 0x01 |
| 2 | 0x00 | 0x11 | 0x01 |
| 3 | 0x00 | 0x12 | 0x01 |

(4) Intermediate object property model

◇ Overview

Definition information for the properties possessed by an intermediate object.

◇ Record format

| Intermediate object ID | Property | | |
|---|---|---|---|
| | Property code (IPC) | Type | Size |

◇ Example of use

When an intermediate object (ID = 1) possesses

・Two unsigned char types with the code (IPC) set at 0x01 and

・One signed long type with the code (IPC) set at 0x02:

| Intermediate object ID | Property | | |
|---|---|---|---|
| | Property code (IPC) | Type | Size |
| 1 | 0x01 | Unsigned char | 2 |
| | 0x02 | Signed long | 4 |

(5) ECHONET property model

◇ Overview

Definition information for the ECHONET properties possessed by an intermediate object.

◇ Record format

| Intermediate object ID | Property | | | | | |
|---|---|---|---|---|---|---|
| | Property code (EPC) | Element number | Type | Access rule | Status change announcement | Size |

◇ Example of use

When an intermediate object (ID = 1) possesses

・One unsigned char type with the code (EPC) set at 0xB0,

・One unsigned char type with the code (EPC) set at 0xC9 as "Element 0" and one unsigned char type with the code (EPC) set at 0xC9 as "Element 1":

| Intermediate object ID | Property | | | | | | |
|---|---|---|---|---|---|---|---|
| | Property code (EPC) | Element number | Type | Access rule | | Status change announcement | Size |
| 1 | 0xB0 | - | Unsigned char | set | get | without | 1 |
| | 0xC9 | 0 | Unsigned char | set | get | without | 1 |
| | 0xC9 | 1 | Unsigned char | set | get | without | 1 |

(6) Identical value type conversion table model

◇ Overview

Conversion information for identical value type properties for which no conversion is needed.

◇ Record format

| Intermediate object ID | IPC | EPC | Element number |
|---|---|---|---|

◇ Example of use

When identical value conversions are to be performed

・between IPC = 0x10 of an intermediate object (ID = 1) and EPC = 0x20 and

・between IPC = 0x11 of the intermediate object (ID = 1) and EPC = 0x30 (Element No.3):

| Intermediate object ID | IPC | EPC | Element number |
|---|---|---|---|
| 1 | 0x10 | 0x20 | - |
| | 0x11 | 0x30 | 3 |

(7) Mapping type conversion table model

◇ Overview

Conversion information for mapping type properties for which a mapping table-based value conversion is to be performed.

◇ Record format

| Intermediate object ID | MapID | IPC0･･･IPCn | IFLG | EPC0:ELE0･･･ EPCm:ELEm | EFLG |
|---|---|---|---|---|---|

・MapID: Mapping type conversion table number (which is used during the table generation process)

・IPC0…IPCn: n intermediate object property code (IPC) values.

・IFLG: Priority flag for the case where the mapping cannot be uniquely determined during an IPC to EPC conversion.

・EPC0: ELE0…EPCm: ELEm: m pairs of ECHONET property code value and element number. Element numbers are specified in the case of an array.

・EFLG: Priority flag for the case where the mapping cannot be uniquely determined during an EPC to IPC conversion.

◇ Example of use

Mapping between IPC = 0x22 {1, 2, 3, 4, 5, 6, 7, 8} and EPC = 0xAA {A, A, A, B, B, B, C, C}

The IDT value is 1, 5 and 8 when the EDT value is A, B and C, respectively.

(Priority is given to EFLG=1 mapping relationships.)

| Intermediate object ID | MapID | IPC=0x22 | IFLG | EPC=0xAA | EFLG |
|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | A | 1 |
| | | 2 | 1 | A | 0 |
| | | 3 | 1 | A | 0 |
| | | 4 | 1 | B | 0 |
| | | 5 | 1 | B | 1 |
| | | 6 | 1 | B | 0 |
| | | 7 | 1 | C | 0 |
| | | 8 | 1 | C | 1 |

(8) Function type conversion table model

◇ Overview

Conversion information for function type properties for which a function table-based value conversion is to be performed.

◇ Record format

| Intermediate object ID | IPC0･･･IPCn | EPC0:ELE0･･･ EPCm:ELEm | I2E_FUNC | E2I_FUNC |
|---|---|---|---|---|

・IPC0…IPCn: n intermediate object property code (IPC) values

・EPC0: ELE0…EPCm: ELEm: m pairs of ECHONET property code value and element number. Element numbers are specified in the case of an array.

・I2E_FUNC: The name of the function for IPC to EPC conversions which is contained

in the download program.

・E2I_FUNC: The name of the function for EPC to IPC conversions which is contained in the download program.

◇ Example of use

Conversion between
EPC = 0xB3 {10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 1A, 1B, 1C, 1D, 1E, 1F} and
IPC = 0x11 {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F}
using
FUNC1(X) {return X-0x10;} and
FUNC2(X) {return X＋0x10;}

| Intermediate object ID | IPC0･･･IPCn | EPC0:ELE0･･･ EPCm:ELEm | I2E_FUNC | E2I_FUNC |
|---|---|---|---|---|
| 1 | 0x11 | 0xB3 | FUNC2 | FUNC1 |

## 8.10.3 Program format specifications

Fig. 8.38 (the hatched part) shows the format of the main body of the program in an Interpreter Method-based program execution environment.

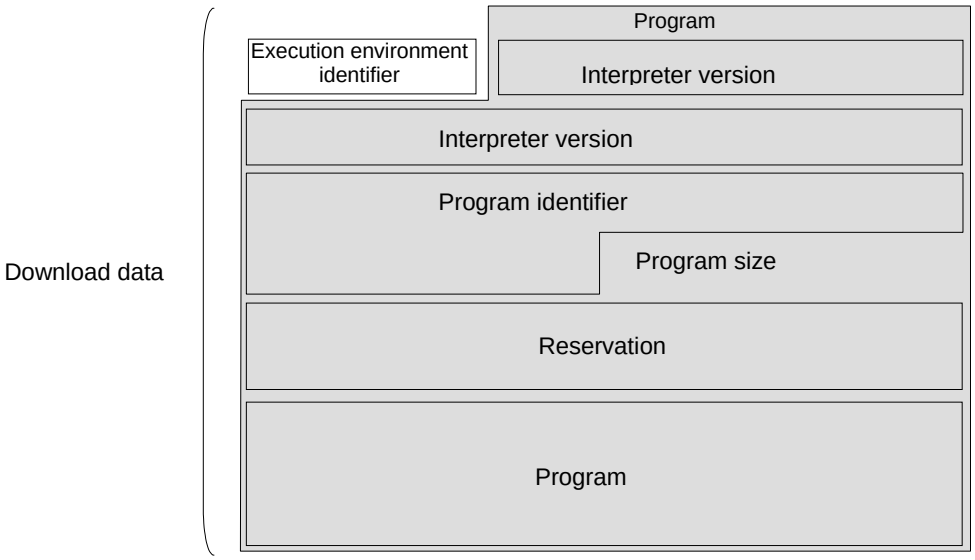



**Fig. 8.38 Format of the Main Body of the Program**

(1) Interpreter version (3 bytes)

First byte : 0x00 (fixed)

Second byte : Major version of the specifications (binary value). In communications based on this version of the ECHONET Specification, the value “0x01” shall be used.

Third byte : Minor version of the specifications (binary value). In communications based on this version of the ECHONET Specification, the value “0x00” shall be used.

(2) Manufacturer code (4 byte)

First byte: 0x00 (fixed)

Second to fourth bytes: Code indicating the manufacturer of the ECHONET-ready equipment (The code value is specified by the ECHONET Consortium.)

(3) Program identifier (6 bytes).

Same as the program identifier for the program download request and response commands.

(4) Program size (2 bytes)

1 byte or 2 bytes: Program size (in bytes)。

(5) Reservation area (20 bytes)

(6) Program

The program as generated according to the “Specifications for the language of the download program” and the “Interpreter API specifications” and then converted according to the “Program compression and uncompression specifications.”

## 8.10.4 Specifications for the language of the download program

### 8.10.4.1 Overview

The download program shall use a Reversed Polish Notation-based stack language.
The API definition shall be as shown in Fig. 8.39.

```
FUNC1(arg1 arg2 arg3 <name> --ret1 ret2, comment)
```

**Fig. 8.39 API Definition**

This is called a stack diagram. “FUNC1” is the API name. The part to the left of “--” indicates the input (arguments) and the part to the right of “--” indicates the output (return values). The part that follows “,” indicates comments. If FUNC1 is executed after stacking arg2 on arg1 and then arg3 on arg2, ret2 will be stacked on ret1 after the execution. “<name>” means that the next stack stacked after the execution of FUNC1 is used. When there is no description to the left and right of “--,” there is no input or output.

The program to execute FUNC1 after stacking arg2 on arg1 and then arg3 on arg2 shall be as shown in Fig. 8.40.

```
arg1 arg2 arg3 FUNC1 name
```

**Fig. 8.40 API Program Description Method**

Fig. 8.41 shows the stack before the execution of FUNC1 API and the stack after the execution.

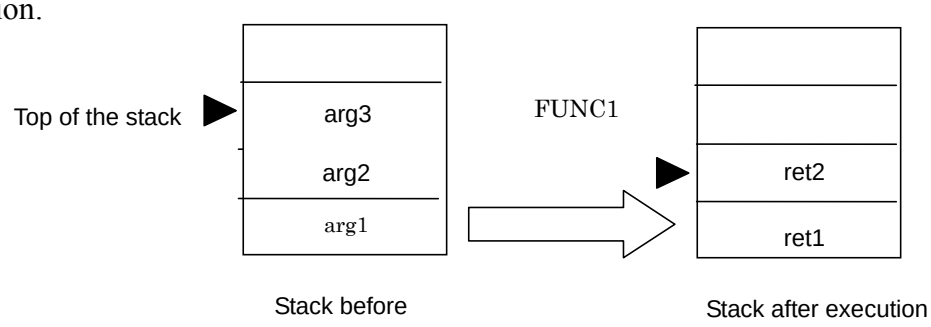



**Fig. 8.41 Stacks before and after Execution of API**

### 8.10.4.2 Characters

(1) Character code
The ASCII code (0x20 to 0x7E) shall be used. Uppercase characters and lowercase characters shall not be discriminated from each other.

(2) Space
“Space” (0x20).

(3) Lexical conversion

The download program shall consist of tokens partitioned by spaces.

(4) Token

A token shall consist of "data" (8.10.4.3), "user-defined names" (8.10.4.4), "interpreter basic API" (8.10.5) and "interpreter ECHONET API" (8.10.6).

### 8.10.4.3 Data

The data shall consist of 16-bit values.

In the case where 32-bit values are used, they shall be represented in "8 bits×4" byte arrays and the data shall be arranged in the big endian order.

### 8.10.4.4 User-defined names

User-defined names defined by "Variable definition" (8.10.5.1), "Array definition" (8.10.5.2), "Constant definition" (8.10.5.3) and "Function definition" (8.10.5.4) described in "8.10.5 Interpreter Basic API specifications."

These names shall not be so defined that an overlap occurs (interpreter basic API and interpreter ECHONET API).

