Part III

ECHONET Lite Communications Equipment Specifications

Revision History

・Version 1.00 Draft	9 March 2011	Released / Open to the Consortium Members
・Version 1.00	30 June 2011	Open to the Consortium Members
	3 September 2012	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.

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

The original language of the ECHONET Lite Specifications is Japanese. This English version is a translation of the Japanese version; in case of any queries about the English version, refer to the Japanese version.

Contents

Chapter 1. Overview of ECHONET Lite Communications Equipment Specifications

1.1 Concept

Part 3 defines the specifications for ECHONET Lite nodes, ECHONET Lite gateways and ECHONET Lite middleware adapters as communications equipment. Detailed requirements are also defined for the interfaces between ECHONET Lite device and ECHONET Lite middleware adapters and equipment adapters as well as their respective functions.

1.2 Overview of Communications Equipment Specifications for ECHONET Lite Nodes

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

- Lower communication layer
- ECHONET Lite Communication Middleware

Specifications for communication equipment of the ECHONET Lite nodes are not defined except the above-mentioned factors.

1.3 Overview of Communications Equipment Specifications for ECHONET Lite Gateways

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

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

The specifications for the ECHONET Lite gateway are described in Chapter 2.

1.4 Overview of Communications Equipment Specifications for ECHONET Lite Middleware Adapters

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

For the purposes of this Specification, an "ECHONET Lite middleware adapter" is defined as an adapter for pieces of equipment that do not have an ECHONET Lite communications processing section and ECHONET Lite lower communication layer to connect them to the ECHONET Lite network. Therefore, an ECHONET Lite middleware adapter must meet the following requirements:

* ECHONET Lite communications processing section
* Lower communication layer
* ECHONET Lite middleware adapter communication software

ECHONET Lite middleware adapter communication software is a software program for communication between an ECHONET Lite middleware adapter and a piece of equipment (ECHONET Lite-ready equipment). The specifications for this software are defined in Chapter 3.

Chapter 2. ECHONET Lite Gateway

2.1. Concept

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

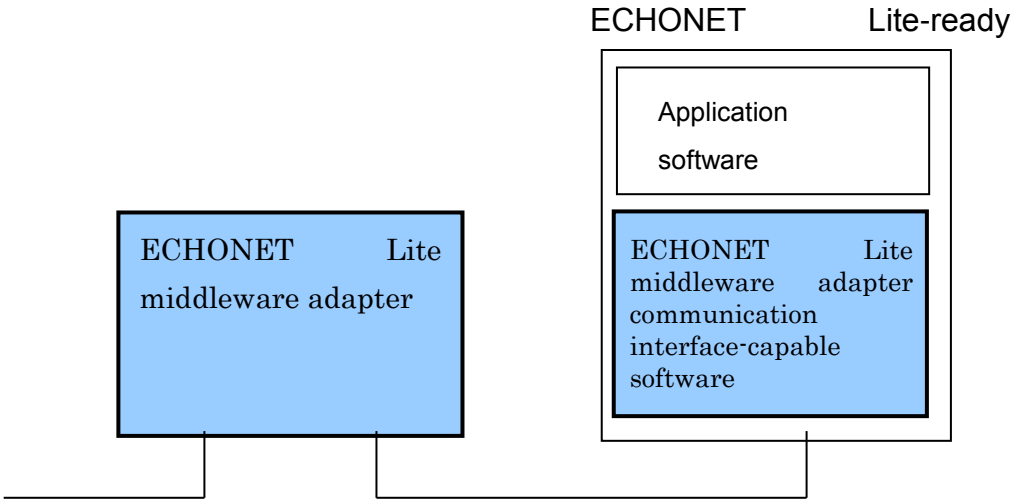
When the system is installed in an ordinary residence, we recommend that users prepare a security function, including a verification function and access control function, for the gateway application to ensure the security of the ECHONET Lite domain. The functional definitions in such a case are described in Part 4.

Chapter 3. ECHONET Lite Middleware Adapters

3.1. Concept

For the purposes of this chapter, an “ECHONET Lite middleware adapter” shall be defined as an adapter to connect a piece of equipment that does not have the ECHONET Lite communications processing section to the ECHONET Lite network (see Fig. 3.1).

As shown in Fig. 3.1, equipment to which an ECHONET Lite middleware adapter can be connected is called “ECHONET Lite-ready equipment.”
An ECHONET Lite middleware adapter shall be configured so as to minimize the burden on ECHONET Lite-ready equipment in relation to network-related processing and the increase in ECHONET Lite-ready equipment costs (software and hardware costs).



ECHONET Lite network ECHONET Lite middleware adapter communication interface

Fig. 3.1 ECHONET Lite Middleware Adapter and ECHONET Lite-ready Equipment

The device specifications and electrical and logical specifications for ECHONET Lite middleware adapters are provided in Sections 3.1 through 3.5. The mechanical, physical and logical characteristics specifications for ECHONET Lite middleware adapter communication interface software are provided in Section 3.6.

In this version of the ECHONET Lite Specification, two types of protocol specifications for ECHONET Lite middleware adapter communication interface software are provided to

minimize the burden on ECHONET Lite-ready equipment in relation to network-related processing:

(1) Object generation type

(2) Peer-to-peer type

Whether or not an ECHONET Lite middleware adapter can be connected to a piece of ECHONET Lite-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 3.7.

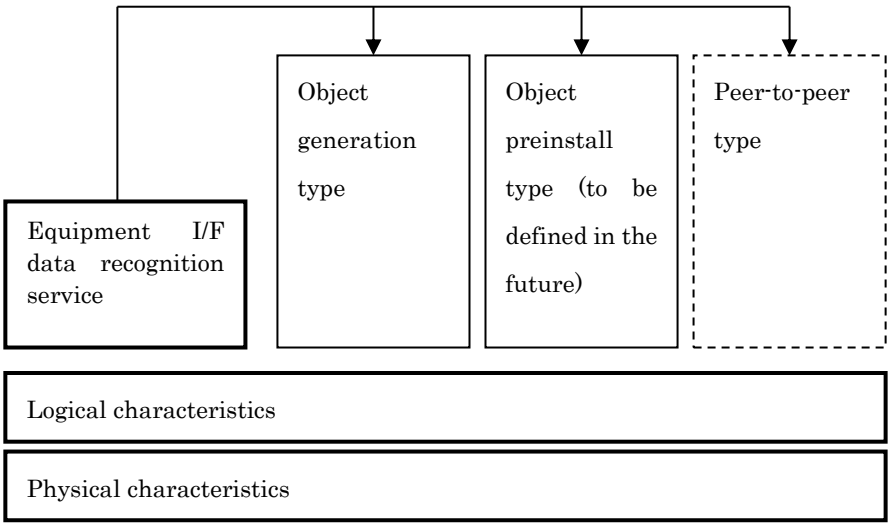


Fig. 3.2 ECHONET Lite Middleware Adapter Communication Software Hierarchy

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

Table 3.1 shows examples of acceptable combinations (i.e. combinations with which communication is possible) of types supported by ECHONET Lite middleware adapters and types supported by ECHONET Lite-ready equipment.

Table 3.1 Acceptable Combinations of Types Supported by ECHONET Lite Middleware Adapters and Types Supported by ECHONET Lite-ready Equipment

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

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

3.1.1. Anticipated configurations for ECHONET Lite middleware adapters (commentary)

To accommodate the implementation methods for ECHONET Lite middleware adapters, several ECHONET Lite middleware adapter communication software protocols have been defined. Fig. 3.3 shows the anticipated ECHONET Lite middleware adapter configurations.

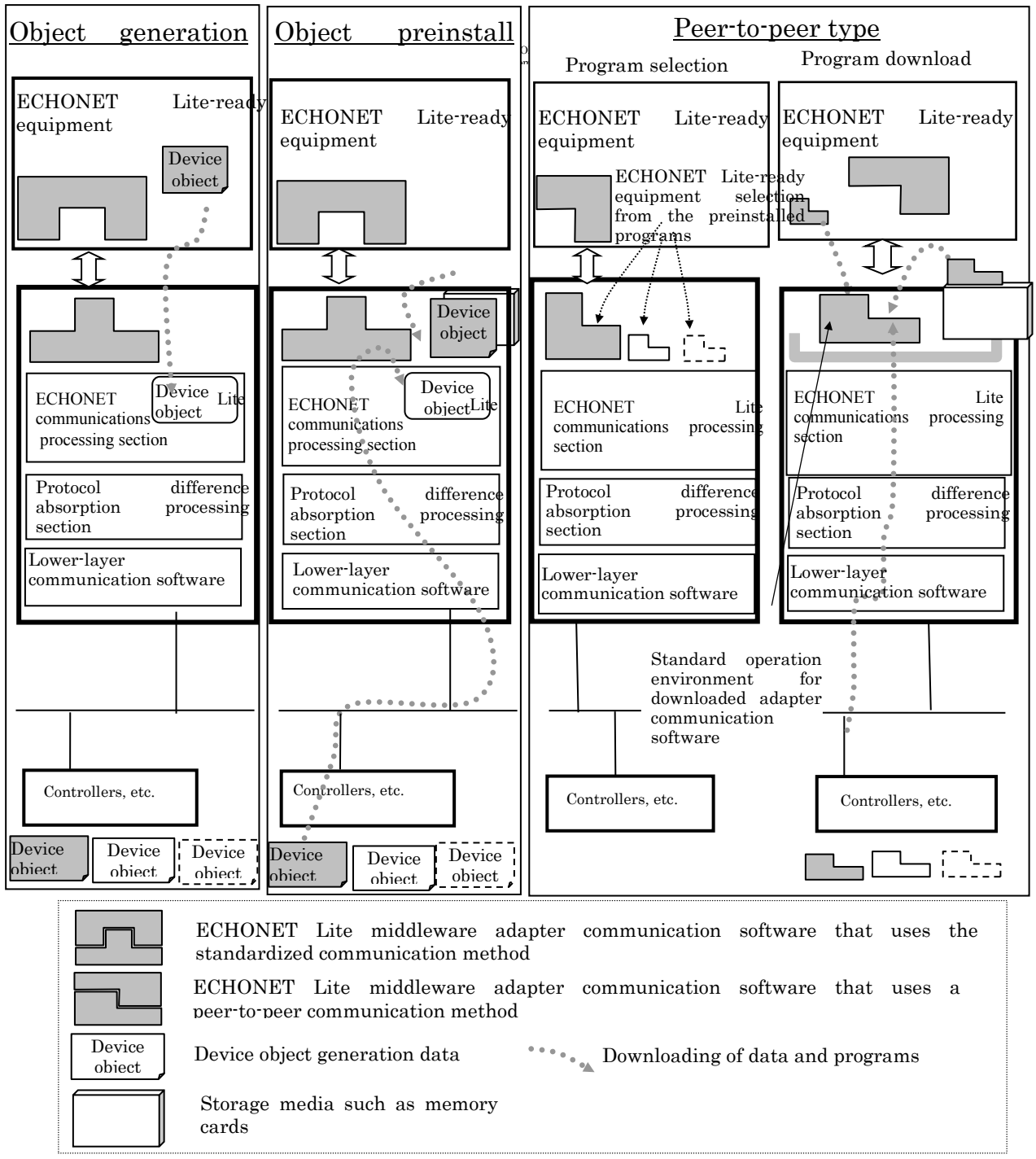


Fig. 3.3 Anticipated Configurations for ECHONET Lite Middleware Adapter

As shown in Fig. 3.3, the following three implementation methods for ECHONET Lite middleware adapters are anticipated:

(1) Object generation type

Data is exchanged between the ECHONET Lite middleware adapter and ECHONET

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

(2) Object preinstall type

Data is exchanged between the ECHONET Lite middleware adapter and ECHONET Lite-ready equipment using a standardized communication method. The assumption is that the object generation data is either preinstalled in the ECHONET Lite 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 Lite middleware adapter and ECHONET Lite-ready equipment using a user-defined communication method. The following two approaches are anticipated.

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

The other is the “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 define the following so that when an ECHONET Lite middleware adapter is connected to a piece of ECHONET Lite-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 Lite middleware adapter by acquiring object generation data from the ECHONET Lite-ready equipment. This version of the ECHONET Lite 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 Lite Specification specifies the communication method to download the ECHONET Lite middleware adapter communication software from ECHONET Lite-ready equipment. This version of the ECHONET Lite Specification does not specify requirements for communication methods to download software from equipment other than ECHONET Lite-ready equipment.

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

A service to acquire ECHONET Lite-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 Lite middleware adapter to the ECHONET Lite-ready equipment. After this inquiry, the ECHONET Lite middleware adapter executes the ECHONET Lite middleware adapter communication software based on the acquired data.

.

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

Fig. 3.4 illustrates the process of executing the equipment interface data recognition service. For details, see Fig. 3.13.

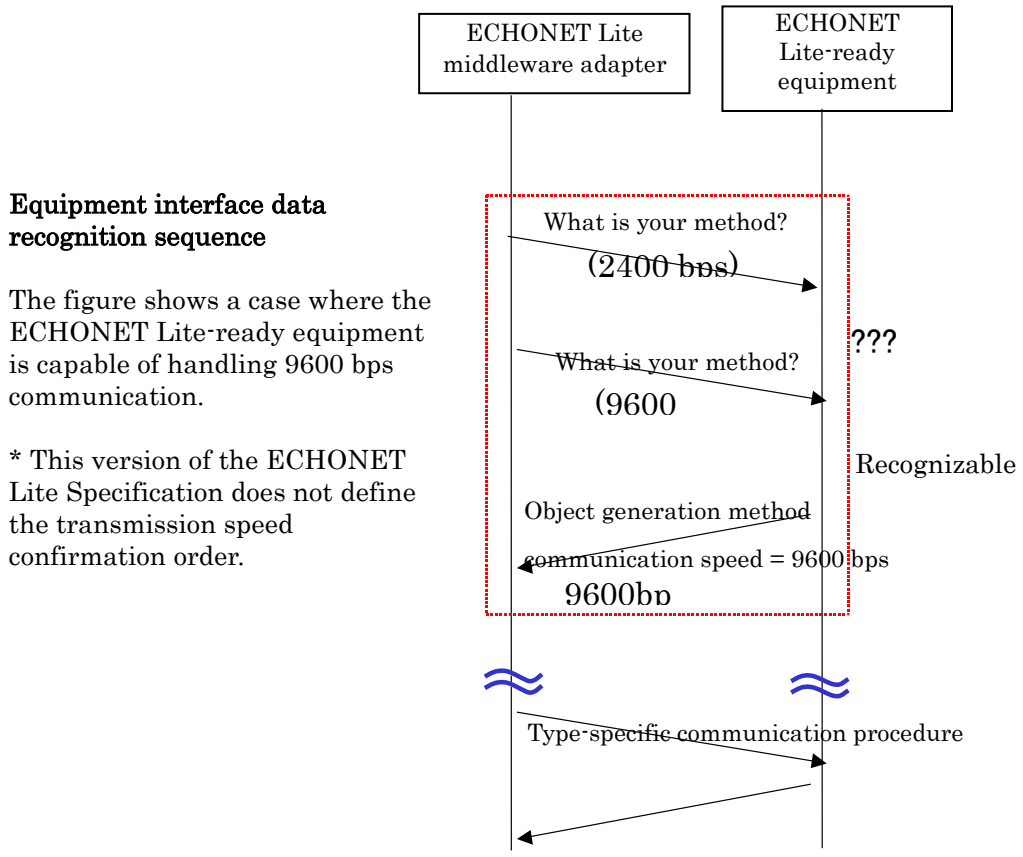


Fig. 3.4 ECHONET Lite-ready Equipment Recognition Process

3.2. Definitions of Functions

The functions required for an ECHONET Lite 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 Lite lower-layer communication software, which means that a transceiver capable of handling the ECHONET Lite lower-layer communication protocol is required as a function.

(2) ECHONET Lite communications processing function

A function to perform the processing specified in “Part 2, Chapter 5: Specifications for Processing at the ECHONET Lite Communications Processing Section.” This function is performed by the ECHONET Lite communications processing section.

(3) ECHONET Lite middleware adapter communication interface function

This function, defined in “3.6 ECHONET Lite Middleware Adapter Communication Software Specifications,” performs, between the ECHONET Lite-ready equipment application software and ECHONET Lite communications processing section, the necessary communications processing between the ECHONET Lite middleware adapter and ECHONET Lite-ready equipment, which is defined in the “ECHONET Lite Middleware Adapter Communication Interface Specifications.” This function is performed by the ECHONET Lite middleware adapter communication software.

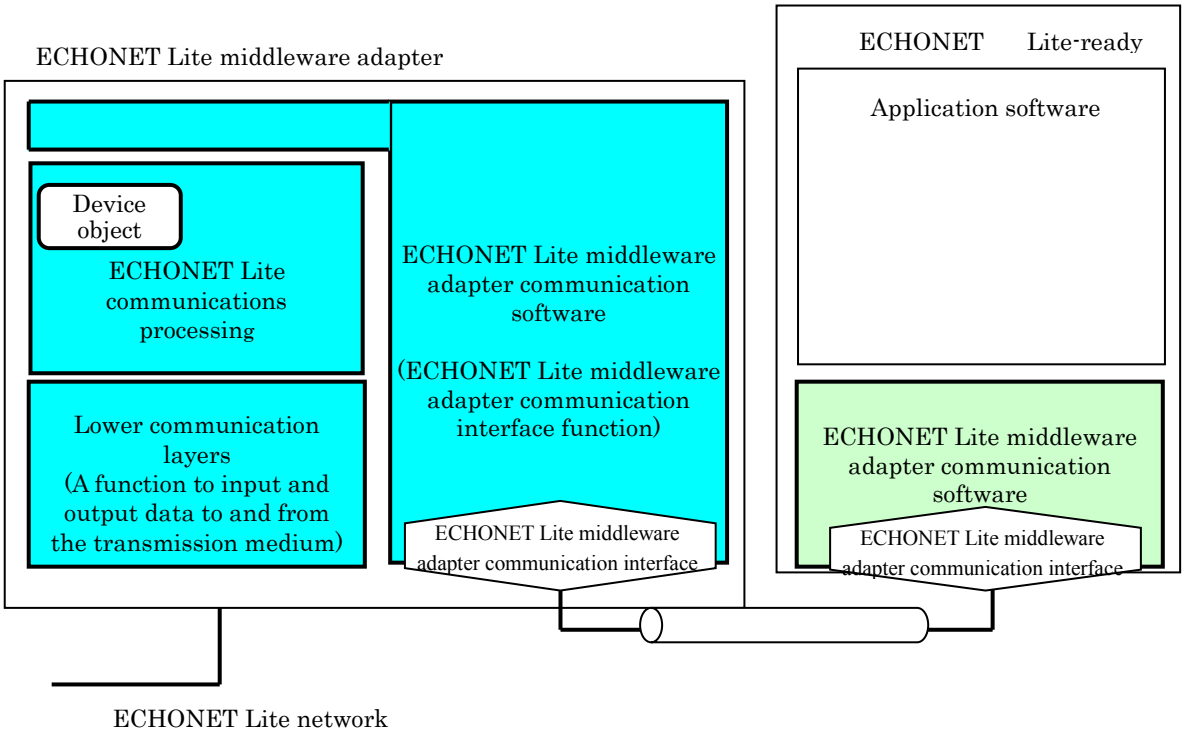


Fig. 3.5 ECHONET Lite Middleware Adapter Functions

3.3. Mechanical and Physical Characteristics

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

3.3.1. Shape

No specifications are defined for shape, with the exception of the connection block to the ECHONET Lite-ready equipment (ECHONET Lite 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 lower communication layer.

3.3.2. Display section

Where LEDs are provided as a means of indicating the operation status of an ECHONET Lite 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 lit followed by approximately 0.5 second unlit

* Rapid blinking - repeated lit-unlit sequence with approximately 0.5 second lit followed by approximately 0.5 second unlit

(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).

3.4. Electrical Characteristics

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

3.5. Logical Requirements

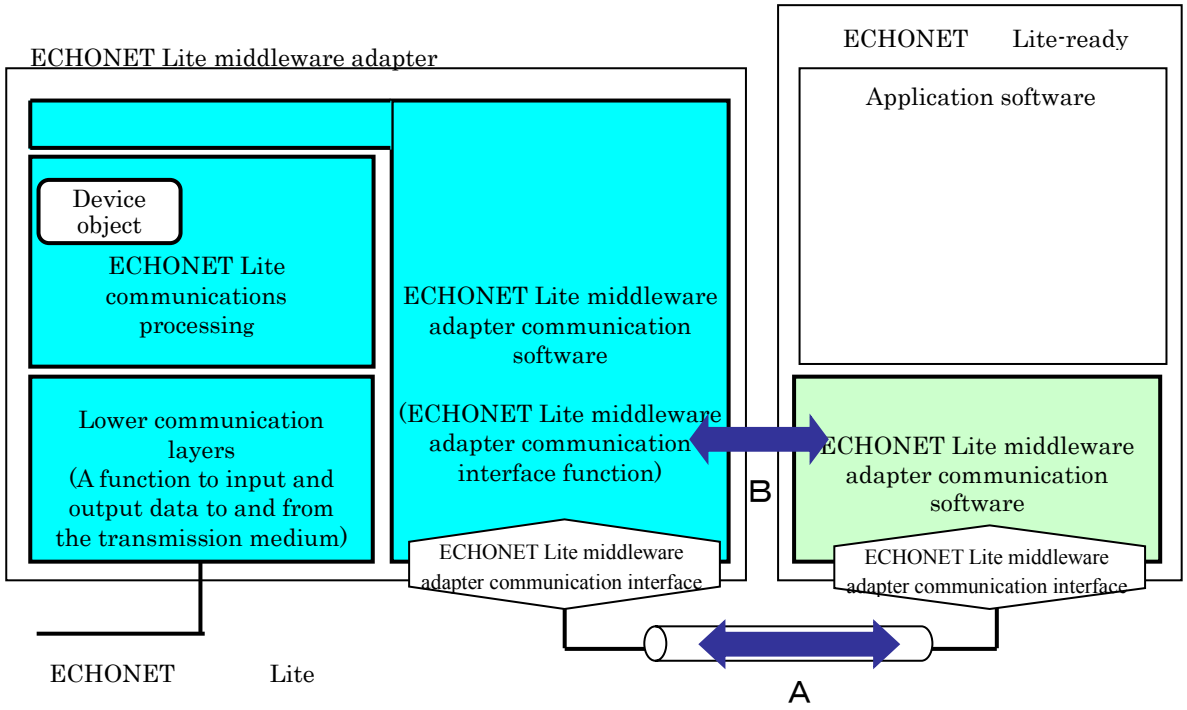
The logical requirements for ECHONET Lite middleware adapter communication software shall be as defined in Section 6. The logical requirements for lower communication layers are based on the specification of the lower communication layer used for communication.

3.6. ECHONET Lite Middleware Adapter Communication Software Specifications

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

3.6.1. ECHONET Lite middleware adapter communication interfaces – overview

Fig. 3.6 shows how the ECHONET Lite middleware adapter communication software allows the ECHONET Lite middleware adapter and the equipment to communicate with each other.



