Part V

ECHONET Common Lower-Layer Communication Interface Specifications

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

・Version1.0	March 18th	2000	Released / Open to consortium members
	July	2000	Open to the public
・Version1.01	May 23rd	2001	Open to consortium members Version 1.0 addendum & corrigendum
・Version2.00	August 7th	2001	Open to consortium members

Since the power line A and power line B methods were integrated into a single power line method (based on the power line A method), the associated descriptions were corrected accordingly.
The following Table-of-Contents entries were revised:

	Revised entry	Revision/addition
1	4.2.1	Descriptions were changed because the power line A and power line B methods were integrated into a single method.
2	4.2.10	Descriptions were changed because the power line A and power line B methods were integrated into a single method.
3	4.2.15	Descriptions were changed because the power line A and power line B methods were integrated into a single method.

・Version 2.01	December 19th	2001	Open to consortium members
・Version 2.10 Preview	December 28th	2001	Open to consortium members
・Version 2.10 Draft	February 15th	2002	Open to consortium members
・Version 2.10	March 7th	2002	Open to consortium members

The following Table-of-Contents entries were revised:

	Revised entries in the Table of Contents	Revision/addition
1	2.1	- The following interfaces were added in accordance with the revision to the state transition stipulated in Part 2: "Request for lower-layer communication software mounting information", "request for complete initialization", "request for communication stop", and "request for complete stop"
2	2.1	- The request named "request for reset" was renamed to "request for warm start" in accordance with the revision to the state transition stipulated in Part 2.
3	2.2	- The detailed interface descriptions were changed in accordance with the revision to the state transition stipulated in Part 2.

•Version 2.11 April 26th 2002 Open to consortium members

The following Table-of-Contents entries were revised:

	Revised entry	Revision/addition
1	4.2.2	- The type of buffers sbuf and rbuf in the structure used was changed from short to unsigned char*.
2	4.2.20	- CLC_ADAPTER_ERROR (4) was added as a return value.
3	4.2.21	- CLC_ADAPTER_ERROR (4) was added as a return value.
4	4.2.22	- CLC_ADAPTER_ERROR (4) was added as a return value.
5	4.2.24	- The type of syntax argument node_id was changed from unsigned char to unsigned char*.

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

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

	Revised entry	Revision/addition
1	1.2	**Addition of Fig. 1.1 New Transmission Media**
2	2.1	Addition of the following: (22) Lower-layer communication software address table data acquisition (23) Master router notification (24) Hardware address data acquisition
3	2.2	Addition of explanation under the following headings: (22) Lower-layer communication software address table data acquisition (23) Master router notification (24) Hardware address data acquisition
4	3.1	Addition of “Lower-layer communication software address table data acquisition,” “Master router notification” and “Hardware address data acquisition” to Table 3.1
5	3.2	Addition of explanation of “lower-layer communication software address table data acquisition,” “master router notification” and “hardware address data acquisition”
6	4.1	Corrections to Table 4.1
7	4.1	Addition of the following to Table 4.1: ClcLowGetAddressTableData ClcLowSetMasterRouterFlag ClcLowGetHardwareAddress
8	4.2.29	Addition of explanation of ClcLowGetAddressTableData
9	4.2.30	Addition of explanation of ClcLowSetMasterRouterFlag
10	4.2.31	Addition of explanation of ClcLowGetHardwareAddress

•Version 3.00 August 29th 2002 Open to consortium members
•Version 3.00 Draft November 8th 2002 Open to consortium members
•Version 3.10 December 18th 2002 Open to consortium members
•Version 3.11 March 7th 2003 Open to consortium members
•Version 3.12 May 22 th 2003 Open to consortium members

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

	Revised entry	Revision/addition
1	3.2	Deletion of the hardware type data from the output data described in "(24) Hardware address data acquisition"
2	4.2.32	Corrections to the explanation
3	2.1 3.1 3.2	Addition of "Lower-layer communication software table data size acquisition"

•Version 3.20 Draft October 17th 2003 Open to consortium members
The parts to which changes have been made are as follows:

	Revised entry	Revision/addition
1	2.1, 2.2, 3.1 and 3.2	Addition of interfaces
2	4.1	Addition of functions
3	4.2.18	Argument correction (ClcLowGetProData)
4	4.2.31	Master router setting function name changed
5	4.2.33	Addition
6	4.2.34	Addition
7	4.2.35	Addition

•Version 3.20 December 12th 2006 Open to the Public
The parts to which changes have been made are as follows:

	Revised entry	Revision/addition
1	3.1	Corrections to Table 3.1
2	4.1	Corrections to Table 4.1
3	4.2.8	Corrections to the explanation
4	4.2.18	Corrections to the explanation
5	4.2.31	Corrections to the explanation
6	4.2.33	NodeID list changed to a table form Corrections to the explanation
7	4.2.34	Function name for "ClcGetMasterRouterInfo" changed to "Master router data acquisition function"

• Version 3.30 December 2nd 2004 Open to consortium members.
The parts to which changes have been made are as follows:

	Revised entry	Revision/addition
1	1.2	• Explanations about IEEE802.11/11b were added in "Fig. 1.1. Positioning of the Common Lower-layer Communication Interface."
2	4.2	The values "0x91" to "0x9F" for IEEE802.11/11b were added to the explanations about device_id in 4.2.1 and 4.2.13 to 4.2.35.
3	4.2.1	"ClcGetDeviceID" in "(3) Syntax" was changed to "ClcGetDevID."

• Version 3.40 Draft December 28th 2004 Open to consortium members.
The parts to which changes have been made are as follows:

	Revised entry	Revision/addition
1	1.2	Explanations about the Power Line Communication Protocol C and D Systems were added in Fig. 1.1. The layout was changed.
2	4.2.1 4.2.13 4.2.14 4.2.15 4.2.16 4.2.17 4.2.18 4.2.19 4.2.20 4.2.21 4.2.22 4.2.23 4.2.24 4.2.25 4.2.26 4.2.27 4.2.28 4.2.29 4.2.30 4.2.31 4.2.32 4.2.33 4.2.34 4.2.35	"Power Line Communication Protocol System" in the explanations about device_num was changed to "Power Line Communication Protocol A and D Systems." The value "0xA1" for the Power Line Communication Protocol C System was added.

• Version 3.40 February 3rd 2005 Open to consortium members.
• Version 3.41 May 11th 2005 Open to consortium members.
• Version 3.2 October 13th 2005 Open to the public.
• Version 3.42 October 27th 2005 Open to consortium members.
The parts to which changes have been made are as follows:

	Revised entry	Revision/addition
1	4.2.18	-Descriptions relating to LOW_PRO_DATA were changed to make them consistent with the corresponding descriptions in Part 6. -Explanations about LOW_PRO_DATA relating to members were changed. -The array sizes of mac_ad and mac_mask were changed.

• Version 3.50 Draft August 3rd 2006 Open to consortium members.
• Version 3.50 September 20th 2006 Open to consortium members.
• Version 3.51 Draft February 2nd 2007 Open to consortium members.
• Version 3.60 March 5th 2007 Open to consortium members.
December 11th 2007 Open to the public.

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

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

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

Contents

Chapter 1 Overview

1.1 Basic Concept

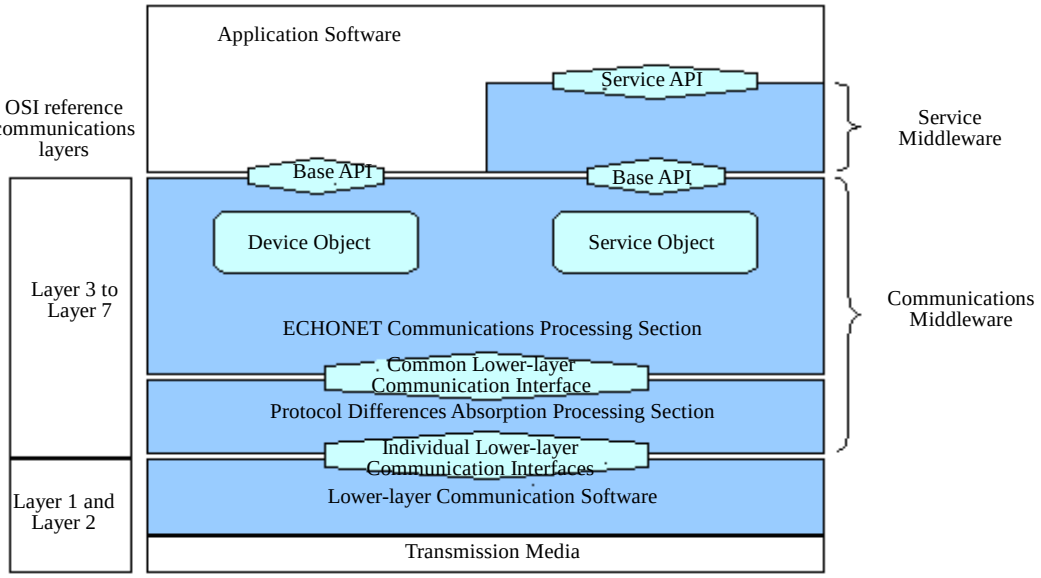
The ECHONET Common Lower-Layer Communication Interface Specifications in Part 5 are provided to specify a software interface to implement processing and information exchange between the ECHONET Communications Processing Block and the Protocol Difference Absorption Processing Block, which are described in Fig. 1.1 on the next page. The Common Lower-Layer Communication Interface makes it possible to describe the processing specifications for the ECHONET Communications Processing Block in common form without regard for differences in lower-layer communications software specifications. The Common Lower-Layer Communication Interface specifications provide Level 1 and Level 2 function rules for cases in which input/output data items and concrete language are specified with regard to APIs, based on the assumption that they are supported by the Protocol Difference Absorption Processing Block. Levels 1 and 2 of the Common Lower-Layer Communication Interface specifications are based on the concept of Levels 1 and 2 of the basic API.

1.2 Positioning on Communication Layers

The interface specifications described in this section are provided in a form that permits the absorption of differences in the lower-layer communication software so that the ECHONET Communications Processing Block may control the portion under the Protocol Difference Absorption Processing Block without regard to differences in the lower-layer communication software.

The shaded area in Fig. 1.1 shows the positioning of the Common Lower-Layer Communication Interface.

The Common Lower-Layer Communication Interface is positioned between the Protocol Difference Absorption Processing Block and the ECHONET Communications Processing Block to implement mutual exchange.



Lower-layer Communication Software Supported by the Current Version

Simbol	Name of Lower-layer Communication Software	Transmission Medium
A	Power Line Communication Protocol A System Power Line Communication Protocol D System	Power distribution lines
B	Low-power Radio	Low-power radio
C	Extended HBS	Twisted-pair cables
D	IrDA Control	Infrared
E	LonTalk®	Low-power radio
F	Bluetooth® (UDP/IP)	Low-power radio (BT)
G	Ethernet IEEE802.3 (UDP/IP)	Ethernet
H	IEEE802.11 IEEE802.11b (UDP/IP)	Low-power radio (WLAN)
I	Power Line Communication Protocol C System	Power distribution lines

and other countries.
Bluetooth is a registered trademark of Bluetooth SIG, Inc.
Ethernet is a registered trademark of the Xerox Corporation.
All other trademarks are properties of their respective owners.

Fig. 1.1 Positioning of Common Lower-Layer Communication Interface

Chapter 2 ECHONET Common Lower-Layer Communication Interface Function Specifications

2.1 List of ECHONET Common Lower-Layer Communication Interface Functions

Table 2.1 shows a list of ECHONET Common Lower-Layer Communication Interface functions supported by the Protocol Difference Absorption Processing Block, which shall be provided with these functions. The interface function specifications are stated in the next section.

