Part VI
ECHONET Discrete Lower-layer Communication Interface Specification

Release information (as of August 7th 2001)

a) Version1.0	March 18th	2000 released	Open to the consortium member
	July	2000	Open to the public
b) Version1.01	May 23rd	2001	Open to the public
c) Version2.00	August 7th	2001	Open to the consortium member

Notes: On and after Version2.00, Powerline communication protocol has drawn together as Powerline communication A.

Contents

Chapter 1 Overview

1.1 Basic Concept

The ECHONET Discrete Lower-layer Communication Interface Specification is provided to specify software interface to implement processing and information exchange between the Protocol Difference Absorption Processing Block and individual Lower-layer Communication Software described in Section 1.2 “Positioning on Communication Layers”. This specification provides the rules on functions for levels 1 and 2 of the discrete lower-layer communication interface specification for cases in which input/output data items and concrete language are specified with regard to APIs, on the assumption they are supported by all Lower-layer Communication Software. APIs specified as “Required” at each level are mandatory for function implementation (mounted as an API) since they are specified as a standard. On the other hand, APIs specified as “Optional” must implement their functions with this API, but the function may not necessarily be implemented. Levels 1 and 2 of the discrete lower-layer communication interface specification are based on the concept of levels 1 and 2 of the basic API.

1.2 Positioning on Communication Layers

The shaded area in Fig. 1.1 shows the positioning of the discrete lower-layer communication interface. This interface is positioned between the Protocol Difference Absorption Processing Block and the Lower-layer Communication Software and implements processing calls and information exchange, thereby connecting the communications middleware and the Lower-layer Communication Software.

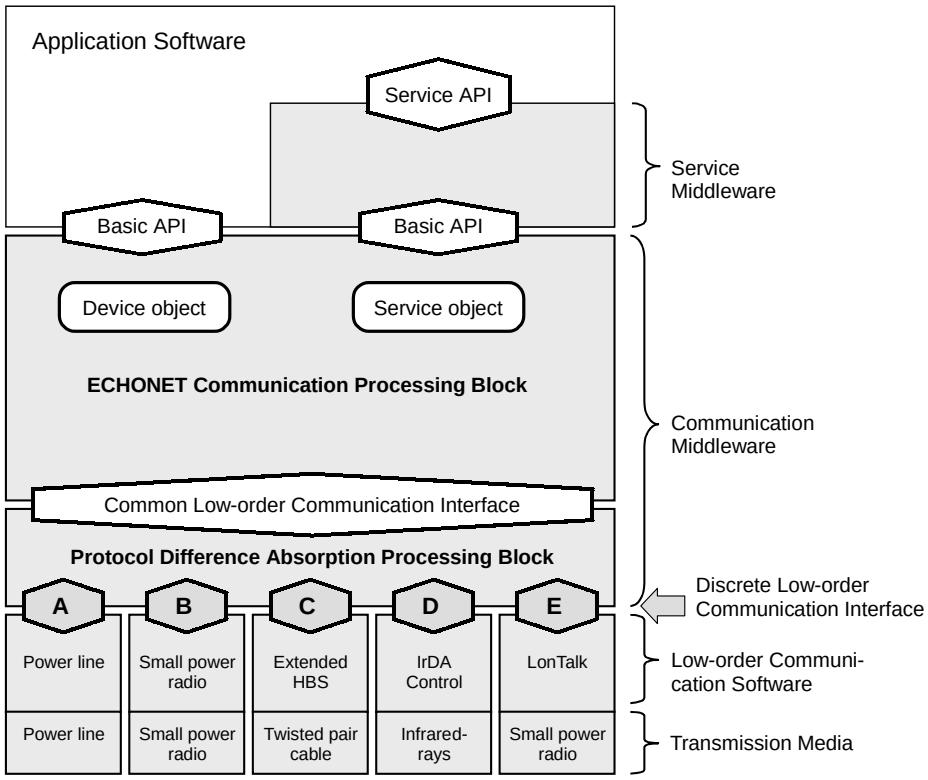

Fig. 1.1 Positioning of Discrete Lower-layer Communication Interface

Chapter 2 ECHONET Discrete Lower-layer Communication Interface Function Specification

2.1 List of ECHONET Discrete Lower-layer Communication Interface Functions

Table 2.1 shows a list of ECHONET discrete lower-layer communication interface functions supported by all Lower-layer Communication Software. Each Lower-layer Communication Software shall be provided with these functions. The functional details of each API are explained in the following section.

Table 2.1 List of ECHONET Discrete Low-layer Communication Interface Functions

No.	API name	Function	Remarks
1	Request for initialization	Requests initialization of Lower-layer Communication Software.	
2	Request for operation start	Requests that Lower-layer Communication Software start operation.	
3	Fault notice	Notifies Lower-layer Communication Software of fault (error) status of high-order layer from Protocol Difference Absorption Processing Block.	
4	Request for reset	Asks Lower-layer Communication Software for reset processing.	
5	Request for suspension	Asks Lower-layer Communication Software to suspend operation.	
6	Request for operation restart	Asks Lower-layer Communication Software to restart operation.	
7	Profile acquisition	Gets profile data of Lower-layer Communication Software.	
8	Status acquisition	Gets status, such as processing fault or address redundancy, from Lower-layer Communication Software.	
9	Request for data transmission	Asks Lower-layer Communication Software to send data.	
10	Transmission result acquisition	Requests data transmission result from Lower-layer Communication Software.	
11	Request for transmission stop	Requests that data transmission be stopped.	
12	Request for received data	Asks Lower-layer Communication Software for received data.	
13	Address information acquisition	Gets information on Mac addresses and house codes recognized by Lower-layer Communication Software.	
14	Address information setting	Sets Mac addresses, house codes, etc. for Lower-layer Communication Software.	
15	Request for Mac address translation	Requests translation of NodeID information into corresponding Mac address.	
16	Request for NodeID translation	Requests translation of Mac address into corresponding NodeID.	
17	Request for broadcast destination acquisition	Requests target NodeID for broadcast.	

2.2 ECHONET Discrete Lower-layer Communication Interface Detailed Specification

(1) Request for initialization

Requests initialization of Lower-layer Communication Software. Upon receiving this request, the Lower-layer Communication Software is initialized using the specified information. However, normal operation is started when “Request for operation start” is received.

(2) Request for operation start

Requests that Lower-layer Communication Software start operation.