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

B: ECHONET Lite middleware adapter communication software protocol

Fig. 3.6 Data Communication Between the Equipment and ECHONET Lite Middleware Adapter

3.6.2. Mechanical and physical characteristics of ECHONET Lite middleware adapter communication interfaces

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

(1) Transmission medium

The recommended transmission medium for ECHONET Lite 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 which must be guaranteed shall be 2 m at the maximum.

(3) Connection style

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

(4) Connector shape

It is recommended that an 8-pin PH connector be used on the ECHONET Lite-ready equipment side. These specifications do not define requirements for the connector on the ECHONET Lite 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 Lite 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 Lite 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 Lite-ready devices (MA9/MA9B sockets) and ECHONET Lite 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 Lite Specification defines connectors for power feed capacities of less than 2000 mVA as MA9 connectors and connectors for power feed capacities of 2000 mVA 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.

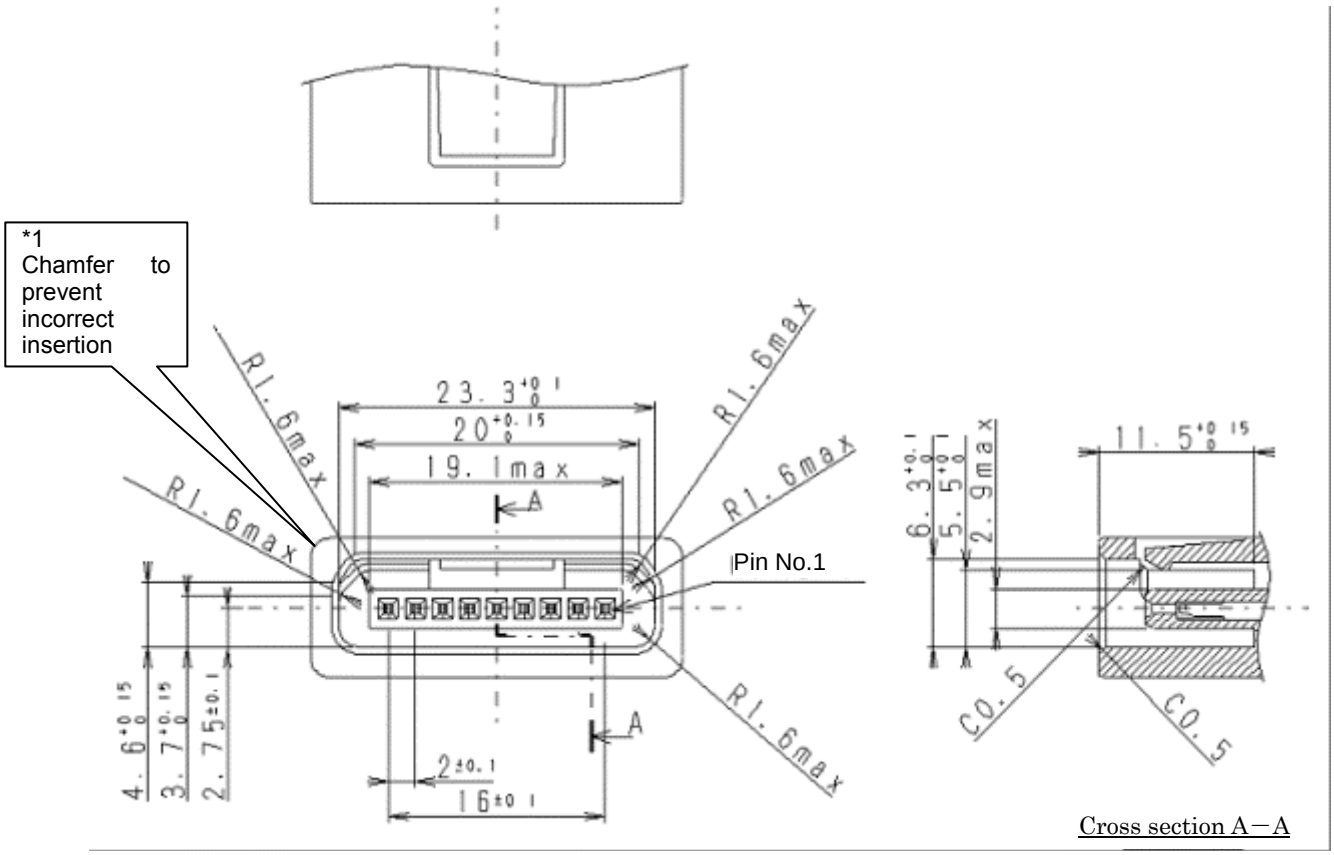
MA9/MA9B Connector Specifications

● 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 Lite Middleware Adapter Specifications. (Details of the WakeUP function are to be determined.)
Connector pin spacing	2 mm	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	0.5 ADC	The rated current is the current for the supply of power to home appliance devices in ECHONET Lite.
Rated voltage	15 VDC	The rated voltage is the voltage for the supply of power to home appliance devices in ECHONET Lite.
Operating temperature range	–20ºC to +85ºC	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	–40ºC to +85ºC	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 Up to two times the initial specification value - Insulation resistance Initial specification value. - Withstand voltage Initial specification value. - External appearance There shall be no crack, deformation or other abnormal condition.	As measured/observed after leaving the connector in an 85±2ºC environment for 500 hours and then in a normal-temperature, normal-humidity environment for 30 minutes.
Low temperature resistance	- Contact resistance Up to two times the initial specification value. - Insulation resistance Initial specification value. - Withstand voltage Initial specification value. - External appearance There shall be no crack, deformation or other abnormal condition.	As measured/observed after leaving the connector in a 40±2ºC environment for 500 hours and then in a normal-temperature, normal-humidity environment for 30 minutes.
Humidity resistance	- Contact resistance Up to two times the initial specification	As measured/observed after leaving the connector in a 60±2ºC, 90 to 95%RH

	value. - Insulation resistance Initial specification value. - Withstand voltage Initial specification value. - External appearance There shall be no crack, deformation or other abnormal condition.	environment for 500 hours and then in a normal-temperature, normal-humidity environment for 30 minutes.
Contact resistance	10 mΩ or less	The inter-pin contact resistances (excluding the conductor resistances) measured with suitable connectors connected. Measurement frequency: 1000 Hz Measurement current: 100 mA or less
Insulation resistance	1000 MΩ or more	As measured after applying a voltage of 500 VDC 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 500 VAC between the conduction points for 1 minute. Breaking current: 2 mA
Insulation	Insulation distance: 2.5 mm or more See *2 in Fig. 3.7(b).	There must be an insulation distance of 2.5 mm or more between the MA9 plug pins (charging section) and the connector opening (outer wall) (supplementary insulation).
Material	RoHS directive-compliant housing UL94 V-0 or higher Must contain PBT glass contact (conduction part) Copper alloy	
Shape	Power Feed Class 1: Power feed capacities of less than 2000 mVA: MA9 socket (ECHONET Lite-ready equipment side) MA9 plug (ECHONET Lite middleware adapter side) Connection Power Feed Class 1: Power feed capacities of 2000 mVA or more: MA9B socket (ECHONET Lite-ready equipment side) MA9B plug (ECHONET Lite middleware adapter side) Connection	 The shape must be as shown in Fig. 3.7(a). The shape must be as shown in Fig. 3.7(b). The shape must be as shown in Fig. 3.7(c). The shape must be as shown in Fig. 3.7(d). The shape must be as shown in Fig. 3.7(e). The shape must be as shown in Fig. 3.7(f).
Mechanism to prevent incorrect insertion	See *1 in Fig. 3.7(a).	A mechanism to prevent users from incorrectly inserting connectors must be provided.
Live insertion and removal function	See *1 in Fig. 3.7(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.	The unlocking force must be 20 to 40 N and it must be possible to confirm that the connector

	See *1 in Fig. 3.7(c).	has been firmly inserted.
Waterproofing	Groove for O-ring See *2 in Fig. 3.7(c).	A groove for installing an O-ring must be provided to make the connector water-proof and splash-proof.



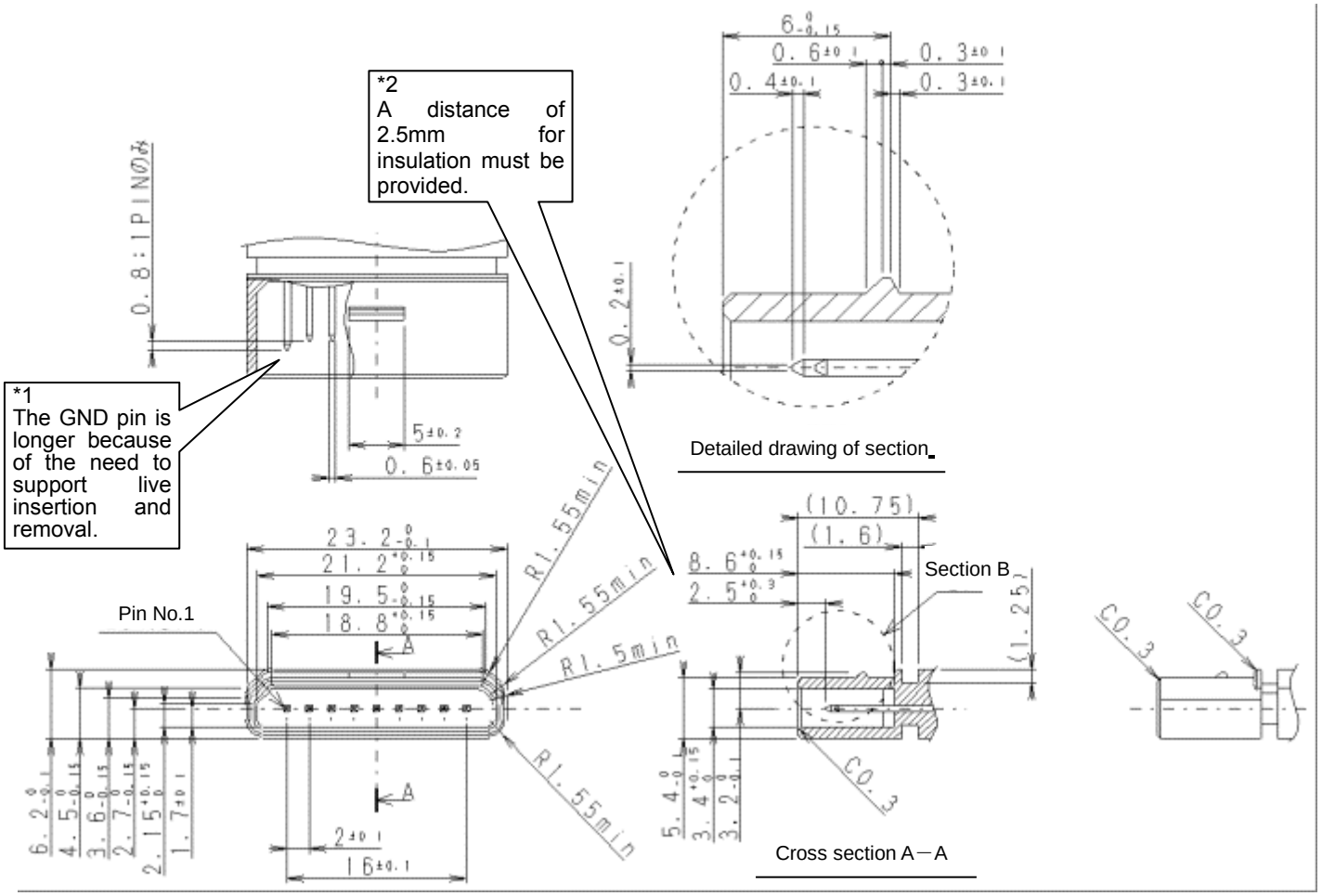


Fig. 3.7(b) MA9 Plug (ECHONET Lite middleware adapter side)

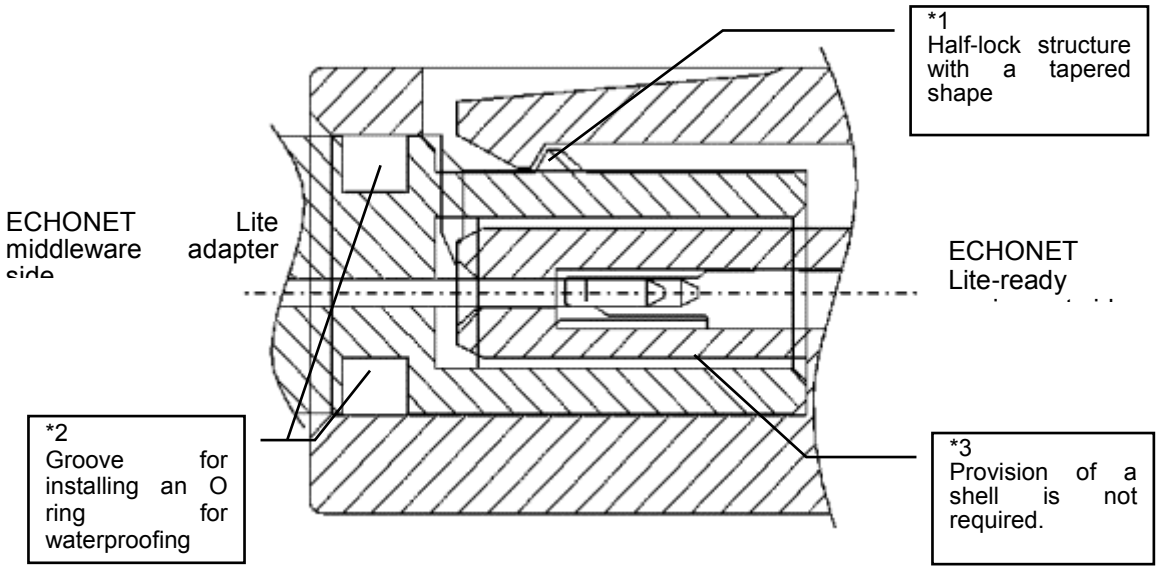
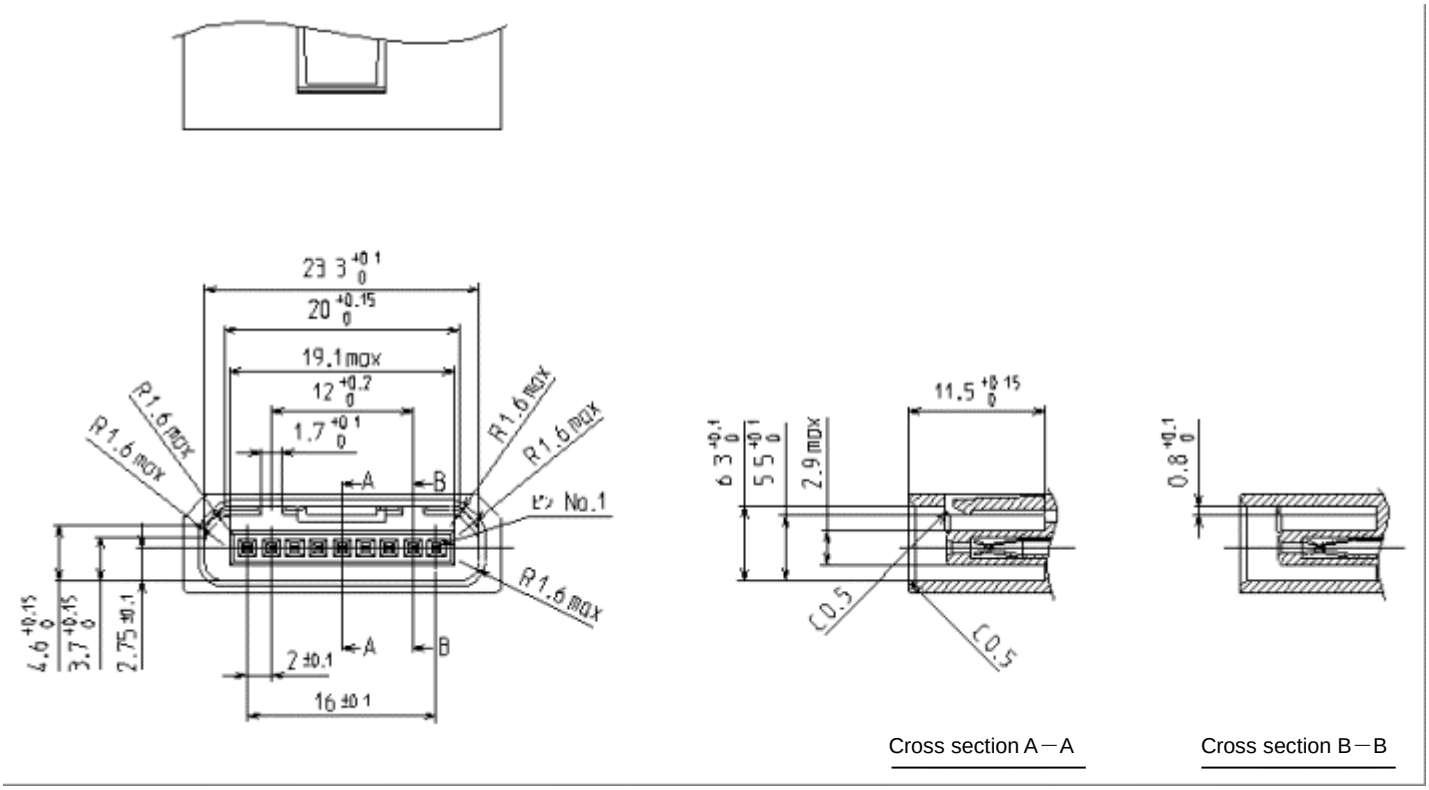


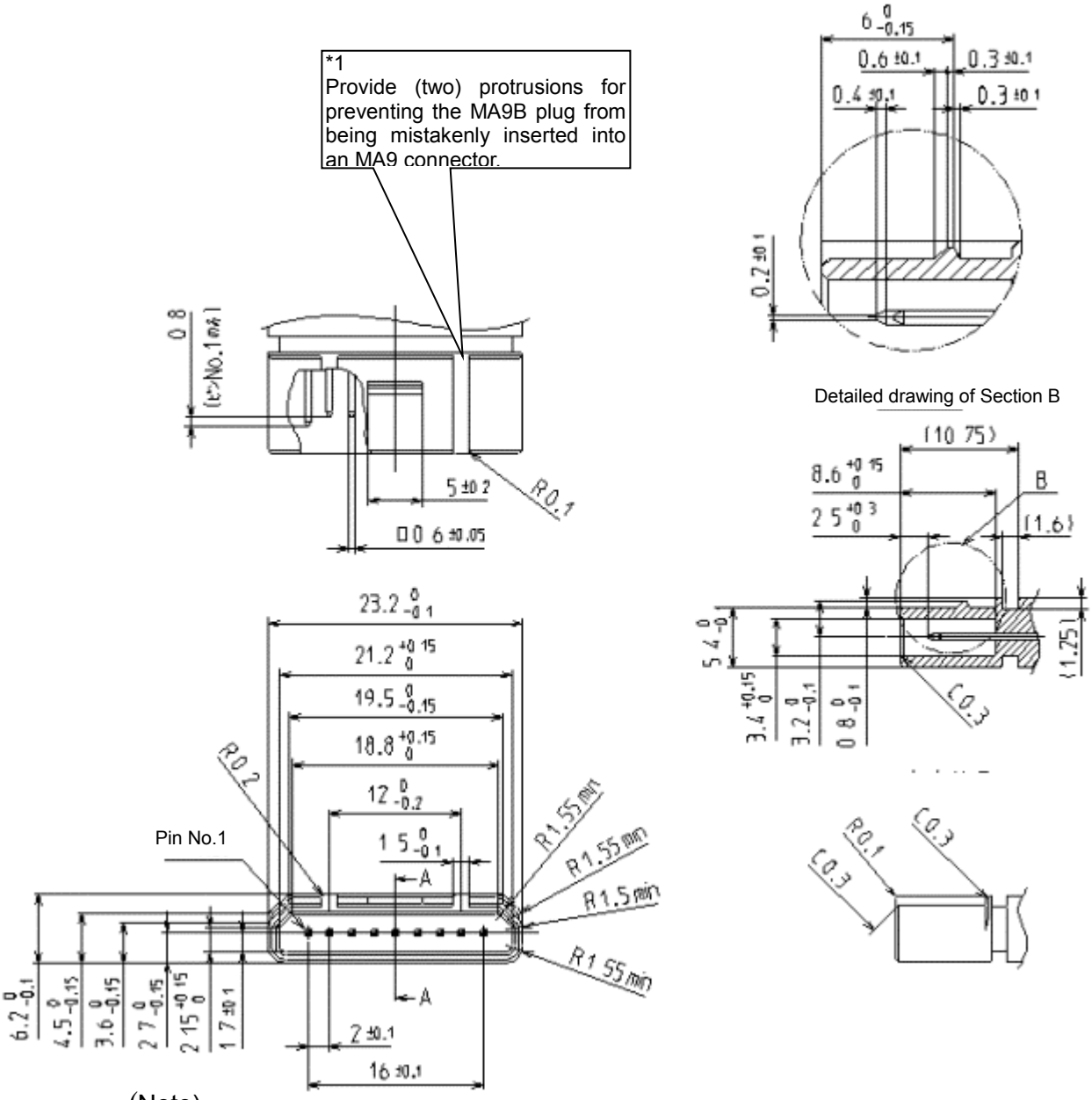
Fig. 3.7(c) MA9 Connector (connection)



(Note)

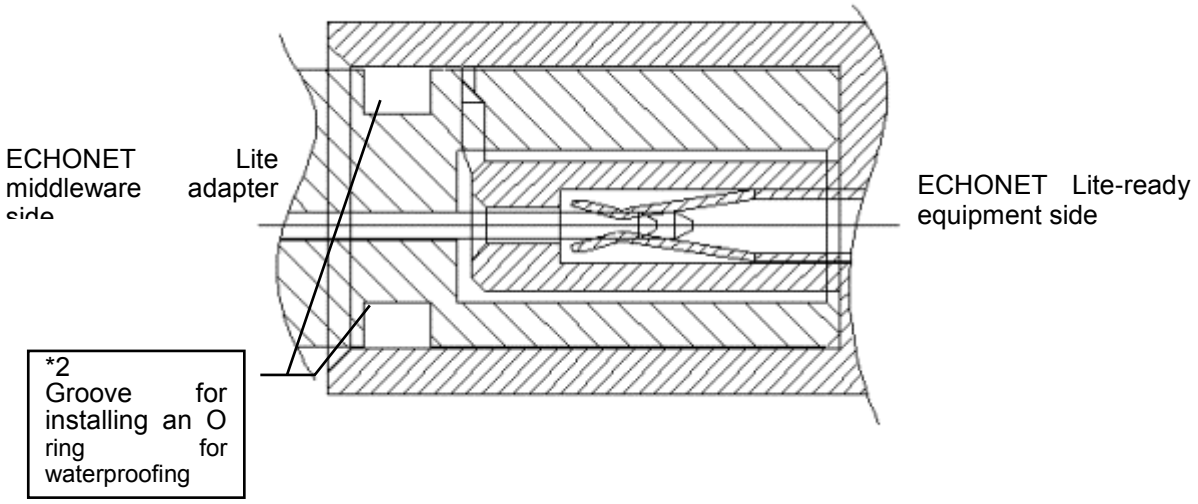
Because the shape of MA9B sockets is the same as that of MA9 sockets except for the (two) concave grooves shown in the “cross section B－B” diagram, it is possible to connect an MA9B socket to an MA9 plug instead of an MA9B plug.