(1) Request for lower-layer communication software mounting information
(2) Request for initialization
(3) Request for operation start
(4) Fault notice
(5) Request for warm start
(6) Request for suspension
(7) Request for operation restart
(8) Request for protocol difference absorption processing block profile data acquisition
(9) Request for lower-layer communication software profile data acquisition
(10) Request for protocol difference absorption processing block status data acquisition
(11) Request for lower-layer communication software status data acquisition
(12) Request for data transmission
(13) Transmission result acquisition
(14) Request for transmission stop
(15) Request for received data
(16) Request for node ID acquisition
(17) Request for node ID setup
(18) Request for complete initialization
(19) Request for communication stop
(20) Request for complete stop
(21) Stop notice
(22) Lower-layer communication software address table data size acquisition
(23) Lower-layer communication software address table data acquisition
(24) Master router notification
(25) Hardware address data acquisition
(26) NodeID list acquisition
(27) Master router data acquisition
(28) Hardware address conversion request

2.2 ECHONET Common Lower-Layer Communication Interface Function Detailed Specifications

This section explains ECHONET Common Lower-Layer Communication Interface functions supported by the Protocol Difference Absorption Processing Block. For state transitions of this processing block and the lower-layer communication software, refer to the associated lower-layer communication software descriptions in Part 3.

(1) Request for lower-layer communication software mounting information

Requests that the protocol difference absorption processing block furnish information about lower-layer communication software (the number of mounted lower-layer communication software programs and their IDs).

(2) Request for initialization

Requests that the protocol difference absorption processing block and lower-layer communication software effect initialization by performing a cold start and switch to the communication stop state. Here, the MAC address retained by the lower-layer communication software is discarded/updated.

(3) Request for operation start

Requests that the protocol difference absorption processing block and lower-layer communication software switch from the communication stop state to the normal operation state.

(4) Fault notice

Notifies the Protocol Difference Absorption Processing Block of the fault (error) status of the high-order layer from the ECHONET Communications Processing Block.

(5) Request for warm start

Requests that the protocol difference absorption processing block and lower-layer communication software effect initialization by performing a warm start and switch to the communication stop state. Here, the MAC address retained by the lower-layer communication software remains unchanged.

(6) Request for suspension

Requests that the protocol difference absorption processing block and lower-layer communication software switch from the normal operation state to the suspension state.

(7) Request for operation restart

Requests that the protocol difference absorption processing block and lower-layer communication software exit the suspension state and enter the normal operation state.

(8) Request for protocol difference absorption processing block profile data acquisition

Asks the Protocol Difference Absorption Processing Block for profile data of the Protocol Difference Absorption Processing Block.

The profile data requested by this function consists of static information about the protocol difference absorption processing block, such as the development manufacturer code and version number.

(9) Request for lower-layer communication software profile data acquisition

Asks the Protocol Difference Absorption Processing Block for profile data of the lower-layer communication software.

The profile data requested by this function consists of static information for lower-layer communication software, such as the software development manufacturer code and version number.

(10) Request for protocol difference absorption processing block status data acquisition

Requests that the protocol difference absorption processing block furnish status data. The status data requested by this function consists of dynamic information about the protocol difference absorption processing block, such as information about abnormality and processing status.

(11) Request for lower-layer communication software status data acquisition

Asks the Protocol Difference Absorption Processing Block for status data of the lower-layer communication software.

The status data requested by this function consists of dynamic information for lower-layer communication software, such as information about abnormality and processing status.

(12) Request for data transmission

Requests that the Protocol Difference Absorption Processing Block send the specified ECHONET data.

(13) Transmission result acquisition

Requests that the protocol difference absorption processing block furnish information about the status of the data transmission process requested immediately before this request.

(14) Request for transmission stop

Requests that the protocol difference absorption processing block stop the data transmission process performed by the lower-layer communication software in compliance with the data transmission request issued immediately before this request.

(15) Request for received data

Requests that the Protocol Difference Absorption Processing Block deliver the received data.

(16) Request for node ID acquisition

Requests the node ID information that is retained by the protocol difference absorption processing block.

(17) Request for node ID setup

Sets the NodeID information for the protocol difference absorption processing block.

(18) Request for complete initialization

Requests that the protocol difference absorption processing block cold-start the lower-layer communication software and then place it in the communication stop state. Here, the house code information and MAC address will be acquired again.

(19) Request for communication stop

Requests that the protocol difference absorption processing block place the lower-layer communication software in the communication stop state.

(20) Request for complete stop

Requests that the protocol difference absorption processing block place the lower-layer communication software in the stop state.

(21) Stop notice

The Protocol Difference Absorption Processing Block notifies the ECHONET Communications Processing Block that the lower-layer communication software has switched to the stop state.

(22) Lower-layer communication software address table data size acquisition

Acquires the number of pairs of lower-layer address table data from the lower-layer communication software.

(23) Lower-layer communication software address table data acquisition

Acquires the lower-layer address table data from the lower-layer communication software.

(24) Master router notification

Notifies the lower-layer communication software as to whether or not the home node is a master router.

(25) Hardware address data acquisition

Acquires the hardware address data from the lower-layer communication software.

(26) NodeID list acquisition

Acquires the NodeID list from the lower-layer communication software.

(27) Master router data acquisition

Acquires the master router data from the lower-layer communication software.

(28) Hardware address conversion request

Requests the lower-layer communication software to provide the hardware address that corresponds to the NodeID sent to the lower-layer communication software.

Chapter 3 Level 1 ECHONET Common Lower-Layer Communication Interface Specifications

3.1 List of Level 1 ECHONET Common Lower-Layer Communication Interface Services

For each service listed in Table 3.1, the Level 1 ECHONET common lower-layer communication interface prescribes the data to be exchanged between the ECHONET Communications Processing Block and Protocol Difference Absorption Processing Block. For mounting in compliance with the Level 1 ECHONET common lower-layer communication interface specifications, the input/output data items stipulated in the next section shall be provided. However, two or more services may be integrated into a single service, and a single service may be divided into two or more services. Further, two or more data items may be processed as a single data item, and a single data item may be processed as two or more data items.

Table 3.1 List of Level 1 ECHONET Common Lower-Layer Communication Interface Services (1/2)

No.	Service name	Function outline	Mounting specification
1	Request for lower-layer communication software mounting information	Requests the number of mounted (accessible) lower-layer communication software programs and their types.	Required
2	Request for initialization	Requests that a specified protocol difference absorption processing block and lower-layer communication software effect initialization by performing a cold start.	Required
3	Request for operation start	Requests that a specified protocol difference absorption processing block and lower-layer communication software start running.	Required
4	Fault notice	Notifies Protocol Difference Absorption Processing Block of fault (error) status of high-order layer from the ECHONET Communications Processing Block.	Optional
5	Request for warm start	Requests that a specified protocol difference absorption processing block and lower-layer communication software effect initialization by performing a warm start.	Required
6	Request for suspension	Requests that a specified protocol difference absorption processing block and lower-layer communication software suspend operation.	Optional
7	Request for operation restart	Requests that a specified protocol difference absorption processing block and lower-layer communication software resume operation.	Optional
8	Request for protocol difference absorption processing block profile data acquisition	Obtains static information for Protocol Difference Absorption Processing Block.	Required
9	Request for lower-layer communication software profile data acquisition	Obtains static information for lower-layer communication software.	Required

Table 3.1 List of Level 1 ECHONET Common Lower-Layer Communication Interface Services (2/2)

No.	Service name	Function outline	Mounting specification
10	Request for protocol difference absorption processing block status data acquisition	Obtains dynamic status (processing fault, etc.) of Protocol Difference Absorption Processing Block.	Optional
11	Request for lower-layer communication software status data acquisition	Obtains dynamic status (processing fault, address redundancy, etc.) of lower-layer communication software.	Required
12	Request for data transmission	Requests data transmission from Protocol Difference Absorption Processing Block.	Required
13	Transmission result acquisition	Requests data transmission result from Protocol Difference Absorption Processing Block.	Optional
14	Request for transmission stop	Requests that the protocol difference absorption processing block stop a data transmission.	Optional
15	Request for received data	Requests received data from Protocol Difference Absorption Processing Block.	Required
16	Request for node ID acquisition	Makes a request for acquiring a node ID retained by a protocol difference absorption processing block.	Required
17	Request for node ID setup	Sets NodeID for Protocol Difference Absorption Processing Block.	Optional
18	Request for complete initialization	Requests that a specified protocol difference absorption processing block and lower-layer communication software effect initialization by performing a cold start. Here, the house code information will be acquired again.	Optional
19	Request for communication stop	Requests that a specified protocol difference absorption processing block and lower-layer communication software switch to the communication stop state.	Optional
20	Request for complete stop	Requests that a specified protocol difference absorption processing block and lower-layer communication software switch into stop state.	Optional
21	Stop notice	Protocol Difference Absorption Processing Block notifies the ECHONET Communications Processing Block that the lower-layer communication software has switched to stop state.	Optional
22	Lower-layer communication software address table data size acquisition	Acquires the number of pairs of lower-layer address table data from the lower-layer communication software.	Optional
23	Lower-layer communication software address table data acquisition	Acquires the lower-layer address table data from the lower-layer communication software.	Optional
24	Master router notification	Notifies the lower-layer communication software as to whether or not the home node is a master router.	Optional
25	Hardware address data acquisition	Acquires the hardware address data from the lower-layer communication software.	Optional
26	NodeID list acquisition	Acquires the NodeID list from the lower-layer communication software.	Optional
27	Master router data acquisition	Acquires the master router data from the lower-layer communication software.	Optional
28	Hardware address conversion request	Requests the lower-layer communication software to provide the hardware address that corresponds to the NodeID sent to the lower-layer communication software.	Optional

3.2 Level 1 ECHONET Common Lower-Layer Communication Interface Detailed Specifications

Input/output data is stipulated in accordance with the services described in Table 3.1 in the previous section. In the following tables, references to data input/output direction are made relative to the ECHONET Communications Processing Block. More specifically, the term “input” denotes the transfer of data from the ECHONET Communications Processing Block to a Protocol Difference Absorption Processing Block, and the term “output” indicates the transfer of data from a Protocol Difference Absorption Processing Block to the ECHONET Communications Processing Block. When these data transfer operations can be performed, the Level 1 ECHONET common lower-layer communication interface specifications are complied with. The data transfer method (the use of a structure, the delivery of data exchange buffer pointer information, etc.) is not stipulated here.

(1) Request for lower-layer communication software mounting information (mandatory function for mounting)

Requests the number of mounted (accessible) lower-layer communication software programs and their types (power line, low-power RF, etc.). Table 3.2 shows the data specifications.

Table 3.2 Input/Output Data List for Lower-Layer Communication Software Type Request Service

Direction	Data name	Contents and condition	Remarks
Input	–		
Output	device_num	- Shows the number of mounted lower-layer communication software programs.	Optional
Output	device_id	- Indicates the type of the lower-layer communication software. - The power line lower-layer communication software, specific low-power RF lower-layer communication software, extended HBS lower-layer communication software, LonTalk®-dependent lower-layer communication software, IrDA-dependent lower-layer communication software, and other similar software shall be distinguishable from each other. - When two or more lower-layer communication software programs are supported, the return of two or more responses shall be achievable.	Required
Output	Return Value	TRUE: normal; FALSE: abnormal.	Optional

(2) Request for initialization (mandatory function for mounting)

Requests that the specified lower-layer communication software effect initialization by performing a cold start and switch to the communication stop state, and that the associated protocol difference absorption processing block effect initialization. Within a series of requested processes, the MAC address information is acquired again. When the lower-layer communication software has house code information, this information remains unchanged. Table 3.3 shows the input/output specifications.

Table 3.3 Input/Output Data List for Initialization Request Service

Direction	Data name	Contents and condition	Remarks
Input	device_id	- Specifies lower-layer communication software to be initialized. - Options to be provided for initializing all lower-layer communication software programs simultaneously and initializing a specific lower-layer communication software program.	Required
Input	p_init	- Specifies the initialization parameters. - The parameters include the outgoing data maximum retention time and incoming data maximum retention time. However, the details vary with the lower-layer communication software to be initialized.	Required
Output	Return Value	TRUE: Successful initialization, FALSE: Failed initialization.	Optional

(3) Request for operation start (mandatory function for mounting)

Requests that the specified lower-layer communication software and associated protocol difference absorption processing block start running. Table 3.4 shows the input/output specifications.

Table 3.4 Input/Output Data List for Operation Start Request Service

Direction	Data name	Contents and condition	Remarks
Input	device_id	- Specifies lower-layer communication software that should start running. - Options to be provided for starting the operations of all lower-layer communication software programs simultaneously and for starting the operation of a specific lower-layer communication software program.	Optional
Output	Return Value	TRUE: Successful operation start, FALSE: Failed operation start.	Optional

(4) Fault notice

Notifies Protocol Difference Absorption Processing Block of the fault (error) status of the high-order layer from the ECHONET Communications Processing Block. Table 3.5 shows input/output specifications.