(3) Fault notice

Notifies Lower-layer Communication Software of fault (error) status of high-order layer from Protocol Difference Absorption Processing Block.

(4) Request for reset

Requests that Lower-layer Communication Software reset. Upon receiving this request, the Lower-layer Communication Software executes reset processing and waits in the initial status.

(5) Request for suspension

Asks Lower-layer Communication Software to suspend operation. Upon receiving this request, the Lower-layer Communication Software shall not accept (stop) requests other than “Request for operation restart” or “Request for reset” from the Protocol Difference Absorption Processing Block, even if they come from the Protocol Difference Absorption Processing Block proper or communication line. When the Lower-layer Communication Software receives this request, it shall enter a stop status after completing the series of transmit/receive processing if it is transmitting or receiving data. (Data transmission/reception shall not be stopped midway.)

(6) Request for operation restart

Requests that Lower-layer Communication Software clear suspension status and restart operation. Upon receiving this request, the Lower-layer Communication Software restarts the stopped operation.

(7) Profile acquisition

Requests profile data for Lower-layer Communication Software. The profile data requested by this function shall be profile information such as the software developer name and version No.

(8) Status acquisition
Requests status data for Lower-layer Communication Software. The status data requested by this function shall be status information such as fault status or processing status.

(9) Request for data transmission
Requests that Lower-layer Communication Software send specified ECHONET data.

(10) Transmission result acquisition
Asks Lower-layer Communication Software for transmission result of data requested by “Request for data transmission”.

(11) Request for transmission stop
Requests that Lower-layer Communication Software stop transmission of data requested by “Request for data transmission”.

(12) Request for received data
Requests that Lower-layer Communication Software deliver received data.

(13) Address information acquisition
Asks Lower-layer Communication Software for address information, such as recognized Mac addresses or house codes.

(14) Address information setting
Sets address information, such as Mac addresses or house codes, for Lower-layer Communication Software.

(15) Request for physical address translation
Delivers NodeID information to Lower-layer Communication Software and requests Mac address of corresponding communications software.

(16) Request for NodeID translation
Delivers Mac address information to Lower-layer Communication Software and requests corresponding NodeID information (the value converted in accordance with the rules of translation for each Lower-layer Communication Software).

(17) Request for broadcast destination acquisition
Delivers broadcast specification addresses (information in 2nd byte of DEA at broadcast specification) to Lower-layer Communication Software and requests target NodeID information for broadcast (the value extracted in accordance with the broadcast group specification for each Lower-layer Communication Software).

Chapter 3 Level 1 ECHONET Discrete Lower-layer Communication Interface Specification

3.1 List of Level 1 ECHONET Discrete Lower-layer Communication Interfaces

Table 3.1 shows a list of level 1 ECHONET discrete lower-layer communication interfaces supported by the Lower-layer Communication Software. Mounted APIs conforming to level 1 should be provided with the input/output data items specified in the next section. Details of each data item and multiple data items may be implemented as a single data item, or a single data item may be divided into multiple data items. Argument names shall be indicated for reference. Function explanation and input/output items for each API are specified on and after the next page.

Table 3.1 List of Level 1 ECHONET Discrete Lower-layer Communication Interfaces (1/2)

No.	API name	Function outline	Mounting specification
1	Request for Lower-layer Communication Software type	Requests type and ID of Lower-layer Communication Software.	Required
2	Request for initialization	Requests initialization of Lower-layer Communication Software.	Required
3	Request for operation start	Requests that Lower-layer Communication Software start operation.	Required
4	Fault notice	Notifies Lower-layer Communication Software of fault (error) status of high-order layer from Protocol Difference Absorption Processing Block.	Optional
5	Request for reset	Asks Lower-layer Communication Software for reset processing.	Optional
6	Request for suspension	Asks Lower-layer Communication Software to suspend operation.	Optional
7	Request for operation restart	Asks Lower-layer Communication Software to restart operation.	Optional
8	Profile acquisition	Gets profile data of Lower-layer Communication Software.	Required
9	Status acquisition	Gets Lower-layer Communication Software information, such as processing fault and address redundancy.	Required
10	Request for data transmission	Asks Lower-layer Communication Software to transmit data.	Required
11	Transmission result acquisition	Asks Lower-layer Communication Software for data transmission result.	Required
12	Request for transmission stop	Asks Lower-layer Communication Software to stop data transmission.	Optional
13	Request for received data	Asks Lower-layer Communication Software to deliver received data.	Required

Table 3.1 List of Level 1 ECHONET Discrete Low-layer Communication Interfaces (2/2)

No.	API name	Function outline	Mounting specification
14	Address information acquisition	Gets address information, such as Mac addresses or house codes, recognized by Lower-layer Communication Software.	Required
15	Address information setting	Sets address information such as Mac addresses and house codes for Lower-layer Communication Software.	Optional
16	Request for physical address translation	Requests translation of NodeID into corresponding Mac address.	Optional
17	Request for NodeID translation	Requests translation of Mac address into corresponding NodeID.	Optional
18	Request for broadcast destination acquisition	Requests target NodeID for broadcast.	Optional

3.2 Level 1 ECHONET Discrete Lower-layer Communication Interface Detail Specification

Data inputs/outputs are shown below for each API indicated in Table 3.1 in the previous section. In the following table, “Input” indicates that data is transferred from the Protocol Difference Absorption Processing Block to the Lower-layer Communication Software (input viewed from the Lower-layer Communication Software), while “Output” indicates that data is transferred from the Lower-layer Communication Software to the Protocol Difference Absorption Processing Block (output viewed from the Lower-layer Communication Software). Regarding mounting, the contents of this data should be provided as inputs/outputs, but the transfer method (for example, using structures or transferring pointer information for transfer buffer) is not specified for level 1.

The ECHONET discrete lower-layer communication interface shown in level 1 indicates the inputs/outputs of data that can be based on the same specification even if the Lower-layer Communication Software type is different. Accordingly, regarding interfaces other than “Request for Lower-layer Communication Software type”, the argument to indicate the type of Lower-layer Communication Software is set as an input. However, every input is of “Optional” specification, so ordinary nodes (i.e., nodes, such as routers, in which multiple Lower-layer Communication Software does not exist) may be operated without specification.

(1) Request for Lower-layer Communication Software (mandatory function to be mounted)

Requests type of Lower-layer Communication Software (power line, low-power radio, etc.). Table 3.2 shows input/output specifications.