Fig. 3.7(d) MA9B Socket (ECHONET Lite-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. 3.7(e) MA9B Plug (ECHONET Lite middleware adapter side)



(Note)
The drawing of the connection for "cross section A－A" is not provided, because it is the same as Fig. 3.7(c).

Fig. 3.7(f) MA9B Connector (connection, cross section B–B)

(5) Relationship between the connector pins and signals

Fig. 3.8 shows the (recommended) PH connector pin assignment and the (recommended) pin assignment for the case where MA9/MA9B connectors are used.

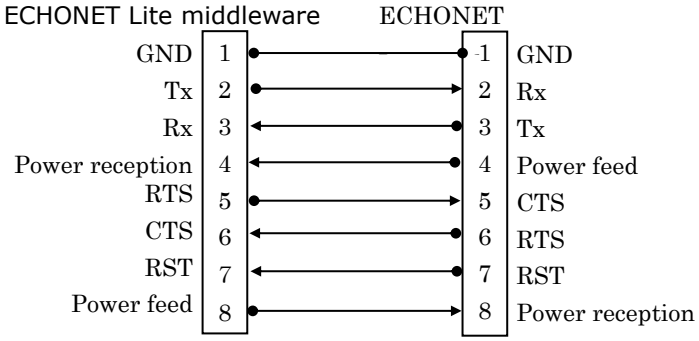


Fig. 3.8(a) ECHONET Lite Middleware Adapter Communication Interface (Recommended) Pin Assignment

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

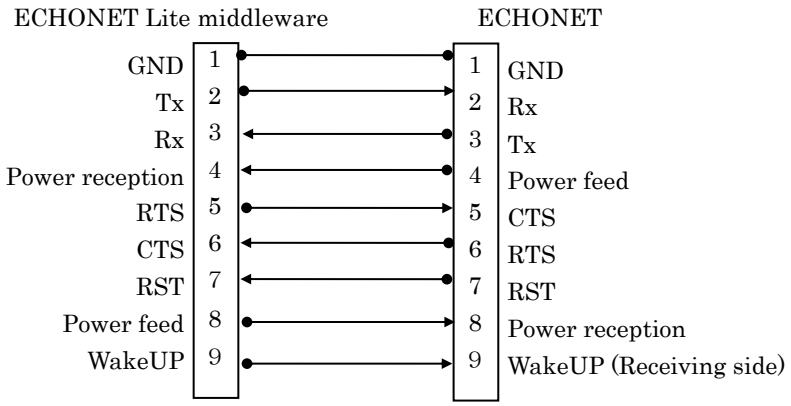


Fig. 3.8(b) ECHONET Lite Middleware Adapter Communication Interface (Recommended) Pin Assignment 2

○ Receiving and Supplying Power (See "3.6.3 (5) Specifications for supplying power.")

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

○ RST (reset)

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

ECHONET Lite-ready equipment: optional
ECHONET Lite middleware adapter: compulsory

○ RTS/CTS (See "3.6.4 (1) Control method.")
ECHONET Lite-ready equipment side: optional
ECHONET Lite middleware adapter side: compulsory

3.6.3. Electrical characteristics

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

(1) Cable characteristic impedance

Not specified.

(2) Signal transmission speed

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

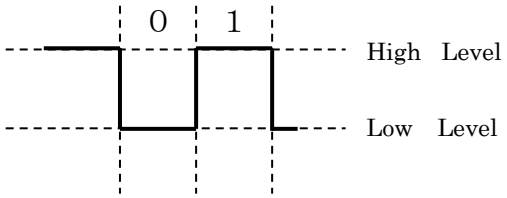


Fig. 3.9 (1) Logic Level

(4) Output pin specifications

Type 1: open collector
Type 2: 3.3V CMOS

(5) Specifications for supplying power

Table 3.2 Specifications for Supplying Power – ECHONET Lite-ready Equipment (Class 1)

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

Table 3.3 Specifications for Supplying Power – ECHONET Lite-ready Equipment (Class 2)

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

Table 3.4 Specifications for Supplying Power – ECHONET Lite-ready Equipment (Class 3)

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

Table 3.5 Specifications for Supplying Power – ECHONET Lite 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 Lite middleware adapter from the ECHONET Lite-ready equipment shall be implemented in the ECHONET Lite middleware adapter. Implementation in the ECHONET Lite-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 Lite middleware adapter” and “reset start,” respectively.

3.6.4. Logical requirements

This section defines the logical requirements for ECHONET Lite 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 Lite 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. 3.10) 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

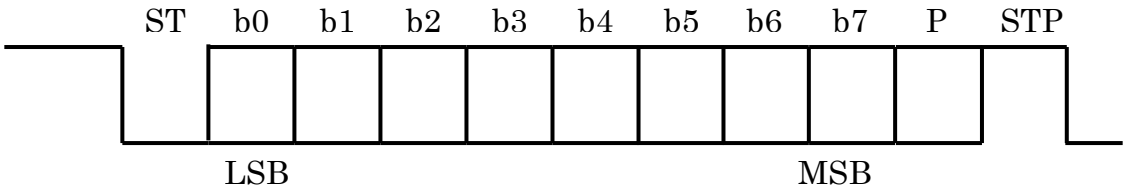


Fig. 3.10 Character Composition

(3) Timing requirements

The timing requirements for execution of the equipment I/F data recognition service shall be as shown in Fig. 3.10 and Table 3.6. The adapter shall send a request frame and the ECHONET Lite-ready equipment shall receive it and send back a response frame.

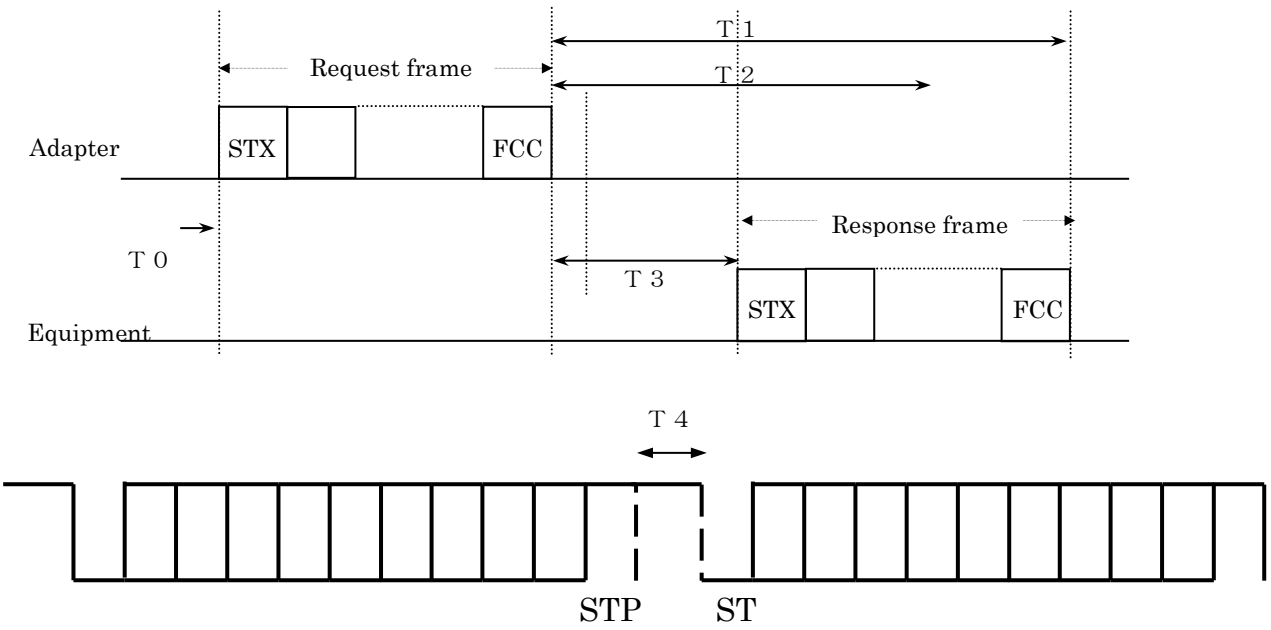


Fig. 3.11 Timing Requirements

Table 3.6 Timing Requirements

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

(4) Frame composition

Each ECHONET Lite 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 “3.6.5 ECHONET Lite 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 Lite middleware adapter communication interface STX and the FCC. The specific content varies depending on what type is selected for the ECHONET Lite middleware adapter communication interface (for details, refer to “3.6.6 ECHONET Lite 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 ms 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.

3.6.5. ECHONET Lite middleware adapter communication software protocols

The ECHONET Lite middleware adapter communication software protocol differs depending on the type of ECHONET Lite 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

3.7. Equipment Interface Data Recognition Service

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

3.7.1. Frame composition for the equipment interface data recognition service

Fig. 3.12 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 Lite middleware adapter communication interface protocol data.

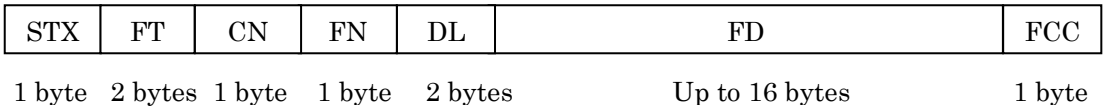


Fig. 3.12 Equipment Interface Data Recognition Service Software Protocol

(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 Lite middleware adapter communication interfaces).

Table 3.7 Equipment Interface Data Recognition Service Command Codes

4 lowest-order bits \ Four 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 Lite-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.

3.7.2. Commands for the equipment interface data recognition service

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

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

These commands are used to allow the ECHONET Lite middleware adapter to acquire information on the type of ECHONET Lite middleware adapter communication interface implemented in the ECHONET Lite-ready equipment.

◇ Direction of request commands

ECHONET Lite middleware adapter → ECHONET Lite-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**

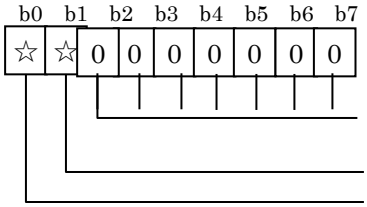
◇ Format for response commands

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

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

DL : n + 2

FD(0) : Middleware adapter type of ECHONET Lite-ready equipment
(0x00 if the function is not implemented)



b2 to b7: Reserved for future use (fixed at 0)
b1: Object generation type (0: not present, 1: present)
b0: Peer-to-peer type implementation (0: not present, 1: present)

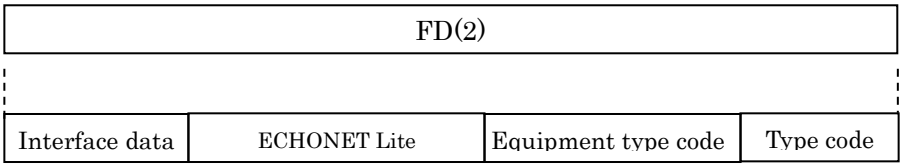
FD(1): transmission speed

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)

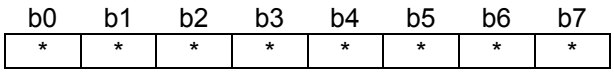
FD(2): Method-specific definition area (n bytes)

– Object generation type: Not present (bytes)

– Peer to peer type



Interface data (1 byte)



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 Lite manufacturer code (3 bytes)

as per the ECHONET Lite 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 Lite middleware adapter shall notify the ECHONET Lite-ready equipment as to whether or not it can handle the communication method specified by the ECHONET Lite-ready equipment using an equipment interface data recognition notification. This notification shall be made within 300 ms (Ti) after the receipt of the equipment interface data. If the ECHONET Lite middleware adapter can handle the communication method specified by the ECHONET Lite-ready equipment, it shall so notify and wait for an equipment interface data recognition notification acceptance response. If the ECHONET Lite middleware adapter cannot handle the communication method specified by the ECHONET Lite-ready equipment, it shall so notify and enter the “connection not possible” state.

Upon receipt of an equipment interface data recognition notification, the ECHONET Lite-ready equipment shall check the content of the notification, and if the value indicates that the ECHONET Lite middleware adapter can handle the communication method specified by the ECHONET Lite-ready equipment, the ECHONET Lite-ready equipment shall send within 300 ms (Ti) an equipment interface data recognition notification acceptance response to the ECHONET Lite middleware adapter and enter the “unconfirmed” state. If the value indicates that the ECHONET Lite middleware adapter cannot handle the communication method specified by the ECHONET Lite-ready equipment, the ECHONET Lite-ready equipment shall enter the state to wait for an equipment interface data request.

If the ECHONET Lite middleware adapter receives an equipment interface data recognition notification acceptance response within 300 ms after the transmission of an equipment interface data recognition notification, it shall enter the “unconfirmed” state.

If the ECHONET Lite middleware adapter does not receive an equipment interface data recognition notification acceptance response within 300 ms after the transmission of an equipment interface data recognition notification, it shall send an equipment interface data request again.

◇ Direction of request commands

ECHONET Lite middleware adapter → ECHONET Lite-ready equipment

◇ Format for request commands

1 byte	2 bytes	1 byte	1 byte	2 bytes	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 in case multiple specified
FCC : 0x**

◇ Format for response commands

1 byte	2 bytes	1 byte	1 byte	2 bytes	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**

3.7.3. Equipment interface data recognition service sequence

Fig. 3.13 shows the operation sequence of the service. Fig. 3.14 shows how the ECHONET Lite middleware adapter and ECHONET Lite-ready equipment change their states. As shown in these figures, both the ECHONET Lite middleware adapter and ECHONET Lite-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 Lite-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 Lite-ready equipment and ECHONET Lite 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. 3.14.

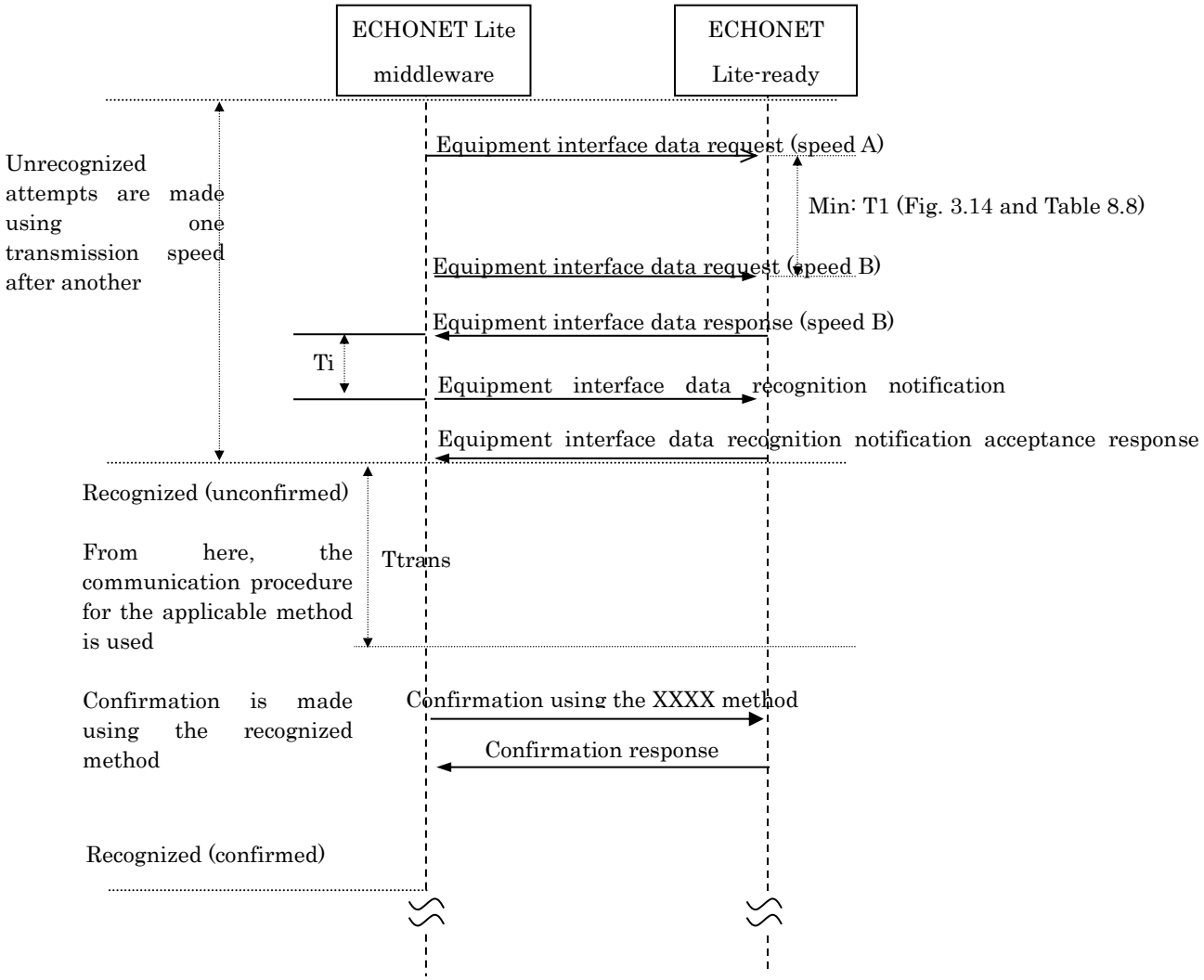


Fig. 3.13 Operation Sequence

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

Ti: Max. 300 ms

If the adapter cannot properly recognize the response, it shall send the recognition service

again after the T1 period (Fig. 3.10) has elapsed.

3.7.4. Status change diagram for all types

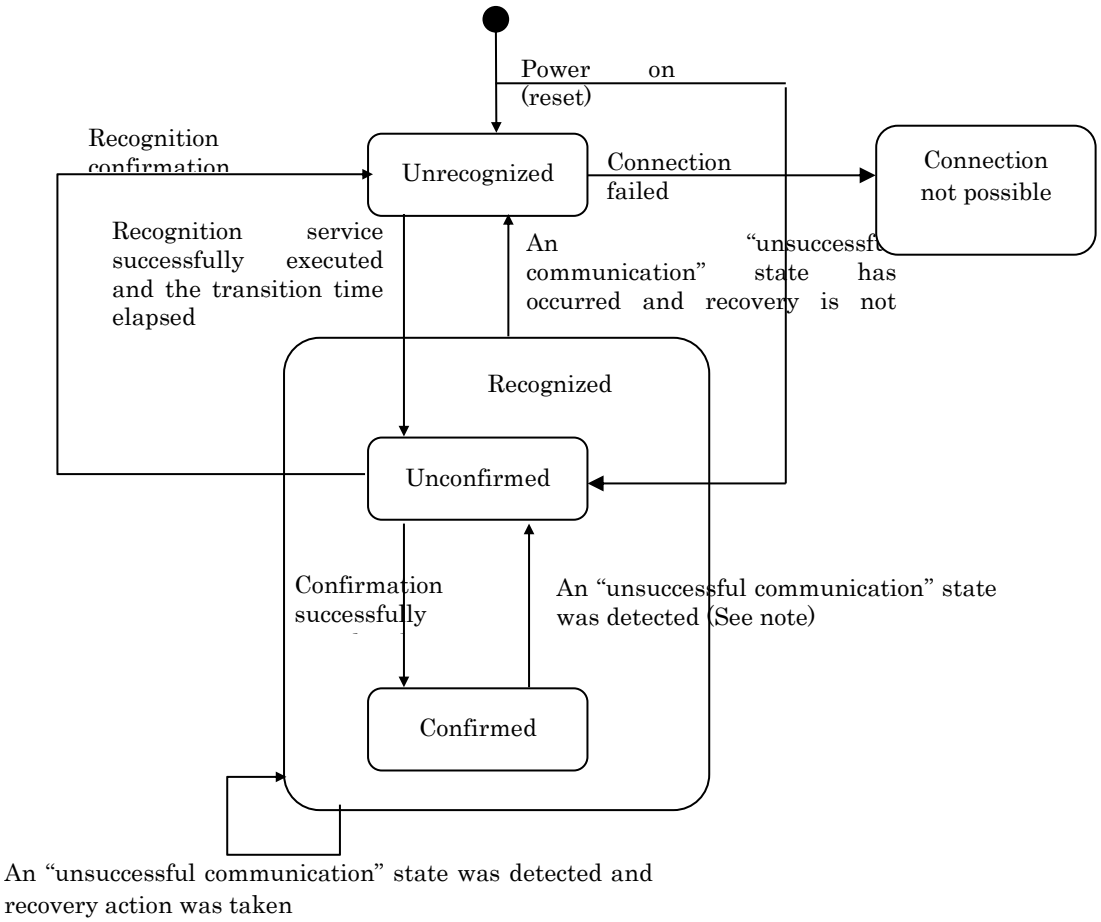


Fig. 3.14 Status Change Diagram

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

(Note) The ECHONET Lite 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 3.8 Definition of States

State	Definition		Notes
Unrecognized	The process of recognizing that the communication method has not been completed After the detection of an "unsuccessful communication" state	ECHONET Lite-ready equipment side: Waiting for a recognition service request from the adapter. If the equipment interface data recognition notification from the ECHONET Lite middleware adapter indicates that the adapter cannot handle the communication method specified by the ECHONET Lite-ready equipment, the ECHONET Lite-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 Lite-ready equipment, the ECHONET Lite-ready equipment will send an equipment interface data recognition notification acceptance response and enter the "unconfirmed" state.	
		ECHONET Lite middleware adapter side: Attempting to make a recognition service request to the ECHONET Lite-ready equipment. 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, ...). 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 Lite-ready equipment to so notify, and after receiving the corresponding acceptance response, the "unconfirmed" state is entered. If communication using the received information is not possible, an equipment interface data recognition notification is sent to the ECHONET Lite-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. 3.15) 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 Lite middleware adapter and the ECHONET Lite-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. ECHONET Lite middleware adapter side: 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.	

3.7.5. Error processing

If a state should occur that falls under any one of (a) to (c) below, the ECHONET Lite middleware adapter shall give the node profile object or device object a setting value indicating the occurrence of such an abnormal state and inform the error to the other nodes. In cases where an LED is provided as an indication means (display section) as described in Section 3.3.2, 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 Lite-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 three 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 Lite-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.

3.8. Communication Protocols for Object Generation Type

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

Middleware adapters are divided into the following two 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 Lite network. A basic middleware adapter is not capable of handling, on the middleware adapter interface, address information on other nodes located in the ECHONET Lite 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 Lite-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 Lite 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 three device objects and managing them. In the case where the number of device objects that the basic middleware adapter can manage is three, it is necessary to set aside at least 1 KB of memory space to store property values.
To allow the basic middleware adapter to generate and manage four or more device objects, it is necessary to use optional commands and set aside at least 2 KB of memory space to store property values.

A basic middleware adapter must implement object classes such as a node profile object, which required in the Part 2 of the ECHONET Lite specification.

3.8.1. Frame composition for object generation type interfaces

Fig. 3.15 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 Lite middleware adapter communication interface protocol data. The composition shown in Fig. 3.15 is the same as that for the equipment interface data recognition service.