Table 3.5 Input/Output Data List for Fault Notice Service

Direction	Data name	Contents and condition	Remarks
Input	trouble_no	- Reports a trouble number indicating an abnormal state.	Required
Output	Return Value	TRUE: Fault notice acceptable, FALSE: Fault notice not acceptable	Optional

(5) Request for warm start (mandatory function for mounting)

Requests that the specified lower-layer communication software and associated protocol difference absorption processing block effect initialization by performing a warm start and then switch to the communication stop state. Within a series of requested processes, the house code information and MAC address information remain unchanged. Table 3.6 shows the input/output specifications.

Table 3.6 Input/Output Data List for Warm Start Request Service

Direction	Data name	Contents and condition	Remarks
Input	device_id	- Specifies lower-layer communication software to be warm-started. - Options to be provided for warm-starting all lower-layer communication software programs simultaneously and for warm-starting a specific lower-layer communication software program.	Required
Output	Return Value	TRUE: Warm start request accepted; FALSE: Request denied.	Optional

(6) Request for suspension

Requests that the specified lower-layer communication software and associated protocol difference absorption processing block switch into suspension state. Table 3.7 shows the input/output specifications.

Table 3.7 Input/Output Data List for Suspension Request Service

Direction	Data name	Contents and condition	Remarks
Input	device_id	- Specifies lower-layer communication software to be suspended. - Options to be provided for suspending all lower-layer communication software programs simultaneously and for suspending a specific lower-layer communication software program.	Required
Output	Return Value	TRUE: Suspension acceptable, FALSE: Suspension not acceptable	Optional

(7) Request for operation restart

Requests that the specified lower-layer communication software and associated protocol difference absorption processing block exit suspension state and enter normal operation state. Table 3.8 shows the input/output specifications.

Table 3.8 Input/Output Data List for Operation Restart Request Service

Direction	Data name	Contents and condition	Remarks
Input	device_id	- Specifies lower-layer communication software whose operation is to be resumed. - Options to be provided for resuming the operations of all lower-layer communication software programs simultaneously and for resuming the operation of a specific lower-layer communication software program.	Required
Output	Return Value	TRUE: Successful restart, FALSE: Restart disable (including failure)	Optional

(8) Request for protocol difference absorption processing block profile data acquisition (mandatory function for mounting)

Requests profile data for the protocol difference absorption processing block associated with the specified lower-layer communication software. The data requested by this service consists of static information about the protocol difference absorption processing block, such as the manufacturer code and version number. Table 3.9 shows the input/output specifications.

Table 3.9 Input/Output Data List for Protocol Difference Absorption Processing Block Profile Data Acquisition Request Service

Direction	Data name	Contents and condition	Remarks
Input	device_idinfo	- Specifies lower-layer communication software associated with the protocol difference absorption processing block to be targeted for acquisition. - Options to be provided for specifying all lower-layer communication software programs and for specifying a specific lower-layer communication software program.	Required
Output	version_No	- Presents version information for the protocol difference absorption processing block.	Optional
Output	company_name	- Shows the manufacturer code.	Optional
Output	rwlen	- Presents buffer size information.	Optional
Output	Return Value	TRUE: Normal, FALSE: Error	Optional

(9) Request for lower-layer communication software profile data acquisition (mandatory function for mounting)

Requests profile data for specified lower-layer communication software. Profile data requested by this function consists of static information for lower-layer communication software, such as the software development manufacturer code and version number. Table 3.10 shows the input/output specifications.

Table 3.10 Input/Output Data List for Lower-Layer Communication Software Profile Data Acquisition Request Service

Direction	Data name	Contents and condition	Remarks
Input	device_id	- Specifies lower-layer communication software to be targeted for acquisition. - Options to be provided for specifying all lower-layer communication software programs and for specifying a specific lower-layer communication software program.	Required
Output	version_No	- Version information for lower-layer communication software	Optional
Output	company_name	- Manufacturer information	Optional
Output	mac_address	- MAC address size information	Optional
Output	rwlen	- Buffer size information	Optional
Output	broad	- Broadcast function available/unavailable	Optional
Output	baud	- Transmission rate	Optional
Output	Return Value	TRUE: Normal, FALSE: Error	Optional

(10) Request for protocol difference absorption processing block status data acquisition

Asks the Protocol Difference Absorption Processing Block for Protocol Difference Absorption Processing Block status data. The status data requested by this function consists of dynamic data such as error status and processing status. Table 3.11 shows input/output specifications.

Table 3.11 Input/Output Data List for Protocol Difference Absorption Processing Block Status Data Acquisition Request Service

Direction	Data name	Contents and condition	Remarks
Input	device_id	- Specifies lower-layer communication software associated with the protocol difference absorption processing block to be targeted for acquisition. - Options to be provided for specifying all lower-layer communication software programs and for specifying a specific lower-layer communication software program.	Required
Output	state	- Presents state transition information.	Required
Output	trouble_no	- Fault information for Protocol Difference Absorption Processing Block.	Optional
Output	upper_trouble	- Information recognized as fault in high-order layer	Optional
Output	Low_trouble	- Information recognized as fault in low-order layer	Optional
Output	Return Value	TRUE: Normal, FALSE: Error	Optional

(11) Request for lower-layer communication software status data acquisition (mandatory function for mounting)

Asks the Protocol Difference Absorption Processing Block for lower-layer communication software status data. The status data requested by this function consists of dynamic data such as error status and processing status. Table 3.12 shows input/output specifications.

Table 3.12 Input/Output Data List for Lower-Layer Communication Software Status Data Acquisition Request Service

Direction	Data name	Contents and condition	Remarks
Input	device_id	- Specifies lower-layer communication software to be targeted for acquisition. - Options to be provided for specifying all lower-layer communication software programs and for specifying a specific lower-layer communication software program.	Required
Output	state	- State transition information. Status recognition shall be achievable as stipulated in Part 3.	Required
Output	trouble_no	- Fault information for lower-layer communication software	Optional
Output	upper_trouble	- Information recognized as fault in high-order layer by lower-layer communication software.	Optional
Output	Return Value	TRUE: Normal, FALSE: Error	Optional

(12) Request for data transmission (mandatory function for mounting)

Requests transmission of specified ECHONET data by specified lower-layer communication software. Table 3.13 shows the input/output specifications.

Table 3.13 Input/Output Data List for Data Transmission Request Service

Direction	Data name	Contents and condition	Remarks
Input	device_id	- Specifies lower-layer communication software to be targeted for transmission. - Individual lower-layer communication software programs must be recognizable.	Required
Input	send_data	- Indicates the requested outgoing data in ECHONET data format. - The format (ECHONET frame) used must be acceptable between the ECHONET communications processing blocks.	Required
Input	dnode_id	- Indicates intra-subnet transmission destination node ID information and transmission type (broadcast or individual).	Required
Output	Return Value	TRUE: Normal, FALSE: Error	Optional

(13) Transmission result acquisition

Requests that the protocol difference absorption processing block associated with specified lower-layer communication software furnish the transmission result of the data requested by a “request for data transmission”. Table 3.14 shows the input/output specifications.

Table 3.14 Input/Output Data List for Transmission Result Acquisition Service

Direction	Data name	Contents and condition	Remarks
Input	device_id	- Specifies lower-layer communication software to be targeted for transmission result acquisition. - Individual lower-layer communication software programs must be recognizable.	Required
Output	result	- Information on transmitting status, normal termination of transmission, termination due to transmission error, or transmission stopping status.	Required
Output	Return Value	TRUE: Normal, FALSE: Error	Optional

(14) Request for transmission stop

Requests that the specified lower-layer communication software stop an ongoing data transmission process. Table 3.15 shows the input/output specifications.

Table 3.15 Input/Output Data List for Transmission Stop Request Service

Direction	Data name	Contents and condition	Remarks
Input	device_id	- Specifies lower-layer communication software to be targeted for transmission stop. - Individual lower-layer communication software programs must be recognizable.	Required
Output	Return Value	TRUE: Stop success, FALSE: Stop failure (already transmitted)	Optional

(15) Request for received data (mandatory function for mounting)

Requests data received by specified lower-layer communication software. Table 3.16 shows the input/output specifications.

Table 3.16 Input/Output Data List for Received Data Request Service

Direction	Data name	Contents and condition	Remarks
Input	device_id	- Specifies lower-layer communication software to be targeted for the received data request. - Individual lower-layer communication software programs must be recognizable.	Required
Output	receive_data	- Indicates received data in ECHONET data format. - The format (ECHONET frame) used must be acceptable between the ECHONET communications processing blocks.	Required
Output	snode_id	- Indicates node ID information for the intra-subnet transmission source.	Required
Output	Return Value	TRUE: Normal, FALSE: Error (error indication code, such as no received data)	Optional

(16) Request for node ID acquisition (mandatory function for mounting)

Requests node ID information corresponding to MAC address retained by specified lower-layer communication software. Table 3.17 shows the input/output specifications.

Table 3.17 Input/Output Data List for Node ID Acquisition Request Service

Direction	Data name	Contents and condition	Remarks
Input	device_id	- Specifies lower-layer communication software to be targeted for node ID acquisition. - Individual lower-layer communication software programs must be recognizable.	Required
Output	nodeID	- Presents node ID information.	Required
Output	Return Value	TRUE: Normal; FALSE: Abnormal (code indicating an undefined node ID, improper device ID selection, or other abnormality).	Optional

(17) Request for node ID setup

Sets node ID information for protocol difference absorption processing block associated with specified lower-layer communication software. The lower-layer communication software changes the MAC address in accordance with this node ID setting. Table 3.18 shows the input/output specifications.

Table 3.18 Input/Output Data List for Node ID Setup Request Service

Direction	Data name	Contents and condition	Remarks
Input	device_id	- Specifies lower-layer communication software to be targeted for node ID setup. - Individual lower-layer communication software programs must be recognizable.	Required
Input	nodeID	- Presents node ID information.	Required
Output	Return Value	TRUE: Normal, FALSE: Error (error indication code such as set disable)	Optional

(18) Request for complete initialization

Requests that the specified lower-layer communication software and associated protocol difference absorption processing block effect initialization by performing a cold start and then switch to communication stop state. Within a series of requested processes, the house code information and MAC address information are acquired again.

Table 3.19 Input/Output Data List for Complete Initialization Request Service

Direction	Data name	Contents and condition	Remarks
Input	software_id	- Specifies lower-layer communication software to be targeted for complete initialization. - Options to be provided for completely initializing all lower-layer communication software programs simultaneously and completely initializing a specific lower-layer communication software program.	Required
Input	p_init	- Specifies initialization parameters. - Parameters include outgoing data maximum retention time and incoming data maximum retention time. However, details vary with the lower-layer communication software to be initialized.	Required
Output	Return Value	TRUE: Initialization successful; FALSE: Initialization not successful.	Optional

(19) Request for communication stop

Requests that the specified lower-layer communication software and associated protocol difference absorption processing block switch to the communication stop state. Table 3.20 shows the input/output specifications.

Table 3.20 Input/Output Data List for Communication Stop Request Service

Direction	Data name	Contents and condition	Remarks
Input	software_id	- Specifies lower-layer communication software to be targeted for communication stop. - Options to be provided for stopping the communication of all lower-layer communication software programs simultaneously and for stopping the communication of a specific lower-layer communication software program.	Required
Output	Return Value	TRUE: Request accepted; FALSE: Request denied:	Optional

(20) Request for complete stop

Requests that the specified lower-layer communication software and associated protocol difference absorption processing block switch into stop state. Table 3.21 shows the input/output specifications.

Table 3.21 Input/Output Data List for Complete Stop Request Service

Direction	Data name	Contents and condition	Remarks
Input	software_id	- Specifies lower-layer communication software to be targeted for complete stop. - Options to be provided for stopping all lower-layer communication software programs simultaneously and for stopping a specific lower-layer communication software program.	Required
Output	Return Value	TRUE: Request accepted; FALSE: Request denied:	Optional

(21) Stop notice

Notifies the ECHONET Communications Processing Block that the lower-layer communication software and the corresponding Protocol Difference Absorption Processing Block have switched to stop state. Table 3.22 shows the input and output specifications.