Table 3.2 List of Low-layer Communication Software Type Request API Input/Output Data

Direction	Data name	Contents and condition	Remarks
Input	–		
Output	device_id	Specifies type and ID of Lower-layer Communication Software. Identification of power line communications software, specific low-power radio communications software, extended HBS communications software, LON communications software, IrDA Control communications software, etc. shall be enabled.	Required
Output	Return Value	TRUE: Successful initialization, FALSE: Failed initialization	Optional

(2) Request for initialization (mandatory function to be mounted)

Request for initialization of Lower-layer Communication Software. Upon receiving this request, the lower-layer communication is initialized using the specified information. However, normal operation is executed when “Request for operation start” is received. Table 3.3 shows input/output specifications.

Table 3.3 List of Initialization Request API Input/Output Data

Direction	Data name	Contents and condition	Remarks
Input	device_id	Specifies type of Lower-layer Communication Software.	Optional
Input	sfholdtime	Maximum holding time for transmission data. This is the maximum transmission waiting time when network is busy and data cannot be transmitted. If transmission cannot be performed after this time elapses, this attempt is abandoned.	Optional
Input	rfholdtime	Maximum holding time for received data. This is the maximum holding time for received data when the received data is not read by the Protocol Difference Absorption Processing Block (i.e., when the received data cannot be delivered to the Protocol Difference Absorption Processing Block) after receipt of data. If the received data cannot be delivered after this time elapses, it is abandoned.)	Optional
Input	sfbuf	Information on buffer size and buffer position for transmission data to be received from Protocol Difference Absorption Processing Block by Lower-layer Communication Software.	Optional
Input	rfbuf	Information on buffer size and buffer position for received data to be delivered to Protocol Difference Absorption Processing Block by Lower-layer Communication Software.	Optional
Input	snfbuf	Information on buffer size and buffer position for transmission data to be sent to ECHONET of Lower-layer Communication Software.	Optional
Input	rnfbuf	Information on buffer size and buffer position for receiving data from ECHONET of Lower-layer Communication Software.	Optional
Input	low_mode	Specifies operation mode for Lower-layer Communication Software such as test mode or on-network data monitoring mode	Optional
Input	mac_ad	Mac address information	Optional
Input	mac_len	Mac address size information	Optional
Input	housecode	House code information	Optional
Input	lowinit	Initialization parameter that differs for each Lower-layer Communication Software	Optional
Output	Return Value	TRUE: Successful initialization, FALSE: Failed initialization	Optional

(3) Request for operation start (mandatory function to be mounted)

Requests that Lower-layer Communication Software start operation. Table 3.4 shows input/output specifications.

Table 3.4 List of Operation Start Request API Input/Output Data

Direction	Data name	Contents and condition	Remarks
Input	device_id	Specifies type of Lower-layer Communication Software.	Optional
Output	Return Value	TRUE: Successful operation start, FALSE: Failed operation start	Optional

(4) Fault notice

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

Table 3.5 List of Fault Notice API Input/Output Data

Direction	Data name	Contents and condition	Remarks
Input	device_id	Specifies type of Lower-layer Communication Software.	Optional
Input	htrouble_no	Notifies trouble No.	Required
Output	Return Value	TRUE: Successful notice, FALSE: Failed notice	Optional

(5) Request for reset

Asks Lower-layer Communication Software to reset. Upon receiving this request, the Lower-layer Communication Software executes reset processing and waits in the initial status. Table 3.6 shows input/output specifications.

Table 3.6 List of Reset Request API Input/Output Data

Direction	Data name	Contents and condition	Remarks
Input	device_id	Specifies type of Lower-layer Communication Software.	Optional
Output	Return Value	TRUE: Reset acceptable, FALSE: Not acceptable	Optional

(6) Request for suspension

Asks Lower-layer Communication Software to suspend operation. Upon receiving this request, the Lower-layer Communication Software shall not accept requests other than “Request for operation restart (LowWakeup)” or “Request for reset (LowReset)” from the Protocol Difference Absorption Processing Block, even if they come from the Protocol Difference Absorption Processing Block or the lower-layer communication driver. Table 3.7 shows input/output specifications.

Table 3.7 List of Suspension Request API Input/Output Data

Direction	Data name	Contents and condition	Remarks
Input	device_id	Specifies type of Lower-layer Communication Software.	Optional
Output	Return Value	TRUE: Suspension acceptable, FALSE: Not acceptable	Optional

(7) Request for operation request

Asks Lower-layer Communication Software to clear suspension status and restart operation. Table 3.8 shows input/output specifications.

Table 3.8 List of Operation Restart Request API Input/Output Data

Direction	Data name	Contents and condition	Remarks
Input	device_id	Specifies type of Lower-layer Communication Software.	Optional
Output	Return Value	TRUE: Successful restart, FALSE: Restart disabled (including failure)	Optional

(8) Profile acquisition (mandatory function to be mounted)

Asks Lower-layer Communication Software for profile data. The profile data to be requested by this function shall be static information, such as software developer name, version No., etc. Table 3.9 shows input/output specifications.

Table 3.9 List of Profile Acquisition API Input/Output Data

Direction	Data name	Contents and condition	Remarks
Input	device_id	Specifies type of Lower-layer Communication Software.	Optional
Output	kind	Lower-layer medium type information Identification information on power line, low-power radio, extended HBS, IrDA Control, and LON	Required
Output	mac_ad	Mac address information	Required
Output	housecode	House code information	Required
Output	version_No	Version information for Lower-layer Communication Software	Optional
Output	maker	Maker information	Optional
Output	srlen	Transmittable/receivable data length	Optional
Output	broad	Broadcast function available/unavailable	Optional
Output	baud	Transmission rate	Optional
Output	chmac_info	Physical address to NodeID translation information (translation function address information, etc.)	Optional
Output	chnode_info	NodeID to physical address translation information (translation function address information, etc.)	Optional
Output	chbroad_info	NodeID to broadcast destination physical address translation information (translation function address information, etc.)	Optional
Output	Return Value	TRUE: Normal, FALSE: Error	Optional

(9) Status acquisition (mandatory function to be mounted)

Asks Lower-layer Communication Software for status data for Lower-layer Communication Software. The status data to be requested by this function shall be static information such as fault status or processing status. Table 3.10 shows input/output specifications.