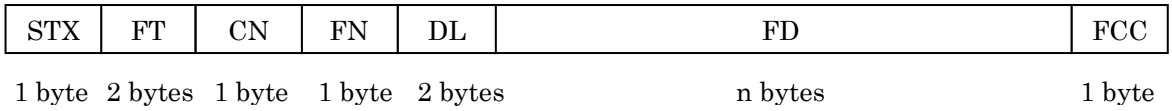


Fig. 3.15 Object Generation Type ECHONET Lite Middleware Adapter Communication Software Protocol

(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 Lite frame
0x04: Advanced regular ECHONET Lite 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 Lite middleware adapter communication interfaces). This version of the ECHONET Lite Specification defines the commands shown in Table 3.9. Codes with no specific function assigned shall be reserved for future use.

Table 3.9 Object Generation Type Interface Command Codes

Frame Type (FT)	Command Number Code (CN)	Command Name	
Equipment interface confirmation mode 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 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 Object construction frame (0x0002) is used	0x00	Equipment inquiry request	Required
	0x80	Equipment inquiry response	Required
	0x01	Equipment inquiry completion notification	Required
	0x81	Equipment inquiry completion notification acceptance response	Required
	0x02	Adapter startup notification	Required
	0x82	Adapter startup notification acceptance response	Required
	0x03	Equipment inquiry request with object specified	Optional*1
	0x83	Equipment inquiry response with object specified	Optional*1
ECHONET Lite communication mode Basic regular ECHONET Lite 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 four 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 Lite-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 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.

3.8.2. 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 section defines internal services for adapters that instruct the adapter on how to handle services received from other nodes such as Set (equipment status alteration) and Get (equipment status confirmation). These internal services are IASet, IASetup, IAGet and IAGetup. When a Set is received, an IASet or IASetup shall be performed. When a Get is received, an IAGet or IAGetup shall be performed.

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

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

3.8.2.1 IASet

IASet 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 Lite-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. Fig. 3.16 shows how this works.

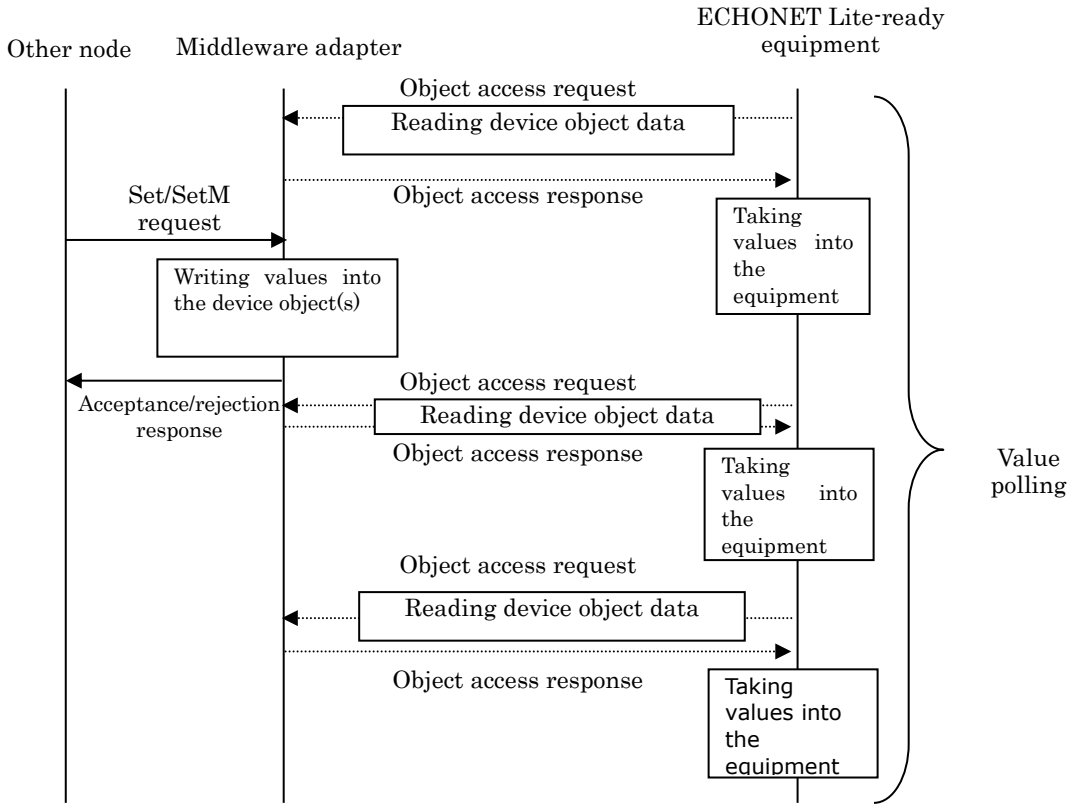


Fig. 3.16 Operation of IASet

3.8.2.2 IASetup

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

The reception (acceptance) response to the other node is made upon reception of a status alteration response sent by the ECHONET Lite-ready equipment in response to the IASetup. Fig. 3.17 shows how this works. If no equipment status access response 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.

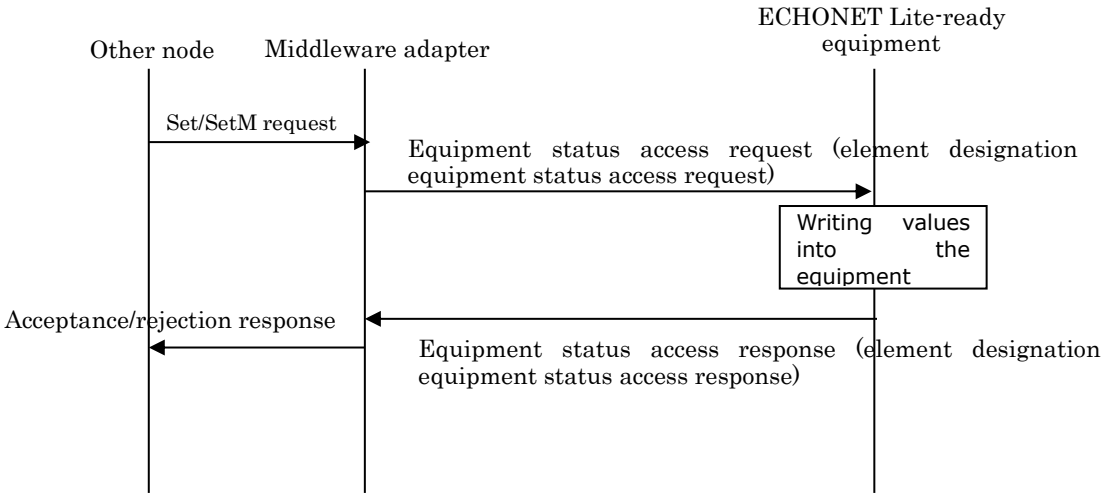


Fig. 3.17 Operation of IASetup

3.8.2.3 IAGet

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

The ECHONET Lite-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.

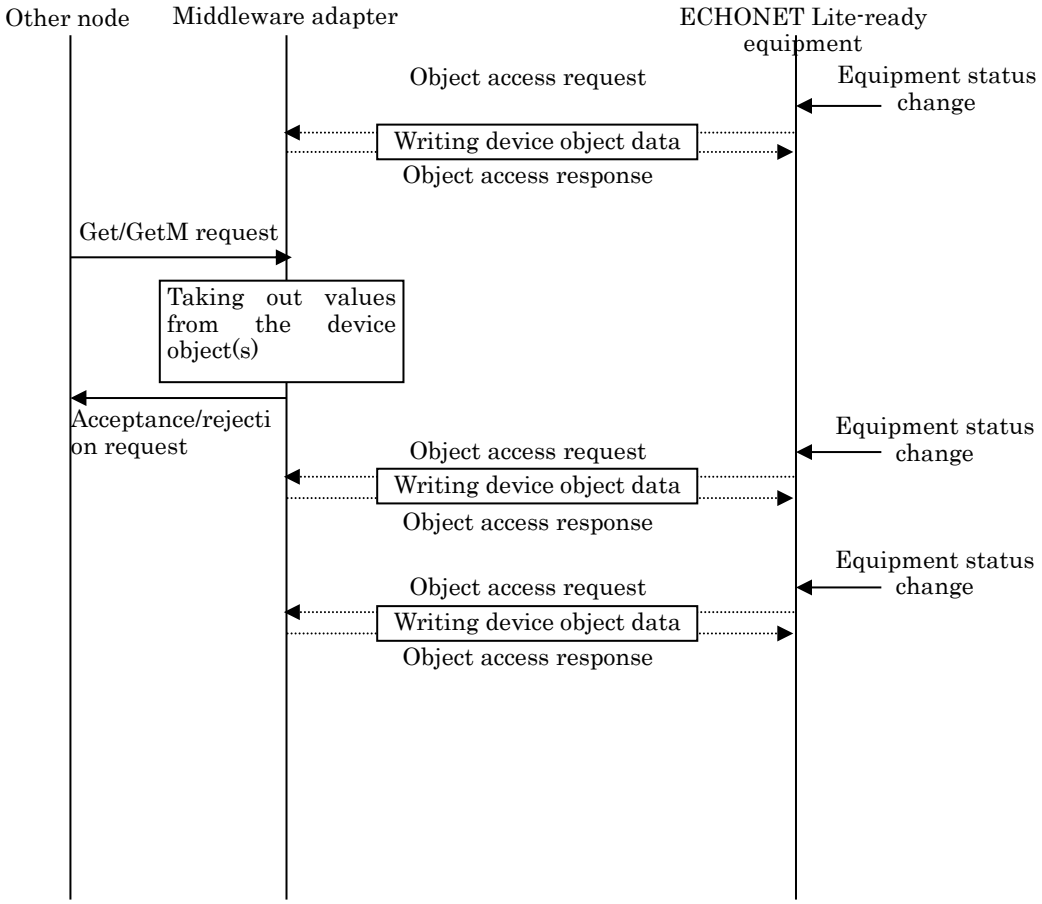


Fig. 3.18 Operation of IAGet

3.8.2.4 IAGetup

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

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

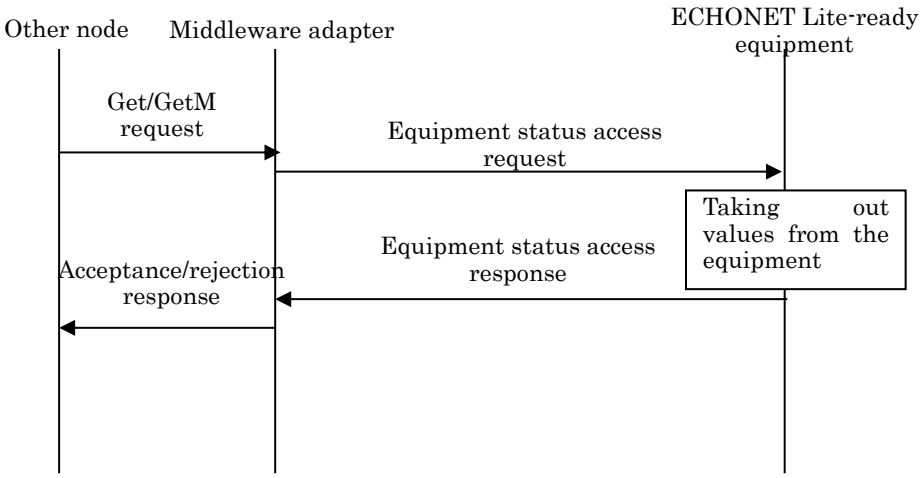


Fig. 3.19 Operation of IAGetup

3.8.2.5 Status change announcement and Internal services of adapters

In the case that the adapter's internal service is IAGet and IAGetup, the simultaneous announce to be notified in the domain must be made upon the time when the property that corresponding bit of status change announcement property map property is set at one bit (set in status change) changes its value and, if the adapter's internal service is IAGet, middleware adapter shall announce simultaneously to be notified in the domain. If the adapter's internal service is IAGetup, ECHONET Lite-ready equipment shall announce simultaneously in the domain.

3.8.3. ECHONET Lite middleware adapter status changes for object generation type interfaces

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

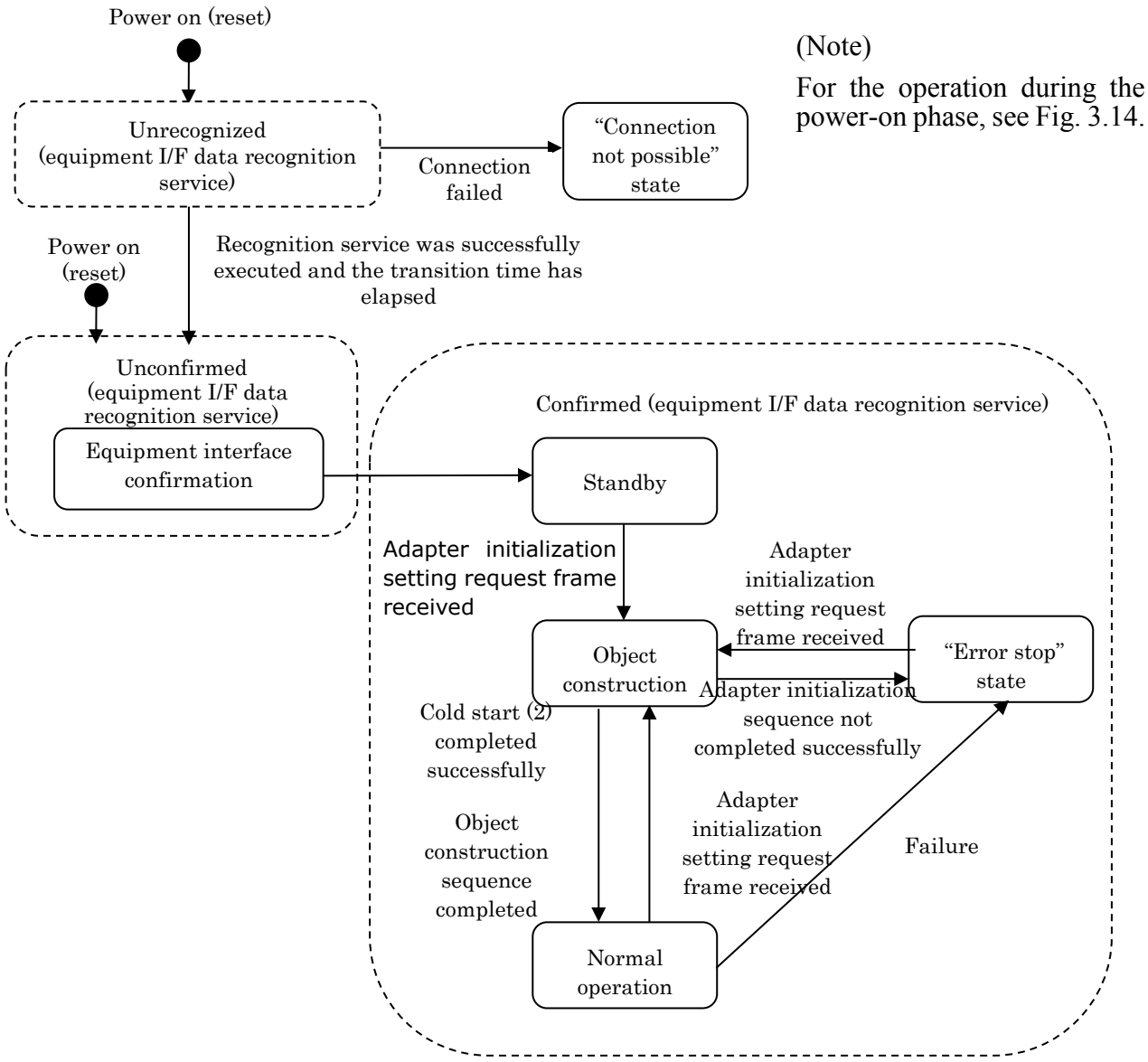


Fig. 3.20 Middleware Adapter Status Changes

3.8.3.1. "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 Lite 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 the specified number of times, a shift is made to the "unrecognized" state (equipment interface data).

3.8.3.2. "Standby" state

The state of waiting for an initialization request frame after completion of the confirmation of the interface state with the ECHONET Lite-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 Lite network are discarded.

3.8.3.3. "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 Lite 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.

3.8.3.4. "Error Stop" state

If the ECHONET Lite 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 Section 3.8.3 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)

3.8.3.5. “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 Lite-ready equipment. Normal ECHONET Lite communication is only possible in this state. During this time, object construction mode-related frames are not accepted.

3.8.4. Commands for object generation type interfaces

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

3.8.4.1. Equipment interface confirmation mode

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

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

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

◇ Direction of request commands

ECHONET Lite middleware adapter → ECHONET Lite-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: 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: Reserved for future use

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

n is the value 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**

3.8.4.2. Adapter initialization mode

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

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

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 Lite-ready equipment → ECHONET Lite 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 start If the basic middleware adapter already contains a device object, the adapter performs with the object. If no device object exists, an equipment inquiry request is made, and the device object generation is performed.
0x0002	Equipment data disposal start If the basic middleware adapter already contains a device object, the adapter performs with the object discarded.
0x0003	Equipment data retention start The same process as 0x0001 is performed for interoperability with ECHONET-ready equipment
0x0004	Equipment data disposal start The same process as 0x0002 is performed for interoperability with ECHONET-ready equipment
0x0005	Equipment data retention start The same process as 0x0001 is performed for interoperability with ECHONET-ready equipment
0x0006	Equipment data disposal start The same process as 0x0002 is performed for interoperability with ECHONET-ready equipment
0x0007 to 0xFFF	Reserved for future use

0x0003 to 0x0006 (compatibility with ECHONET) are start modes used for ECHONET, and are not used for ECHONET Lite. If these values are received, an appropriate process is needed, such as to ignore them.

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
0x02								

FT : 0x0001

CN : 0x81

FN : Value at the time of the request

DL : 0x0002 / 0x000B

FD(0): Response result

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

* See 6.11.1 of Part 2, “Node Identification Number” property of “Node Profile Class: Detailed Specifications.”

0x00: Node identification number has not been set

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

* See 6.11.1 of Part 2, “Node Identification Number” property of “Node Profile Class: Detailed Specifications.” If Node Identification Number is longer than 8 bytes, 0x00 is set.

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 Lite-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 Lite-ready equipment must return an adapter initialization completion notification acceptance response when it receives an adapter initialization completion notification.

◇ Direction of request commands

ECHONET Lite middleware adapter → ECHONET Lite-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

Value of FD(0)	Meaning
0x0000	Notification acceptance
0xFFFF	Other error

FCC: 0x**

3.8.4.3. Object construction mode

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

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

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

The ECHONET Lite-ready equipment that receives an equipment inquiry request returns, in the form of an equipment inquiry 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 three. Information on three pieces of equipment can be sent in a single response or divided into two or three pieces and sent in two or three responses, respectively.

◇ Direction of request commands

ECHONET Lite middleware adapter → ECHONET Lite-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 inquiry response frame

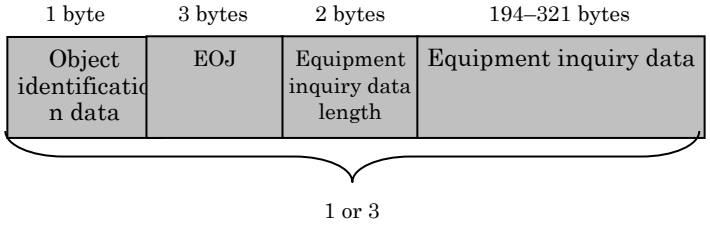
FD(2): Object data

Refer to Object Data Format.

FCC: 0x**

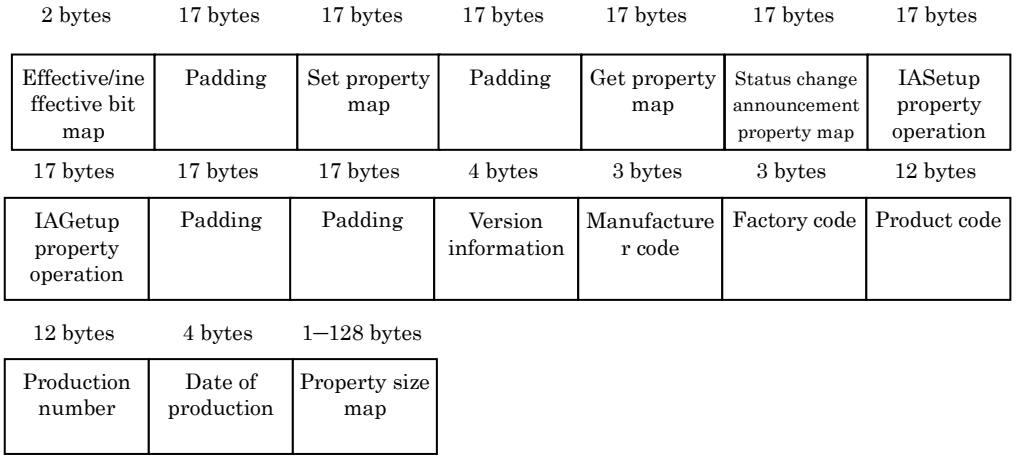
Object Data Format

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). b7 to b4: Total number of objects (1: 0001, 2: 0010, 3: 0011) b3 to b0: Object identification numbers (1: 0001, 2: 0010, 3: 0011)
EOJ	3	ECHONET object code for generated construction method A device object
Equipment inquiry data length	2	Indicates the size of the equipment inquiry data.
Equipment inquiry data	194–321	Refer to Equipment Inquiry Data Format.

Equipment Inquiry Data Format



Name	Size (in bytes)	Explanation
Effective/ineffective bit map	2	Indicates the meaningful part in the data that follows: b15: (Ineffective) b14: Set property map b13: (Ineffective) b12: Get property map b11: Status change announcement property map b10: IASetup property operation map b9: IAGetup property operation map b8: (Ineffective) b7: (Ineffective)

		b6: Version information b5: Manufacturer code b4: Factory code b3: Product code b2: Production number b1: Date of production b0: Property size map The items set to 1 are effective.
(Padding)	17	(Arbitrary values) (for compatibility with ECHONET)
Set property map	17	Indicates the property that accepts Set. The format shall be as per the ECHONET Lite Specification.
(Padding)	17	(Arbitrary values) (for compatibility with ECHONET)
Get property map	17	Indicates the property that accepts Get. The format shall be as per the ECHONET Lite Specification.
Status change announcement property map	17	Indicates the property that performs status change announcements. The format shall be as per the ECHONET Lite 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 Lite-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 Lite-ready equipment side, then the response is returned. The format shall be subject to the same rules as the other property maps.
(Padding)	17	(Arbitrary values) (for compatibility with ECHONET)
(Padding)	17	(Arbitrary values) (for compatibility with ECHONET)
Version information	4	Indicates the specification version used. The format shall be as per the ECHONET Lite Specification.
Manufacturer code	3	Indicates the manufacturer of the ECHONET Lite-ready equipment. The format shall be as per the ECHONET Lite Specification.
Factory code	3	Indicates the vendor-dependent code that indicates the factory that manufactured the ECHONET Lite-ready equipment. The format shall be as per the ECHONET Lite Specification.
Product code	12	Indicates the vendor-dependent ECHONET Lite-ready equipment product code. The format shall be as per the ECHONET Lite Specification.
Production number	12	Indicates the vendor-dependent ECHONET Lite-ready equipment production number. The format shall be as per the ECHONET Lite Specification.
Date of production	4	Indicates the date on which the ECHONET Lite-ready equipment was manufactured. The format shall be as per the ECHONET Lite Specification.
Property size map	1 to 128	Indicates the size of each property in bytes. For the properties that are present, the sizes are listed starting from the one with the smallest EPC.