Table 3.22 Stop Notice Service Input/Output Data

Direction	Data name	Contents and condition	Implementation specification
Output	software_id	・Indicates lower-layer communication software that has switched to stop state.	Required
Input	Return Value	TRUE: notice received, FALSE: notice cannot be received	Optional

(22) Lower-layer communication software address table data size acquisition (optional)

Acquires the number of pairs of address data from the lower-layer communication software (lower-layer address tables). Table 3.23 shows the input and output specifications.

Table 3.23 Input and Output Data for the Lower-Layer Communication Software Address Table Data Size Acquisition Service

Direction	Data name	Content and conditions	Remarks
Input	device_id	* Specifies the lower-layer communication software ID acquired by the lower-layer communication software type request service.	Optional
Output	data_number	* Indicates the number of pairs of address data in the lower-layer address tables.	Required
Output	Return Value	TRUE: normal, FALSE: abnormal	Optional

(23) Acquires the lower-layer address table data from the lower-layer communication software. The data consists of data pairs and the number of data pairs. Each data pair contains the hardware type, hardware address and MAC address data and a flag indicating whether or not the home node is a master router.

Table 3.24 Input and Output Data for the Lower-Layer Communication Software Address Table Data Acquisition Service

Direction	Data name	Content and conditions	Remarks
Input	device_id	* Specifies the lower-layer communication software ID acquired by the lower-layer communication software type request service.	Optional
Output	data_number	* Indicates the number of pairs of address data in the lower-layer address table data.	Required
Output	hardwareaddress	* Gives a list of the hardware addresses that correspond to the address tables in the lower-layer address table data.	Required
Output	node_id	* Gives a list of the NodeIDs of the address tables in the lower-layer address table data.	Required
Output	masterrouter_flag	* If the node (address table in the lower-layer address table data) is a master router, “1” is given. Otherwise, “0” is given.	Required
Output	Return Value	TRUE: normal, FALSE: abnormal	Optional

(24) Master router notification (optional)

* Sets a flag in the lower-layer communication software to indicate whether or not the home node is a master router.

Table 3.25 Input and Output Data for the Master Router Notification Service

Direction	Data name	Content and conditions	Remarks
Input	device_id	* Specifies the lower-layer communication software ID acquired by the lower-layer communication software type request service.	Optional
Output	masterRouter_Flag	* "1" is specified if it is a master router. Otherwise, "0" is specified.	Required
Output	Return Value	TRUE: normal, FALSE: abnormal	Optional

(25) Hardware address data acquisition (optional)

Acquires the hardware address data from the lower-layer communication software. The output data is a hardware address.

Table 3.26 Input and Output Data for the Hardware Address Data Acquisition Service

Direction	Data name	Content and conditions	Remarks
Input	device_id	* Specifies the lower-layer communication software ID acquired by the lower-layer communication software type request service.	Optional
Output	hardwareaddress	* Gives the hardware addresses in the lower-layer address table data.	Required
Output	Return Value	TRUE: normal; FALSE: abnormal	Optional

(26) NodeID list acquisition

Acquires the NodeID list from the lower-layer communication software.

Table 3.27 Input and Output Data for the NodeID List Acquisition Service

Direction	Data name	Content and conditions	Remarks
Input	device_id	* Specifies the lower-layer communication software ID acquired by the lower-layer communication software type request service.	Optional
Output	node_id_list	* Gives a list of NodeIDs held by the lower-layer communication software.	Required

(27) Master router data acquisition

Acquires the master router data from the lower-layer communication software.

Table 3.28 Input and Output Data for the Master Router Data Acquisition Service

Direction	Data name	Content and conditions	Remarks
Input	device_id	* Specifies the lower-layer communication software ID acquired by the lower-layer communication software type request service.	Optional
Output	result,	* Indicates whether or not a master router is present	Required
Output	master_node_id	* Master router node ID	Required

(28) Hardware address conversion request

Requests the lower-layer communication software to provide the hardware address that corresponds to the NodeID sent to the lower-layer communication software.

Table 3.29 Input and Output Data for the Hardware Address Conversion Request Service

Direction	Data name	Content and conditions	Remarks
Input	device_id	* Specifies the lower-layer communication software ID acquired by the lower-layer communication software type request service.	Optional
Input	node_id	* Indicates the NodeID for which conversion is to be performed.	Required
Output	hardwareaddress	* Pointer to the hardware address	Required
Output	hardwareaddress_len	* Pointer to the hardware address size	Required

Chapter 4 Level 2 ECHONET Common Lower-Layer Communication Interface Specifications

This chapter provides API detailed specifications in light of the interchangeability of the software to be developed using this interface as the Level 2 ECHONET Common Lower-Layer Communication Interface. The specifications provided in this chapter cover cases in which API processing is mounted in the Protocol Difference Absorption Processing Block (a form in which processing of the Protocol Difference Absorption Processing Block is called by the ECHONET Communications Processing Block).

The Level 2 ECHONET Common Lower-Layer Communication Interfaces intended for the ANSI Standard C language (hereafter referred to as C language) are specified in ECHONET Standard Version 2.10.

4.1 List of Level 2 ECHONET Common Lower-Layer Communication Interface Functions for C Language

The following twenty-two functions are specified as functions of the Level 2 ECHONET Common Lower-Layer Communication Interface for C language. “Level 2 Optional” need not be mounted. (For example, function No.10 need not be mounted in the Protocol Difference Absorption Processing Block based on the specifications allowing discrete lower-layer communication software operation to be started in the same way when the Protocol Difference Absorption Processing Block operation start function is executed.) The functions shown in this section shall be implemented to conform to Level 2.

Table 4.1 List of Level 2 ECHONET Common Lower-Layer Communications Interface Functions for C Language

No.	Function name	Name	Remarks
1	ClcGetDevID	Lower-layer communication software mounting information request function	Required
2	ClcInit	Initialization request function	Required
3	ClcRequestRun	Operation start request function	Required
4	ClcSetTrouble	High-order layer fault notice function	Optional
5	ClcStart	Warm start request function	Required
6	ClcSuspend	Operation suspension request function	Optional
7	ClcWakeUp	Operation restart request function	Optional
8	ClcGetProData	Protocol difference absorption processing block profile data acquisition request function	Required
9	ClcGetStatus	Protocol difference absorption processing block status data acquisition request function	Optional
10	ClcInitAll	Complete initialization request function	Optional
11	ClcStop	Communication stop request function	Optional
12	ClcHalt	Complete stop request function	Optional
13	ClcLowInit	Lower-layer communication software initialization request function	Optional
14	ClcLowRequestRun	Lower-layer communication software operation start request function	Optional
15	ClcLowStart	Lower-layer communication software warm start request function	Optional
16	ClcLowSuspend	Lower-layer communication software suspension request function	Optional
17	ClcLowWakeUp	Lower-layer communication software operation restart request function	Optional
18	ClcGetLowProData	Lower-layer communication software profile data acquisition request function	Required
19	ClcGetLowStatus	Lower-layer communication software status data acquisition request function	Required
20	ClcSendData	Data transmission request function	Required
21	ClcGetSemdResult	Transmission result request function	Optional
22	ClcSendCancel	Transmission stop request function	Optional

No.	Function name	Name	Remarks
23	ClcReceiveData	Received data request function	Required
24	ClcGetNodeID	Node ID acquisition request function	Required
25	ClcSetNodeID	Node ID setup request function	Optional
26	ClcLowInitAll	Lower-layer communication software complete initialization request function	Optional
27	ClcLowStop	Lower-layer communication software communication stop request function	Optional
28	ClcLowHalt	Lower-layer communication software complete stop request function	Optional

4.2 C Language-oriented Level 2 ECHONET Common Lower-Layer Communication Interface Detailed Specifications

This section describes the details of each function shown in Table 4.1 for the following seven items:

(1) Name
Indicates function name.

(2) Function
Explains function.

(3) Syntax
Indicates function syntax.

(4) Explanation
Provides detailed specifications for arguments and variables.

(5) Return value
Indicates return value.

(6) Structure
Indicates any structure specifications.

(7) Notes/restrictions
Indicates any relevant precautions or restrictions.

4.2.1 ClcGetDevID

(1) Name

Lower-layer communication software mounting information function

(2) Function

Requests lower-layer communication software ID information indicating the number and type of lower-layer communication software programs that can be operated via a protocol difference absorption processing block.

(3) Syntax

```
BOOL   ClcGetDeviceID (
           unsigned char    *device_num    /*  [OUT] Information on the number of operable
                                               lower-layer communication software */
           unsigned char    *device_idset  /*  [OUT] Information on operable lower-layer
                                               communication software ID */
)
```

(4) Explanation

*device_num : Pointer to the number of operative lower-layer communication software programs.

*device_idset : Pointer to ID information for operative lower-layer communication software. In the place indicated by the pointer, items of information exist, the number of which is specified by device_num. The relationship between lower-layer communication software types and IDs is as indicated below:

Power Line Communication Protocol A and D Systems	0x11 – 0x1F
Specific low-power RF	0x31 – 0x3F
Extended HBS	0x41 – 0x4F
IrDA_Control	0x51 – 0x5F
LonTalk®	0x61 – 0x6F
Bluetooth™	0x61 – 0x6F
Ethernet	0x81 – 0x8F
IEEE802.11/11b	0x91 to 0x9F
Power Line Communication Protocol C System	0xA1

(5) Return value

0: Failed acquisition

1: Successful acquisition

(6) Structure

None

(7) Notes/restrictions

It is presumed that this function is called prior to the initialization request function

(ClcInit) and operation start request function (ClsRequestRun).

4.2.2 ClcInit

(1) Name
Initialization request function

(2) Function
Requests that all existing lower-layer communication software programs and associated protocol difference absorption processing block be initialized (cold start) and the MAC address acquired again. Upon receipt of this request, the protocol difference absorption processing block cold-starts all lower-layer communication software programs that can be cold-started, places them in the communication stop state, and is initialized in accordance with the initialization parameter.

(3) Syntax

```
BOOL   ClcInit (
           CLC_INIT_DATA  *init_data      /*    [IN] Pointer to initialization parameter */
)
```

(4) Explanation
*init_data : Pointer to initialization parameter for Protocol Difference Absorption Processing Block

(5) Return value
0: Failed initialization
1: Successful initialization

(6) Structure

```
typedef strut {
    short  sbuf_len; /*        Transmitting buffer size */
    unsigned char   *sbuf;   /*      Pointer to transmitting buffer */
    short  rbuf_len; /*        Receiving buffer size */
    unsigned char   *rbuf    /*      Pointer to receiving buffer */
    short  sholdtime,        /*      Maximum holding time for data transmitted by Protocol
                                        Difference Absorption Processing Block */
    short  rholdtime,        /*      Maximum holding time for data received by Protocol
                                        Difference Absorption Processing Block */
    unsigned char   clc_mode,      /*  Operation mode specifications */
                    0x00    Normal operation mode
                    0x01    Test/maintenance mode (details not specified) */
} CLC_INIT_DATA
```

(7) Notes/restrictions
If all lower-layer communication software programs have already been cold-started or warm-started, this function returns “Failed initialization”.

4.2.3 ClcRequestRun

(1) Name

Operation commencement request function

(2) Functions

Requests the protocol difference absorption processing section to start operating and make all of the lower-layer communication software programs present to start operating. As soon as the protocol difference absorption processing section receives this request, it shifts all lower-layer communication software programs to the “normal operation” state.

(3) Syntax

BOOL ClcRequestRun (void)

(4) Explanation

N/A

(5) Return value

0: Failed to start

1: Started successfully

(6) Structure used

None

(7) Notes and restrictions

This function is effective only for lower-layer communication software programs that are in the “not communicating” state. The return value will be “1” as long as there is one successful shift.

4.2.4 ClcSetTrouble

(1) Name

Fault notice function

(2) Function

Notifies Protocol Difference Absorption Processing Block of fault (error) status of application software and ECHONET Communications Processing Block.