Table 3.10 List of Status Acquisition API Input/Output Data

Direction	Data name	Contents and condition	Remarks
Input	device_id	Specifies type of Lower-layer Communication Software.	Optional
Output	state	Transition state information (information on stop status, initializing status, completion of initialization, normal processing status, error stop status, etc.)	Required
Output	upper_trouble	Information recognized as fault of high-order layer	Optional
Output	low_trouble	Information recognized as fault of Lower-layer Communication Software	Optional
Output	low_mode	Information on operation mode (monitoring mode, test mode, etc.)	Optional
Output	Return Value	TRUE: Normal, FALSE: Error	Optional

(10) Request for data transmission (mandatory function to be mounted)

Asks Lower-layer Communication Software to send specified ECHONET data. Table 3.11 shows input/output specifications.

Table 3.11 List of Data Transmission Request API Input/Output Data

Direction	Data name	Contents and condition	Remarks
Input	device_id	Specifies type of Lower-layer Communication Software.	Optional
Input	send_data	Information on transmission request data of ECHONET data format	Required
Input	d_add	Transmitting destination Mac address information	Required
Input	mac_len	Transmitting source Mac address size information	Optional
Input	broad	Information on selection of broadcast/individual	Optional
Output	Return Value	TRUE: Normal, FALSE: Error	Optional

(11) Transmission result acquisition

Asks Lower-layer Communication Software for transmission result of data requested by "Request for data transmission". Table 3.12 shows input/output specifications.

Table 3.12 List of Transmission Result Acquisition API Input/Output Data

Direction	Data name	Contents and condition	Remarks
Input	device_id	Specifies type of Lower-layer Communication Software	Optional
Output	result	Information on transmitting status, normal termination of transmission, abnormal termination of transmission, or stop of transmission	Required
Output	Return Value	TRUE: Normal, FALSE: Error	Optional

(12) Request for transmission stop

Asks Lower-layer Communication Software to stop transmission of data requested by "Request for data transmission". Table 3.13 shows input/output specifications.

Table 3.13 List of Transmission Stop Request API Input/Output Data

Direction	Data name	Contents and condition	Remarks
Input	device_id	Specifies type of Lower-layer Communication Software.	Optional
Output	Return Value	TRUE: Successful stop, FALSE: Failure to stop (already transmitted)	Optional

(13) Request for data reception (mandatory function to be mounted)

Asks Lower-layer Communication Software for received data. Table 3.14 shows input/output specifications.

Table 3.14 List of Data Reception Request API Input/Output Data

Direction	Data name	Contents and condition	Remarks
Input	device_id	Specifies type of Lower-layer Communication Software.	Optional
Output	receive_data	Received data information	Required
Output	s_add	Transmitting source Mac address information	Required
Output	mac_len	Transmitting source Mac address size information	Optional
Output	Return Value	TRUE: Normal, FALSE: Error (error indication code such as "No received data")	Optional

(14) Address information acquisition (mandatory function to be mounted)

Requests address information recognized by Lower-layer Communication Software. Table 3.15 shows input/output specifications.

Table 3.15 List of Address Information Acquisition API Input/Output Data

Direction	Data name	Contents and condition	Remarks
Input	device_id	Specifies type of Lower-layer Communication Software.	Optional
Output	mac_ad	Mac address information	Required
Output	mac_len	Mac address size information	Optional
Output	houscode	House code information	Optional
Output	Return Value	TRUE: Normal; FALSE: Error (error indication code such as “NodeID not set” or “Specified device_id error”)	Optional

(15) Address information setting

Sets address information for Lower-layer Communication Software. Table 3.16 shows input/output specifications.

Table 3.16 List of Address Setting API Input/Output Data

Direction	Data name	Contents and condition	Remarks
Input	device_id	Specifies type of Lower-layer Communication Software.	Optional
Input	mac_ad	Mac address information	Required
Input	mac_len	Mac address size information	Optional
Input	houscode	House code information	Optional
Output	Return Value	TRUE: Normal, FALSE: Error (error indication code such as “Set disable”)	Optional

(16) Request for physical address translation

Delivers NodeID information to Lower-layer Communication Software and requests physical address from corresponding Lower-layer Communication Software. Table 3.17 shows input/output specifications.

Table 3.17 List of Physical Address Translation Request API Input/Output Data

Direction	Data name	Contents and condition	Remarks
Input	device_id	Specifies type of Lower-layer Communication Software.	Optional
Input	node_id	NodeID information to request translation.	Required
Output	mac_ad	Mac address information corresponding to specified NodeID information	Required
Output	mac_len	Mac address size information	Optional
Output	Return Value	TRUE: Normal, FALSE: Error (error indication code such as “Translate disable”	Optional

(17) Request for NodeID translation

Delivers physical address information to Lower-layer Communication Software and requests corresponding NodeID information (the value translated in accordance with the rules of translation for each Lower-layer Communication Software). Table 1.18 shows input/output specifications.

Table 1.18 List of NodeID Translation Request API Input/Output Data

Direction	Data name	Contents and condition	Remarks
Input	device_id	Specifies type of Lower-layer Communication Software.	Optional
Input	mac_ad	Mac address information to request translation	Required
Output	mac_len	Mac address size information	Optional
Output	node_id	NodeID information corresponding to specified Mac address information	Required
Output	Return Value	TRUE: Normal, FALSE: Error (error indication code such as “translate disable”)	Optional

(18) Request for broadcast destination acquisition

Delivers broadcast specified address information (DEA 2nd byte information when broadcast is specified) to Lower-layer Communication Software and requests target NodeID information for broadcast (the value extracted in accordance with the broadcast group specification for each Lower-layer Communication Software). Table 3.19 shows input/output specifications.

Table 3.19 List of Broadcast Destination Acquisition Request API Input/Output Data

Direction	Data name	Contents and condition	Remarks
Input	device_id	Specifies type of Lower-layer Communication Software.	Optional
Input	broad_adinfo	Broadcast address information to request target NodeID information for broadcast	Required
Output	node_num	Number information on target NodeID for broadcast	Required
Output	node_idinfo	Information on target NodeID for broadcast	Required
Output	Return Value	TRUE: Normal, FALSE: Error (error indication code such as “translate disable”)	Optional

Chapter 4 Level 2 ECHONET Discrete Lower-layer Communication Interface Specification

This Section provides the API detailed specification in consideration of the interchangeability of the software to be developed by using this interface as the level 2 ECHONET discrete lower-layer communication interface. The level 2 ECHONET discrete lower-layer communication interfaces intended for ANSI Standard C language (hereafter referred to as C language) are specified as the API specification of the level 2 ECHONET discrete lower-layer communication interface in the ECHONET Standard V 1.0 β.