(Note)

*1: The format shall be as specified in Supplement 1, APPENDIX Detailed Requirements for ECHONET Device objects, “Property map description format.” (The size shall be fixed

at 17 bytes.)

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

The basic middleware adapter sends an equipment inquiry completion notification to the ECHONET Lite-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 inquiry response, “invalid” is returned and a shift is made to the “error stop” state.

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

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

◇ Direction of request commands

ECHONET Lite middleware adapter → ECHONET Lite-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 inquiry 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

FCC: 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 a device object or the process of constructing a device object has been successfully completed, the basic middleware adapter sends an adapter startup notification to the ECHONET Lite-ready equipment.

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

If the construction of a 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 Lite-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 Lite middleware adapter → ECHONET Lite-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 : 0x02

FN : 0x**

DL : 0x0002

FD(0): Initialization processing result

Value of FD(0)	Meaning
0x0000	Startup completed successfully
0x0011	Startup failed

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 : 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 inquiry request with object specified” and “equipment inquiry response with object specified” commands (optional)

If no device object is present inside, the basic middleware adapter will send an “equipment inquiry request with object specified” to the ECHONET Lite-ready equipment. In response to this request, the ECHONET Lite-ready equipment will return device information in the form of an “equipment inquiry response with object specified.”

When the ECHONET Lite-ready equipment receives an “equipment inquiry 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 inquiry response with object specified.” The task of acquiring and managing object information is the responsibility of the ECHONET Lite middleware adapter. The ECHONET Lite-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 Lite middleware adapter through responses to

requests from the ECHONET Lite middleware adapter. The ECHONET Lite middleware adapter identifies and manages objects using these object identification numbers.

(Note) *: Required in the case of an adapter capable of generating four or more device objects or a piece of ECHONET Lite-ready equipment capable of requesting the generation of four or more device objects.

◇ Direction of request commands

ECHONET Lite middleware adapter → ECHONET Lite-ready equipment

◇ Format for request commands

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

STX: 0x02

FT : 0x0002

CN : 0x03

FN : 0x**

DL : 0x0001

FD(0): Object identification number (used in a sequential order starting with "0x01")

FCC: 0x**

Value of FD(0)	Meaning
0x0000	Startup completed successfully
0x0011	Startup failed

◇ Format for response commands

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

STX: 0x02

FT : 0x0002

CN : 0x83

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

DL : 0x****

FD(0): Response result

Name	Size (bytes)	Explanation
Result	2	Response result

		0x0000: Acceptance response 0x0001: Acceptance response (network out of operation) 0x0011: Rejection response 0x0012: The specified device object does not exist. 0x0101: Status discrepancy error (“device interface confirmation” state (“unconfirmed”)) 0x0103: Status discrepancy error (“standby” state) 0x0105: Status discrepancy error (“error stop” state) 0xFFFF: Other error

FD(1): Number of objects sent in the frame

Fixed at "0x01"

FD(2) : Object data

Refer to "Object Data Format."

FCC : 0x**

Object Data Format

2 bytes	3 bytes	2 bytes	194-289 bytes
Object identification information	EOJ	Equipment inquiry data length	Equipment inquiry data

Name	Size (bytes)	Explanation
Object identification information	2	First byte: object identification number Second byte: total number of objects
EOJ	3	ECHONET object code value for the device object to be constructed
Equipment inquiry data length	2	Indicates the size of the equipment inquiry data.
Equipment inquiry data	194 - 289	Refer to “Equipment Inquiry Data Format.”

Equipment Inquiry Data Format

2 bytes	17 bytes	17 bytes	17 bytes	17 bytes	17 bytes	17 bytes
Effective/ineffective bit map	(Padding)	Set property map	(Padding)	Get property map	Status change announcement property map	IASetup property operation map

17 bytes	17 bytes	17 bytes	4 bytes	3 bytes	3 bytes	12 bytes
IAGetup property operation map	(Padding)	(Padding)	Version information	Manufacturer code	Factory code	Product code

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

Name	Size (bytes)	Explanation
Effective/ineffective bit map	2	Indicates the meaningful ones of the following data: b15: (ineffective) b14: Set property map b13: (ineffective) b12: Get property map b11: Status change announcement property map b10: IASetup property operation map b9: IAGetup property operation map b8: (ineffective) b7: (ineffective) b6: Version information b5: Manufacturer code b4: Factory code b3: Product code b2: Production number b1: Date of manufacture b0: Property size map The items for which "1" is specified are effective.
(Padding)	17	(Arbitrary values) (For compatibility with ECHONET)
Set property map	17	Indicates the properties that accept Set.*1
(Padding)	17	(Arbitrary values) (For compatibility with ECHONET)
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
(Padding)	17	(Arbitrary values) (For compatibility with ECHONET)
(Padding)	17	(Arbitrary values) (For compatibility with ECHONET)
Version information	4	Indicates the specification version used. The format shall be as per the ECHONET Lite Specification.
Manufacturer code	3	Indicates the manufacturer of the ECHONET Lite-ready equipment. The format shall be as per the ECHONET Lite Specification.
Factory code	3	A vendor-dependent code indicating the factory that manufactured the ECHONET Lite-ready equipment. The format shall be as per the ECHONET Lite Specification.
Product code	12	A vendor-dependent ECHONET Lite-ready equipment product code. The format shall be as per the ECHONET Lite Specification.
Production number	12	A vendor-dependent ECHONET Lite-ready equipment production number. The format shall be as per the ECHONET Lite Specification.
Date of production	4	Indicates the date on which the ECHONET Lite-ready equipment was manufactured. The format shall be as per the ECHONET Lite Specification.
Property size map	1 to 128	Indicates, in bytes, the size of each property of the specified device objects. 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.)

3.8.4.4. ECHONET Lite communication mode

(1) Equipment status access request and response commands (Required)

If a Set is performed for a property whose internal service is IASetup, or if a Get is performed for a property whose internal service is IAGetup, the basic middleware adapter reports the Set or Get information to the ECHONET Lite-ready equipment by means of an equipment status access request.

Upon reception of the equipment status access request, the ECHONET Lite-ready equipment returns to the basic middleware adapter a corresponding response (when it is IASetup) or the status value that corresponds to the specified property (when it is IAGetup) (equipment status access response).

◇ Direction of request commands

ECHONET Lite middleware adapter → ECHONET Lite-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 Lite-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 Lite-ready equipment (referenced or altered)
Result	2	Response result 0x0000: Acceptance response 0x0011: Rejection response 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 Lite-ready equipment, and, if the adapter's internal service for the target property for notification is IAGet, 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, it shall return an equipment status notification response to the ECHONET Lite-ready equipment and announce the value to be notified in the domain.

◇ Direction of request commands

ECHONET Lite-ready equipment → ECHONET Lite 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 : 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 Lite-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)

FC: 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 0x0000: Acceptance response 0x0011: Rejection response (network not operating) 0x0012: Rejection response (other) 0x0101: Status discrepancy error (“equipment interface confirmation” state (“unconfirmed”)) 0x0102: status discrepancy error (“external equipment data confirmation (reference)” state) 0x0103: status discrepancy error (“standby” state) 0x0104: status discrepancy error (“object construction” state) 0x0105: status discrepancy error (“error stop” state) 0xFFFF: other error

FD(1): FD(1) EOJ for the ECHONET Lite-ready equipment (reported)

FCC: 0x**

(3) Element designation equipment status access request and response commands

This command is used by the ECHONET specification, and is not prescribed in this specification.

(4) Element designation equipment status notification request and response commands

This command is used by the ECHONET specification, and is not prescribed in this specification.

(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 Lite-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, an object access response shall be returned to the ECHONET Lite-ready equipment with the “Result” set to “rejection response.”

◇ Direction of request commands

ECHONET Lite-ready equipment → ECHONET Lite 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 Lite-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 Lite-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 0x0000: Acceptance response 0x0001: Acceptance response (network not operating) 0x0011: Rejection response 0x0101: Status discrepancy error (“equipment interface confirmation” state (“unconfirmed”)) 0x0102: Status discrepancy error (“external equipment data confirmation (reference)” state) 0x0103: Status discrepancy error (“standby” state) 0x0104: Status discrepancy error (“object construction” state) 0x0105: Status discrepancy error (“error stop” state) 0xFFFF: Other error

FD(1): Object information

Name	Size (in bytes)	Explanation
EOJ	3	EOJ for the ECHONET Lite-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 Lite middleware adapter to read values from and set values in the ECHONET Lite-ready equipment for all properties for which specific services are specified. Specifically, these commands allow the ECHONET Lite middleware adapter to:

- Read the values of properties stored in the ECHONET Lite-ready equipment for all properties for which the IAGet services are specified.
- Read the values of properties stored in the ECHONET Lite middleware adapter for all properties for which the IASet services are specified.
- Notify the equipment of the values of properties stored in the ECHONET Lite middleware adapter for all properties for which the IAGet services are specified.
- Notify the equipment of the values of properties stored in the ECHONET Lite middleware adapter for all properties for which the IASet services are specified.

When the ECHONET Lite-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 Lite-ready equipment → ECHONET Lite middleware adapter

◇ Format for request commands

1 byte	2 bytes	1 byte	1 byte	2 bytes	1 byte	3 bytes	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 services are specified

0x01: Notification request for the properties for which the IAGet services are specified

0x02: Read request for the properties for which the IASet services are specified

0x03: Notification request for the properties for which the IASet services are specified

0x02–0xFF: Reserved for future use

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 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 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 the above)	
Other than above	Reserved for future use	

FD(1): This is present when FD(0) of the request message is "0x00" or "0x02"

(i.e. a read request) and FD(0) of the response message is "0x0000" ("request accepted").

When FD(0) of the request message is 0x00:

Lists the property values of the properties held by the ECHONET

Lite-ready equipment as the IAGet 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) of the request message is 0x02:

Lists the property values of the properties held by the ECHONET Lite-ready equipment as the IASet 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 Lite middleware adapter is equipped with a function that enables it to handle messages which contain plural properties, these commands allow the ECHONET Lite middleware adapter to read values from and set values in the ECHONET Lite-ready equipment for all properties specified in a message. Specifically, these commands allow the ECHONET Lite middleware adapter to:

- Read the values of the target properties stored in the ECHONET Lite-ready equipment in response to a read request from another node for the properties for which the IAGetup services are specified.
- Notify the ECHONET Lite-ready equipment of the requested values in response to a write request from another node for the properties for which the IASetup services are specified.

When the ECHONET Lite-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 plural properties specified in the message using the equipment status access request command.

◇ Direction of request commands

ECHONET Lite-ready equipment → ECHONET Lite middleware adapter

◇ Format for request commands

1 byte	2 bytes	1 byte	1 byte	2 bytes	1 byte	3 bytes	2 bytes	1-256 bytes	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 services are specified

0x01: Write request for the values of the properties for which the IASetup services are specified

0x02 to 0xFF: Reserved for future use

FD(1): Object code (EOJ)

FD(2): Two bytes. The first byte shall be fixed at “0x00” and the second byte shall indicate the number of EPC code values.

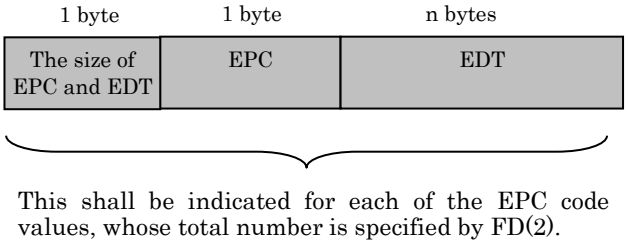
FD(3): The composition differs between whether FD(0) is 0x00 or 0x01:

When FD(0) 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 0x01:

The following shall be indicated for each of the EPC code values, whose total number is specified by FD(2):



FCC: 0x**

◇ Format for response commands

1 byte	2 bytes	1 byte	1 byte	2 bytes	2 bytes	2 bytes	n bytes	1 byte
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

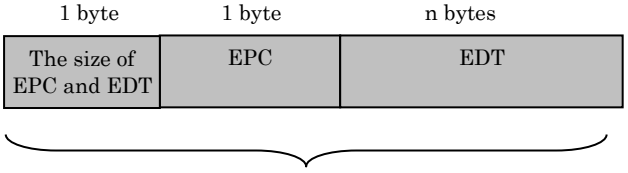
FD(0) value	Content	Remark
0x0000	Request accepted (normal)	
0x0011	Request rejected (The object specified in the request does not exist.)	
0xFFFF	Request rejected (Reasons other than the above)	
Other than above	Reserved for future use	

FD(1): This is present when FD(0) of the response message is “0x0000” (“request accepted”).

Two bytes. The first byte shall be fixed at “0x00” and the second byte shall indicate the number of EPC code values.

FD(2): This is present when FD(0) of the response message is “0x0000” (“request accepted”) and FD(0) of the request message is “0x00.”

The following shall be indicated for each of the EPC code values, whose total number is specified by FD(2):



FCC: 0x**

(8) “Equipment status notification request (all)” and “equipment status notification response (all)” commands (optional)

These commands allow the ECHONET Lite-ready equipment to notify values to the ECHONET Lite middleware adapter for all properties for which specific services are specified. Specifically, these commands allow the ECHONET Lite-ready equipment to:

- Request the notification, throughout the ECHONET Lite, of the values of all properties for which the IAGet services are specified.
- Request the notification, throughout the ECHONET Lite, of the values of all properties for which the IASet services are specified.
- Request the notification, throughout the ECHONET Lite, of the values of all properties for which the IAGetup services are specified.
- Request the notification, throughout the ECHONET Lite, of the values of all properties for which the IASetup services are specified.

When the basic middleware adapter receives an equipment status notification request (all)

from the ECHONET Lite-ready equipment, it shall return an equipment status notification response (all) to the ECHONET Lite-ready equipment and announce in the domain the target properties for notification.

For the properties for which IAGet and IASet are specified, the ECHONET Lite middleware adapter shall overwrite the values it contains with the received values.

◇ Direction of request command

ECHONET Lite-ready equipment → ECHONET Lite middleware adapter

◇ Format for request commands

1 byte	2 bytes	1 byte	1 byte	2 bytes	1 byte	3 bytes	n bytes	1 byte	n bytes	1 byte
STX	FT	CN	FN	DL	FD(0)	FD(1)	FD(2)	FD(3)	FD(4)	FCC

STX: 0x02

FT : x0003

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 services are specified

0x01: Notification request for the properties for which the IASet services are specified

0x02: Notification request for the properties for which the IAGetup services are specified

0x03: Notification request for the properties for which the IASetup services are specified

0x04–0xFF: Reserved for future use

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 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

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 : 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: 0x0000: Acceptance response 0x0011: Rejection response (“network out of operation”) 0x0012: Rejection response (others) 0x0101: Status discrepancy error (“device interface confirmation” state (“unconfirmed”)) 0x0103: Status discrepancy error (“standby” state) 0x0104: Status discrepancy error (“object construction” state) 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 Lite-ready equipment to read values from and set values in the ECHONET Lite middleware adapter for all properties for which specific services are specified. Specifically, these commands allow the ECHONET Lite-ready equipment to:

- Read the values of the properties stored in the ECHONET Lite middleware adapter for which the IASet services are specified.
- Write the values of properties in the ECHONET Lite middleware adapter for all properties for which the IAGet services are specified.

When the ECHONET Lite 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 Lite-ready equipment in the form of an “object access response (all)”.

◇ Direction of request command

ECHONET Lite-ready equipment → ECHONET Lite middleware adapter

◇ Format for request commands

1 byte	2 bytes	1 byte	1 byte	2 bytes	1 byte	3 bytes	n bytes	1 byte
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 services are specified

0x01: Property value write request for the properties for which the IAGet services are specified

0x02–0xFF: Reserved for future use

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 Lite-ready equipment as the IAGet 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	n bytes	1 byte
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:

		0x0000: Acceptance response 0x0001: Acceptance response (“network out of operation”) 0x0011: Rejection response 0x0101: Status discrepancy error (“device interface confirmation” state (“unconfirmed”)) 0x0103: Status discrepancy error (“standby” state) 0x0104: Status discrepancy error (“object construction” state) 0x0105: Status discrepancy error (“error stop” state) 0xFFFF: Other error

FD(1): This is present when FD(0) of the response message is “0x0000” or “0x0001” (i.e. an acceptance response) and FD(0) of the request message is “0x00.”

Lists the property values of the properties held by the ECHONET Lite-ready equipment as the IASet 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 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**

3.8.4.5. 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 Lite-ready equipment, a communication error notification frame must be sent as response data.

◇ Direction of request commands

ECHONET Lite middleware adapter → ECHONET Lite-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	Error 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 the above

FN : 0x**

DL : 0x0000

FCC : 0x**

3.8.5. Communication sequences (Object generation type)

This section defines the communication sequences for object generation type methods used for ECHONET Lite middleware adapter communication interfaces. A detailed sequence-by-sequence explanation is given below.

Table 3.10 Communication Sequences (Object Generation Type)

No.	Sequence Name	Description	Note
1	Data confirmation sequence	Sequence allowing the ECHONET Lite middleware adapter to confirm the interface method and objects with the ECHONET Lite-ready equipment	
2	Initialization sequence	Sequence allowing the ECHONET Lite-ready equipment to perform an initialization for the ECHONET Lite middleware adapter	
3	Object construction mode operation sequence	Sequence allowing the ECHONET Lite middleware adapter to acquire object information from the ECHONET Lite-ready equipment and generate internal objects	
4	Equipment status access request sequence	Sequence allowing the ECHONET Lite middleware adapter to report a Set/Get request (where there is a property in the ECHONET Lite-ready equipment)	
5	Equipment status notification request sequence	Sequence allowing the ECHONET Lite-ready equipment to write values and send notifications to the outside (where there is a property in the ECHONET Lite middleware adapter)	
6	Element designation equipment status access request sequence	Sequence allowing the ECHONET Lite middleware adapter to report a Set/Get request (where there is a property in the ECHONET Lite-ready equipment) (element designation)	
7	Element designation equipment status notification request sequence	Sequence allowing the ECHONET Lite-ready equipment to write values and send notifications to the outside (where there is a property in the ECHONET Lite middleware adapter) (element designation)	
8	Object access request sequence	Sequence allowing the ECHONET Lite-ready equipment to access properties held by the ECHONET Lite middleware adapter	

3.8.5.1. Equipment interface confirmation mode

(1) Data confirmation sequences (Required)

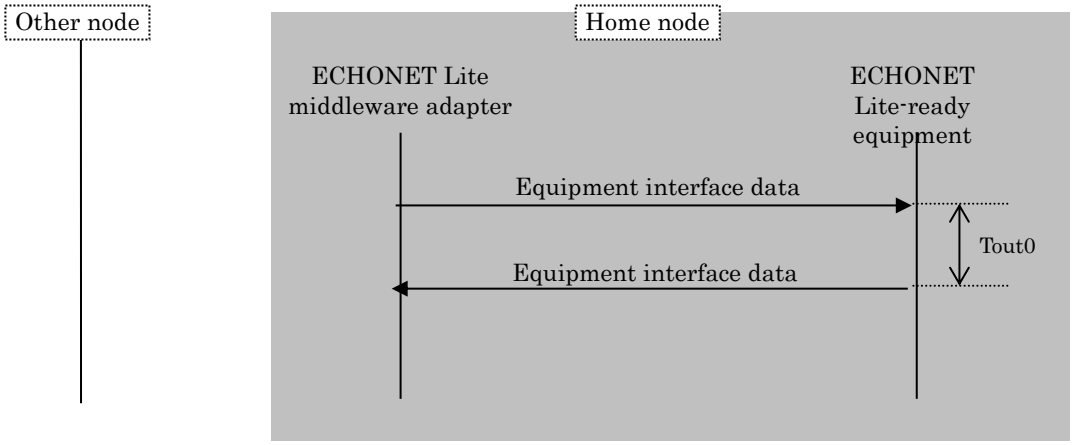


Fig. 3.21 Data Confirmation Sequence

[Equipment Interface Data Confirmation Sequence for the Equipment Side]

The ECHONET Lite-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 Lite-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 Lite-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 three times, the ECHONET Lite-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 Lite-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 Lite-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 Lite-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 three times, the ECHONET Lite-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 Lite-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 Lite-ready equipment to confirm the objects and communication method for the equipment adapter interface maintained. If no object exists, an inquiry 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.

3.8.5.2. Adapter initialization mode

(1) Initialization sequence (required)

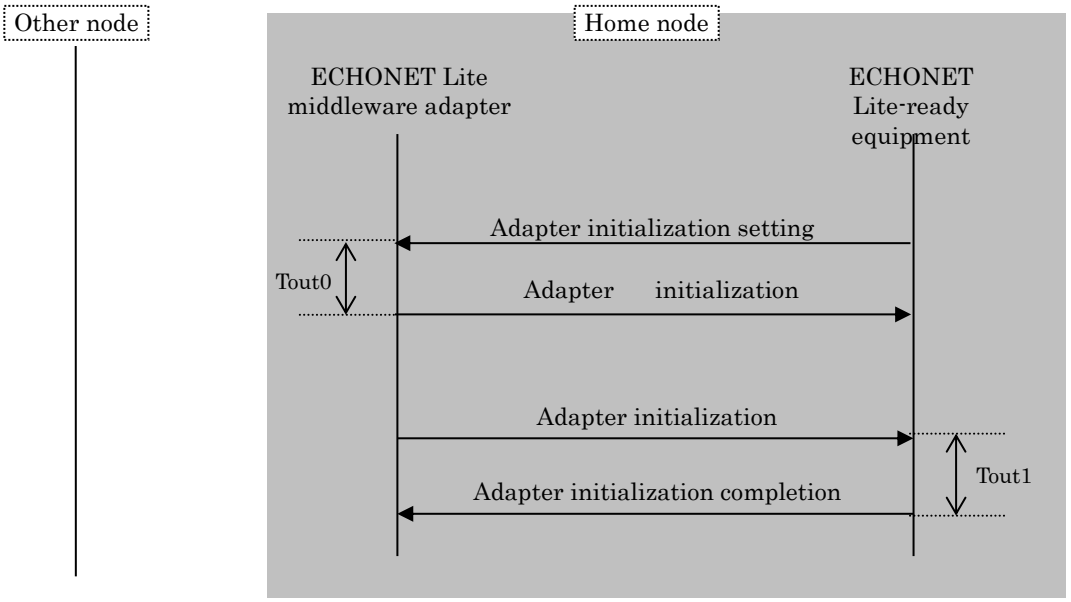


Fig. 3.22 Initialization Sequence

[Initialization Sequence for the Equipment Side]

After a shift to the “object construction” state, the ECHONET Lite-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 re-sent, but only once. If no initialization acceptance response comes in after this retransmission, the ECHONET Lite-ready equipment starts operating in the stand-alone mode.