## 8.10.5 Interpreter Basic API specifications

### 8.10.5.1 Variable definition (VARIABLE)

(1) Function: Defines 16-bit variable names.

(2) Stack diagram VARIABLE (<name>--, defines the variable)

(3) Example description

Adapter's description

```
VARIABLE n
```

Description in the C language

```
int n;
```

### 8.10.5.2 Array (byte array) definition (CREATE – ALLOT)

(1) Function: Defines a byte array name.

(2) Stack diagram CREATE (<name>--, defines the array name)
ALLOT (n --, allots n bytes)

(3) Example description

Adapter's description

```
CREATE buf 10 ALLOT
```

Description in the C language

```
Char buf[10];
```

### 8.10.5.3 Constant definition (CONSTANT)

(1) Function: Defines the constant name.

(2) Stack diagram CONSTANT (n <name>--, defines the constant name).

(3) Example description

Adapter's description

```
128 CONSTANT MAX_CHARS
```

Description in the C language

```
#define MAX_CHARS 128;
```

#### 8.10.5.4 Function definition (: 〜 ;)

(1) Function: Defines the function name.

(2) Stack diagram : (<name>-- , starts the function definition)

; ( -- , ends the function definition)

(3) Example description

Adapter's description

```
:calc(q --ret)    \q indicates the input (argument), and ret indicates the output (return value).
```

Description in the C language

```
Int calc(int q)
{
    n=q+128;
    return n;
}
```

#### 8.10.5.5 Assigning values to variables (!)

(1) Function: Assigns a 16-bit value to the variable.

(2) Stack diagram ! (x addr -- , assigns x to addr).

(3) Example description

Adapter's description

```
34 n !
```

Description in the C language

```
n = 34;
```

#### 8.10.5.6 Retrieving values from variables (@)

(1) Function: Retrieves the 16-bit value from the variable and stores it at the top of the stack.

(2) Stack diagram @ (addr -- x, retrieves the value from addr).

(3) Example description

Adapter's description

```
n@
```

Description in the C language

```
n;
```

#### 8.10.5.7 Assigning 1-byte data to byte arrays (C!)

(1) Function: Assigns 1-byte data to a byte array.

(2) Stack diagram C! (byte addr -- , assigns byte to addr).

(3) Example description

Adapter's description

```
34 buf 3 + C!
```

Description in the C language

```
Buf[3] = 34;
```

#### 8.10.5.8 Retrieving 1-byte data (C@)

(1) Function: Retrieves 1-byte data and stores it at the top of the stack.

(2) Stack diagram C@ (addr -- byte, retrieves the value from addr).

(3) Example description

Adapter's description

```
buf3 + C@
```

Description in the C language

```
Buf[3];
```

#### 8.10.5.9 Four arithmetic operations (+ − * / MOD)

(1) Function: Performs addition/subtraction/multiplication/division/remainder calculation and stores the result at top of the stack.

(2) Stack diagram ＋ (n1 n2 -- add , calculates n1＋n2)
－ (n1 n2 -- sub , calculates n1－n2)
* (n1 n2 -- mul , calculates n1 * n2)
/ (n1 n2 -- div , calculates n1 / n2)
MOD (n1 n2 -- rem , calculates n1 % n2)

(3) Example description

Adapter's description

```
1 1 +
1 1 –
1 1 *
1 1 /
1 1 MOD
```

Description in the C language

```
1 +1;
```

```
1 –1;
1 *1;
1 /1;
1 % 1;
```

### 8.10.5.10 Bit shifting (LSHIFT RSHIFT)

(1) Function: Performs left/right shifting and stores the result at the top of the stack.

(2) Stack diagram LSHIFT (n1 s -- n2 , shifts n1 to the left by s bits)
RSHIFT (n1 s -- n2 , shifts n1 to the right by s bits)

(3) Example description

Adapter's description

```
n @ 4 LSHIFT    ¥ left shifting of n
n @ 4 RSHIFT    ¥ logical right shifting of n (If n is FFFF, the calculation result
                will be 0FFF.)
```

Description in the C language

```
n « 4
n » 4
```

* Logical right shifting: If signed, 0s are entered from the left.

### 8.10.5.11 Comparative operators (= > < >= <= <>)

(1) Function: Compares two values and stores TRUE or FALSE at the top of the stack if the result is true or false, respectively.

(2) Stack diagram = (n1 n2 -- flg , if n1 = n2, flg = TRUE)
> (n1 n2 -- flg , if n1 > n2, flg = TRUE)
< (n1 n2 -- flg , if n1 < n2, flg = TRUE)
>= (n1 n2 -- flg , if n1 >= n2, flg = TRUE)
<= (n1 n2 -- flg , if n1 <= n2, flg = TRUE)
<> (n1 n2 -- flg , if n1 <> n2, flg = TRUE)

(3) Example description

Adapter's description

```
2 2 =
2 2 >
2 2 >
2 2 >=
2 2 <=
2 2 <>
```

Description in the C language

```
2  == 2
2  > 2
2  > 2
2 >= 2
2 <= 2
2 <> 2
```

(4) Notes

Comparisons of 2 values shall be comparisons of signed data.
Comparisons of unsigned data shall be as follows:

【Program】

("a" and "b" shall be compared as unsigned. TRUE shall be returned if "a" is larger and FALSE shall be returned if "a" and "b" are equal or "b" is larger.)

```
a @ 0 < b @ 0 < AND a @ 0 >= b @ 0 >= AND OR @ TRUE = IF
a @ b @ >
ELSE
b @ a @ >
THEN
```

【Explanation】

In the case of a comparison of 2 unsigned values, "first bit = 1" (negative if signed) is larger than "first bit = 0" (positive if signed). This means that the "larger or smaller" relationship will be reversed between the "signed" and "unsigned" cases in the case of a comparison of values with different signs. In the case of a comparison of values with the same sign, the "larger or smaller" relationship will be the same between the "signed" and "unsigned" cases and the program described above will apply.

### 8.10.5.12 True/false value (TRUE FALSE)

(1) Function: Stores the true/false value at the top of the stack.

(2) Stack diagram TRUE ( -- －1 , TRUE value, －1)

FALSE ( -- 0 , TRUE value 0)

(3) Example description

Adapter's description

```
2 TRUE =
2 FALSE =
```

Description in the C language

```
2 == true
2 == true
```

### 8.10.5.13 Logical operators (AND OR NOT XOR)

(1) Function: Performs a logical operation and stores the result at the top of the stack.

(2) Stack diagram AND (n1 n2 -- n3, bit-by-bit logical product)

OR (n1 n2 -- n3, bit-by-bit logical sum)

NOT (n1 -- n2, all-bit inversion)

XOR (n1 n2 -- n3, bit-by-bit exclusive disjunction)

(3) Example description

Adapter's description

```
01AND
01OR
n@NOT
01XOR
```

Description in the C language

```
0&1
0|1
~n
0^1
```

### 8.10.5.14 Conditional statements (IF – ELSE – THEN)

(1) Function: Switches to the appropriate processing according to the condition.

(2) Stack diagram IF (flg -- , executes the succeeding code if flg is TRUE)
ELSE ( -- , executes the succeeding code if flg is FALSE)
THEN ( -- , finishes IF..ELSE..)

(3) Example description

Adapter's description(A)

```
n @ 128 = IF
      256 q !
ELSE
      64 q !
THEN
```

Description in the C language(A)

```
If(n==128) {
    q = 256;
} else{
    q = 64;
}
```

Adapter's description(B)

```
: SAMPLE_IF ( X -- )
      DUP 1 = IF 11 SWAP THEN
      DUP 2 = IF 12 SWAP THEN
      DUP 3 = IF 13 SWAP THEN
      DROP
;
```

Description in the C language(B)

```
Int sample_if() {
      If (x == 1) return 11;
      If (x == 2) return 12;
      If (x == 3) return 13;
}
```

### 8.10.5.15 Loop (BEGIN ー WHILE ー REPEAT)

(1) Function: Repeats the processing

(2) Stack diagram BEGIN ( -- , starts the loop)
WHILE (flg -- , continues executing the loop as long as flg is TRUE)
REPEAT ( -- , ends the loop)

(3) Example description

Adapter's description

```
VARIABLE q
VARIABLE n
0 q !
10 n !
BEGIN
n @ 0 > WHILE
   q @ n @ + q !
   n @ 1 –
   n !
REPEAT
```

Description in the C language

```
int q;
int n;
q=0;
n=10;
while( n>0 ){
   q = q + n;
   n--;
}
```

### 8.10.5.16 Conversion of radix (HEX, DECIMAL, BINARY)

(1) Function: Specifies the radix

(2) Stack diagram HEX ( -- , treats the succeeding data as hexadecimal data)
DECIMAL ( -- , treats the succeeding data as decimal data
BINARY ( -- , treats the succeeding data as binary data)

(3) Example description

Adapter's description

```
HEX       ¥ to hexadecimal
DECIMAL   ¥ to decimal
BINARY    ¥ to binary
```

(4) Notes

The default is DECIMAL (radix 10).

It is possible to provide a description at any desired place in the program.

### 8.10.5.17 Manipulating the stack (DUP PICK DROP SWAP ROLL)

(1) Function: Manipulates the stack.

(2) Stack diagram DUP (n -- n n , duplicates the top layer of the stack)

PICK (xn … x1 x0 n -- xn … x1 x0 xn, copies the nth layer of the stack)

DROP (n -- , deletes the top layer of the stack)

SWAP (n1 n2 -- n2 n1 , replaces the 2 top layers)

ROLL (xn … x1 x0 n -- xn-1 … x1 x0 xn, rotates the nth layer of the stack)

(3) Example description

Adapter's description

```
2 DUP          ¥2        →2 2
3 2 1 2 PICK   ¥3 2 1    →3 2 1 3
2 DROP         ¥2        →
2 1 SWAP       ¥2 1      →1 2
3 2 1 2 ROLL   ¥3 2 1    →2 1 3
```

### 8.10.5.18 Comment (\ (…))

(1) Function: Writes comments in the source code. These comments are not interpreted by the interpreter.

(2) Stack diagram ¥ ( -- , treats the succeeding data as a comment)

( ( -- , starts the comment)

) ( -- , ends the comment)

(3) Example description

Adapter's description

```
¥ this line is a comment line
2 3 ( this is also a comment ) ＋
```

Description in the C language

```
// this line is a comment line
2 ＋ /* this also a comment */ 3
```

### 8.10.5.19 LONG type data calculations (OP_LONG)

(1) Function: Performs long type data (4-byte data) calculations.

(2) Stack diagram OP_LONG ( addr1 [addr2 | n] addr3 type -- result )

(3) Explanation addr1 : "Address 1" to store the calculation target data.

addr2 : "Address 2" to store the calculation target data (This is not necessary in the case of NOT).

n : Number of bits by which left-shifting (LSHIFT) or right-shifting (RSHIFT) is performed.

addr3 : Address to store the calculation result data.

type : Type of calculation

result : Address to store the calculation result data (same as "addr3").

(4) Notes

"addr1," "addr2," "addr3" and "result" store long-type values in the first 4 bytes in the big endian order.

TRUE is represented by 0xffffffff and FALSE is represented by 0x00000000.

### 8.10.5.20 Converting LONG values into TRUE/FALSE values (CONV_ARRD_TO_TF)

(1) Function: Converts 4-byte data into a 2-byte TRUE or FALSE value.