4.1 List of Level 2 ECHONET Discrete Lower-layer Communication Interfaces

The following 18 functions are specified as functions of level 1 ECHONET discrete lower-layer communication interfaces. “Level 2 Optional” may not be mounted as a function. For function implementation, the functions shown in this section shall be implemented so that they conform to level 3. Function names are indicated as reference, and the names at linkage shall be changeable. Detailed specifications for each discrete lower-layer communication interface are provided in the next section.

Table 1.17 List of Level 2 ECHONET Discrete Low-layer Communication Interface Functions (1/2)

No.	API name	API function name	Function	Remarks
1	Request for Lower-layer Communication Software type	LowGetDevID	Requests type and ID of Lower-layer Communication Software.	Required
2	Request for initialization	LowInit	Requests initialization of Lower-layer Communication Software.	Required
3	Request for operation start	LowRequestRun	Requests that Lower-layer Communication Software start operation.	Required
4	Fault notice	LowSetTrouble	Notifies Lower-layer Communication Software of fault (error) status of high-order layer from Protocol Difference Absorption Processing Block.	Optional
5	Request for reset	LowReset	Asks Lower-layer Communication Software for reset processing.	Required
6	Request for suspension	LowSuspend	Asks Lower-layer Communication Software to suspend operation.	Required
7	Request for operation restart	LowWakeup	Asks Lower-layer Communication Software to restart operation.	Required

Table 1.17 List of Level 2 ECHONET Discrete Low-layer Communication Interface Functions (2/2)

No.	API name	API function name	Function	Remarks
8	Profile acquisition	LowGetProData	Gets profile data (static information) of Lower-layer Communication Software.	Required
9	Status acquisition	LowGetStatus	Gets dynamic status (processing fault, address redundancy, etc.) of Lower-layer Communication Software.	Required
10	Request for data transmission	LowSendData	Asks Lower-layer Communication Software to send data.	Required
11	Transmission result acquisition	LowGetSendResult	Asks Lower-layer Communication Software for data transmission result.	Required
12	Request for transmission stop	LowSendCancel	Asks Lower-layer Communication Software to stop data transmission.	Required
13	Request for received data	LowReceiveData	Asks Lower-layer Communication Software to exchange received data.	Required
14	Address information acquisition	LowGetAddress	Gets address information, such as Mac addresses or house codes, recognized by Lower-layer Communication Software.	Required
15	Address information setting	LowSetAddress	Sets such address information as Mac addresses and house codes for Lower-layer Communication Software.	Required
16	Request for physical address translation	LowReqToMac	Requests translation of NodeID into corresponding Mac address.	Optional
17	Request for NodeID translation	LowReqToID	Requests translation of Mac address into corresponding NodeID.	Optional
18	Request for broadcast destination acquisition	LowReqBcastAD	Requests target NodeID for broadcast.	Optional

4.2 Level 2 ECHONET Discrete Lower-layer Communication Interface Detail Specification

This section provides detailed specifications for each function shown in Table 4.1 with regard to the following seven items:

(1) Name
Indicates the function name.

(2) Function
Explains the function.

(3) Syntax
Indicates syntax of the function.

(4) Explanation
Explains detailed specifications of arguments and variables.

(5) Return value
Indicates return value.

(6) Structure
Specifies structure if it exists.

(7) Notes/restrictions
Indicates any notes or restrictions.

4.2.1 LowGetDevID

(1) Name

Request for Lower-layer Communication Software

(2) Function

Requests DeviceID information to indicate the identification for multiple Lower-layer Communication Software types. It is assumed that this request will be called ahead of "Initialization request function: Lowinit" and "Request for operation start: LowReqeustRun".

(3) Syntax

```
BOOL    LowInit (
        unsigned char   *dev_id     /*[OUT]  Type information for Lower-layer
                                               Communication Software */
)
```

(4) Explanation

dev_id : Identification information for Lower-layer Communication Software

Power line A system	0x11～0x1F
Power line B system	0x21～0x2F
Specific low-power radio	0x31～0x3F
Extended HBS	0x41～0x4F
IrDA_Control	0x51～0x5F
LON	0x61～0x6F

(5) Return value

0: Failed acquisition

1: Successful acquisition

(6) Structure

None

(7) Notes

None

(8) Restrictions

None

4.2.2 LowInit

(1) Name

Initialization request function

(2) Function

Requests initialization of Lower-layer Communication Software. Upon receiving this request, the Lower-layer Communication Software initializes the Lower-layer Communication Software using the specified information. However, normal operation is started when “Request for operation start: LowRequestRun” is received.

(3) Syntax

```
BOOL   LowInit (
       unsigned char device_id,          /*[IN] Lower-layer Communication Software type
                                              ID */
       LOW_INIT_DATA  *init_data,        /*[IN] Pointer to initialization parameter (1) */
       void     *low_init                /*[IN] Pointer to initialization parameter (2) */
)
```

(4) Explanation

device_id : Identification information for Lower-layer Communication Software

Power line A system	0x11～0x1F
Power line B system	0x21～0x2F
Specific low-power radio	0x31～0x3F
Extended HBS	0x41～0x4F
IrDA_Control	0x51～0x5F
LON	0x61～0x6F

*init_data : Pointer to initialization parameter of the common specification item

*low_init : Pointer to initialization parameter that differs with each Lower-layer Communication Software. Contents of parameter are specified for each discrete Lower-layer Communication Software. (See Section 4.2.)

(5) Return value

0: Failed initialization

1: Successful initialization

(6) Notes

None

(7) Structure

```
typedef strut {
        short           sfholdtime,     /* Maximum holding time for transmission data */
        short           rfholdtime,     /* Maximum holding time for received data */
        unsigned char   low_mode,       /* Operation mode specification */
        short           mac_len,        /* Mac address length */
        unsigned char   mac_ad[6],      /* Mac address */
} LOW_INIT_DATA
```

Except for mac_ad[6], when there is no initialization data, set NULL.

When mac_len is set to NULL, mac_ad[6] has no significance. (When mac_len is NULL, the mac address is not set.)