After receiving an initialization acceptance response, the ECHONET Lite-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 Lite-ready equipment terminates the initialization sequence and starts operating in the stand-alone mode.

If an initialization setting completion notification comes in, the ECHONET Lite-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 Lite-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 re-sent, 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.

3.8.5.3. Object construction mode

Fig. 3.23 shows the sequence for the object construction mode for the case where the device objects are retained, and Fig. 3.24 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 a device object

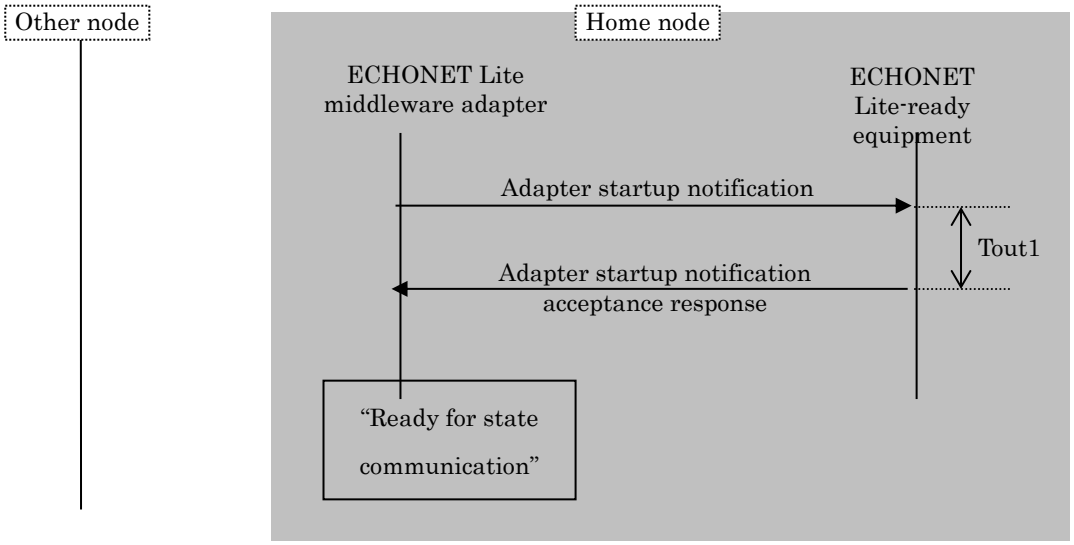


Fig. 3.23 Object Operation Sequence (1)

[Object construction sequence for the adapter side]

The adapter shall first send an adapter startup notification to the ECHONET Lite-ready equipment. The adapter shall then confirm that an adapter startup notification acceptance response from the ECHONET Lite-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

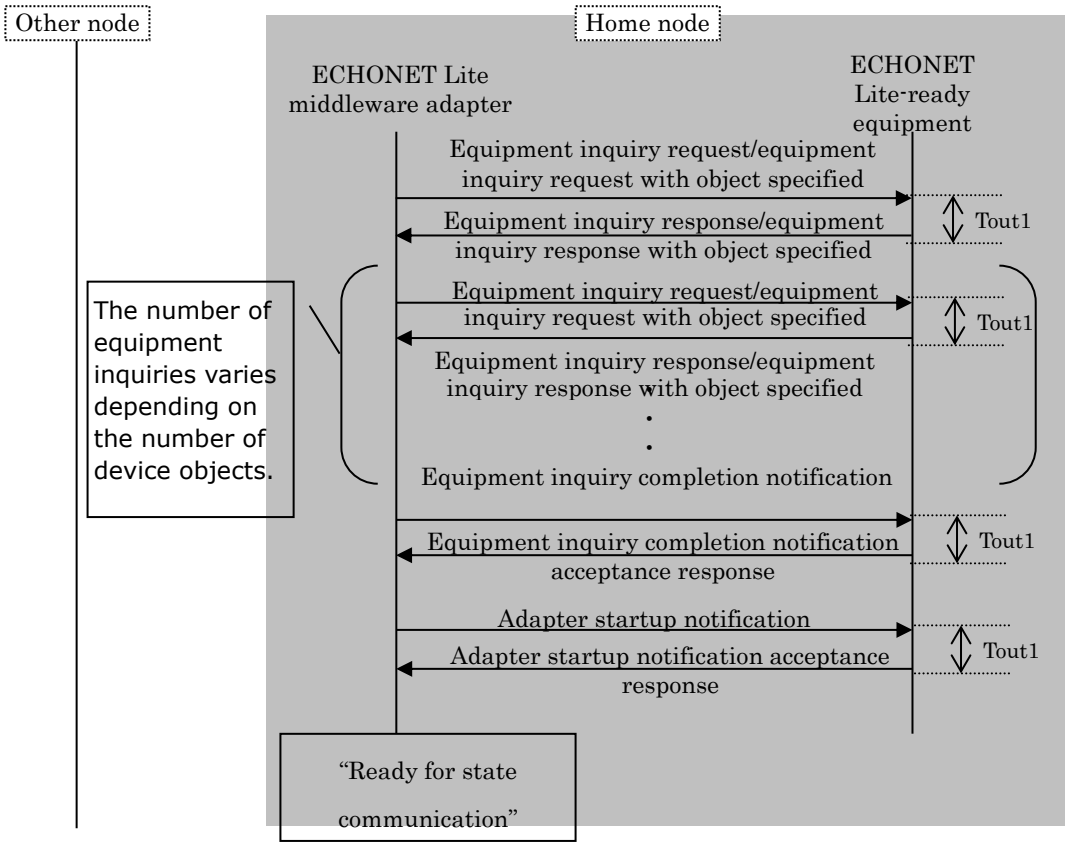


Fig. 3.24 Object Operation Sequence (2)

[Object construction sequence for the adapter side]

The adapter shall first send an equipment inquiry request (equipment inquiry request with object specified) to the ECHONET Lite-ready equipment. The adapter shall then confirm that an equipment inquiry response (equipment inquiry response with object specified) from the ECHONET Lite-ready equipment has been received within the Tout1 period after the transmission of the request. If there are two 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 inquiry completion notification to the ECHONET Lite-ready equipment. The adapter shall then confirm that an equipment inquiry completion notification acceptance response from the ECHONET Lite-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 Lite-ready equipment. The adapter shall then confirm that an adapter startup notification acceptance response from the ECHONET Lite-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 inquiry request (equipment inquiry request with object specified), it shall send an equipment inquiry response (equipment inquiry response with object specified) within the Tout1 period. If the number of implemented pieces of object data is two or more, the processing shall be repeated. If the equipment receives an equipment inquiry completion notification, it shall send an equipment inquiry 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.

3.8.5.4. ECHONET Lite communication mode

There are three command sequences for the ECHONET Lite communication mode: “equipment status access request” (Fig. 3.25), “equipment status notification request” (Fig. 3.26) and “object access request” (Fig. 3.27).

(1) Equipment status access request sequence (Required)

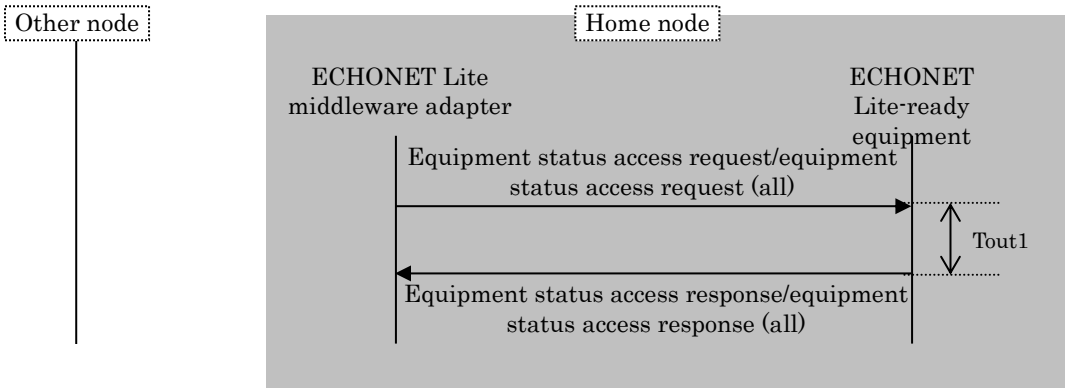


Fig. 3.25 Equipment Status Access Request Sequence

[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 Lite-ready equipment. The adapter shall then confirm that an equipment status access response (equipment status access response (all)) from the ECHONET Lite-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)

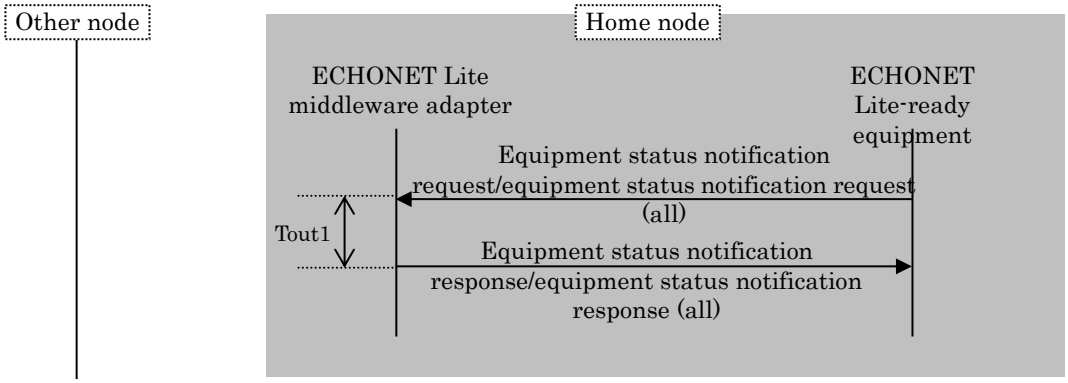


Fig. 3.26 Equipment Status Notification Request Sequence

[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 Lite-ready equipment, it shall send an equipment status notification response (equipment status notification response (all)) to the ECHONET Lite-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) Object access request sequence (Required)

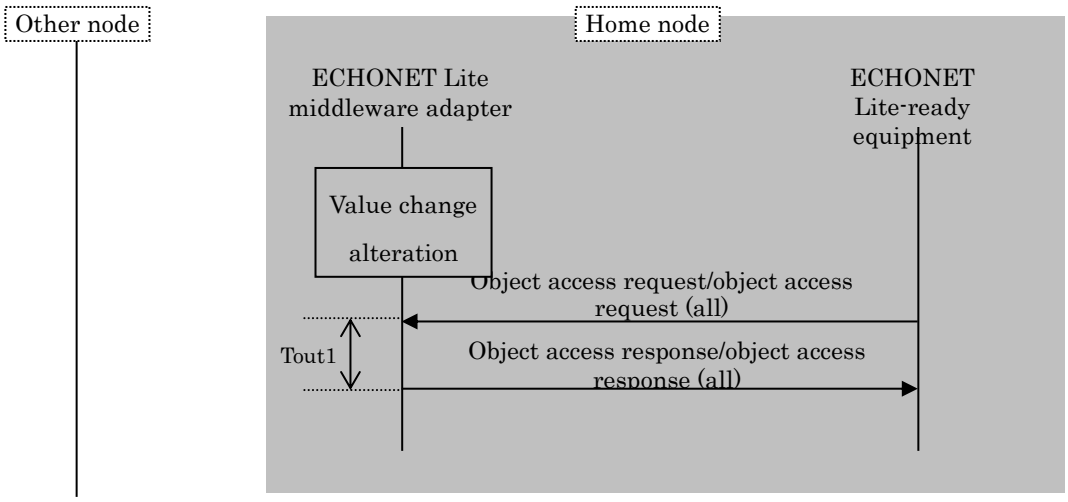


Fig. 3.27 Object Access Request Sequence

[Object access request sequence for the adapter side]

If the adapter receives an object access request (object access request (all)) from the ECHONET Lite-ready equipment, it shall send an object access response (object access response (all)) to the ECHONET Lite-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.

3.8.5.5. Communication error processing sequence for the equipment side

If a communication error occurs while receiving a frame from the middleware adapter, the ECHONET Lite-ready equipment discards that frame and sends a communication error notification frame.

If the ECHONET Lite-ready equipment receives a communication error frame from the middleware adapter as a response to a command that the ECHONET Lite-ready equipment sent, the ECHONET Lite-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.

3.8.5.6. Communication error processing sequence for the adapter side

If a communication error occurs while receiving a frame from the ECHONET Lite-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 Lite-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 Section 3.3.3.

3.8.5.7. Number of frames that can be sent simultaneously

In the case of a basic middleware adapter, the ECHONET Lite middleware adapter and ECHONET Lite-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.

3.8.5.8. Handling of messages which contain plural properties

When a basic middleware adapter equipped with a function that enables it to handle messages which contain plural properties receives a composite Get request message from another node in the ECHONET Lite, the following processing shall be performed:

(1) If the properties specified in the 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 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 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 messages which contain plural properties receives a composite Set request message from another node in the ECHONET Lite, the following processing shall be performed:

(1) If the properties specified in the 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 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.

3.8.5.9. Timeout

The timeout values shall be as follows:

Table 3.11 Timeout Values

Symbol	Name	Timeout Value	Explanation
Tout0, Tout1	Maximum response time 1	3 sec.	The maximum time the middleware adapter or ECHONET Lite-ready equipment can take to send back a response after receiving a request from the ECHONET Lite-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 Lite-ready equipment)	3 sec.	The maximum time the ECHONET Lite-ready equipment can take to send back an equipment interface data confirmation response to the middleware adapter.
Tout40	Minimum object data	6 sec.	The maximum time the ECHONET Lite-ready equipment can

	disposal waiting time		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 Lite-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

3.8.6. 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 Lite-ready equipment to the adapter:

Low = “operation stop” state

Low → High = reset start

ECHONET Lite-ready equipment: Compulsory

Middleware adapter: Compulsory

○ Transmission speeds: 2400 bps, 9600 bps, 19200 bps, 28800 bps, 38400 bps, 57600 bps, 115200 bps

3.9. Communication Protocols for the “Peer-to-Peer Type”

Communication protocols for the “Peer-to-Peer Type” allow an ECHONET Lite middleware adapter and a piece of ECHONET Lite-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

3.9.1. Program Selection Method

The Program Selection Method is a method in which a peer-to-peer type communication method that supports ECHONET Lite-ready equipment (ECHONET Lite middleware

adapter communication software) is implemented in advance in ECHONET Lite middleware adapters. This ECHONET Lite Specification does not specify requirements for the Program Selection Method.

3.9.2. Program Download Method

In the Program Download Method, a peer-to-peer type communication method that supports ECHONET Lite-ready equipment (ECHONET Lite middleware adapter communication software ("the program")) is obtained from outside by means of downloading. This version of the ECHONET Lite Specification specifies the requirements for protocols to download the program from ECHONET Lite-ready equipment. This version of the ECHONET Lite Specification also defines the (recommended) specifications for program execution environments to execute the downloaded program in the ECHONET Lite middleware adapter (see Fig. 3.28).

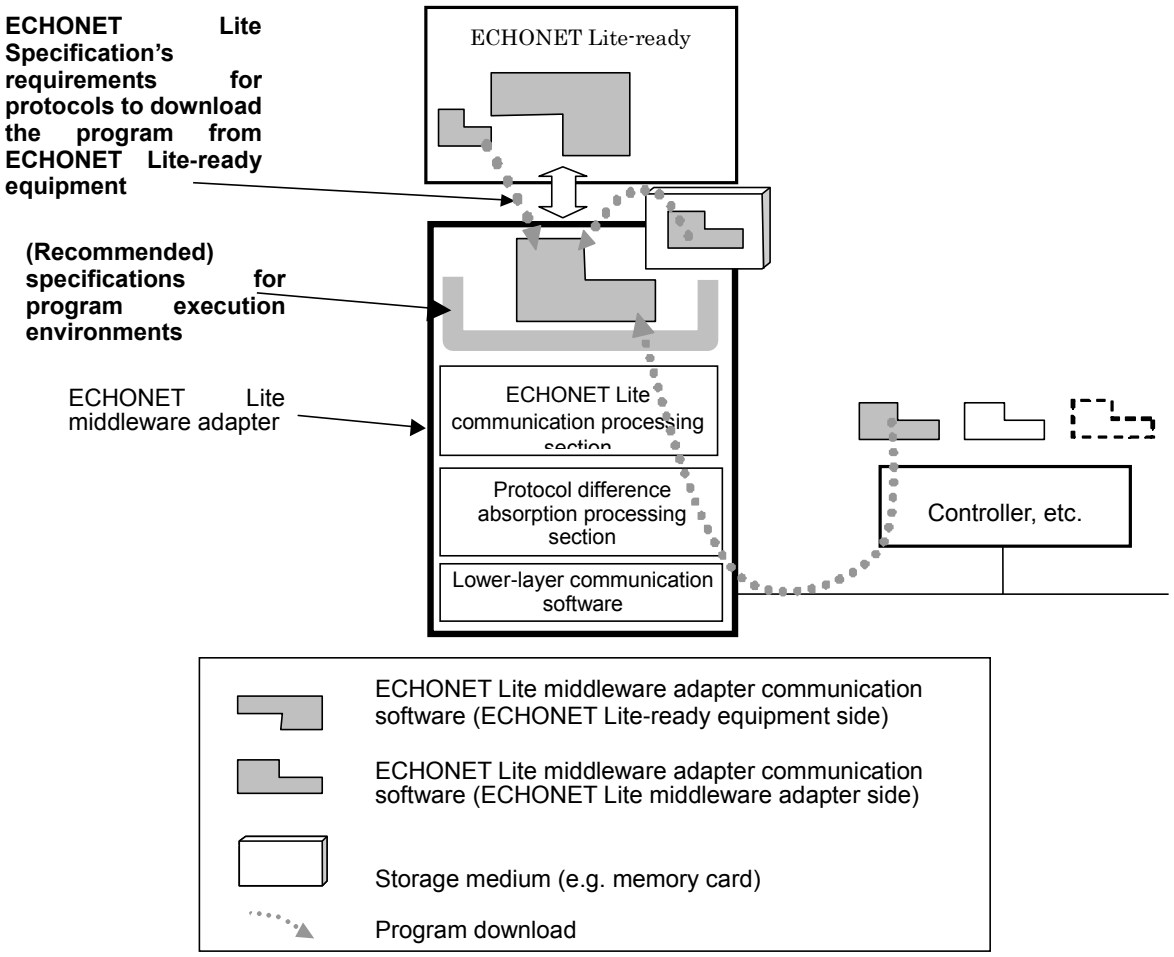


Fig. 3.28 Scope of the ECHONET Lite Specification with Regard to the Program Download Method

3.9.3. Protocols to download the program from ECHONET Lite-ready equipment

This section specifies the requirements for protocols to download the program from ECHONET Lite-ready equipment via the ECHONET Lite middleware adapter communication interface.

3.9.3.1. 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. 3.29. 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 Lite middleware adapter transmission interface protocol.

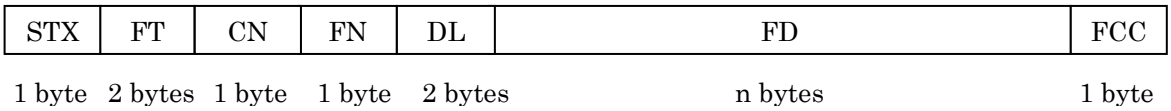


Fig. 3.29 Composition of the Frame for the Program Download Method

(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 Lite 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 Lite middleware adapter communication interface. In communications based on this version of the ECHONET Lite Specification, the commands listed in Table 3.12 shall be used. The code values to which no command is assigned are reserved for future use.

Table 3.12 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 the same number as the corresponding request frame.

In the case of ECHONET Lite-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 order shall be big-endian.

(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 two or more bytes of data, the data order shall be big-endian. 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."

3.9.3.2. Program download commands

This section specifies the requirements for the commands for the program download protocol for the ECHONET Lite middleware adapter communication interface.

(1) Program download request / program download response commands (Required)

◇ Direction of request commands

ECHONET Lite middleware adapter → ECHONET Lite-ready equipment

◇ Format for request commands

1 byte	2 bytes	1 byte	1 byte	2 bytes	2 bytes	7 bytes	9 bytes	1 byte
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):

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):

manufacturer code (3 bytes)

Program identifier (6 bytes)

* If a download has already been performed and the ECHONET Lite middleware adapter already contains the program, the above shall be filled with that data.

* If no download has been performed and the ECHONET Lite middleware adapter does not contain the program, all of the above shall be filled with "0xff."

FCC: 0x**

◇ Response command

a) When FD(0) of the request command is "0x00" (Request for acquisition of the data to transmit)

1 byte	2 bytes	1 byte	1 byte	2 bytes	1 byte	2 bytes	9 bytes	2 bytes	1 byte
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 Lite 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 Lite manufacturer code (3 bytes)

Program identifier (6 bytes)

* "Program identifier" is an identifier that allows the program to be uniquely

identified at the ECHONET Lite-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 FD(0) of the request command is other than "0x00" (Split data sequence number)

1 byte	2 bytes	1 byte	1 byte	2 bytes	2 bytes	Up to 128	1 byte
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 Lite middleware adapter → ECHONET Lite-ready equipment

◇ Notification command

1 byte	2 bytes	1 byte	1 byte	2 bytes	1 byte	1 byte
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

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

STX : 0x02
FT : 0x0100
CN : 0x81
FN : 0x**
DL : 0x0000
FCC : 0x**

3.9.3.3. Download data format specifications

Fig. 3.30 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.

1 byte	Max 65534 bytes (= 65535 - 1)
Program execution environment identifier value	Main body of the program

Fig. 3.30 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 Lite Specification, the program execution environment identifier value specified in Table 3.13 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 3.13 Program Execution Environment Identifier Code

Execution environment name	Program execution environment identifier value
Interpreter Method (as defined in Section 3.10)	0x00

3.9.3.4. Program download communication sequence

This section specifies the requirements for the communication sequence for the program

download protocol for the ECHONET Lite middleware adapter communication interface.

Fig. 3.31 shows the communication sequence for the program download protocol. If the communication method is “Peer-to-Peer” (i.e. b0 of FD(0) of the equipment interface data response command is 1) and the ECHONET Lite-ready equipment is to retain the downloaded data (i.e. b3 (interface information) of FD(2) of the equipment interface data response command is 1), the ECHONET Lite 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.