(2) Stack diagram CONV_ADOR_TO_TF ( addr -- val )

(3) Explanation addr : Address at which 4-byte data is stored.

val : 0 (FALSE: when all of the first 4 bytes of "addr" are 0)

: －1 (TRUE: when any of the first 4 bytes of "addr" is not 0)

## 8.10.6 Interpreter ECHONET API specifications

### 8.10.6.1 INT_ECHONET

(1) Function: Initializes the ECHONET communication processing section.

(2) Stack diagram INIT_ECHONET ( mode nretry -- result )

(3) Explanation mode : Start mode
0 (Warm start)
1 (Cold start (1))
2 (Cold start (2))
3 (Cold start (3))
result : Result.
0 (Successful completion)
-1 (Abnormal completion)

(4) Notes

- Performs processing that corresponds to the initialization of the ECHONET communication processing section (MidStart, MidReset, MidInit, MidInitAll) and the starting of operation (MidRequestRun).
- This API is called only once after the ECHONET object initialization and conversion table initialization processes.

### 8.10.6.2 SET_COM_PARAM

(1) Function: Sets an ECHONET middleware communication interface parameter. This API specifies the length of time [ms] to wait before making a retry after an error.

(2) Stack diagram :SET_COM_PARAM ( r_time -- , sets a communication parameter)

(3) Explanation r_time : Length of time [ms] to wait before making a retry after an error (default value = 1000ms).

(4) Notes

#### 8.10.6.3 SET_UART_RV_MODE

(1) Function: Sets the criterion to determine the end of the process to receive a message from ECHONET-ready equipment.

(2) Stack diagram

SET_UART_RV_MODE ( addr) (val | len) mode -- )

(3) Explanation (addr) : Array address of the completion code for the case where "mode" = 2

This may be omitted when "mode" = 1.

Val : Length of time (in milliseconds) to wait when "mode" = 1 to determine that the reception process has been completed

Len : Length of completion code for the case where "mode" = 2

Mode : Determination mode

1 (time-based determination)

2 (completion code-based determination)

(4) Notes
- (addr) is not necessary when "mode" = 1
- (val | len) shall be set to val when "mode" = 1 and to len when "mode" = 2.

#### 8.10.6.4 CREATE_MNG_TABLES

(1) Function: Creates management tables to store object information, property information and property-related information.

(2) Stack diagram CREATE_MNG_TABLES ( n_obj n_ipc n_epc n_epcm n_irel n_mrel n_frel -- )

(3) Explanation
n_obj : Number of home appliance device objects

n_ipc : Number of home appliance device side definition properties

n_epc : Number of non-array type ECHONET properties

n_epcm : Number of array type ECHONET properties

n_irel : Number of identical value type property relationships

n_mrel : Number of mapping type property relationships

n_frel : Number of function type property relationships

(4) Notes
- This API is called only once before executing the API (RGST_XXX) to register object information, property information and property-related information.

#### 8.10.6.5 RGST_NODE

(1) Function: Registers other nodes.

(2) Stack diagram RGST_NODE ( node_id EA -- )

(3) Explanation
node_id : Node ID
EA : ECHONET address

(4) Notes
- The argument node_id is an identifier to uniquely identify nodes within the download program. This is different from the ECHONET's NodeID
- The home node is automatically generated at the time of interpreter startup with node_id set to 0.

#### 8.10.6.6 RGST_OBJ

(1) Function: Registers intermediate objects.

(2) Stack diagram RGST_OBJ ( obj_id node_id obj_class_group obj_class instance -- )

(3) Explanation
obj_id : Intermediate object ID
Node_id : ECHONET node ID
obj_class_group : ECHONET object class group
obj_class : ECHONET object class
instance : ECHONET object instance:

(4) Notes
- When registering an object with the home node, node_id shall be set to 0.
- The node profile object is automatically generated in the home node (node_id = 0) at the time of interpreter startup with obj_id set to 0
- When this API is called, the following properties of the node profile object shall be set:
  - Home node instance list S
  - Home node class list S
  - Number of instances of the home node
  - Number of classes of the home node
  - Home node instance list
  - Home node class list

### 8.10.6.7 RGST_EPC

(1) Function: Registers non-array type ECHONET properties.

(2) Stack diagram RGST_EPC ( obj_id epc type rule anno keep_edt size -- )

(3) Explanation

epc : ECHONET property code (EPC)

type : Property type

0 (signed char)
1 (signed short)
2 (signed long)
3 (unsigned char)
4 (unsigned short)
5 (unsigned long)
6 (no data type）

rule : Processable access rules (The following shall be specified using OR.)

0x0001 (Set)
0x0002 (Get)
0x0004 (Anno）

anno : Whether or not to make status change announcements

1 (With announcements)
0 (Without announcements)

keep_edt : Property value retention flag

1 (Retain the property values)
0 (Do not retain the property values)

size : Data area size (in bytes)

(4) Notes

・The compulsory properties and error description property (0x89) of the node profile object (obj_id = 0) are automatically generated at the time of interpreter startup.

・The property maps for the target objects are also set.

・Relationship between “rule” and ESV:

Set: SetI (0x60), SetC (0x61)
Get: Get (0x62)
Anno: INF_REQ (0X63)

・The argument keep_edt has been set for “service that does not retain values in the adapter” of the “object generation method.” For properties for which keep_edt is set to “Do not retain the property values,” all requests go up with CHK_RV_IPC because there is no “middleware return.” For properties for which

keep_edt is set to “Retain the property values,” only Set requests go up with CHK_RV_IPC.

### 8.10.6.8 RGST_EPCM

(1) Function: Registers array type ECHONET properties.

(2) Stack diagram RGST_EPCM ( obj_id epc type rule anno keep_edt size member_no -- )

(3) Explanation

obj_id : Intermediate object ID

epc : ECHONET property code (EPC)

type : Property type

0 (signed char)

1 (signed short)

2 (signed long)

3 (unsigned char)

4 (unsigned short)

5 (unsigned long)

6 (no data type)

rule : Processable access rules (The following shall be specified using OR.)

0x0001 (Element designation Set)

0x0002 (Element designation Get)

0x0010 (Element designation presence confirmation request)

0x0040 (Element designation notification request)

anno : Whether or not to make status change announcements

1 (With announcements)

0 (Without announcements)

keep_edt : Property value retention flag

1 (Retain the property values)

0 (Do not retain the property values)

size : Data area size (in bytes)

member_no : Element number to register

(4) Notes

・The property maps for the target objects are also set.

・Relationship between “rule” and ESV:

Element designation Set: SetMI (0x64), SetMC (0x65)

Element designation Get: GetM (0x66

Element designation presence confirmation request: CheckM (0x6c)

Element designation notification request: INFM_REQ (0x67)

#### 8.10.6.9 ADD_EPC_MEMBER

(1) Function: Adds an array element, with element number specified, to an array type ECHONET property registered using RGST_EPCM.

(2) Stack diagram ADD_EPC_MEMBER ( obj_id epc member_no -- )

(3) Explanation
- obj_id : Intermediate object ID
- epc : ECHONET property code (EPC)
- member_no : Element number to add

(4) Notes

#### 8.10.6.10 RGST_IPC

(1) Function: Registers intermediate object properties.

(2) Stack diagram RGST_IPC ( obj_id ipc type size -- )

(3) Explanation
- obj_id : Intermediate object ID
- ipc : intermediate object property code (IPC)
- type : Property type

0 (signed char)
1 (signed short)
2 (signed long)
3 (unsigned char)
4 (unsigned short)
5 (unsigned long)
6 (no data type)

- size : Data area size (in bytes)

(4) Notes

### 8.10.6.11 RGST_INDENTICAL_PROP

(1) Function: Identical value type conversion table registration

(2) Stack diagram RGST_IDENTICAL_PROP ( ipc (ele) epc obj_id -- )

(3) Explanation
- ipc : : intermediate object property code (IPC)
- (ele) : ECHONET property element number
  This is specified only when the ECHONET property is an array type property.
- epc : ECHONET property code (EPC)
- obj_id : Intermediate object ID

(4) Notes

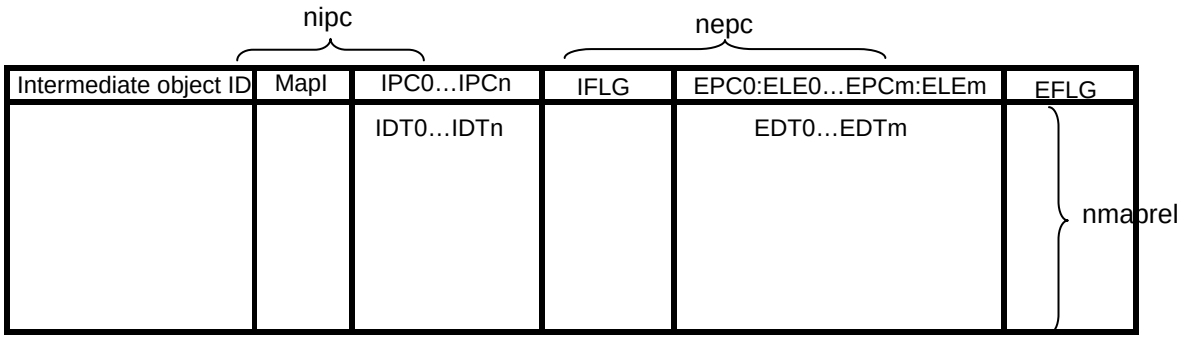
### 8.10.6.12 RGST_MAP_PROP_REL

(1) Function: Creates mapping type conversion tables.

(2) Stack diagram RGST_ MAP_PROP_REL ( map_id ipc0...ipcn (ele0) epc0...(elem) epcm nmparel nipc nepc obj_id -- )

(3) Explanation
- map_id : Mapping relationship ID to create.
- ipc0...ipcn : n intermediate object property code values.
- (ele0)epc0...(elem)epcm
  : m pairs of ECHONET property code value and element number.
  Element number is specified only when the ECHONET property is an array type property.
- nmparel : Number of mapping records to create
- nipc : Number of intermediate object properties
- nepc : Number of ECHONET properties
- obj_id : Intermediate object ID

(4) Notes
- This function is only capable of creating mapping type conversion tables.
- For the generation of table data, RGST_MAP_PROP_VAL and RGST_MAP_PROP_VAL_PR are used.



### 8.10.6.13 RGST_MAP_PROP_VAL

(1) Function: Registers property associations in mapping type conversion tables.

(2) Stack diagram RGST_ MAP_PROP_VAL ((val0...vall) idt0...idtn edt0...edtm map_id -- )

(3) Explanation

(val0...vall) : Property value for an escape
The value at the time when idt/edt is "escaped." "val = 0x0000" represents "NO CARE." "val = 0xFFFF" indicates that the idt and edt values are 0xFFFF.
The order of arrangement of (val0…val1) corresponds to the order of "idt0…idtn" and "edt0…edtm" whose value is 0xFFFF.
This may be omitted when no escape code is used.

idt0...idtn : When the n intermediate object property values are 0xFFFF or when the address of the array that stores the property values is 0xFFFF, it shall be regarded as an escape code and the corresponding escape property value shall be referenced.

edt0...edtm : When the m ECHONET property code values are 0xFFFF or when the address of the array that stores the properties is 0xFFFF, it shall be regarded as an escape code and the corresponding escape property value shall be referenced.