(8) Restrictions

None

4.2.3 LowRequestRun

(1) Name

Operation start request function

(2) Function

Requests that Lower-layer Communication Software start operation. Upon receiving this request, the Lower-layer Communication Software starts operation.

(3) Syntax

```
BOOL   LowRequestRun (
           unsigned char device_id  /*[IN]   Lower-layer Communication Software type ID */
)
```

(4) Explanation

device_id : Identification information for Lower-layer Communication Software
(For specific details of codes, see the LowInit function.)

(5) Return value

0: Failure to start

1: Successful start

(6) Notes

After execution of Lowinit, this function must be called before data is transmitted or received.

(7) Structure

None

(8) Restrictions

None

4.2.4 LowSetTrouble

(1) Name

High-order layer fault notice function

(2) Function

Notifies Lower-layer Communication Software of fault (error) status of high-order layer) from Protocol Difference Absorption Processing Block.

(3) Syntax

```
BOOL   LowSetTrouble (
       unsigned char device_id, /*[IN] Lower-layer Communication Software type ID */
       char  htrouble_no        /*[IN] High-order layer trouble No.
)
```

(4) Explanation

When a fault notice is in process, the Lower-layer Communication Software performs the following operations:

- Processing at data reception

  A NAK response is returned. Data is not received, or received data is abandoned.
- Request for transmission from Protocol Difference Absorption Processing Block

  Any request other than “Request for reset” (LowReset) is rejected and an error is returned.

device_id : Identification information for Lower-layer Communication Software (For specific details of codes, see the LowInit function.)

htrouble_no : Trouble No.

-1	Removal of trouble
1	Application software error
2	ECHONET communication processing block error
3	Protocol Difference Absorption Processing Block error

(5) Return value

0: Failed notice

1: Successful notice

(6) Notes

None

(7) Structure

None

(8) Restrictions

None

4.2.5 LowReset

(1) Name

Reset request function

(2) Function

Requests that Lower-layer Communication Software reset. Upon receiving this request, the Lower-layer Communication Software executes reset processing and waits in the initial status.

(3) Syntax

```
BOOL   LowReset (
           unsigned char device_id     /*[IN] Lower-layer Communication Software type ID */
)
```

(4) Explanation

When this request is received, the following processing is performed to reset:

- Clear transmitting/receiving buffer.
- Reset high-order layer fault setting.
- Reset various status and work areas.
- Reset communication hardware block.

device_id : Identification information for Lower-layer Communication Software
(For specific details of codes, see the LowInit function.)

(5) Return value

0: Failed request

1: Successful request

(6) Notes

None

(7) Structure

None

(8) Restrictions

None

4.2.6 LowSuspend

(1) Name

Suspension request function

(2) Function

Requests that Lower-layer Communication Software suspend operation. Upon receiving this request, the Lower-layer Communication Software shall not accept requests other than “Request for operation restart: LowWakeup” or “Request for reset: LowReset” from the Protocol Difference Absorption Processing Block, even if they come from the Protocol Difference Absorption Processing Block or the lower-layer communication driver.

(3) Syntax

```
BOOL    LowSuspend (
        unsigned char device_id      /*[IN] Lower-layer Communication Software type ID */)
```

(4) Explanation

When this request is received, the transmission processing series is terminated and operation stop processing is executed if data is being transmitted. If data is being received, the received data is abandoned and processing is terminated. The following operations are performed in the operation stop status:

- Data reception

  All data is abandoned. Upon receipt of data, no response shall be made in the case of such a medium as returns ACK/NAK.

- Request from Protocol Difference Absorption Processing Block

  Requests other than “Request for operation restart” (LowWakeup) and “Request for reset” (LowReset) are rejected, and an error code is returned.

device_id : Identification information for Lower-layer Communication Software
(For specific details of codes, see the LowInit function.)

(5) Return value

0: Failed suspension

1: Successful suspension

(6) Notes

None

(7) Structure

None

(8) Restrictions

None

4.2.7 LowWakeup

(1) Name

Lower-layer Communication Software operation restart function

(2) Function

Requests that Lower-layer Communication Software clear suspension status and restart operation. Upon receiving this request, the Lower-layer Communication Software restarts operation immediately.

(3) Syntax

```
BOOL    LowWakeup (
        unsigned char device_id     /*[IN] Lower-layer Communication Software type ID */)
```

(4) Explanation

device_id : Identification information for Lower-layer Communication Software
(For specific details of codes, see the LowInit function.)

(5) Return value

0: Failure to restart

1: Successful restart

(6) Notes

None

(7) Structure

None

(8) Restrictions

None

4.2.8 LowGetProData

(1) Name

Lower-layer Communication Software profile data acquisition function

(2) Function

Gets profile data for Lower-layer Communication Software and addresses of specific processing functions to be used in Protocol Difference Absorption Processing Block. The profile data requested by this function consists of information on property values of the profile object, such as the software developer name and version No.

(3) Syntax

```
BOOL   LowGetProData (
       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] Physical address translation function address*/
       unsigned char (**chnodefunc) (unsigned char *mac),
                    /*[OUT] Physical address to NodeID translation 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 (For specific details of codes, see the LowInit function.)

pro_data : Profile data of specified Lower-layer Communication Software

chmacfunc : Address of translation function from NodeID to physical address native to Lower-layer Communication Software is returned.
For a medium where a match exists between NodeID and physical address or for a medium that performs simple linear translation, NULL is returned.
Argument specifications of functions to be delivered are as follows:
node_id : [in] NodeID to be translated
mac : [out] Physical address that was translated

chnodefunc : Translation function address from physical address native to Lower-layer Communication Software to NodeID is returned. For a medium where a match exists between NodeID and physical address or for a medium that performs simple linear translation, NULL is returned.
Argument specifications of functions to be delivered are as follows:
mac : [in] Physical address to be translated
This function returns NodeID as a return value.

broadfunc : Broadcast destination acquisition function address is returned.
For Lower-layer Communication Software with a broadcast function, NULL is returned.
Argument specifications of functions to be delivered are as follows:
bcast : [in] Except 0xff (simultaneous broadcast), observe codes for DEA broadcast.

map	: [out]	Address to bit map of broadcast destination node is returned. Relationship between broadcast destination 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) Notes

None

(7) Structure