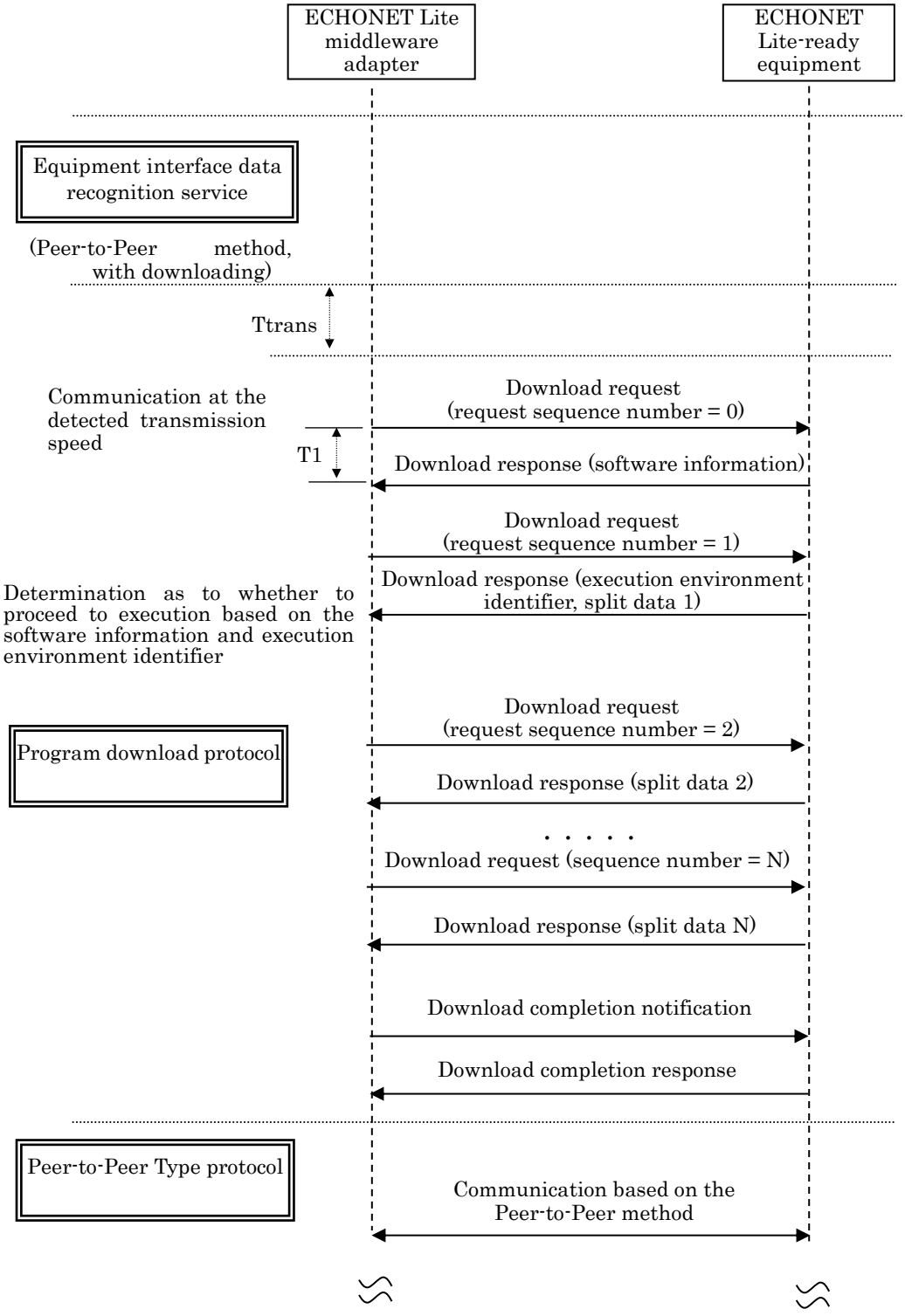


Fig. 3.31 Program Download Communication Sequence

Ttrans : The period of time for the transition to the relevant method. This period must be 500ms or longer.

Ti : The period of time to wait for a response. This period must be 300ms or shorter.

3.9.3.5. 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 Lite Specification does not specify requirements for the method of indicating errors, but, in the case where LEDs are used to indicate errors, it is recommended that the requirements specified in Part 7, "3.3.2 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 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.

3.10. (Recommended) Specifications for Interpreter Method-based Program Execution Environments for the "Program Download Method" for the "Peer-to-Peer Type"

3.10.1. 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 Lite 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 four 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

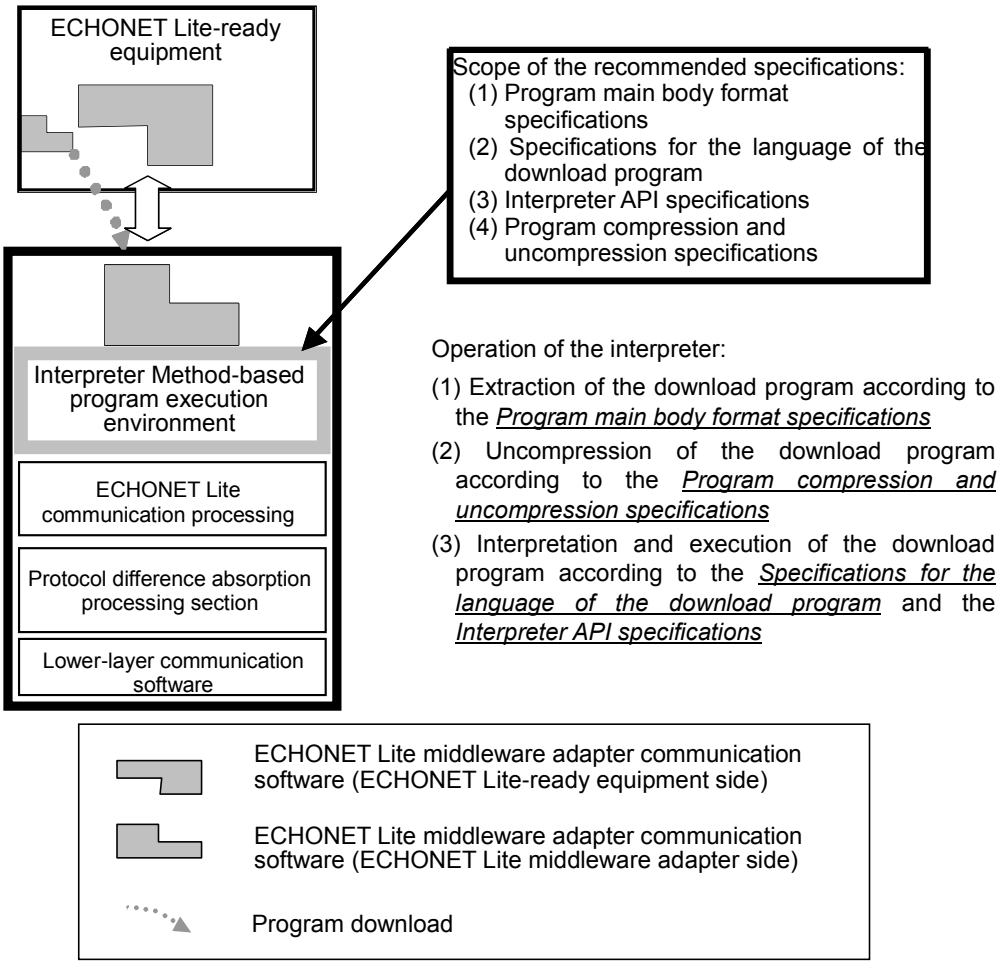


Fig. 3.32 Scope of the Recommended Specifications

3.10.2. Overview of Interpreter Method-based program execution environments

3.10.2.1. Roles of the download program and the interpreter

Fig. 3.33 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 Lite 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 Lite communication processing section API.

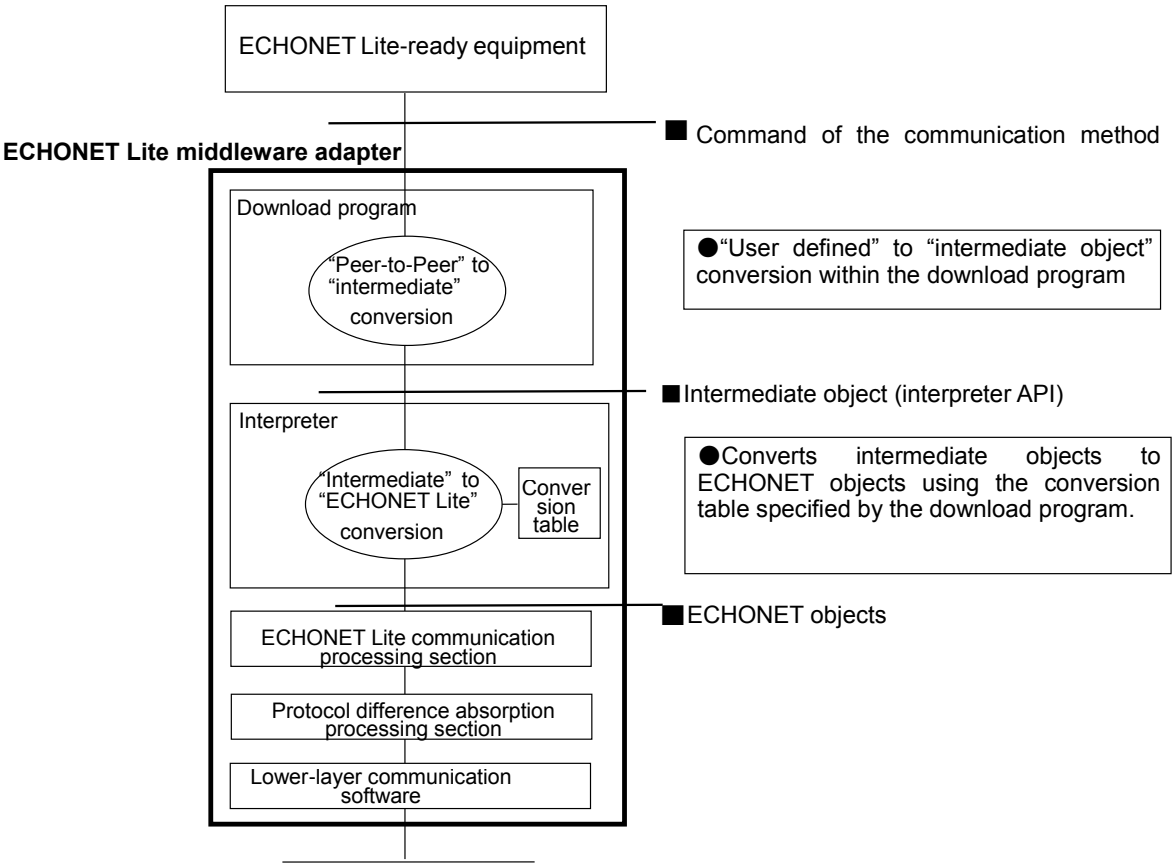


Fig. 3.33 Roles of the Download Program and the Interpreter

3.10.2.2. Operation procedure

Fig. 3.34 shows the operation procedure for an ECHONET Lite middleware adapter equipped with an Interpreter Method-based program execution environment.

(1) Program download

Downloading of the program from the ECHONET Lite-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 Lite initialization

The ECHONET Lite initialization API is called from the download program.

(5) Transmission and reception of ECHONET Lite commands and Peer-to-Peer commands

All communications that use ECHONET Lite shall be performed using the conversion table.

Communications with ECHONET Lite-ready equipment and the interpretation of the data shall be performed directly by the download program.

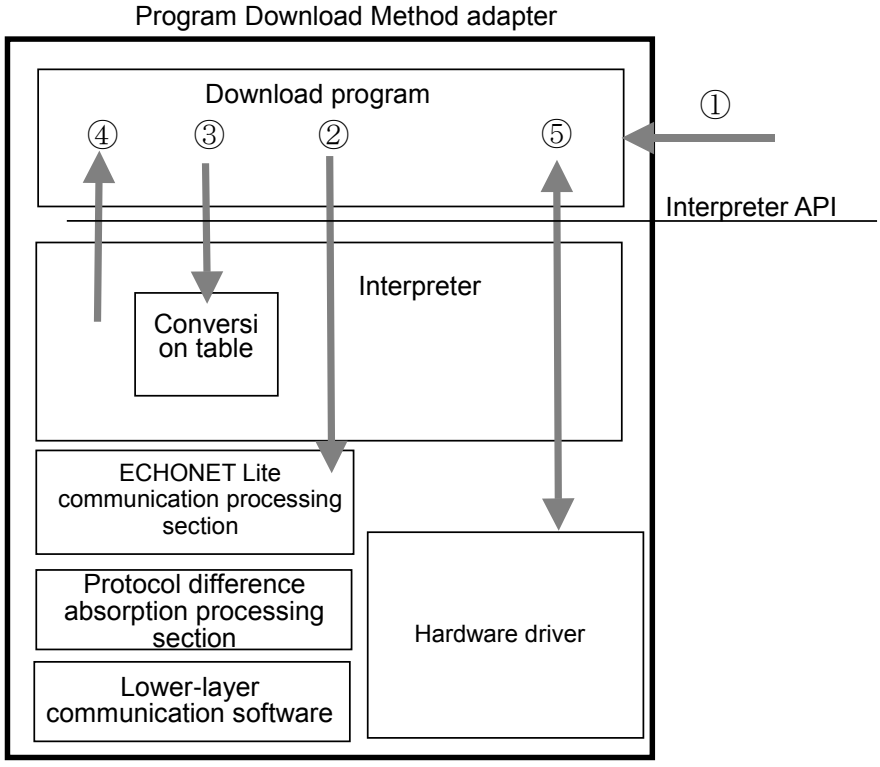


Fig. 3.34 Operation Procedure

3.10.2.3 Conversion table model

(1) Structure of the conversion table model

Fig. 3.35 shows the model of the conversion table used in this method.

- The “ECHONET Lite node” has two or more (n) intermediate objects.
- The intermediate objects are identified by ID and have ECHONET Lite 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 three types of associations between properties: 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.

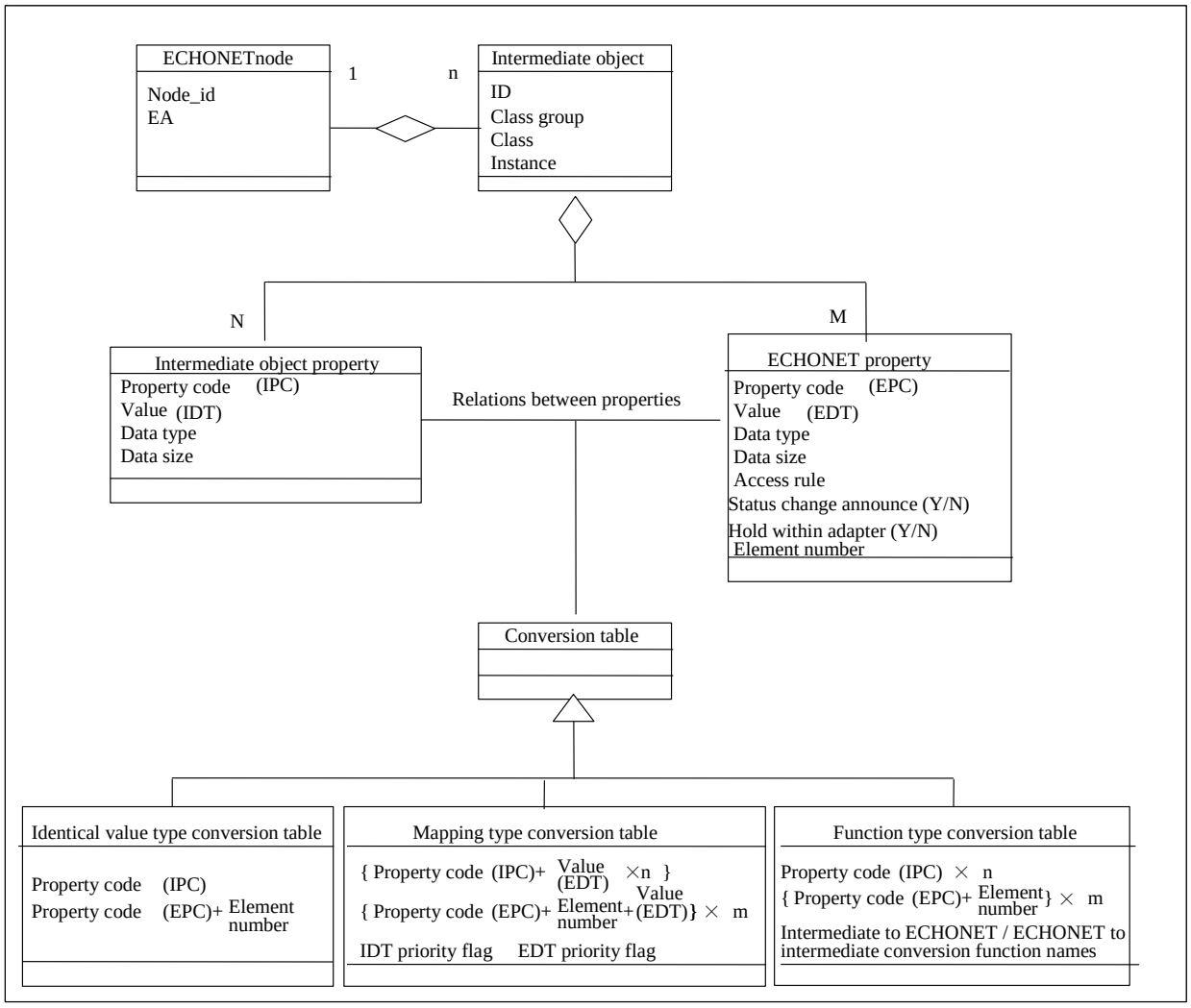


Fig. 3.35 Overview of the Conversion Table Model

(2) ECHONET Lite node

◇ Overview

Definition information for the ECHONET Lite nodes possessed by the adapter.

◇ Record format

Node ID	Communication address

* “Node ID” is an identifier that uniquely identifies the node within the adapter.

* “Communication address” is an address used for communication to lower layers.

◇ Example of use

In the case when the home node is the communication address A and the controller node is the communication address B:

(where, the communication address A / B has an actual address value)

NetID	Communication address
0	Communication address A

1	Communication address B

* 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 values

・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 values

・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

3.10.3. Program format specifications

Fig. 3.36 (the hatched part) shows the format of the main body of the program in an Interpreter Method-based program execution environment.

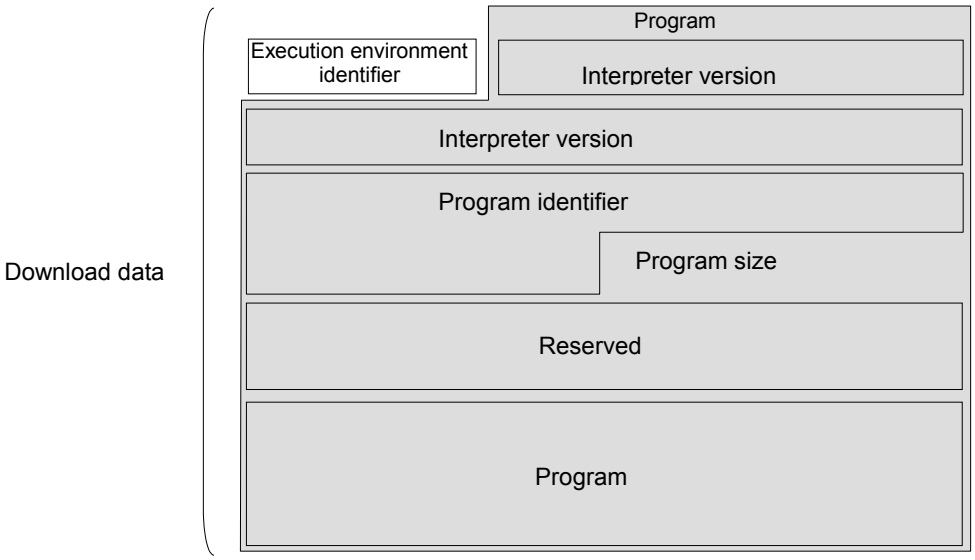


Fig. 3.36 Format of the Main Body of the Program

(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 Lite 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 Lite 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 Lite-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) Reserved 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.”

3.10.4. Specifications for the language of the download program

3.10.4.1. Overview

The download program shall use a Reversed Polish Notation-based stack language.
The API definition shall be as shown in Fig. 3.37.

```
FUNC1(arg1 arg2 arg3 <name> --ret1 ret2, comment)
```

Fig. 3.37 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. 3.38.

```
arg1 arg2 arg3 FUNC1 name
```

Fig. 3.38Fig. 3.40 API Program Description Method

Fig. 3.39 shows the stack before the execution of FUNC1 API and the stack after the execution.

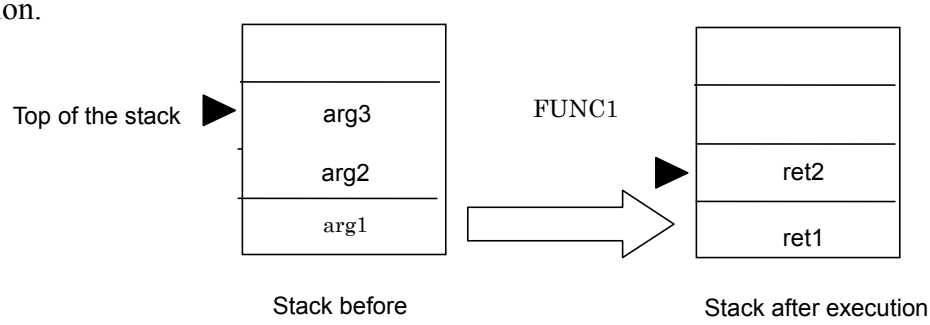


Fig. 3.39 Stacks before and after Execution of API

3.10.4.2. 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" (3.10.4.3), "user-defined names" (3.10.4.4), "interpreter basic API" (3.10.5) and "interpreter ECHONET Lite API" (3.10.6).

3.10.4.3. 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 order shall be big-endian.

3.10.4.4. User-defined names

User-defined names defined by "Variable definition" (3.10.5.1), "Array definition" (3.10.5.2), "Constant definition" (3.10.5.3) and "Function definition" (3.10.5.4) described in "3.10.5 Interpreter Basic API specifications."

These names shall not be so defined that an overlap occurs (interpreter basic API and interpreter ECHONET Lite API).

3.10.5. Interpreter Basic API specifications

3.10.5.1. 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 C language

```
int n;
```

3.10.5.2. 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 C language

```
Char buf[10];
```

3.10.5.3. 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 C language

```
#define MAX_CHARS 128;
```

3.10.5.4. 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 C language

```
Int calc(int q)
{
    n=q+128;
    return n;
}
```

3.10.5.5. 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 C language

```
n = 34;
```

3.10.5.6. 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 C language

```
n;
```

3.10.5.7. 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 C language

```
Buf[3] = 34;
```

3.10.5.8. 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 C language

```
Buf[3];
```

3.10.5.9. Four arithmetic operations (+ – * / MOD)

(1) Function Performs addition/subtraction/multiplication/division/remainder calculation and stores the result at the 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 C language

```
1 +1;
1 –1;
1 *1;
1 /1;
1 % 1;
```

3.10.5.10. 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 C language

```
n « 4
n » 4
```

* Logical right shifting: If signed, 0s are entered from the left.

3.10.5.11. 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 C language

```
2  == 2
2  > 2
2  > 2
2 >= 2
```

```
2 <= 2
2 <> 2
```

(4) Notes

Comparisons of two 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 two 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.

3.10.5.12. 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 C language

```
2 == true
2 == true
```

3.10.5.13. 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 C language

```
0&1
0|1
~n
0^1
```

3.10.5.14. 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 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 C language (B)

```
Int sample_if() {
      If (x == 1) return 11;
      If (x == 2) return 12;
      If (x == 3) return 13;
}
```

3.10.5.15. Loop (BEGIN – WHILE – 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 C language

```
int q;
int n;
q=0;
n=10;
while(n>0){
   q = q + n;
   n--;
}
```

3.10.5.16. 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.