Map_id : ID of the mapping type conversion table to register.
The ID generated with RGST_MAP_PROP_REL shall be specified.

(4) Notes

・How to use "NO CARE"
In the case of the example shown in the table below, IPC0 is 3 if EPC0 is 2, regardless of the EPC1 value (NOCARE). In other cases, the IPC0 value is associated using the ECP0 and EPC1 values.

| Intermediate object ID | MapID | IPC0 | IFLG | EPC0 | EPC1(val) | EFLG |
|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | | 2 | 1 | 1 | 2 | 1 |
| | | 3 | 1 | 2 | 0xFFFF(0x0000) | 1 |
| | | 4 | 1 | 3 | 1 | 1 |
| | | 5 | 1 | 3 | 2 | 1 |

If "val" is 0xFFFF, EPC0 = 2 and EPC1 = 0xFFFF is converted to IPC0 = 3.

| Intermediate object ID | MapID | IPC0 | IFLG | EPC0 | EPC1(val) | EFLG |
|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | | 2 | 1 | 1 | 2 | 1 |
| | | 3 | 1 | 2 | 0xFFFF(0xFFFF) | 1 |
| | | 4 | 1 | 3 | 1 | 1 |
| | | 5 | 1 | 3 | 2 | 1 |

・If the conversion condition matches 2 or more lines, priority shall be given to the one that appeared first.

・Registration of unsigned char type (1-byte) IPC and unsigned long type (4-byte) EPC (mapping relationship):

```
HEX
CREATE LONG_EPC 4 ALLOT
12 LONG_EPC 0 + C!
34 LONG_EPC 1 + C!
56 LONG_EPC 2 + C!
78 LONG EPC 3 + C!
0 1 5 4 RGST_EPC    \ unsigned long, 4 bytes
.....
41 LONG_EPC 1 RGST_MAP_PROP_VAL
.....
```

### 8.10.6.14 RGST_MAP_PROP_VAL_PR

(1) Function: Assigns, when a unique association cannot be determined, a flag indicating the association to which priority is given, and registers the property association in the mapping conversion table.

(2) Stack diagram RGST_MAP_PROP_VAL_PR ( (val0...vall) i_pflg e_pflg idt0...idtn edt0...edtm map_id -- )

(3) Explanation

(val0...vall) : Property value for an escape
The value at the time when idt/edt is "escaped." "val = 0x0000" represents "NO CARE." "val = 0xFFFF" indicates that the idt and edt values are 0xFFFF.
The order of arrangement of (val0…val1) corresponds to the order of "idt0…idtn" and "edt0…edtm" whose value is 0xFFFF.
This may be omitted when no escape code is used.

i_pflg : Priority flag for IPC to EPC conversion
(If there are two or more associations, priority is

given to the value to which 1 is assigned.)

e_pflg : Priority flag for EPC to IPC conversion. (If there are two or more associations, priority is given to the value to which 1 is assigned.)

idt0...idtn : When the n intermediate object property values are 0xFFFF or when the address of the array that stores the property values is 0xFFFF, it shall be regarded as an escape code and the corresponding escape property value shall be referenced.

edt0...edtm : When the m ECHONET property code values are 0xFFFF or when the address of the array that stores the properties is 0xFFFF, it shall be regarded as an escape code and the corresponding escape property value shall be referenced.

map_id : ID of the mapping type conversion table to register.
The ID generated with RGST_MAP_PROP_REL shall be specified.

(4) Notes

・How to use “NO CARE”

In the case of the example shown in the table below, IPC0 is 3 if EPC0 is 2, regardless of the EPC1 value (NOCARE). In other cases, the IPC0 value is associated using the ECP0 and EPC1 values.

| Intermediate object ID | MapID | IPC0 | IFLG | EPC0 | EPC1(val) | EFLG |
|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | | 2 | 1 | 1 | 2 | 1 |
| | | 3 | 1 | 2 | 0xFFFF(0x0000) | 1 |
| | | 4 | 1 | 3 | 1 | 1 |
| | | 5 | 1 | 3 | 2 | 1 |

If “val” is 0xFFFF, EPC0 = 2 and EPC1 = 0xFFFF is converted to IPC0 = 3.

| Intermediate object ID | MapID | IPC0 | IFLG | EPC0 | EPC1(val) | EFLG |
|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | | 2 | 1 | 1 | 2 | 1 |
| | | 3 | 1 | 2 | 0xFFFF(0xFFFF) | 1 |
| | | 4 | 1 | 3 | 1 | 1 |
| | | 5 | 1 | 3 | 2 | 1 |

・How to use i_pflg and e_pflg

In the case of the example shown in the table below, the IDT values that correspond to EDT = A are 1, 2 and 3, but 1 is associated first because the EFLG is on.

| Intermediate object ID | MapID | 0x22(=IPC) | IFLG | 0xAA(=EPC) | EFLG |
|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | A | 1 |
| | | 2 | 1 | A | 0 |
| | | 3 | 1 | A | 0 |
| | | 4 | 1 | B | 0 |
| | | 5 | 1 | B | 1 |
| | | 6 | 1 | B | 0 |
| | | 7 | 1 | C | 0 |
| | | 8 | 1 | C | 1 |

・Use as a kind of “IF ELSE”

In the case of the example shown in the table below, IPC0 = 5 if EPC0 = 2 except for ECP1 = 1.

| Intermediate object ID | MapID | IPC0 | IFLG | EPC0 | EPC1 | EFLG |
|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | | 2 | 1 | 1 | 2 | 1 |
| | | 3 | 1 | 1 | 3 | 1 |
| | | 4 | 1 | 2 | 1 | 1 |
| | | 5 | 1 | 2 | - | 1 |
| | | 6 | 1 | 3 | 1 | 1 |
| | | 7 | 1 | 3 | 2 | 1 |
| | | 8 | 1 | 3 | 3 | 1 |

・If the conversion condition matches 2 or more lines, priority shall be given to the one that appeared first.

・Registration of unsigned char type (1-byte) IPC and unsigned long type (4-byte) EPC (mapping relationship):

```
HEX
CREATE LONG_EPC 4 ALLOT
12 LONG_EPC 0 + C!
34 LONG_EPC 1 + C!
56 LONG_EPC 2 + C!
78 LONG EPC 3 + C!
0 1 5 4 RGST_EPC    \ unsigned long, 4 bytes
.....
41 LONG_EPC 1 RGST_MAP_PROP_VAL_PR
.....
```

### 8.10.6.15 RGST_FUNC_PROP

(1) Function: Registers property associations in a function type conversion table.

(2) Stack diagram RGST_FUNC_PROP ( idt2edt_func edt2idt_func ipc0...ipcn (ele0) epc0...(elem) epcm nipc nepc obj_id -- )

(3) Explanation

idt2edt_func : Name of the function to convert intermediate property values into ECHONET property values

edt2idt_func : Name of the function to convert ECHONET property values into intermediate property values

ipc0...ipcn : n intermediate object property code values

(ele0)epc0...(elem)epcm : m pairs of ECHONET property code value and element number
Element number (ele) is specified only when the ECHONET property is an array type property.

Nipc : Number of intermediate object properties

Nepc : Number of ECHONET properties

obj_id : Intermediate object ID

(4) Notes

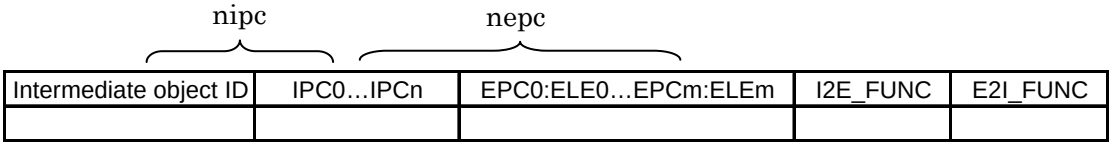
・Method to describe idt2edt_func (When the conversion function name is “I2E”)

```
C" I2E" FIND DROP
```

・Method to describe edt2idt_func (When the conversion function name is “E2I”)

```
C" E2I" FIND DROP
```

・Meanings of nipc and nepc




### 8.10.6.16 SET_IPC

(1) Function: Specifies intermediate object property values and writes the values to the corresponding EPC. Performs the status notification service in the case where status change notification processing has been specified

(2) Stack diagram SET_IPC ( idt ipc obj_id -- )

(3) Explanation
idt : Address of the array in which the intermediate object property value or the property value is stored
ipc : Intermediate object property code
obj_id : Intermediate object ID

(4) Notes
・The node profile object can be accessed by specifying obj_id = 0.
・When writing the value “0x12345678” to unsigned long type IPC:

```
HEX
CREATE LONG_IPC 4 ALLOT
12 LONG_IPC 0 + C!
34 LONG_IPC 1 + C!
56 LONG_IPC 2 + C!
78 LONG IPC 3 + C!
0 1 5 4 RGST_IPC   \ unsigned long, 4 bytes
.....
LONG_IPC 1 0 SET_IPC     \ ipc=1 , obj_id=0
.....
```

### 8.10.6.17 SET_SEND_IPC

(1) Function: Specifies intermediate object property values, writes the values to the corresponding ECHONET properties and sends them to the ECHONET.

(2) Stack diagram SET_SEND_IPC ( idt ipc obj_id isv dst_id -- )

(3) Explanation
idt : Address of the array in which the intermediate object property value or the property value is stored
ipc : Intermediate object property code
obj_id : Intermediate object ID
isv : Intermediate object service code (0x00**)
Specifies a service with a combination of the 4 highest-order bits and 4 lowest-order bits.
4 highest-order bits:
0x1* (Setting; no response required)
0x2* (Setting; response required)
0x3* (Acquisition)
0x4* (Notification)

4 lowest-order bits
0x1* (Request)
0x2* (Response; no response required)
0x3* (Response; response required)
0x4* ("Response not possible" response)

dst_id : Destination address ID or broadcast type:
Destination address ID: 0x0000 － 0x7fff
Broadcast type: 0x8000 － 0xffff

(4) Notes

・The node profile object can be accessed by specifying obj_id = 0.

・Relationship between isv and ESV (in the case of non-element EPC)

| Four highest-order bits | Four lowest-order bits | | | |
|---|---|---|---|---|
| | 0x*1 | 0x*2 | 0x*3 | 0x*4 |
| 0x1* | SetI | - | - | SetI_SNA |
| 0x2* | SetC | Set_Res | - | SetC_SNA |
| 0x3* | Get | Get_Res | - | Get_SNA |
| 0x4* | INF_REQ | INF | INFC | INF_SNA |

・Relationship between isv and ESV (in the case of element EPC)

| Four highest-order bits | Four lowest-order bits | | | |
|---|---|---|---|---|
| | 0x*1 | 0x*2 | 0x*3 | 0x*4 |
| 0x1* | SetMI | - | - | SetMI_SNA |
| 0x2* | SetMC | SetM_Res | - | SetMC_SNA |
| 0x3* | GetM | GetM_Res | - | GetM_SNA |
| 0x4* | INFM_REQ | INFM | INFMC | INFM_SNA |

・For isv, the hatched parts in the table are compulsory.

・In the case where an individual address is specified with dst_id, the "src_id_buf" of CHK_RV_IPC shall be assigned.
For broadcast addresses, ECHONET broadcast type code ＋ broadcast target code (the 15 lowest-order bits) shall be used.