(3) Syntax

```
BOOL   ClcSetTrouble (
       char htrouble_no    /*          [IN] High-order layer trouble No. /*
```

(4) Explanation

htrouble_no : Trouble No.
-1 Trouble removed
1 Application software is abnormal
2 ECHONET Communications Processing Block error

(5) Return value

0: Failed notice reception
1: Successful notice reception

(6) Structure

None

(7) Notes/restrictions

While an abnormality is reported, the protocol difference absorption processing block performs the following operations:

- Data reception process

  After notifying the lower-layer communication software of an abnormality in the higher-layer operation, the protocol difference absorption processing block refrains from performing data reception or discards received data.

- Data transmission request from ECHONET communication control processing block

  The protocol difference absorption processing block causes an error to be returned.

4.2.5 ClcStart

(1) Name

Warm-start request function

(2) Function

Requests that all existing lower-layer communication software programs and associated protocol difference absorption processing block be initialized (warm start) while retaining the MAC address. Upon receipt of this request, the protocol difference absorption processing block warm-starts all lower-layer communication software programs that can be warm-started and places them in the communication stop state.

(3) Syntax

BOOL ClcStart (void)

(4) Explanation

None

(5) Return value

0: Failed request
1: Successful request

(6) Structure

None

(7) Notes/restrictions

If all lower-layer communication software programs are already cold-started or warm-started, this function returns “Failed request”.

Upon receipt of this request, the protocol difference absorption processing block performs the following processes:

- Clears transmitting and receiving buffers
- Resets higher-layer fault setup
- Resets various status/work areas

4.2.6 ClcSuspend

(1) Name

Suspension request function

(2) Function

Requests that all existing lower-layer communication software programs and associated protocol difference absorption processing block suspend operation. Upon receipt of this request, the protocol difference absorption processing block places in suspension state all lower-layer communication software programs that can be suspended.

(3) Syntax

BOOL ClcSuspend (void)

(4) Explanation

None

(5) Return value

0: Failed suspension
1: Successful suspension

(6) Structure

None

(7) Notes/restrictions

If all lower-layer communication software programs are in a state other than normal operation, this function returns “Failed suspension”.

If the lower-layer communication software and protocol difference absorption processing block are in the midst of data transmission when this request is received, they terminate the series of transmission processes and switch into suspension state. If they are in the midst of data reception, on the other hand, they discard the received data and terminate the process.

The following operations are performed while in suspension state:

- Data reception

  No data is to be received.

- Data transmission request from ECHONET communication control processing block

  An error is returned.

4.2.7 ClcWakeUp

(1) Name

Operation restart function

(2) Function

Requests that all existing lower-layer communication software programs and the associated protocol difference absorption processing block exit suspension state and start running again. Upon receipt of this request, the protocol difference absorption processing block places all suspended lower-layer communication software programs in operation restart state.

(3) Syntax

BOOL ClcWakeup (void)

(4) Explanation

None

(5) Return value

0: Failed restart
1: Successful restart

(6) Structure

None

(7) Notes/restrictions

If all lower-layer communication software programs are in a state other than suspension, this function returns “Failed restart”.

4.2.8 ClcGetProData

(1) Name

Protocol difference absorption processing block profile data acquisition request function

(2) Function

Acquires profile data for protocol difference absorption processing block. The profile data requested by this function consists of the static portion of the property information for the protocol difference absorption processing block profile class stipulated in Part 2.

(3) Syntax

```
BOOL   ClcGetProData (
           CLC_PRO_DATA  *pro_data,     /*    [OUT] Pointer to profile data */
```

(4) Explanation

*pro_data : Pointer to profile data for protocol difference absorption processing block.

(5) Return value

0: Failed acquisition
1: Successful acquisition

(6) Structure

```
typedef strut {
        unsigned char     ver[3];       /*  Version No. of Protocol Difference Absorption
                                            Processing Block */
        unsigned char     maker[3];     /*  Manufacturer code */
        short             slen;         /*  Transmittable data length */
        short             rlen;         /*  Receivable data length */
} CLC_PRO_DATA
```

(7) Notes/restrictions

None

4.2.9 ClcGetStatus

(1) Name
Protocol difference absorption processing block status data acquisition request function

(2) Function
Requests status data for protocol difference absorption processing block. The status data that can be acquired by this function consists of dynamic status information such as the retained abnormality state and operation mode.

(3) Syntax
```
BOOL   LowGetStatus (
       CLC_STATUS   *status        /*  [OUT] Status of Protocol Difference
                                       Absorption Processing Block */
```

(4) Explanation
status : Returns status of Protocol Difference Absorption Processing Block.

(5) Return value
0: Failed acquisition
1: Successful acquisition

(6) Structure
```
typedef strut {
        char    upper_trouble;   /*  High-order layer fault code (0 to 127)
                                     No fault and removal of trouble (0) */
        char    clc_mode; /*     Operation mode code
                                     Normal operation (0)
                                     Test mode, such as maintenance (1)
                                     Monitoring mode (2) */
} CLC_STATUS;
```

(7) Notes/restrictions
None

4.2.10 ClcInitAll

(1) Name

Complete initialization request function

(2) Function

Requests that all existing lower-layer communication software programs and the associated protocol difference absorption processing block be initialized (cold start) and that the house code information and MAC address be acquired again. Upon receipt of this request, the protocol difference absorption processing block cold-starts all lower-layer communication software programs that can be cold-started, places them in the communication stop state, and is initialized in accordance with the initialization parameter.

(3) Syntax

```
BOOL   ClcInitAll
       CLC_INIT_DATA  *init_data     /*   [IN] Pointer to initialization parameter */
```

(4) Explanation

*ini_data : Pointer to initialization parameter for protocol difference absorption processing block.

(5) Return value

0: Failed initialization
1: Successful initialization

(6) Structure

```
typedef strut {
    short  sbuf_len; /*       Transmitting buffer size */
    short  *sbuf;    /*       Pointer to transmitting buffer */
    short  rbuf_len; /*       Receiving buffer size */
    short  *rbuf     /*       Pointer to receiving buffer */
    short  sholdtime,        /*    Maximum holding time for data transmitted by Protocol
                                   Difference Absorption Processing Block */
    short  rholdtime,        /*    Maximum holding time for data received by Protocol
                                   Difference Absorption Processing Block */
    unsigned char  clc_mode,    /*  Operation mode specifications */
                   0x00   Normal operation mode
                   0x01   Test/maintenance mode (details not specified) */
} CLC_INIT_DATA
```

(7) Notes/restrictions

If all lower-layer communication software programs are already cold-started, warm-started, or in the communication stop state, this function returns “Failed initialization”.

For lower-layer communication software that does not use house code information, the same process will be performed as in the case of an initialization request.

4.2.11 ClcStop

(1) Name

Communication stop request function

(2) Function

Requests that all existing lower-layer communication software programs and the associated protocol difference absorption processing block stop communications. Upon receipt of this request, the protocol difference absorption processing block places in the communication stop state all lower-layer communication software programs that can be placed in the communication stop state.

(3) Syntax

BOOL ClcStop (void)

(4) Explanation

None

(5) Return value

0: Failed stop
1: Successful stop

(6) Structure

None

(7) Notes/restrictions

If all lower-layer communication software programs are in a state other than normal operation, this function returns “Failed stop”.

If the lower-layer communication software and protocol difference absorption processing block are in the midst of data transmission when this request is received, they terminate the series of transmission processes and switch to the communication stop state. If they are in the midst of data reception, on the other hand, they discard the received data and terminate the process.

The following operations are performed while in the communication stop state:

- Data reception

  No data is to be received.

- Transmission request from ECHONET communication control processing block

  An error is returned.

4.2.12 ClcHalt

(1) Name

Complete stop request function

(2) Function

Requests that all existing lower-layer communication software programs and associated protocol difference absorption processing block stop completely. Upon receipt of this request, the protocol difference absorption processing block places in the stop state all lower-layer communication software programs that can be placed in the stop state.

(3) Syntax

BOOL ClcHalt (void)

(4) Explanation

None

(5) Return value

0: Failed stop
1: Successful stop

(6) Structure

None

(7) Notes/restrictions

If all lower-layer communication software programs are in a state other than normal operation, this function returns “Failed stop”.

If the lower-layer communication software and protocol difference absorption processing block are in the midst of data transmission when this request is received, they terminate the series of transmission processes and switch to the communication stop state. If they are in the midst of data reception, on the other hand, they discard the received data and terminate the process.

The following operations are performed while in the communication stop state:

- Data reception

  No data is to be received.

- Transmission request from ECHONET communication control processing block

  An error is returned.

4.2.13 ClcLowInit

(1) Name

Lower-layer communication software initialization request function

(2) Function

Requests that the specified lower-layer communication software and associated protocol difference absorption processing block be initialized (cold start) and the MAC address acquired again. Upon receipt of this request, the protocol difference absorption processing block cold-starts the specified lower-layer communication software, places it in the communication stop state, and is initialized in accordance with the initialization parameter.

(3) Syntax

```
BOOL   ClcLowInit (
       unsigned char    device_id,          /*[IN] Target software type ID for initialization */
       CLC_INIT_DATA  *clcinit_data /*[IN] Pointer to initialization parameter (1) */
       LOW_INIT_DATA  *lowinit_data,/*[IN] Pointer to initialization parameter (2) */
       void    *low_init                   /*[IN] Pointer to initialization parameter (3) */
```

(4) Explanation

device_id : Identification information for lower-layer communication software to be initialized.

- Power Line Communication Protocol A and D Systems 0x11 – 0x1F
- Low-power RF 0x31 – 0x3F
- Extended HBS 0x41 – 0x4F
- IrDA_Control 0x51 – 0x5F
- LonTalk® 0x61 – 0x6F
- IEEE802.11/11b 0x91 – 0x9F
- Power Line Communication Protocol C System 0xA1

*clcinit_data : Pointer to initialization parameter of Protocol Difference Absorption Processing Block

*lowinit_data : Pointer to initialization parameter of lower-layer communication software common specification item

*low_init : Pointer to initialization parameter that differs with individual lower-layer communication software programs
Contents of parameter are specified for each discrete lower-layer communication software program

(5) Return value

0: Failed initialization
1: Successful initialization

(6) Structure

```
typedef strut {
    short  sbuf_len; /*      Transmitting buffer size */
    short  *sbuf;    /*      Pointer to transmitting buffer */
    short  rbuf_len; /*      Receiving buffer size */
    short  *rbuf     /*      Pointer to receiving buffer */
    short  sholdtime,        /*    Maximum holding time for data transmitted by Protocol
                                   Difference Absorption Processing Block */
    short  rholdtime,        /*    Maximum holding time for data received by Protocol
                                   Difference Absorption Processing Block */
unsigned char       clc_mode,    /*  Operation mode specifications */
                    0x00    Normal operation mode
                    0x01    Test/maintenance mode (details not specified)
} CLC_INIT_DATA

typedef strut {
    short  sfholdtime,       /*    Maximum holding time for data transmitted by lower-layer
                                   communication software */
    short  rfholdtime,       /*    Maximum holding time for data received by lower-layer
                                   communication software */
    unsigned char   low_mode,    /*  Operation mode specifications */
    short  mac_len, /*      MAC address length */
    unsigned char   mac_ad[7],   /*  MAC address */
} LOW_INIT_DATA
```

* Except mac_ad[7], set NULL when initialization data is not found.
* When NULL is set in mac_len, mac_ad[7] is not significant. (When mac_len is set to NULL, this indicates that there is no MAC address setting.)

(7) Notes/restrictions

If the targeted lower-layer communication software is in the cold-start or warm-start state, this function returns “Failed initialization”.

4.2.14 ClcLowRequestRun

(1) Name

Lower-layer communication software operation start request function

(2) Function

Requests that the specified lower-layer communication software and associated protocol difference absorption processing block start running. Upon receipt of this request, the protocol difference absorption processing block places the specified lower-layer communication software in the normal operation state.