```
typedef strut {
        unsigned char   kind;           /* Lower-layer medium types
                                           Power line A: 0x31   Small power radio: 0x32
                                           Power line B: 0x33   Extended HBS: 0x34
                                           IrDAControl: 0x35    LON: 0x36 */
        unsigned char   ver[3];         /* Lower-layer Communication Software version
                                           No. */
        unsinged char   maker[3];       /* Maker code */
        short           mac_len;        /* MAC address length */
        unsigned char   mac_ad[6];      /* MAC address */
        unsigned char   mac_mask[6];    /* 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
```

(8) Restrictions

None

4.2.9 LowGetStatus

(1) Name

Lower-layer Communication Software status acquisition function

(2) Function

Asks Lower-layer Communication Software for status data for Lower-layer Communication Software. The status data obtained by this function is dynamic information, such as error status and processing status.

(3) Syntax

```
BOOL   LowGetStatus (
           unsigned char device_id    /*[IN]    Lower-layer Communication Software type ID */
           LOW_STATUS  *status        /*[OUT] Lower-layer Communication Software status */
)
```

(4) Explanation

device_id : Identification information for Lower-layer Communication Software
(For specific details of codes, see the LowInit function.)

status : Status of Lower-layer Communication Software is returned.

(5) Return value

0: Failed acquisition

1: Successful acquisition

(6) Notes

None

(7) Structure

```
typedef strut {
    char    upper_trouble;  /* High-order layer fault code (0～127)
                               No fault or removal of trouble (0) */
    char    low_trouble;    /* Lower-layer Communication Software block fault
                               code (0～127)
                               No fault or removal of trouble (0) */
    char    low_mode;       /* Operation mode code
                               Normal operation (0)
                               Test mode, such as maintenance (1)
                               Monitoring mode (2) */
    short   state;          /* Lower-layer Communication Software block status
                               LOW_STS_STOP(0)     : Stop status
                               LOW_STS_INI(1)      : Initializing status
                               LOW_STS_RUN(2)      : Normal processing status
                               LOW_STS_ESTOP(3)    : Error stop status */
} LOW_STATUS;
```

(8) Restrictions

None

4.2.10 LowSendData

(1) Name

Data transmission function

(2) Function

Asks Lower-layer Communication Software to transmit specified ECHONET data.

(3) Syntax

```
short   LowSendData (
            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 */
            const unsigned char *da,        /*[IN]  Physical address of transmission destination */
            unsigned char broad,            /*[IN]  Broadcast specification */
)
```

(4) Explanation

Data is transmitted to Lower-layer Communication Software.

The transmission data to be delivered in this case is “EDC+EDATA”. Upon receiving a request for transmission by this function, the Lower-layer Communication Software creates a packet in accordance with the lower-layer medium specification and transmits the packet to the lower-layer medium.

device_id : Identification information for Lower-layer Communication Software (For specific details of codes, see the LowInit function.)

buf : Specifies pointer to transmission data.

snd_sz : Specifies transmission data size.

da : Specifies physical address of transmission destination.
When broadcast is specified in the parameter “bkind”, this parameter is not used.

broad : Specifies broadcast.
No broadcast specification. Regarding others, observe the codes for SEA broadcast.

(5) Return value

LOW_BUFFER_FULL(0) : Buffer full error

LOW_NO_ERROR(1) : Transmission accepted

LOW_BUFFER_SIZE_ERROR(2) : Buffer size error

LOW_STATE_ERROR(3) : Internal error in Lower-layer Communication Software

(6) Notes

None

(7) Structure

None

(8) Restrictions

None

4.2.11 LowGetSendResult

(1) Name

Lower-layer Communication Software transmission result acquisition function

(2) Function

Gets transmission result of data requested by “Request for data transmission (LowSendData)” from Lower-layer Communication Software.

(3) Syntax

```
short    LowGetSendResult (
             unsigned char device_id,   /*[IN]   Lower-layer Communication Software type ID */
             unsigned char *result      /*[OUT]  Transmission result */
)
```

(4) Explanation

Obtains a transmission result from lower-layer communications module.

The result has significance only when the return value is NO_ERROR.

device_id : Identification information for Lower-layer Communication Software (For specific details of codes, see the LowInit function.)

result : Transmission result. 0x00: Successful transmission, 0x01: Failed transmission, 0xFF: No response

(5) Return value

LOW_CANCEL(0)	: Transmission stop
LOW_NO_ERROR(1)	: Normal
LOW_NO_SENDEND(2)	: Transmitting status (transmission not completed)
LOW_INTERNAL_ERROR(3)	: Internal error in Lower-layer Communication Software

(6) Notes

None

(7) Structure

None

(8) Restrictions

None

4.2.12 LowSendCancel

(1) Name

Lower-layer Communication Software transmission stop request function

(2) Function

Asks Lower-layer Communication Software to stop transmission of data requested by "Request for data transmission (LowSendData)".

(3) Syntax

```
short    LowSendCancel (
         unsigned char device_id  /*[IN]   Lower-layer Communication Software type ID */
)
```

(4) Explanation

Asks Lower-layer Communication Software to stop data transmission. Upon receiving this request, the Lower-layer Communication Software abandons all data in the transmitting buffer.

device_id : Identification information for Lower-layer Communication Software
(For specific details of codes, see the LowInit function.)

(5) Return value

LOW_CANCEL(0)	: No execution of stop processing because transmission has been completed
LOW_NO_ERROR(1)	: Normal
LOW_INTERNAL_ERROR(3)	: Internal error in Lower-layer Communication Software

(6) Notes

None

(7) Structure

None

(8) Restrictions

None

4.2.13 LowReceiveData

(1) Name

Data receive request function

(2) Function

Asks Lower-layer Communication Software for received data.

(3) Syntax

```
short    LowReceiveData (
         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  *sa         /*[OUT]   Physical address of transmitting source */
)
```

(4) Explanation

Obtains received data from Lower-layer Communication Software.

device_id : Identification information for Lower-layer Communication Software (For specific details of codes, see the LowInit function.)

buf : Specifies pointer (1st byte: EDC) to receiving buffer.

buf_sz : Specifies receiving buffer size.

Rcv_sz : Returns actual received data size.

Sa : Returns physical address of transmitting source.

(5) Return value

LOW_NO_RECEIVE(0)	: No received data
LOW_NO_ERROR(1)	: Normal (with received data)
LOW_BUFFER_SIZE_ERROR(2)	: Buffer size error
LOW_INTERNAL_ERROR(3)	: Internal error in Lower-layer Communication Software

(6) Notes

None

(7) Structure

None

(8) Restrictions

None

4.2.14 LowGetAddress

(1) Name

Lower-layer medium address information acquisition function

(2) Function

Requests address information recognized by Lower-layer Communication Software.

(3) Syntax