・When writing the value "0x12345678" to unsigned long type IPC:

```
HEX
CREATE LONG_IPC 4 ALLOT
12 LONG_IPC 0 + C!
34 LONG_IPC 1 + C!
56 LONG_IPC 2 + C!
78 LONG IPC 3 + C!
0 1 5 4 RGST_IPC     \ unsigned long, 4 bytes
.....
LONG_IPC 1 0 SET_IPC        \ ipc=1 , obj_id=0
.....
```

### 8.10.6.18 CHK_RV_IPC

(1) Function: Confirms the intermediate object properties whose values have been changed as a result of the reception of a message from the ECHONET.

(2) Stack diagram CHK_RV_IPC ( buf_num src_id_buf obj_id_buf ipc_buf rv_code_buf -- nipc )

(3) Explanation

buf_num : Maximum number of elements of the buffer.

src_id_buf : Buffer address to store the ID indicating the transmission source address internally managed in the interpreter.

obj_id_buf : Buffer address to store the intermediate object ID corresponding to the property for which a message was received.

ipc_buf : Buffer address to store the value of the intermediate object property code for which a message was received.

rv_code_buf : Buffer address to store the service (intermediate object service code (0x00**)) for which a message was received.

4 highest-order bits:

0x1* (Setting; no response required)

0x2* (Setting; response required)

0x3* (Acquisition)

0x4* (Notification)

4 lowest-order bits

0x1* (Request)

0x2* (Response; no response required)

0x3* (Response; response required)

0x4* (“Response not possible” response)

nipc : Number of intermediate object properties whose values have been changed.

(4) Notes

・The “obj_id_buf,” “ipc_buf” and “rv_code_buf” buffers are provided in the download program. Calling this API puts in these buffers the property code and intermediate object ID values of the intermediate objects in which a value change has occurred.

・When the IPC is non-element EPC:

| Four highest-order bits | Four lowest-order bits | | | |
|---|---|---|---|---|
| | 0x*1 | 0x*2 | 0x*3 | 0x*4 |
| 0x1* | SetI | - | - | SetI_SNA |
| 0x2* | SetC | Set_Res | - | SetC_SNA |
| 0x3* | Get | Get_Res | - | Get_SNA |
| 0x4* | INF_REQ | INF | INFC | INF_SNA |

・When the IPC is element EPC:

| Four highest-order bits | Four lowest-order bits | | | |
|---|---|---|---|---|
| | 0x*1 | 0x*2 | 0x*3 | 0x*4 |
| 0x1* | SetMI | - | - | SetMI_SNA |
| 0x2* | SetMC | SetM_Res | - | SetMC_SNA |
| 0x3* | GetM | GetM_Res | - | GetM_SNA |
| 0x4* | INFM_REQ | INFM | INFMC | INFM_SNA |

・For rv_code_buf, the hatched parts in the table are compulsory.

### 8.10.6.19 GET_ IPC

(1) Function: Reads intermediate object property code (IPC) values by performing conversions from the corresponding ECHONET property code (EPC) values.

(2) Stack diagram GET_IPC ( (idt_buf) ipc obj_id -- idt )

(3) Explanation
(idt_buf) : Address of the buffer to store the intermediate object property value.
This shall be specified only when the property to read is an array property.
ipc : Intermediate object property code
idt : Intermediate object property value.
Address of the buffer to store the property value specified by the argument in the case of an array property

(4) Notes
・The “idt_buf” buffer used in the case where the property is an array property shall be provided in the download program.
Calling this API puts the property value specified with ipc in the above-mentioned buffer.
・The node profile object can be accessed by specifying obj_id = 0.
・When reading a value to unsigned long type IPC:

```
HEX
CREATE LONG_IPC 4 ALLOT
0 1 5 4 RGST_IPC     \ unsigned long, 4 bytes
.....
LONG_IPC 1 0 GET_IPC       \ ipc=1 , obj_id=0
```

.....

The newest value is updated in the array LONG_IPC.

### 8.10.6.20 FROM_EQUIPMENT

(1) Function: Receives data from the home appliance device interface.

(2) Stack diagram FROM_EQUIPMENT ( rv_buf buf_size time_out -- rv_code )

(3) Explanation
- rv_buf : Address of the buffer to store the received data
- buf_size : Size of the buffer to store the received data
- time_out : Timeout period [ms]
  (Setting this to 0 specifies the non-blocking mode.)
- rv_code : Reception code
  - −1 : (Reception failed)
  - 0 : (No received data)
  - >0 : (Number of bytes of the received data)

(4) Notes
- The "rv_buf" buffer shall be provided in the download program. Calling this API puts in this buffer the data received from the home appliance device interface.

### 8.10.6.21 TO_EQUIPMENT

(1) Function: Transmits data to the home appliance device interface.

(2) Stack diagram TO_EQUIPMENT ( tr_buf dat_size b_flg -- )

(3) Explanation
- tr_buf : Address of the buffer in which the data to transmit is stored.
- dat_size : Number of bytes of the data to transmit
- b_flg : Blocking mode flag
  - 1 : (Blocking mode flag)
  - 0 : (Non-blocking mode)

(4) Notes
- The "tr_buf" buffer shall be provided in the download program. The data stored in this buffer is transmitted from the home appliance device interface by calling this API.

#### 8.10.6.22 SET_BUF

(1) Function: Sets data in a buffer.
(2) Stack diagram SET_BUF ( dat0...datn tr_buf dat_size -- )
(3) Explanation dat0...datn : Data string to store
dat_size : Address of the buffer to store the data
dat_size : Number of bytes of the data to store
(4) Notes

#### 8.10.6.23 SLEEP

(1) Function: Stands by with the processing stopped for the specified period of time.
(2) Stack diagram SLEEP ( s_time -- )
(3) Explanation s_time : Waiting time [ms]
(4) Notes

#### 8.10.6.24 SET_TIMER

(1) Function: Sets the system timer (0 － 327670 ms).
(2) Stack diagram SET_TIMER ( time -- )
(3) Explanation time : Time [10 ms]
(4) Notes

#### 8.10.6.25 GET_TIMER

(1) Function: Retrieves the system timer value (0 － 327670 ms).
(2) Stack diagram GET_TIMER ( -- time )
(3) Explanation time : Time [10 ms]
(4) Notes

#### 8.10.6.26 INDICATE_STATUS

(1) Function: Indicates, in the display section, the error status.

(2) Stack diagram INDICATE_STATUS ( status -- )

(3) Explanation status : Status:
1 : Error occurred
0 Error resolved

(4) Notes

#### 8.10.6.27 STOP

(1) Function: Stops the operation of the ECHONET middleware adapter.

(2) Stack diagram STOP ( -- )

(3) Explanation None:

(4) Notes

#### 8.10.6.28 RESET

(1) Function: Places the ECHONET middleware adapter in the “unrecognized” state and starts the equipment interface data recognition service.

(2) Stack diagram RESET ( -- )

(3) Explanation None:

(4) Notes

## 8.10.7 Program compression and uncompression specifications

### 8.10.7.1 Overview of program compression

Program compression is a conversion of the program from text to binary format to reduce the size of the program. This shall be done by the ECHONET-ready equipment manufacturer during the program development stage. The compressed program shall be uncompressed in the interpreter of the middleware adapter and executed.

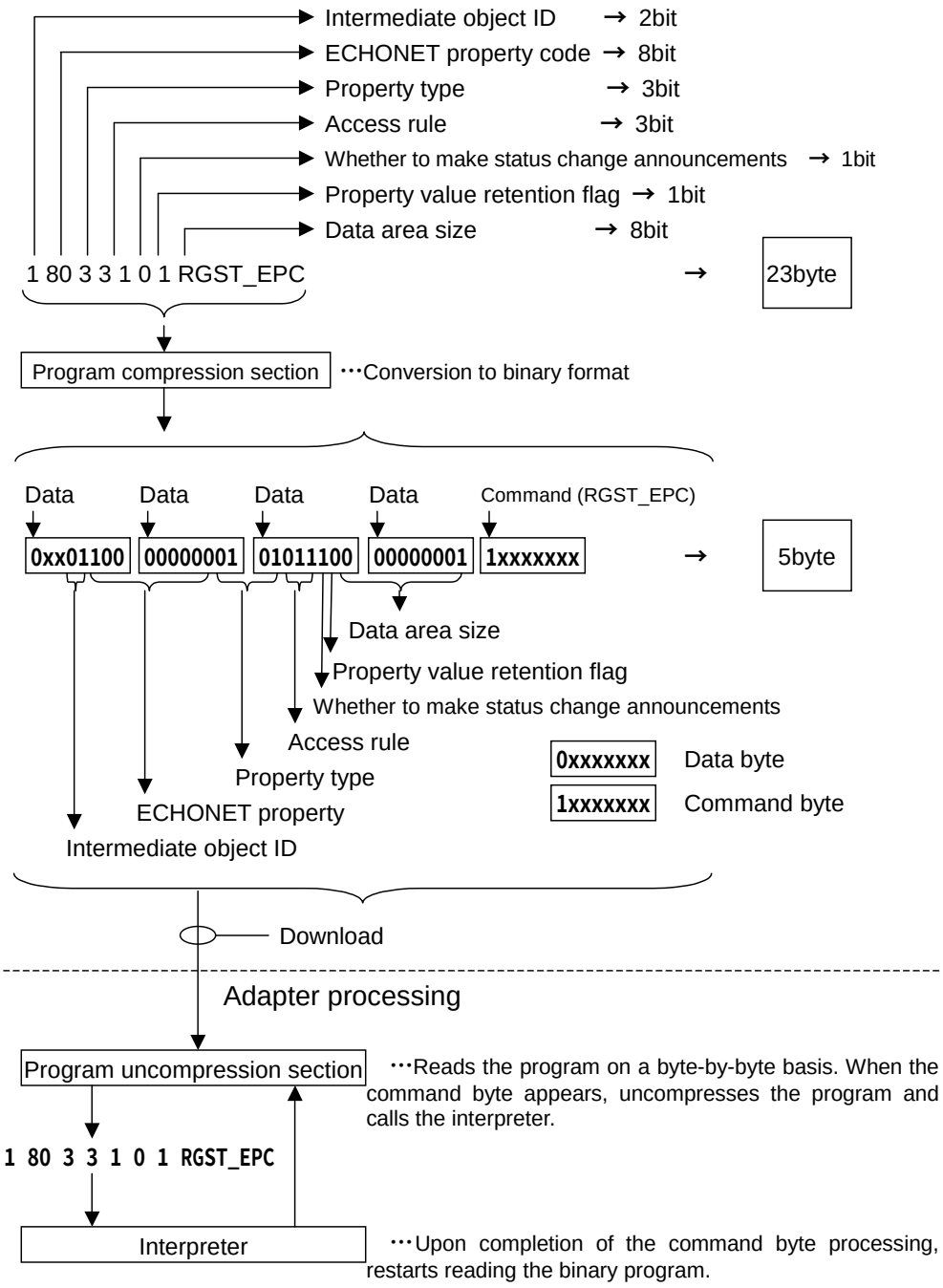



**Fig.8.42 Overview of Program Compression**

Fig. 8.42 shows an overview of the program compression process using RGST_EPC as an example. The RGST_EPC input data sizes are as shown in the figure (The size of the intermediate object ID is 4 bits in practice, but is shown to be 2 bits in the figure). The program compression process generates data in which the values are stored with each piece of the input data assigned with an appropriate number of bits according to the data size, and arranges the data in the second and succeeding bits of the byte. The first bit of each byte is used as a flag to indicate whether the byte is a data byte or a command byte. The bit strings in the data bytes of the compressed program shall be right-justified so that the program can be decoded from the top layer of the stack.