(3) Syntax

```
BOOL   ClcLowRequestRun (
       unsigned char   device_id,    /*  [IN] ID of lower-layer communication
                                          software targeted for start of operation */
```

(4) Explanation

device_id : Identification information for lower-layer communication software targeted for start of operation

Power Line Communication Protocol A and D Systems	
0x11 – 0x1F	
Low-power RF	0x31 – 0x3F
Extended HBS	0x41 – 0x4F
IrDA_Control	0x51 – 0x5F
LonTalk®	0x61 – 0x6F
Bluetooth™	0x71 – 0x7F
Ethernet	0x81 – 0x8F
IEEE802.11/11b	0x91 – 0x9F
Power Line Communication Protocol C System	
0xA1	

(5) Return value

0: Failed start
1: Successful start

(6) Structure

None

(7) Notes/restrictions

If the specified lower-layer communication software is not in the communication stop state, this function returns “Failed start”.

4.2.15 ClcLowStart

(1) Name

Lower-layer communication software warm start request function

(2) Function

Requests that the specified lower-layer communication software be initialized (warm start) while retaining the MAC address. Upon receipt of this request, the protocol difference absorption processing block warm-starts the specified lower-layer communication software and places it in the communication stop state.

(3) Syntax

```
BOOL   ClcLowStart (
       unsigned char   device_id,    /*  [IN] ID of lower-layer communication
                                          software to be warm-started */
```

(4) Explanation

device_id : Identification information for lower-layer communication software to be warm-started

Power Line Communication Protocol A and D Systems	
0x11 – 0x1F	
Low-power RF	0x31 – 0x3F
Extended HBS	0x41 – 0x4F
IrDA_Control	0x51 – 0x5F
LonTalk®	0x61 – 0x6F
Bluetooth™	0x71 – 0x7F
Ethernet	0x81 – 0x8F
IEEE802.11/11b	0x91 – 0x9F
Power Line Communication Protocol C System	
0xA1	

(5) Return value

0: Failed request
1: Successful request

(6) Structure

None

(7) Notes/restrictions

If the targeted lower-layer communication software is in the cold-start or warm-start state, this function returns “Failed request”.

Upon receipt of this request, the protocol difference absorption processing block associated with the specified lower-layer communication software performs the following processes:

- Clears transmitting and receiving buffers
- Resets higher-layer fault setup
- Resets various status/work areas

4.2.16 ClcLowSuspend

(1) Name

Lower-layer communication software suspension request function

(2) Function

Requests that the specified lower-layer communication software and associated protocol difference absorption processing block suspend operation. Upon receipt of this request, the protocol difference absorption processing block places the specified lower-layer communication software in the suspension state.

(3) Syntax

```
BOOL   ClcLowSuspend (
       unsigned char    device_id,        /*[IN] ID of lower-layer communication
                                          software to be suspended */
```

(4) Explanation

device_id : Identification information for lower-layer communication software to be suspended

Power Line Communication Protocol A and D Systems
0x11 – 0x1F
Low-power RF 0x31 – 0x3F
Extended HBS 0x41 – 0x4F
IrDA_Control 0x51 – 0x5F
LonTalk® 0x61 – 0x6F
Bluetooth™ 0x71 – 0x7F
Ethernet 0x81 – 0x8F
IEEE802.11/11b 0x91 – 0x9F
Power Line Communication Protocol C System
0xA1

(5) Return value

0: Failed suspension
1: Successful suspension

(6) Structure

None

(7) Notes/restrictions

If all lower-layer communication software programs are in a state other than normal operation, this function returns “Failed stop”.

If the lower-layer communication software and protocol difference absorption processing block are in the midst of data transmission when this request is received, they terminate the series of transmission processes and switch to the communication stop state. If they are in the midst of data reception, on the other hand, they discard the

received data and terminate the process.

The following operations are performed while in the communication stop state:

- Data reception

No data is to be received.

- Transmission request from ECHONET communication control processing block

An error is returned.

4.2.17 ClcLowWakeUp

(1) Name

Lower-layer communication software operation restart request function

(2) Function

Requests that the specified lower-layer communication software and associated protocol difference absorption processing block exit the suspension state. Upon receipt of this request, the protocol difference absorption processing block places the specified lower-layer communication software in the normal operation state.

(3) Syntax

```
BOOL   ClcLowWakeUp (
           unsigned char    device_id,     /*   [IN] ID of lower-layer communication
                                                  software targeted for an operation restart */
```

(4) Explanation

device_id : Identification information for lower-layer communication software targeted for an operation restart

Power Line Communication Protocol A and D Systems	
0x11 – 0x1F	
Low-power RF	0x31 – 0x3F
Extended HBS	0x41 – 0x4F
IrDA_Control	0x51 – 0x5F
LonTalk®	0x61 – 0x6F
Bluetooth™	0x71 – 0x7F
Ethernet	0x81 – 0x8F
IEEE802.11/11b	0x91 – 0x9F
Power Line Communication Protocol C System	
0xA1	

(5) Return value

0: Failed restart
1: Successful restart

(6) Structure

None

(7) Notes/restrictions

If the specified lower-layer communication software is in a state other than suspension, this function returns “Failed restart”.

4.2.18 ClcGetLowProData

(1) Name

Lower-layer communication software profile data acquisition request function

(2) Function

Acquires profile data for the specified lower-layer communication software and the address for a special process function used by the associated protocol difference absorption processing block. The profile data requested by this function is the property value information for the lower-layer communication software profile class, such as the software development manufacturer name and version number.

(3) Syntax

```
BOOL   ClcLowGetProData (
       unsigned char device_id,          /*   [IN] Lower-layer communication software
                                                 type ID */
       LOW_PRO_DATA  *pro_data,   /*   [OUT] Profile data */
       short (**chmacfunc) (unsigned char node_id, unsigned char *mac),
                         /*              [OUT] NodeID →MAC address conversion
                                                 function address */
       unsigned char (**chnodefunc) (unsigned char *mac),
                         /*              [OUT] MAC address →Node ID conversion
                                                 function address */
       void(**broadfunc) (const char bcast, char map[32])
                         /*              [OUT] Broadcast destination acquisition function
                                                 address */
```

(4) Explanation

device_id : Identification information for lower-layer communication software targeted for profile data acquisition.

Power Line Communication Protocol A and D Systems	
0x11 – 0x1F	
Low-power RF	0x31 – 0x3F
Extended HBS	0x41 – 0x4F
IrDA_Control	0x51 – 0x5F
LonTalk®	0x61 – 0x6F
Bluetooth™	0x71 – 0x7F
Ethernet	0x81 – 0x8F
IEEE802.11/11b	0x91 – 0x9F
Power Line Communication Protocol C System	
0xA1	

*pro_data : Pointer to profile data structure for the specified lower-layer communication software.

**chmacfunc : Pointer to address for the node ID-to-lower-layer communication software specific MAC address conversion function. If the specified lower-layer communication software has a node ID equal to the MAC address or effects simple linear conversion, NULL is returned.

The specifications for the function arguments to be passed are as follows:
Node ID prior to conversion.
*MAC address after conversion.

This function returns the MAC address size (in bytes).

**chnodefunc : Pointer to address for the lower-layer communication software specific MAC address-to-node ID conversion function. If the specified lower-layer communication software has a node ID equal to the MAC address or effects simple linear conversion, NULL is returned.
The specifications for the function argument to be passed is as follows:
*Pointer to MAC address to be converted.
This function returns the node ID derived from conversion.

**broadfunc : Pointer to address of broadcast destination acquisition function.
If the specified lower-layer communication software has broadcast function, NULL is returned.
The specifications for the function arguments to be passed are as follows:

bcast : [in] Complies with DEA broadcast target designation code except for 0xff (simultaneous broadcast).

map : [out] Returns address of broadcast destination node bitmap. The relationship between broadcast destination node addresses and bits is shown below:

map[0]-bit0 : NodeID 0 (0x00)
map[0]-bit1 : NodeID 1 (0x01)
……
map[1]-bit0 : NodeID 8 (0x08)
map[2]-bit1 : NodeID 9 (0x09)
……
map[31]-bit7 : NodeID 255 (0xFF)

(5) Return value

0: Failed acquisition
1: Successful acquisition

(6) Structure

```
typedef strut {
        unsigned char   kind;       /* Low-order medium types
                                       Power line: 0x31
                                       Low-power RF: 0x33
                                       Extended HBS: 0x34
                                       IrDA Control: 0x35
                                       LonTalk®: 0x36 */
        unsigned char   ver[3];     /* Lower-layer communication software version No. */
        unsigned char   maker[3];   /* Manufacturer code */
        short           mac_len;    /* MAC address length */
        unsigned char   mac_ad[7];  /* MAC address */
        unsigned char   mac_mask[7]; /* MAC address mask value */
        short           house_len;  /* House code length */
        short           *housecode; /* Pointer to house code information */
        short           slen;       /* Transmittable data length */
        short           rlen;       /* Receivable data length */
        short           broad;      /* Existence/non-existence of broadcast function
                                       (0: Non-existence, 1: Existence) */
        short           baud;       /* Transmission rate */
} LOW_PRO_DATA
```

(7) Notes/restrictions

None

4.2.19 ClcGetLowStatus

(1) Name

Lower-layer communication software status data acquisition request function

(2) Function

Requests status data for specified lower-layer communication software. The status data that can be acquired by this function consists of dynamic status information such as the retained abnormality state and operation mode.

(3) Syntax

```
BOOL   ClcGetLowStatus (
       unsigned char device_id,      /*  [IN] Lower-layer software type ID */
       LOW_STATUS  *status           /*  [OUT] Lower-layer communication software
                                         status */
```

(4) Explanation

device_id : Identification information for lower-layer communication software targeted for status data acquisition.

- Power Line Communication Protocol A and D Systems
  - 0x11 – 0x1F
- Low-power RF 0x31 – 0x3F
- Extended HBS 0x41 – 0x4F
- IrDA_Control 0x51 – 0x5F
- LonTalk® 0x61 – 0x6F
- Bluetooth™ 0x71 – 0x7F
- Ethernet 0x81 – 0x8F
- IEEE802.11/11b 0x91 – 0x9F
- Power Line Communication Protocol C System
  - 0xA1

*status : Pointer to status data structure for lower-layer communication software.

(5) Return value

0: Failed acquisition
1: Successful acquisition

(6) Structure

```
typedef strut {
        char    upper_trouble;      /*  High-order layer fault code (0 to 127)
                                        No fault or removal of trouble (0) */
        char    low_trouble;        /*  Lower-layer communication software block fault
                                        code (0 to 127)
                                        No fault and removal of trouble (0) */
        char    low_mode;/*         Operation mode code
                                        In normal operation state
                                        In maintenance or other test mode
```

```
                                    In monitoring mode
        short       state;      /*      Lower-layer communication software block status
                                            LOW_STS_STOP    : 0 Stop state
                                            LOW_STS_INI     : 1 Cold start state
                                            LOW_STS_RUN     : 2 Normal operation state
                                            LOW_STS_ESTOP   : 3 Error stop state
                                            LOW_STS_RST     : 4 Warm start state
                                            LOW_STS_CSTOP   : 5 Communication stop state
                                            LOW_STS_SPD     : 6 Suspension state
                                                /*
    } LOW_STATUS;
```

(7) Notes/restrictions

None

4.2.20 ClcSendData

(1) Name

Data transmission function

(2) Function

Requests that ECHONET data be transmitted with the specified lower-layer communication software. The protocol difference absorption processing block associated with the specified lower-layer communication software creates data in compliance with lower-layer communication software specifications and requests that the lower-layer communication software transmit the data.

(3) Syntax

```
short  ClcSendData (
        unsigned char   device_id,          /*  [IN] Lower-layer communication software
                                                type ID */
        const unsigned char *buf,           /*  [IN] Pointer to transmission data */
        short snd_sz,       /*              [IN] Transmission data size */
        unsigned char dnode_id,             /*  [IN] Transmission destination NodeID */
        unsigned char broad,                /*  [IN] Broadcast specifications */
```

(4) Explanation

device_id : Identification information for lower-layer communication software to be used for data transmission.