3.10.5.17. 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 two 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
```

3.10.5.18. 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 C language

```
// this line is a comment line
2 + /* this also a comment */ 3
```

3.10.5.19. 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 big-endian order.
TRUE is represented by 0xffffffff and FALSE is represented by 0x00000000.

3.10.5.20. Converting LONG values into TRUE/FALSE values (CONV_ADDR_TF)

(1) Function Converts 4-byte data into a 2-byte TRUE or FALSE value.

(2) Stack diagram CONV_ADDR_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)

3.10.6. Interpreter ECHONET Lite API specifications

3.10.6.1. INT_ECHONET

(1) Function Initializes the ECHONET Lite 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 Lite 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.

3.10.6.2. SET_COM_PARAM

(1) Function Sets an ECHONET Lite 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 = 1000 ms)

(4) Notes

3.10.6.3. SET_UART_RV_MODE

(1) Function Sets the criterion to determine the end of the process to receive a message from ECHONET Lite-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.

3.10.6.4. 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.

3.10.6.5. RGST_NODE

(1) Function: Registers other nodes.

(2) Stack diagram: RGST_NODE (node_id addr_buf addr_size --)

(3) Explanation

node_id : Node ID

addr_buf : Buffer address containing communication address

addr_size : The number of bytes of communication address

(4) Notes

・The argument node_id is an identifier to uniquely identify nodes within the download program.

・The home node is automatically generated at the time of

interpreter startup with node_id set to 0.

3.10.6.6. 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 Lite 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

3.10.6.7. 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.

3.10.6.8. RGST_EPCM

(1) Function Registers array type ECHONET properties.

This is used in the ECHONET Specification. However, this is not stipulated in the ECHONET Lite Specification.

3.10.6.9. ADD_EPC_MEMBER

(1) Function Adds an array element, with element number specified, to an array type ECHONET property registered using RGST_EPCM.

This is used in the ECHONET Specification. However, this is not stipulated in the ECHONET Lite Specification.

3.10.6.10. 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

3.10.6.11. RGST_IDENTICAL_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

3.10.6.12. 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

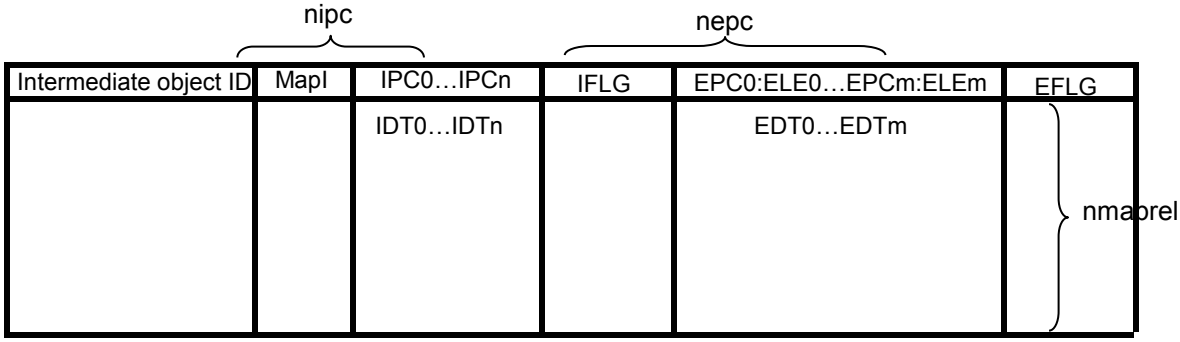
nipc : Number of intermediate object properties

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.



3.10.6.13. 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 two 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
.....
```

3.10.6.14. 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 two 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
.....
```

3.10.6.15. 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

Intermediate object ID	IPC0…IPCn	EPC0:ELE0…EPCm:ELEm	I2E_FUNC	E2I_FUNC

3.10.6.16. 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
```

.....

3.10.6.17. SET_SEND_IPC

(1) Function — Specifies intermediate object property values, writes the values to the corresponding ECHONET properties and sends them to ECHONET Lite.

(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 four highest-order bits and four lowest-order bits.

four highest-order bits:

- 0x1* (Setting; no response required)
- 0x2* (Setting; response required)
- 0x3* (Acquisition)
- 0x4* (Notification)

four 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 Lite 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
.....
```

3.10.6.18. 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 ECHONET Lite.

(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

four highest-order bits:

0x1* (Setting; no response required)

0x2* (Setting; response required)

0x3* (Acquisition)

0x4* (Notification)

four 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.

3.10.6.19. 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.

3.10.6.20. 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.

3.10.6.21. 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.

3.10.6.22. 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

3.10.6.23. 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

3.10.6.24. 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

3.10.6.25. 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

3.10.6.26. 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

3.10.6.27. STOP

(1) Function Stops the operation of the ECHONET Lite middleware adapter.

(2) Stack diagram STOP (--)

(3) Explanation None

(4) Notes

3.10.6.28. RESET

(1) Function Places the ECHONET Lite middleware adapter in the "unrecognized" state and starts the equipment interface data recognition service.

(2) Stack diagram RESET (--)

(3) Explanation None

(4) Notes

3.10.7. Program compression and uncompression specifications

3.10.7.1. 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 Lite-ready equipment

manufacturer during the program development stage. The compressed program shall be uncompressed in the interpreter of the middleware adapter and executed.

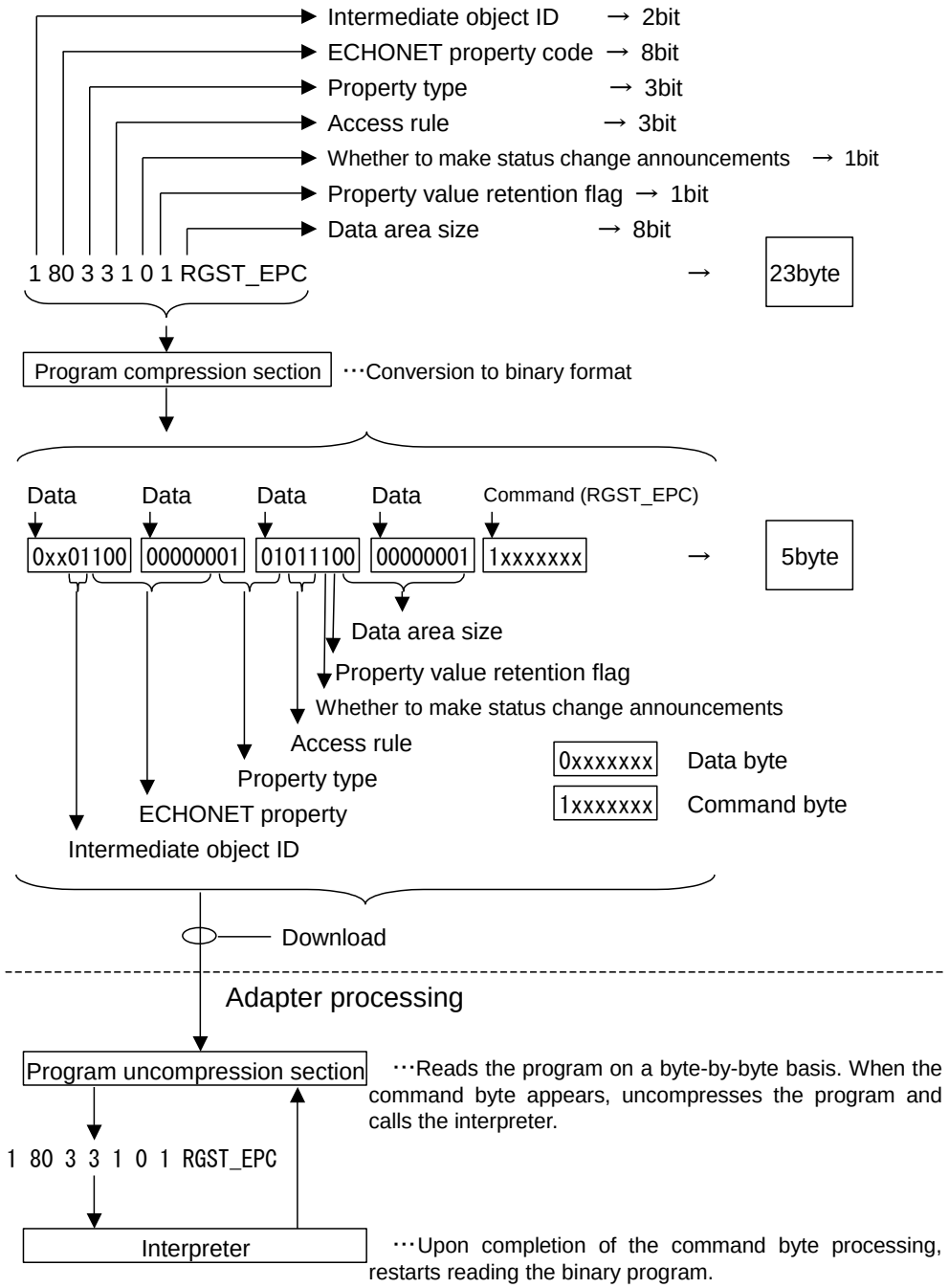


Fig. 3.40 Overview of Program Compression

Fig. 3.40 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 four bits in practice, but is shown to be two bits in the figure). The program compression process generates data in which the values are stored with each piece of the input data assigned 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.

3.10.7.2. Byte specifications

(1) Command bytes

A command byte is a byte whose first bit is 1. The remaining seven 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 3.14 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 seven bits indicate the content of the data. For the data byte generation method, refer to Section 3.10.7.3.

3.10.7.3. 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 Section 3.10.7.1, character strings indicating data are converted into data byte strings in accordance with the following procedure (see Fig. 3.40).

(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. 3.40, the bit pattern "01" is generated for the intermediate object ID (since the value is "1" and the data size is two bits), the bit pattern "10000000" is generated for the ECHONET property code (since the value is "80" and

the data size is eight bits) and the bit pattern “011” is generated for the property type (since the value is “3” and the data size is three 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. 3.40).

The data sizes for individual types of data shall be as specified in Section 3.10.7.4. The data size for types of data that are not specified in Section 3.10.7.4 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 two 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.

3.10.7.4. 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 3.15 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 RGST_EPC RGST_EPCM ADD_EPC_MEMBER RGST_IPC RGST_IDENTICAL_PROP RGST_MAP_PROP_REL RGST_FUNC_PROP SET_IPC SET_SEND_IPC 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 RGST_EPCM ADD_EPC_MEMBER

			RGST_IDENTICAL_PROP RGST_MAP_PROP_REL RGST_FUNC_PROP
Property type	3	0–7	RGST_EPC 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 RGST_EPCM
Property value retention flag	1	0–1	RGST_EPC RGST_EPCM
Data area size (number of bytes)	8	0–255 (byte)	RGST_EPC RGST_EPCM
Element number (array type EPC)	8	0–255	RGST_EPCM ADD_EPC_MEMBER RGST_IDENTICAL_PROP RGST_MAP_PROP_REL RGST_FUNC_PROP
Intermediate object property code (IPC)	8	0–255	RGST_IPC RGST_IDENTICAL_PROP RGST_MAP_PROP_REL RGST_FUNC_PROP SET_IPC SET_SEND_IPC 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 RGST_FUNC_PROP
Number of ECHONET property code values	3	0–7	RGST_MAP_PROP_REL 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 RGST_MAP_PROP_VAL RGST_MAP_PROP_VAL_PR
Intermediate object property value (IDT)	16	0x0000–0xFFFF	RGST_MAP_PROP_VAL RGST_MAP_PROP_VAL_PR SET_IPC SET_SEND_IPC
ECHONET object property value (EDT)	16	0x0000–0xFFFF	RGST_MAP_PROP_VAL 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

			TO_EQUIPMENT 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 RGST_OBJ
Blocking mode flag	1	0–1	TO_EQUIPMENT
Status	1	0–1	INDICATE_STATUS
Type of calculation	5	0–31	OP_LONG
1-byte data	8	0x00–0xFF	SET_BUF
2-byte data	16	0x0000–0xFFFF	(Default)

3.10.7.5. 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

Appendix 1 Reference Document

(1) "PH-CONNECTOR" issued by JST Mfg. Co., Ltd.

Appendix 2 Examples of Interpreter Method Programs

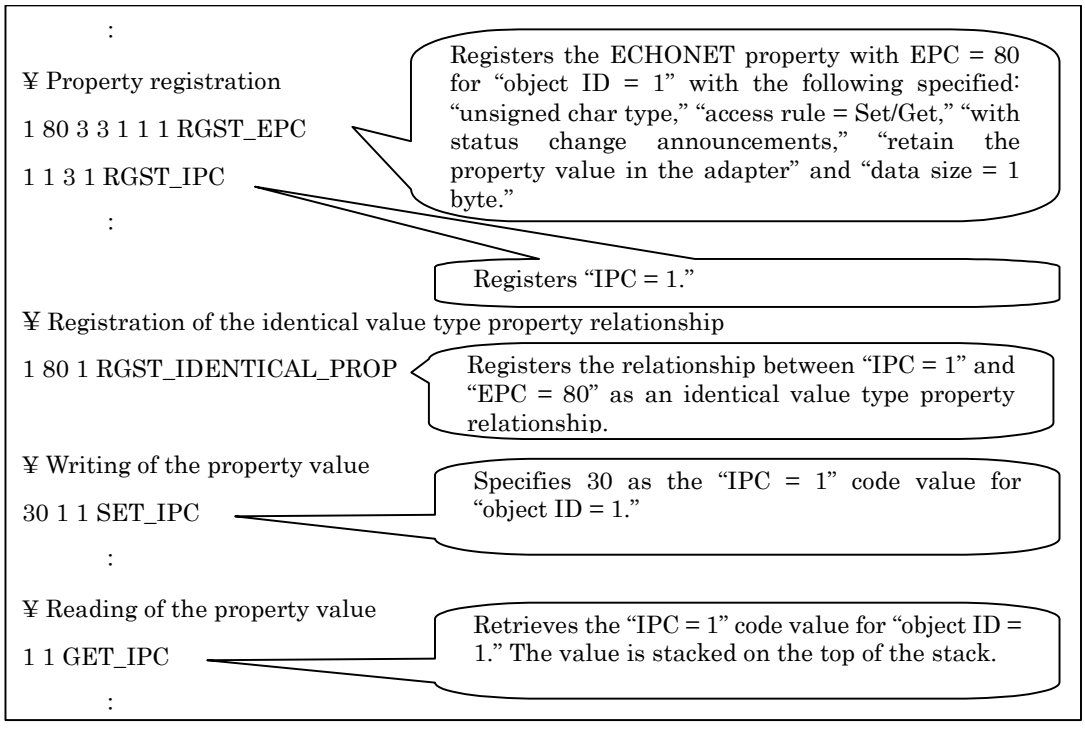
(1) "PH-CONNECTOR" issued by JST Mfg. Co., Ltd.

(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 Lite "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

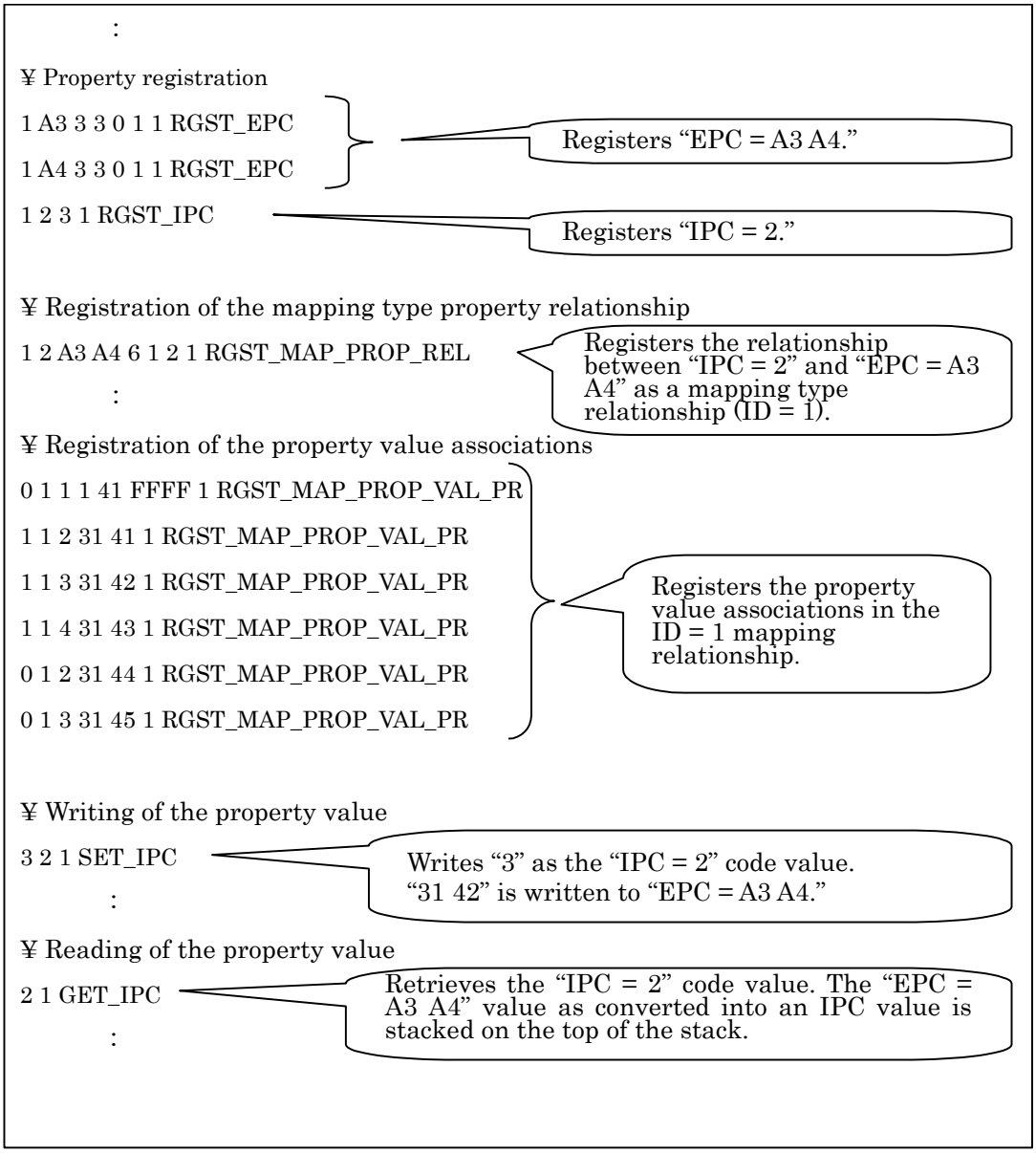
(1.2) Mapping type property conversion

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 Lite "automatic swing of air flow" property (EPC = 0xA3) and the ECHONET Lite "wind direction setting (vertical)" property (EPC = 0xA4) in a 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 Lite "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 C language.

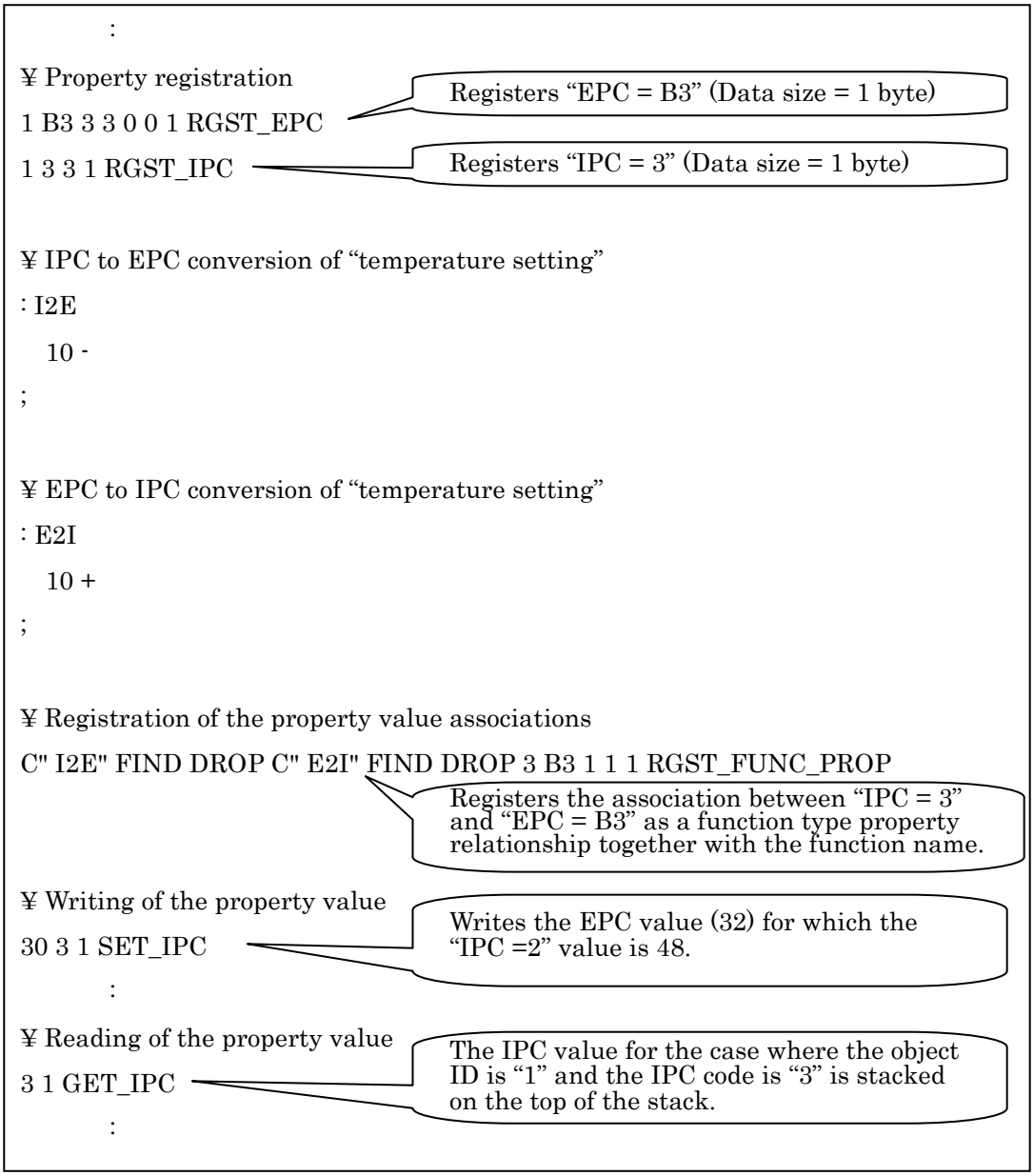
```
¥ IPC to EPC conversion of "temperature setting"
: I2E
  10 -
;

¥ 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 C Language (reference information)

Below is an example property conversion code for the case where the above-mentioned function type relationship exists.



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 ms

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 ECHONET Lite)
 ●● ・・・ ● 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 Lite 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 Lite
 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 Lite
  CNT @ 1 + CNT !            ¥ Increments the loop counter by one
 REPEAT
 PR_REG                      ¥ Equipment status confirmation (Routine
processing)
REPEAT
```