### 8.10.7.2 Byte specifications

(1) Command bytes

A command byte is a byte whose first bit is 1. The remaining 7 bits indicate the type of the command. Command bytes are classified into predefined command bytes and user-defined command bytes.

(a) Predefined command bytes

A predefined command byte is a command byte which has a predefined command byte code value for an API command provided by the interpreter.

(b) User-defined command bytes

A user-defined command byte is a command byte which is assigned, upon the definition of a new variable, function or array by the user using “VARIABLE,” “:” or “CREATE”, to the defined character string. Up to 50 command bytes can be generated in the order of definition starting with “0xCD”.

The table below shows the API commands provided by the interpreter and the corresponding command byte values.

**Table 8.14 API Commands and the Corresponding Command Byte Values**

| API command | Command byte | API command | Command byte |
|---|---|---|---|
| VARIABLE | 0x80 | INIT_ECHONET | 0xA7 |
| : | 0x81 | SET_COM_PARAM | 0xA8 |
| ; | 0x82 | SET_UART_RV_MODE | 0xA9 |
| CREATE | 0x83 | CREATE_MNG_TABLES | 0xAA |
| ALLOT | 0x84 | RGST_OBJ | 0xAB |
| ! | 0x85 | RGST_EPC | 0xAC |
| @ | 0x86 | RGST_EPCM | 0xAD |
| C! | 0x87 | ADD_EPC_MEMBER | 0xAE |
| C@ | 0x88 | RGST_IPC | 0xAF |

| | | | |
|---|---|---|---|
| + | 0x89 | RGST_IDENTICAL_PROP | 0xB0 |
| - | 0x8A | RGST_MAP_PROP_REL | 0xB1 |
| * | 0x8B | RGST_MAP_PROP_VAL | 0xB2 |
| / | 0x8C | RGST_MAP_PROP_VAL_PR | 0xB3 |
| MOD | 0x8D | RGST_FUNC_PROP | 0xB4 |
| LSHIFT | 0x8E | SET_IPC | 0xB5 |
| RSHIFT | 0x8F | SET_SEND_IPC | 0xB6 |
| AND | 0x90 | CHK_RV_IPC | 0xB7 |
| OR | 0x91 | GET_IPC | 0xB8 |
| XOR | 0x92 | FROM_EQUIPMENT | 0xB9 |
| NOT | 0x93 | TO_EQUIPMENT | 0xBA |
| TRUE | 0x94 | SLEEP | 0xBB |
| FALSE | 0x95 | SET_BUF | 0xBC |
| = | 0x96 | SET_TIMER | 0xBD |
| < | 0x97 | GET_TIMER | 0xBE |
| <= | 0x98 | RGST_NODE | 0xBF |
| > | 0x99 | INDICATE_STATUS | 0xC0 |
| >= | 0x9A | STOP | 0xC1 |
| <> | 0x9B | RESET | 0xC2 |
| IF | 0x9C | END_OF_CODE | 0xC3 |
| ELSE | 0x9D | OP_LONG | 0xC4 |
| THEN | 0x9E | CONV_ADDR_TO_TF | 0xC5 |
| BEGIN | 0x9F | | |
| WHILE | 0xA0 | (The code values following the above down to 0xD6 are unused.) | |
| REPEAT | 0xA1 | | |
| DUP | 0xA2 | (0xD7 and succeeding code values are for the user definition option.) | |
| PICK | 0xA3 | | |
| DROP | 0xA4 | | |
| SWAP | 0xA5 | | |
| ROLL | 0xA6 | EXTENSION | 0xFF |

No command byte is assigned to HEX, DECIMAL, BINARY, CONSTANT, C" and FIND, because the processing completes within the compression process.

(2) Data bytes

A data byte is a byte whose first bit is 0. The remaining 7 bits indicate the content of the data. For the data byte generation method, refer to 8.10.7.3.

### 8.10.7.3 Program compression method

The program compression process converts character strings indicating commands and character strings indicating data into command bytes and data byte strings, respectively. The compressed program is suffixed with a command byte indicating the end of the program (END_OF_CODE (0xC3)).

As outlined in 8.10.7.1, character strings indicating data are converted into data byte strings in accordance with the following procedure (See Fig. 8.42).

(i) The character strings indicating data are converted into numerical values.

(ii) Bit patterns are generated by truncating the numerical values according to the data sizes (bit sizes).

(In the case of the example shown in Fig. 8.42, the bit pattern "01" is generated for the intermediate object ID (since the value is "1" and the data size is 2 bits), the bit pattern "10000000" is generated for the ECHONET property code (since the value is "80" and the data size is 8 bits) and the bit pattern "011" is generated for the property type (since the value is "3" and the data size is 3 bits.))

(iii) A data byte string with the value "0" prefixed is generated by sequentially arranging the bit patterns generated in (ii) above in the second and succeeding bits of the byte with the bit patterns right-justified, starting with the data immediately before the command (RGST_EPC in the case of the example shown in Fig. 8.42).

The data sizes for individual types of data shall be as specified in 8.10.7.4. The data size for types of data that are not specified in 8.10.7.4 shall be 16 bits.

The compressed program shall be created as follows:

The character strings shall be read one by one from the beginning of the text format program.

(1) If the character string read is data, skip the data.

(2) If the character string read is "VARIABLE," ":" or "CREATE":

(2-1) Convert the skipped data into a data byte string and add it to the output data.

(2-2) Add the command byte for "VARIABLE," ":" or "CREATE" to the output data.

(2-3) Assign a user-defined command byte to the next character string and add it to the output data.

(3) If the data string read is a command other than those specified in (2) above (including user-defined variables, arrays and functions):

(3-1) Convert the skipped data into a data byte string and add it to the output data.

(3-2) Add the command byte for the command to the output data.

Once all strings have been read, add END_OF_CODE (0xC3) to the output data and output the output data to complete the process.

Note that the above-mentioned procedure is subject to the following 2 exceptions:

(a) For the 'C" FUNC_NAME" FIND DROP' ("FUNC_NAME" indicates the function name) specified when registering a function type property relationship (RGST_FUNC_PROP), the step described in (3) above for the command byte shall not be performed for FUNC_NAME, and the compression processing shall be performed with the command data value treated as 8-bit data.

(b) In the case of a buffer name corresponding to the character string preceding the one immediately before SET_BUF defined with CREATE, the step described in (3) above for the command byte shall not be performed for buffer name, and the compression processing shall be performed with the command byte value treated as 8-bit data.

#### 8.10.7.4 Data Sizes for Individual Types of Data

The table below shows the types of data to which data sizes (bit sizes) other than the default data size (16 bits) are assigned at the time of data byte generation.

**Table 8.15 Data Sizes (Bit Sizes) for Individual Types of Data**

| Data | Bit size | Value range | Command used |
|---|---|---|---|
| Number of retries | 3 | 0～7 | INIT_ECHONET |
| Length of time to wait before making a retry | 15 | 0～32767 (milliseconds) | SET_COM_PRARM |
| Mode for determination of completion | 2 | 1～2 | SET_UART_RV_MODE |
| Length of time to wait to determine that the reception process has been completed | 10 | 0～1023 (milliseconds) | SET_UART_RV_MODE |
| Length of reception completion code | 3 | 0～7 (byte) | SET_UART_RV_MODE |
| Number of home appliance device objects | 4 | 0～15 | CREATE_MNG_TABLES |
| Number of home appliance device side definition properties | 7 | 0～127 | CREATE_MNG_TABLES |
| Number of non-array type ECHONET properties | 7 | 0～127 | CREATE_MNG_TABLES |
| Number of array type ECHONET properties | 7 | 0～127 | CREATE_MNG_TABLES |
| Number of identical value type property relationships | 7 | 0～127 | CREATE_MNG_TABLES |
| Number of mapping type property relationships | 7 | 0～127 | CREATE_MNG_TABLES |

| | | | |
|---|---|---|---|
| Number of function type property relationships | 7 | 0～127 | CREATE_MNG_TABLES |
| Intermediate object ID | 4 | 0～15 | RGST_OBJ<br>RGST_EPC<br>RGST_EPCM<br>ADD_EPC_MEMBER<br>RGST_IPC<br>RGST_IDENTICAL_PROP<br>RGST_MAP_PROP_REL<br>RGST_FUNC_PROP<br>SET_IPC<br>SET_SEND_IPC<br>GET_IPC |
| ECHONET object class group | 8 | 0x00～0xFF | RGST_OBJ |
| ECHONET object class | 8 | 0x00～0xFF | RGST_OBJ |
| ECHONET object instance | 8 | 0x00～0xFF | RGST_OBJ |
| ECHONET property code (EPC) | 8 | 0x00～0xFF | RGST_EPC<br>RGST_EPCM<br>ADD_EPC_MEMBER<br>RGST_IDENTICAL_PROP<br>RGST_MAP_PROP_REL<br>RGST_FUNC_PROP |
| Property type | 3 | 0～7 | RGST_EPC<br>RGST_EPCM |
| Access rule (non-array type EPC) | 3 | 0x00～0x03 | RGST_EPC |
| Access rule (array type EPC) | 7 | 0x00～0x7F | RGST_EPCM |
| Whether or not to make status change announcements | 1 | 0～1 | RGST_EPC<br>RGST_EPCM |
| Property value retention flag | 1 | 0～1 | RGST_EPC<br>RGST_EPCM |
| Data area size (number of bytes) | 8 | 0～255 (byte) | RGST_EPC<br>RGST_EPCM |
| Element number (array type EPC) | 8 | 0～255 | RGST_EPCM<br>ADD_EPC_MEMBER<br>RGST_IDENTICAL_PROP<br>RGST_MAP_PROP_REL<br>RGST_FUNC_PROP |
| Intermediate object property code (IPC) | 8 | 0～255 | RGST_IPC<br>RGST_IDENTICAL_PROP<br>RGST_MAP_PROP_REL<br>RGST_FUNC_PROP |

| | | | |
|---|---|---|---|
| | | | SET_IPC<br>SET_SEND_IPC<br>GET_IPC |
| Intermediate object property type | 3 | 1～7 | RGST_IPC |
| Data area size (number of bytes) | 8 | 0～255 | RGST_IPC |
| Number of intermediate object property code values | 3 | 0～7 | RGST_MAP_PROP_REL<br>RGST_FUNC_PROP |
| Number of ECHONET property code values | 3 | 0～7 | RGST_MAP_PROP_REL<br>RGST_FUNC_PROP |
| Number of mapping records | 8 | 0～255 | RGST_MAP_PROP_REL |
| Mapping type conversion table ID | 7 | 0～127 | RGST_MAP_PROP_REL<br>RGST_MAP_PROP_VAL<br>RGST_MAP_PROP_VAL_PR |
| Intermediate object property value (IDT) | 16 | 0x0000～<br>0xFFFF | RGST_MAP_PROP_VAL<br>RGST_MAP_PROP_VAL_PR<br>SET_IPC<br>SET_SEND_IPC |
| ECHONET object property value (EDT) | 16 | 0x0000～<br>0xFFFF | RGST_MAP_PROP_VAL<br>RGST_MAP_PROP_VAL_PR |
| IPC to EPC conversion priority flag | 1 | 0～1 | RGST_MAP_PROP_VAL_PR |
| EPC to IPC conversion priority flag | 1 | 0～1 | RGST_MAP_PROP_VAL_PR |
| Maximum number of buffer elements | 7 | 0～127 | CHK_RV_IPC |
| (Transmission/receiving) buffer size | 7 | 0～127 | FROM_EQUIPMENT<br>TO_EQUIPMENT<br>SET_BUF |
| Timeout period | 15 | 0～32768 (milliseconds) | FROM_EQUIPMENT |
| Standby period | 15 | 0～32768 (milliseconds) | SLEEP |
| Time | 15 | 0～32760 (x10 milliseconds) | SET_TIMER |
| Start mode | 3 | 0～7 | INIT_ECHONET |
| Node ID | 8 | 0～255 | RGST_NODE<br>RGST_OBJ |
| Blocking mode flag | 1 | 0～1 | TO_EQUIPMENT |
| Status | 1 | 0～1 | INDICATE_STATUS |
| Type of calculation | 5 | 0～31 | OP_LONG |

| | | | |
|---|---|---|---|
| 1byte data | 8 | 0x00～0xFF | SET_BUF |
| 2byte data | 16 | 0x0000～<br>0xFFFF | (Default) |