Power Line Communication Protocol A and D Systems
0x11 – 0x1F
Low-power RF 0x31 – 0x3F
Extended HBS 0x41 – 0x4F
IrDA_Control 0x51 – 0x5F
LonTalk® 0x61 – 0x6F
IEEE802.11/11b 0x91 – 0x9F
Power Line Communication Protocol C System
0xA1

*buf : Specifies pointer to ECHONET data to be transmitted. The ECHONET data to be passed here is one of the data exchanged between ECHONET communications processing blocks as stipulated in Part 2, Section 4.2.

snd_sz : Specifies size of data to be transmitted (outgoing ECHONET data size).

dnode_id : Specifies ID of transmission destination node within local subnet.
When the broadcast specification information (broad) is 0x00, the node ID of the default router or a router in an appropriate path is to be specified. If the broadcast specification information (broad) is 0xFF, this parameter is ignored.

broad : Specifies a broadcast.
0x00 : Specifies no broadcast or simultaneous broadcast within specified subnet.
0xFF : Specifies simultaneous broadcast within domain or within local subnet.

(5) Return value

CLC_BUFFER_FULL	: 0	Buffer full error
CLC_NO_ERROR	: 1	Transmission accepted
CLC_BUFFER_SIZE_ERROR	: 2	Buffer size error
CLC_STATE_ERROR	: 3	Internal error of lower-layer communication software
CLC_ADAPTER_ERROR	: 4	Device adapter processing failed

(6) Structure

None

(7) Notes/restrictions

If the specified lower-layer communication software is not in the normal operation state, this function returns “Internal error of lower-layer communication software”.

4.2.21 ClcGetSendResult

(1) Name

Transmission result acquisition function

(2) Function

Requests the results of the latest ECHONET data transmission that the specified lower-layer communication software performed in accordance with the data transmission function (ClcSendData). The protocol difference absorption processing block associated with the specified lower-layer communication software requests that the lower-layer communication software furnish the transmission results.

(3) Syntax

```
short    ClcGetSendResult (
         unsigned char device_id,      /*  [IN] Lower-layer communication software
                                           type ID */
         unsigned char  *result        /*  [OUT] Transmission result */
```

(4) Explanation

device_id : Identification information for lower-layer communication software targeted for transmission result acquisition.

Power Line Communication Protocol A and D Systems	
0x11 – 0x1F	
Low-power RF	0x31 – 0x3F
Extended HBS	0x41 – 0x4F
IrDA_Control	0x51 – 0x5F
LonTalk®	0x61 – 0x6F
Bluetooth™	0x71 – 0x67
Ethernet	0x81 – 0x8F
IEEE802.11/11b	0x91 – 0x9F
Power Line Communication Protocol C System	
0xA1	

*result : Pointer to the transmission results.

0x00: Successful transmission
0x01: Failed transmission
0xFF: No response

(5) Return value

CLC_CANCEL : 0 Transmission stop
CLC_NO_ERROR : 1 Normal
CLC_NO_SENDEND : 2 Transmitting status (transmission not completed)
CLC_INTERNAL_ERROR: 3 Internal error in lower-layer communication software
CLC_ADAPTER_ERROR : 4 Device adapter processing failed

(6) Structure

None

(7) Notes/restrictions

If the specified lower-layer communication software is not in the normal operation state, this function returns “Internal error of lower-layer communication software”. Note that “result” is meaningful only when the return value is normal (NO_ERROR).

4.2.22 ClcSendCancel

(1) Name

Transmission stop request function

(2) Function

Requests that the specified lower-layer communication software cancel ECHONET data transmission being performed in accordance with the data transmission function (ClcSendData). The protocol difference absorption processing block associated with the specified lower-layer communication software requests that the lower-layer communication software stop the transmission.

(3) Syntax

```
short    ClcSendCancel (
         unsigned char device_id        /*  [IN] Lower-layer communication software
                                            type ID */
```

(4) Explanation

device_id : Identification information for lower-layer communication software targeted for transmission stop request.

Power Line Communication Protocol A and D Systems	
0x11 – 0x1F	
Low-power RF	0x31 – 0x3F
Extended HBS	0x41 – 0x4F
IrDA_Control	0x51 – 0x5F
LonTalk®	0x61 – 0x6F
Bluetooth™	0x71 – 0x7F
Ethernet	0x81 – 0x8F
IEEE802.11/11b	0x91 – 0x9F
Power Line Communication Protocol C System	
0xA1	

(5) Return value

CLC_CANCEL : 0 Stop processing not executed because transmission was completed
CLC_NO_ERROR : 1 Normal
CLC_INTERNAL_ERROR: 3 Internal error in lower-layer communication software
CLC_ADAPTER_ERROR : 4 Device adapter processing failed

(6) Structure

None

(7) Notes/restrictions

If the specified lower-layer communication software is not in the normal operation state, this function returns “Internal error of lower-layer communication software”.

4.2.23 ClcReceiveData

(1) Name

Received-data request function

(2) Function

Requests received ECHONET data retained by specified lower-layer communication software. The protocol difference absorption processing block associated with the specified lower-layer communication software requests that the lower-layer communication software furnish the received data.

(3) Syntax

```
short    ClcReceiveData (
         unsigned char device_id,        /*  [IN] Lower-layer communication software
                                             type ID */
         unsigned char *buf,             /*  [OUT] Pointer to receiving buffer */
         short buf_sz        /*          [IN] Receiving buffer size */
         short  *rcv_cz      /*          [OUT] Received data size */
         unsigned char  *snode_id        /*  [OUT] Transmission source node ID */
```

(4) Explanation

device_id : Identification information for lower-layer communication software targeted for received-data request.

Power Line Communication Protocol A and D Systems	
0x11 – 0x1F	
Low-power RF	0x31 – 0x3F
Extended HBS	0x41 – 0x4F
IrDA_Control	0x51 – 0x5F
LonTalk®	0x61 – 0x6F
Bluetooth™	0x71 – 0x7F
Ethernet	0x81 – 0x8F
IEEE802.11/11b	0x91 – 0x9F
Power Line Communication Protocol C System	
0xA1	

*buf : Specifies pointer to receiving buffer.
buf_sz : Specifies receiving buffer size.
rcv_sz : Returns size of received data.
snode_id : Returns ID of transmission source node within the local subnet. If received data was transmitted from a remote subnet, router node ID is returned.

(5) Return value

CLC_NO_RECEIVE	: 0	No received data
CLC_NO_ERROR	: 1	Normal (with received data)
CLC_BUFFER_SIZE_ERROR	: 2	Buffer size error
CLC_INTERNAL_ERROR	: 3	Internal error in lower-layer communication software

(6) Structure

None

(7) Notes/restrictions

If the specified lower-layer communication software is not in the normal operation state, this function returns “Internal error of lower-layer communication software”.

4.2.24 ClcGetNodeID

(1) Name

Node ID acquisition request function

(2) Function

Requests the node ID information retained by the protocol difference absorption processing block associated with the specified lower-layer communication software.

(3) Syntax

```
BOOL   ClcGetNodeID (
           unsigned char    device_id,    /*  [IN] Lower-layer communication software
                                              type ID */
           unsigned char    *node_id,     /*  [OUT] NodeID */
```

(4) Explanation

device_id : Identification information for lower-layer communication software targeted for node ID information acquisition.

- Power Line Communication Protocol A and D Systems
  - 0x11 – 0x1F
- Low-power RF 0x31 – 0x3F
- Extended HBS 0x41 – 0x4F
- IrDA_Control 0x51 – 0x5F
- LonTalk® 0x61 – 0x6F
- Bluetooth™ 0x71 – 0x7F
- Ethernet 0x81 – 0x8F
- IEEE802.11/11b 0x91 – 0x9F
- Power Line Communication Protocol C System
  - 0xA1

node_id : NodeID code

(5) Return value

0: Failed NodeID acquisition
1: Successful NodeID acquisition

(6) Structure

None

(7) Notes/restrictions

If no node ID is retained, this function returns “Failed NodeID acquisition”.

4.2.25 ClcSetNodeID

(1) Name

Node ID setup request function

(2) Function

Updates node ID information retained by the protocol difference absorption processing block associated with the specified lower-layer communication software.

(3) Syntax

```
short    ClcSetSNodeID (
         unsigned char device_id,      /*  [IN] Lower-layer communication software
                                           type ID */
         unsigned char node_id         /*  [IN] NodeID information */
```

(4) Explanation

Sets NodeID corresponding to lower-layer communication software recognized by the Protocol Difference Absorption Processing Block.

device_id : Identification information for lower-layer communication software targeted for node ID information acquisition.

- Power Line Communication Protocol A and D Systems 0x11 – 0x1F
- Low-power RF 0x31 – 0x3F
- Extended HBS 0x41 – 0x4F
- IrDA_Control 0x51 – 0x5F
- LonTalk® 0x61 – 0x6F
- Bluetooth™ 0x71 – 0x7F
- Ethernet 0x81 – 0x8F
- IEEE802.11/11b 0x91 – 0x9F
- Power Line Communication Protocol C System 0xA1

node_id : NodeID code

(5) Return value

CLC_NO_CHEANGE : 0 Cannot be changed with software
CLC_NO_ERROR : 1 Normal
CLC_INTERNAL_ERROR: 3 Internal error of lower-layer communication software

(6) Structure

None

(7) Notes/restrictions

When node ID information retained by the protocol difference absorption processing block is updated, the protocol difference absorption processing block updates the MAC

address of the associated lower-layer communication software in accordance with the updated node ID value.

4.2.26 ClcLowInitAll

(1) Name

Lower-layer communication software complete initialization request function

(2) Function

Requests that the specified lower-layer communication software and associated protocol difference absorption processing block be initialized (cold start) and that the house code information and MAC address acquired again. Upon receipt of this request, the protocol difference absorption processing block cold-starts the specified lower-layer communication software, places it in the communication stop state, and is initialized in accordance with the initialization parameter.

(3) Syntax

```
BOOL   ClcLowInit (
       unsigned char    device_id,          /*[IN] Target software type ID for initialization */
       CLC_INIT_DATA  *init_data     /*[IN] Pointer to initialization parameter (1) */
       LOW_INIT_DATA  *lowinit_data, /*[IN] Pointer to initialization parameter (2) */
       void    *low_init                   /*[IN] Pointer to initialization parameter (3) */
```

(4) Explanation

device_id : Identification information for lower-layer communication software targeted for complete initialization.

- Power Line Communication Protocol A and D Systems 0x11 – 0x1F
- Low-power RF 0x31 – 0x3F
- Extended HBS 0x41 – 0x4F
- IrDA_Control 0x51 – 0x5F
- LonTalk® 0x61 – 0x6F
- Bluetooth™ 0x71 – 0x7F
- Ethernet 0x81 – 0x8F
- IEEE802.11/11b 0x91 – 0x9F
- Power Line Communication Protocol C System 0xA1

*init_data : Pointer to initialization parameter for protocol difference absorption processing block.

*lowinit_data : Pointer to initialization parameter for lower-layer communication software common specification items.

*low_init : Pointer to initialization parameter that varies with lower-layer communication software. The parameter is stipulated variously for all lower-layer communication software programs.

(5) Return value

0: Failed NodeID acquisition
1: Successful NodeID acquisition

(6) Structure

```
typedef strut {
    short  sbuf_len; /*      Transmitting buffer size */
    short  *sbuf;    /*      Pointer to transmitting buffer */
    short  rbuf_len; /*      Receiving buffer size */
    short  *rbuf     /*      Pointer to receiving buffer */
    short  sholdtime,        /*    Maximum holding time for data transmitted by Protocol
                                   Difference Absorption Processing Block */
    short  rholdtime,        /*    Maximum holding time for data received by Protocol
                                   Difference Absorption Processing Block */
    unsigned char  clc_mode,    /*  Operation mode specifications */
                   0x00   Normal operation mode
                   0x01   Test/maintenance mode (details not specified)
} CLC_INIT_DATA

typedef strut {
    short  sfholdtime,       /*    Maximum holding time for data transmitted by
                                   Lower-layer communication software */
    short  rfholdtime,       /*    Maximum holding time for data received by Lower-layer
                                   communication software */
    unsigned char  low_mode,    /*  Operation mode specifications */
    short  mac_len, /*      MAC address length */
    unsigned char  mac_ad[7],   /*  MAC address */
} LOW_INIT_DATA
```

* Except mac_ad[7], set NULL when initialization data is not found.
* When NULL is set in mac_len, mac_ad[7] is not significant. (When mac_len is set to NULL, this indicates that there is no MAC address setting.)

(7) Notes/restrictions

If the specified lower-layer communication software is in cold start, warm start, or communication stop state, this function returns “Failed initialization”.