### 8.10.7.5 Limitations regarding the use of variables and functions

For certain commands, the target data for processing cannot be specified with a return value of a function or a variable. In such a case, the value shall be specified directly or specified using a constant defined with CONSTANT.

Correct description method(1)

```
HEX
1 80 3 3 1 0 1RGST_EPC
```

Correct description method(1)

```
HEX
3 CONSTANT RULE
1 80 3 RULE 1 0 1RGST_EPC
```

Incorrect description method(1)

```
HEX
VARIABLE RULE
RULE 3 !
1 80 3 RULE @ 1 0 1RGST_EPC
```

Incorrect description method(2)

```
HEX
: FUNC
 3
;
1 80 3 FUNC 1 0 1RGST_EPC
```

The API's for which it is necessary to specify the value directly or using a constant defined with CONSTANT are as follows:

(1) Data processed by INIT_ECHONET

(2) Data processed by CREATE_MNG_TABLES

(3) Data processed by RGST_EPC

(4) Data processed by RGST_EPCM

(5) Data processed by ADD_EPC_MEMBER

(6) Data processed by RGST_IPC

(7) Data processed by RGST_IDENTICAL_PROP

(8) Data processed by RGST_MAP_PROP_REL

(9) Data processed by RGST_FUNC_PROP ('C" FUNC_NAME" FIND DROP' ("FUNC_NAME" indicates the function name)) is excluded

# Chapter 9 NetID Servers

## 9.1 Basic Concept

An ECHONET environment allows different types of networks to connect to each other and operate as a coherent system. A NetID server is a piece of equipment that is designed to be capable of operating in an ECHONET environment to assign each subnet a unique NetID in a domain and perform ECHONET communication processing. However, a NetID server does not necessarily have to be a special piece of equipment dedicated to assigning NetIDs. A PC or controller may have the NetID server functions and serve as a NetID server. In fact, any equipment (regardless of type), such as an air conditioner, may have the NetID server functions and serve as a NetID server. However, only one piece of equipment may serve as a NetID server in a given domain.

A NetID server also has the functions of an ECHONET router, but does not necessarily have to be physically connected to two or more subnets.

## 9.2 Definition of Functions

A NetID server shall meet the following minimum requirements:

(1) If there are two or more subnets within a domain, the NetID server must assign NetIDs in such a way that no overlapping occurs. If the NetID server detects an overlapping of NetIDs, it shall send a NetID to the appropriate ECHONET router or deactivate the ECHONET router.

(2) A NetID server shall have a NetID server profile object.

(3) A NetID server shall have a router profile object. However, it does not necessarily have to be physically connected to two or more subnets.

## 9.3 Mechanical and physical characteristics

Regarding connection to the transmission medium, the specifications provided in Part 3 for the individual lower-layer communication protocols available for NetID servers shall apply. This subsection provides the mechanical and physical specifications for NetID servers used in addition to those described in Part 3.

### 9.3.1 Display section

Where a means to indicate the operation status of a NetID Server is provided, the minimum requirements listed below must be satisfied. For indication using a method not described herein, the specifications for the individual products shall be used. For operation status-related matters, refer to Part 2, Chapter 5.

* Number of LEDs
  One (used to indicate the operation status)
* LED color
  Green
* Status indication
  Normal operation: Lit
  Initial processing: Blinking (Period 2)
  Abnormal state: Blinking (Period 3)
  Not operating: Unlit

● Period 1: Repeated lit-unlit sequence with 2 seconds of lighting followed by 2 seconds without lighting.

● Period 2: Repeated lit-unlit sequence with 2 seconds of lighting followed by 0.5 seconds without lighting.

● Period 3: Repeated lit-unlit sequence with 0.5 seconds of lighting followed by 0.5 seconds without lighting.

Note: “Initial processing” means a cold start or a warm start (i.e. a start whereby hardware reset processing is performed with the previously acquired addresses and initial setting data retained).

## 9.4 Electrical characteristics

Regarding connection to the transmission medium, the specifications provided in Part 3 for the individual transmission medium communication protocols available for NetID servers shall apply.

## 9.5 Logical specifications

Regarding logical specifications for transmission medium communication protocols, the

specifications provided in Part 3 for the individual transmission medium communication protocols available for NetID servers shall apply. Regarding logical specifications for the protocol difference absorption processing section, the requirements specified in Part 2, Chapter 7 (“Specifications for Processing at the Protocol Difference Absorption Processing Section”) shall apply. Regarding routing specifications, the routing requirements specified in Part 2 shall apply.

# Appendix 1 Reference Document

(1) “PH-CONNECTOR” issued by JST Mfg. Co., Ltd.

# Appendix 2 Example Interpreter Method Programs

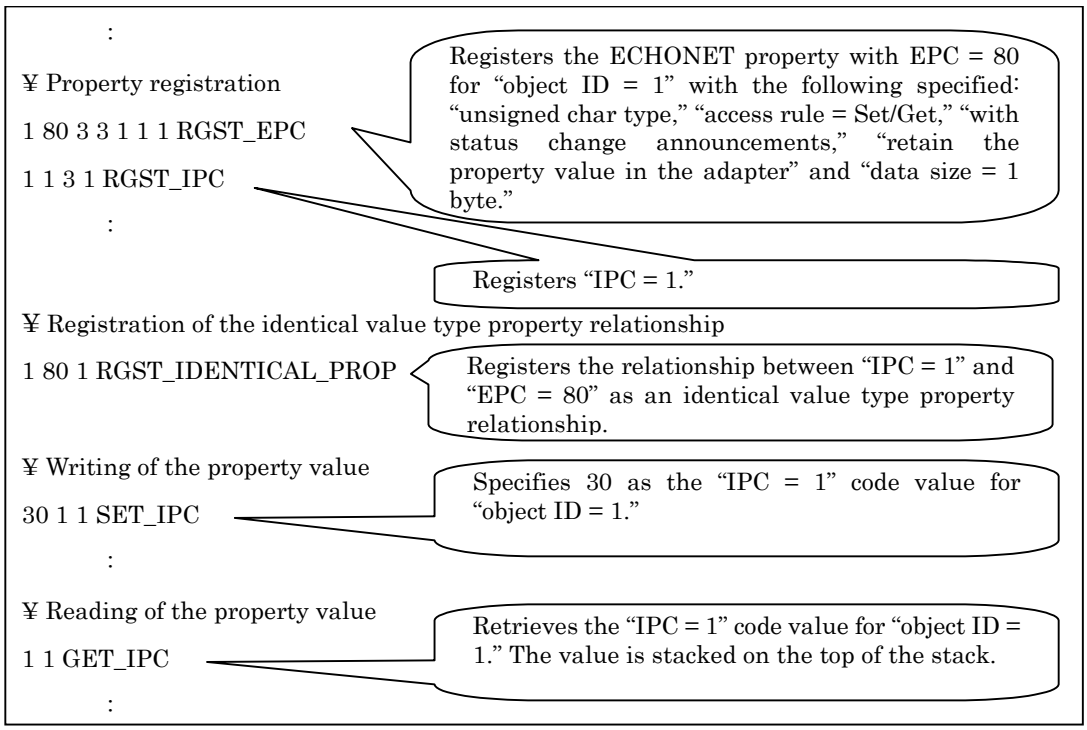
(1) "PH-CONNECTOR" issued by JST Mfg. Co., Ltd.

(1.1) Identical value type property conversion

This section provides an example of an identical value type property relationship conversion. The figure below shows a program code to register the property relationship and a program code to read and write the property value in the case where the intermediate object's "operation status" property (IPC = 0x01) and the ECHONET "operation status" property (EPC = 0x01) are in an identical value type relationship in a home air conditioner.

| Operation status (IPC=0x01) | Operation status (EPC=0x80) |
|---|---|
| 0x31(Power OFF) | 0x31(Power OFF) |
| 0x30(Power ON) | 0x30(Power ON) |

**Relationships between the Property Values**




**Program Code**

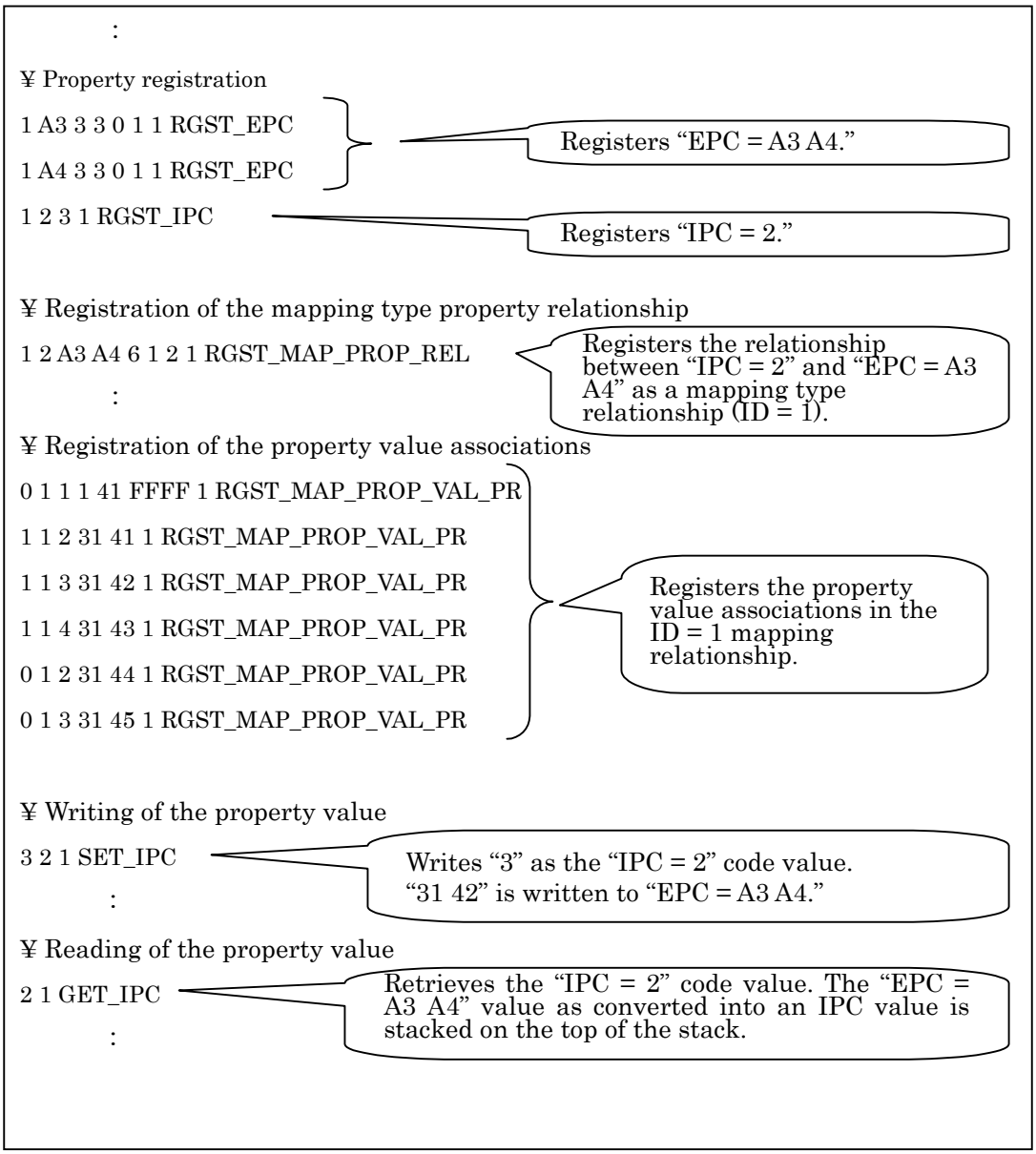
(1.2) Mapping type property conversion

This section provides an example of a mapping type property relationship conversion. In this example, it is assumed that the intermediate object's "wind direction (vertical)" property (IPC = 0x02) is associated with the ECHONET "automatic swing of air flow" property (EPC = 0xA3) and the ECHONET "wind direction setting (vertical)" property (EPC = 0xA4) in an home air conditioner and the relationships between the values are as shown in the table below. Because some of the IPC values are associated with multiple sets of EPC values in this example, the associations to which priority is given are marked with (P).

| Wind direction (vertical) (IPC=0x02) | "Automatic swing of air flow" setting (EPC=0xA3) | Wind direction setting (vertical) (EPC=0xA4) |
|---|---|---|
| 0x01(Swing) | 0x41(Vertical) | (no care) |
| (P) 0x02(Uppermost) | 0x31(OFF) | 0x41(Uppermost) |
| (P) 0x03(Lowermost) | 0x31(OFF) | 0x42(Lowermost) |
| 0x04(Central) | 0x31(OFF) | 0x43(Central) |
| 0x02(Uppermost) | 0x31(OFF) | 0x44 (Midpoint between uppermost and central) |
| 0x03(Lowermost) | 0x31(OFF) | 0x45 (Midpoint between lowermost and central) |

**Relationships between the Property Values**

Below is an example property conversion code for the case where the above-mentioned mapping type relationship exists.




**Program Code**

(1.3) Function type property conversion

This section provides an example of a function type property relationship conversion. This example explains how functions are defined for a relationship between the intermediate object's "temperature setting" property and the ECHONET "temperature setting" property whereby the value of the former property is always higher than the value of the latter property by 10 in a home air conditioner. This section also shows, as reference information, the code as shown in the C language.

```
¥ IPC to EPC conversion of "temperature setting"
: I2E
  10 -
;

¥ EPC to IPC conversion of "temperature setting"
: E2I
  10 +
;
```

**Relationships between the Property Values**

```
unsigned char I2E (unsigned char x)
{
   return (x - 0x10);
}
unsigned char E2I (unsigned char x)
{
   return (x + 0x10);
}
```

**Relationships between the Property Values as Expressed in the C Language (reference information)**

Below is an example property conversion code for the case where the above-mentioned function type relationship exists.

```
        :
¥ Property registration
1 B3 3 3 0 0 1 RGST_EPC
1 3 3 1 RGST_IPC

¥ IPC to EPC conversion of “temperature setting”
: I2E
   10 -
;

¥ EPC to IPC conversion of “temperature setting”
: E2I
   10 +
;

¥ Registration of the property value associations
C" I2E" FIND DROP C" E2I" FIND DROP 3 B3 1 1 1 RGST_FUNC_PROP

¥ Writing of the property value
30 3 1 SET_IPC
        :
¥ Reading of the property value
3 1 GET_IPC
        :
```

Registers “EPC = B3” (Data size = 1 byte)

Registers “IPC = 3” (Data size = 1 byte)

Registers the association between “IPC = 3” and “EPC = B3” as a function type property relationship together with the function name.

Writes the EPC value (32) for which the “IPC =2” value is 48.

The IPC value for the case where the object ID is “1” and the IPC code is “3” is stacked on the top of the stack.

**Program Code**

(2) Example of overall processing

Below is an example of an overall program to perform an “operation status” (Power ON/OFF) property conversion and the communication processing. In this example, it is assumed that the associations between the values are the mapping type associations shown in the table below.

| Operation status (IPC=0x01) | Operation status (EPC=0x80) |
|---|---|
| 0x31(Power OFF) | 0x31(Power OFF) |
| 0x30(Power ON) | 0x30(Power ON) |

```
HEX                                  ¥ Specifies hexadecimal

400 SET_COM_PARAM                    ¥ Length of time to wait before making a retry: 1024
ms
10 1 SET_UART_RV_MODE                ¥ Length of time to wait to determine that the
                                     reception process has been completed: 16 msec

CREATE RV_BUF 16 ALLOT               ¥ Receiving buffer: 22 bytes
CREATE TR_BUF 16 ALLOT               ¥ Transmission buffer: 22 bytes

CREATE SRC_BUF 5 ALLOT               ¥ Transmission source storage buffer
CREATE OBJ_BUF 5 ALLOT               ¥ Received object storage buffer
CREATE IPC_BUF 5 ALLOT               ¥ Received IPC storage buffer
CREATE RV_CD_BUF 5 ALLOT             ¥ Received code storage buffer

VARIABLE ERR_FLG                     ¥ Error flag
0 ERR_FLG !                          ¥Sets the error flag to 0

VARIABLE RV_NUM                      ¥ Received numbers
VARIABLE CNT                         ¥ Loop counter
VARIABLE CNT2                        ¥ Loop counter 2
VARIABLE IDT                         ¥ IDT
VARIABLE FCC                         ¥ FCC

1 1 1 0 0 1 0 CREATE_MNG_TABLES      ¥ Creation of management tables (Object 1, EPC 1)
1 0 1 30 1 RGST_OBJ                  ¥ Object registration
1 80 3 3 1 1 1 RGST_EPC              ¥ “Operating status” (EPC = 80) registration
1 1 3 1 RGST_IPC                     ¥ “Operating status” (IPC = 1) registration
1 1 80 2 1 1 1 RGST_MAP_PROP_REL     ¥ Associates “IPC = 1” with “EPC = 80” in a mapping
                                         type property
                                     ¥ relationship (ID = 1)
1 31 1 RGST_MAP_PROP_VAL             ¥ Power OFF
2 30 1 RGST_MAP_PROP_VAL             ¥ Power ON
```

```
: CALC_FCC                              ¥ FCC calculation at the time of transmission (Sum of
                                          TR_BUF [1] to [14])
  1 CNT2 !
  0 FCC !
  BEGIN
    CNT2 @ 14 <
  WHILE
    TR_BUF CNT2 @ + C@ FCC @ + FCC !        ¥ FCC += TR_BUF[CNT]
    CNT2 @ 1 + CNT2 !
  REPEAT
  FCC @ FF AND
;

: CHK_FCC                               ¥ FCC check at the time of reception (Sum of TR_BUF
                                            [1] to [14])
  1 CNT2 !
  0 FCC !
  BEGIN
    CNT2 @ 14 <
  WHILE
    RV_BUF CNT2 @ + C@ FCC @ + FCC !        ¥ FCC += RV_BUF[CNT]
    CNT2 @ 1 + CNT2 !
  REPEAT
  FCC @ FF AND 0 = IF
    0                                   ¥ OK (return 0)
  ELSE
    1                                   ¥ NG (return 1)
  THEN
;

: PR_ECHO                               ¥ Function definition (Processing performed in
                                          response to the reception of a message from the
                                          ECHONET)
 ●● ・・・ ● TR_BUF 16 SET_BUF          ¥ Transmission buffer setting (operation request)
                                        ¥ The part represented by "●●…●" differs
                                          depending on the home appliance
                                        ¥ device communication specifications.
 IPC_BUF CNT @ + C@ 1 = IF              ¥ When IPC = 1
  1 1 GET_IPC IDT !                     ¥ Acquisition of IDT (IPC = 1)
  IDT @ TR_BUF ● + C!                   ¥ Sets IDT in the ●th byte of the transmission buffer
  CALC_FCC TR_BUF 15 + C!               ¥ FCC setting
  TR_BUF 16 0 TO_EQUIPMENT              ¥ Transmission to equipment
  RV_BUF 16 100 FROM_EQUIPMENT          ¥ Reception from equipment
  2 = IF                                ¥ If a message is received
   CHK_FCC 0 = IF                       ¥ If FCC is OK
    RV_BUF 5 + C@ 0 = IF                ¥ If the response is a normal response
     IDT @ 1 1 SET_IPC                  ¥ IPC set
    THEN
   THEN
```

```
   THEN
  THEN
;


: PR_REG                                ¥ Function definition (Routine processing)
  ● ● ・・・ ● TR_BUF 16 SET_BUF        ¥ Transmission buffer setting (status request)
  TR_BUF 16 0 TO_EQUIPMENT              ¥ Transmission to equipment
  RV_BUF 16 100 FROM_EQUIPMENT          ¥ Reception from equipment
  2 = IF                                ¥ If a message is received
    CHK_FCC 0 = IF                      ¥ If FCC is OK
      RV_BUF ● + C@ 1 1 SET_IPC         ¥ IDT setting from the ●th byte of the transmission
                                          buffer
    THEN
  THEN
;

1 3 INIT_ECHONET                        ¥ ECHONET initialization

PR_REG                                  ¥ Initial value setting (Routine processing)

BEGIN
  TRUE
WHILE
  5 SRC_BUF OBJ_BUF IPC_BUF RV_CD_BUF CHK_RV_IPC
                                        ¥ Reception from the ECHONET
  RV_NUM !                              ¥ "Number of received messages" setting
  0 CNT !                               ¥ Counter setting
  BEGIN
    CNT @ RV_NUM @ <
  WHILE
    PR_ECHO                             ¥ Processing performed in response to the reception of
                                          a message from the ECHONET
    CNT @ 1 + CNT !                     ¥ Increments the loop counter by one
  REPEAT
  PR_REG                                ¥ Equipment status confirmation (Routine processing)
REPEAT
```