For lower-layer communication software that does not use house code information, the same process will be performed as in the case of an initialization request.

4.2.27 ClcLowStop

(1) Name

Lower-layer communication software communication stop request function

(2) Function

Requests that the specified lower-layer communication software and associated protocol difference absorption processing block stop communications. Upon receipt of this request, the protocol difference absorption processing block places the specified lower-layer communication software in the communication stop state.

(3) Syntax

```
BOOL   ClcLowStop (
       unsigned char    device_id      /*   [IN] Lower-layer communication software ID
                                            */
```

(4) Explanation

device_id : Identification information for lower-layer communication software targeted for communication stop.

Power Line Communication Protocol A and D Systems
0x11 – 0x1F

Low-power RF	0x31 – 0x3F
Extended HBS	0x41 – 0x4F
IrDA_Control	0x51 – 0x5F
LonTalk®	0x61 – 0x6F
Bluetooth™	0x71 – 0x7F
Ethernet	0x81 – 0x8F
IEEE802.11/11b	0x91 – 0x9F

Power Line Communication Protocol C System
0xA1

(5) Return value

0: Failed restart
1: Successful restart

(6) Structure

None

(7) Notes/restrictions

If all lower-layer communication software programs are in a state other than normal operation, this function returns “Failed stop”.

If the lower-layer communication software and protocol difference absorption processing block are in the midst of data transmission when this request is received, they terminate the series of transmission processes and switch to the communication stop state. If they are in the midst of data reception, on the other hand, they discard the received data and terminate the process.

The following operations are performed while in the communication stop state:

- Data reception
  No data is to be received.
- Transmission request from ECHONET communication control processing block
  An error is returned.

4.2.28 ClcLowHalt

(1) Name

Lower-layer communication software deactivation request function

(2) Functions

Makes a request to deactivate the specified lower-layer communication software programs and the protocol difference absorption processing section. Upon receiving this request, the protocol difference absorption processing section shifts the specified lower-layer communication software programs to the “stop” state.

(3) Syntax

```
BOOL  ClcLowHalt (
unsigned char   device_id    /*[IN] lower-layer communication software ID  */
)
```

(4) Explanation

device_id: Information for identifying the lower-layer communication software programs to be deactivated

Power Line Communication Protocol A and D Systems	
0x11 – 0x1F	
Low-power RF	0x31 – 0x3F
Extended HBS	0x41 – 0x4F
IrDA_Control	0x51 – 0x5F
LonTalk®	0x61 – 0x6F
Bluetooth™	0x71 – 0x7F
Ethernet	0x81 – 0x8F
IEEE802.11/11b	0x91 – 0x9F
Power Line Communication Protocol C System	
0xA1	

(5) Return value

0: Deactivation failed

1: Deactivated successfully

(6) Structure used

None

(7) Notes and restrictions

The return value for this function will be “0” when the specified lower-layer communication software programs are in the “cold start” or “warm start” state.

If this request is received while an electronic message is being transmitted, the lower-layer communication software programs and protocol difference absorption processing section will shift to the “stop” state after completing the series of processing for that message. If this request is received while an electronic message is being received, the message will be discarded and the processing will be terminated.

The operation during the “stop” state is as follows:

* Reception of electronic messages:
Electronic messages are not received.

* Electronic message transmission requests from the ECHONET communication control processing section:
Returns errors.

4.2.29 ClcLowGetAddressTableDataSize

(1) Name

Lower-layer communication software address table data size acquisition function

(2) Functions

Acquires the number of pairs of lower-layer address table data from the lower-layer communication software.

(3) Syntax

BOOL

/*[IN] lower-layer communication software ID */

/*[OUT] number of pairs (destination) */

(4) Explanation

The output data consists of a pointer to the number of pairs of data.

device_id : Lower-layer communication software identification information

Power Line Communication Protocol A and D Systems	0x11 to 0x1F
Low-power RF	0x31 to 0x3F
Extended HBS	0x41 to 0x4F
IrDA_Control	0x51 to 0x5F
LonTalk®	0x61 to 0x6F
Bluetooth™	0x71 to 0x7F
Ethernet	0x81 to 0x8F
IEEE802.11/11b	0x91 – 0x9F
Power Line Communication Protocol C System	0xA1

data_number : Pointer to the number of pairs of address tables in the lower-layer address table data

(5) Return value

0: abnormal

1: normal

(6) Structure used

None

(7) Notes and restrictions

4.2.30 ClcLowGetAddressTableData

(1) Name

Lower-layer communication software address table data acquisition function

(2) Functions

Acquires the lower-layer address table data from the lower-layer communication software.

(3) Syntax

BOOL

/*[IN] lower-layer communication software ID */

/*[IN/OUT] number of pairs (destination) */

/*[OUT] address table structure */

(4) Explanation

The input data (data_number) is a pointer to the number of pairs of address tables acquired using ClcLowGetAddressTableDataSize. The output data comprises array data of a structure that consists of a flag to indicate whether or not the node is a master router, the NodeID, the hardware address of each data pair and the number of pairs of address tables stored.

device_id : Lower-layer communication software identification information

Power Line Communication Protocol A and D Systems	0x11 to 0x1F
Low-power RF	0x31 to 0x3F
Extended HBS	0x41 to 0x4F
IrDA_Control	0x51 to 0x5F
LonTalk®	0x61 to 0x6F
Bluetooth™	0x71 to 0x7F
Ethernet	0x81 to 0x8F
IEEE802.11/11b	0x91 to 0x9F
Power Line Communication Protocol C System	0xA1

data_number : Pointer to the number of pairs of address tables in the lower-layer address table data

addresstable: Head pointer to the array of address table structure that contains a flag

to indicate whether or not the node is a master router, the NodeID and the hardware address in the lower-layer address table data

(5) Return value

0: abnormal

1: normal

(6) Structure used

/* hardware address data size */

/* hardware address (the data is stored in such a manner that the lower bytes are filled first) */

/* NodeID*/

/* Identifier to indicate whether or not the node is a master router. The value is “1” if it is a master router and “0” otherwise. */

(7) Notes and restrictions

None

4.2.31 ClcLowSetMasterRouterFlag

(1) Name

Master router setting function

(2) Functions

Notifies the lower-layer communication software as to whether or not the home node is a master router.

(3) Syntax

BOOL

/*[IN] lower-layer communication software ID */
/*[IN] master router identification flag */

(4) Explanation

device_id: Lower-layer communication software identification information

Power Line Communication Protocol A and D Systems	0x11 to 0x1F
Low-power RF	0x31 to 0x3F
Extended HBS	0x41 to 0x4F
IrDA_Control	0x51 to 0x5F
LonTalk®	0x61 to 0x6F
Bluetooth™	0x71 to 0x7F
Ethernet	0x81 to 0x8F
IEEE802.11/11b	0x91 to 0x9F
Power Line Communication Protocol C System	0xA1

masterRouter_Flag: “1” is specified if the node is a master router.
Otherwise, “0” is specified.

(5) Return value

0: abnormal
1: normal

(6) Structure used

None

(7) Notes and restrictions

None

4.2.32 ClcLowGetHardwareAddress

(1) Name

Hardware address data acquisition function

(2) Functions

Acquires the hardware address data from the lower-layer communication software.

(3) Syntax

```
BOOL
  ClcLowGetHardwareAddress (
  unsigned char device_id,                /*[IN]  lower-layer communication software ID */
  unsigned char* hardwareaddresssize, /*[OUT] hardware address size*/
  unsigned char* hardwareaddress       /*[OUT] hardware address */
```

(4) Explanation

The output data is a hardware address.

device_id : Lower-layer communication software identification information

Power Line Communication Protocol A and D Systems	0x11 to 0x1F
Low-power RF	0x31 to 0x3F
Extended HBS	0x41 to 0x4F
IrDA_Control	0x51 to 0x5F
LonTalk®	0x61 to 0x6F
Bluetooth™	0x71 to 0x7F
Ethernet	0x81 to 0x8F
IEEE802.11/11b	0x91 to 0x9F
Power Line Communication Protocol C System	0xA1

hardwareaddresssize: Pointer to the size of the hardware addresses in the lower-layer address table data

hardwareaddress: Pointer to the array of hardware addresses corresponding to the address tables in the lower-layer address table data

(5) Return value

0: abnormal
1: normal

(6) Structure used
None

(7) Notes and restrictions
None

4.3.33 ClcGetNodeIDList

(1) Name

NodeID list acquisition function

(2) Functions

Acquires the NodeID list from the lower-layer communication software.

(3) Syntax

BOOL

/*[IN] lower-layer communication software ID */

/*[OUT] NodeID list */

(4) Explanation

device_id : Lower-layer communication software identification information

Power Line Communication Protocol A and D Systems	0x11 to 0x1F
Low-power RF	0x31 to 0x3F
Extended HBS	0x41 to 0x4F
IrDA_Control	0x51 to 0x5F
LonTalk®	0x61 to 0x6F
Bluetooth™	0x71 to 0x7F
Ethernet	0x81 to 0x8F
IEEE802.11/11b	0x91 to 0x9F
Power Line Communication Protocol C System	0xA1

*node_id_list: Pointer to the 32-element array (NodeID list) held by the lower-layer communication software. The data is as follows:

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

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

26th byte	30	31	32	33	34	35	36	37
27th byte	28	29	2A	2B	2C	2D	2E	2F
28th byte	20	21	22	23	24	25	26	27
29th byte	18	19	1A	1B	1C	1D	1E	1F
30th byte	10	11	12	13	14	15	16	17
31st byte	08	09	0A	0B	0C	0D	0E	0F
32nd byte	00	01	02	03	04	05	06	07

(5) Return value

0: abnormal

1: normal

(6) Structure used

None

(7) Notes and restrictions

None

4.2.34 ClcGetMasterRouterInfo

(1) Name

Master router data acquisition function

(2) Functions

Acquires the master router data from the lower-layer communication software.

(3) Syntax

BOOL

/*[IN] lower-layer communication software ID */

/*[OUT] whether or not a master router is present */

/*[OUT] master router Node ID */

(4) Explanation

device_id: Lower-layer communication software identification information

Power Line Communication Protocol A and D Systems	0x11 to 0x1F
Low-power RF	0x31 to 0x3F
Extended HBS	0x41 to 0x4F
IrDA_Control	0x51 to 0x5F
LonTalk®	0x61 to 0x6F
Bluetooth™	0x71 to 0x7F
Ethernet	0x81 to 0x8F
IEEE802.11/11b	0x91 to 0x9F
Power Line Communication Protocol C System	0xA1

*result : Pointer to indicate whether or not a master router is present

0x00: not present

0x01: present

* master_node_id: Pointer to the master router NodeID value

(5) Return value

0: abnormal

1: normal

(6) Structure used

None

(7) Notes and restrictions

None

4.2.35 ClcLowReqToHardwareAddress

(1) Name

Hardware address conversion request function

(2) Functions

Requests the lower-layer communication software to provide the hardware address that corresponds to the NodeID sent to the lower-layer communication software.

(3) Syntax

```
BOOL  ClcLowGetHardwareAddress (
unsigned char   device_id,   /*[IN]  lower-layer communication software ID*/
unsigned char   node_id,    /*[IN]  NodeID for which conversion is to be performed */
unsigned char*  hardwareaddress, /*[OUT] pointer to the hardware address */
unsigned char*  hardwareaddress_len/*[OUT] pointer to the hardware address size */
)
```

(4) Explanation

device_id : Lower-layer communication software identification information

Power Line Communication Protocol A and D Systems	0x11 to 0x1F
Low-power RF	0x31 to 0x3F
Extended HBS	0x41 to 0x4F
IrDA_Control	0x51 to 0x5F
LonTalk®	0x61 to 0x6F
Bluetooth™	0x71 to 0x7F
Ethernet	0x81 to 0x8F
IEEE802.11/11b	0x91 to 0x9F
Power Line Communication Protocol C System	0xA1

node_id : NodeID for which conversion is to be performed

hardwareaddress : A pointer to the converted hardware address is returned.

hardwareaddress_len : A pointer to the size of the converted hardware address is returned.

(5) Return value

0: abnormal

1: normal

(6) Structure used

None

(7) Notes and restrictions

None