```
BOOL   LowGetAddress (
           unsigned char device_id,        /*[IN]   Lower-layer Communication Software type ID
                                                    */
           unsigned char *mac_ad,          /*[OUT]  MAC address */
           short          mac_len;            /* MAC address length */
           unsigned char  mac_ad[6];          /* MAC address */
           unsigned char  mac_mask[6];        /* MAC address mask value */
           short          house_len;          /* House code length */
           short          *housecode;         /* Pointer to house code information */
)
```

(4) Explanation

Requests MAC address and house code recognized by Lower-layer Communication Software.

device_id	: Identification information for Lower-layer Communication Software (For specific details of codes, see the LowInit function.)
mac_ad	: Gets MAC address.
mac_len	: MAC address size. Set a value of 0x01 or more.
housecode	: Gets house code.
housecode_len	: House code size. In the case of 0x00, no house code is set.

(5) Return value

0: Failed address acquisition

1: Successful address acquisition

(6) Notes

None

(7) Structure

None

(8) Restrictions

None

4.2.15 LowSetAddress

(1) Name

Lower-layer medium address setting function

(2) Function

None

(3) Syntax

```
short    LowSetAddress (
         unsigned char device_id,        /*[IN]  Lower-layer Communication Software type ID
                                                 */
         short          mac_len;          /* MAC address length */
         unsigned char  mac_ad[6];        /* MAC address */
         unsigned char  mac_mask[6];      /* MAC address mask value */
         short          house_len;        /* House code length */
         short          *housecode;       /* Pointer to house code information */
)
```

(4) Explanation

Sets MAC address and house code of Lower-layer Communication Software.

device_id	: Identification information for Lower-layer Communication Software (For specific details of codes, see the LowInit function.)
mac_ad	: Sets MAC address.
mac_len	: MAC address size In the case of 0x00, there is no request for MAC address setting.
housecode	: Sets house code.
housecode_len	: House code size In the case of 0x00, there is no request for house code setting.

(5) Return value

LOW_NO_CHANGE(0)	: Unchangeable with software
LOW_NO_ERROR(1)	: Normal
LOW_INTERNAL_ERROR(3)	: Internal error in Lower-layer Communication Software

(6) Notes

None

(7) Structure

None

(8) Restrictions

None

4.2.16 LowReqToMac

(1) Name

Physical address translation request function

(2) Function

Translates NodeID delivered to Lower-layer Communication Software into corresponding physical address.

When NodeID code = physical address code, this function is not required.

(3) Syntax

```
BOOL   LowReqToMac (
           unsigned char device_id,     /*[IN]    Lower-layer Communication Software type ID
                                                  */
           unsigned char   node_id,     /*[IN]    Target NodeID code for translation */
           unsigned char   *mac,        /*[OUT]   Translated physical address */
           short     mac_len            /*[OUT]   Translated physical address size */
)
```

(4) Explanation

device_id : Identification information for Lower-layer Communication Software (For specific details of codes, see the LowInit function.)

node_id : Target NodeID code for translation

*mac : Translated physical address code

mac_len : Translated physical address code size

(5) Return value

0: Failed translation

1: Successful translation

(6) Notes

None

(7) Structure

None

(8) Restrictions

None

4.2.17 LowReqToID

(1) Name

NodeID translation request function

(2) Function

Translates physical address delivered to Lower-layer Communication Software into corresponding NodeID.

When NodeID code = physical address code, this function is not required.

(3) Syntax

```
BOOL   LowReqToMac (
           unsigned char device_id,     /*[IN]    Lower-layer Communication Software type ID
                                                  */
           unsigned char   mac,         /*[IN]    Target physical address for translation */
           unsigned char   *node_id,    /*[OUT]   Translated NodeID code */
)
```

(4) Explanation

device_id : Identification information for Lower-layer Communication Software (For specific details of codes, see the LowInit function.)

mac : Translated physical address code

*node_id : Target NodeID code for translation

(5) Return value

0: Failed translation

1: Successful translation

(6) Notes

None

(7) Structure

None

(8) Restrictions

None

4.2.18 LowReqBcastID

(1) Name

NodeID translation request function

(2) Function

Extracts target NodeID from group specification codes at broadcast function delivered to Lower-layer Communication Software.

This function is not required when the Lower-layer Communication Software has a broadcast function and the group specification code is equal to the DEA group specification code at broadcast specification or the inside (Lower-layer Communication Software) has a simple translation function.

(3) Syntax

```
BOOL   LowReqBcastID (
           unsigned char device_id,   /*[IN]   Lower-layer Communication Software type ID
                                                */
           unsigned char   bcast,     /*[IN]   Target broadcast group specification code */
           unsigned char   map        /*[OUT]  Address information for transmitting
                                                destination node */
)
```

(4) Explanation

device_id : Identification information for Lower-layer Communication Software (For specific details of codes, see the LowInit function.)

bcast : Target broadcast group specification code (DEA 2nd type group specification code for broadcast specification, except 0xff (simultaneous broadcast))

map : Target NodeID code for translation. Returns address for bit map of transmitting destination node. 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 translation

1: Successful translation

(6) Notes

None

(7) Structure

None

(8) Restrictions

None

4.2.19 Noise resistance communication information acquisition

Note: Only the power line B system is supported.

(1) Prototype

```
BOOL  PLCGetN_Information (
          unsigned char   *Tone_Code,   //[out] Tone position code
          short           *bps          //[out] Lower-layer medium transmission
                                        rate
)
```

(2) Data

*Tone_Code : Tone position code

*bps : Lower-layer medium transmission rate

(3) Function

Gets noise resistance communication information of PLC layer 1.

(4) Return value

1: TRUE

0: FALSE

4.2.20 Noise resistance communication information setting

Note: Only the power line B system is supported.

(1) Prototype

```
BOOL  PLCSetN_Information (
          unsigned char   *Tone_Code,   //[out] Tone position code
          short           *bps          //[out] Lower-layer medium transmission rate
)
```

(2) Data

*Tone_Code : Sets tone position code.

*bps : Specifies transmission rate of lower-layer medium.

(3) Function

Notifies (sets) PLC layer 1 of noise resistance communication information.

(4) Return value

1: TRUE

0: FALSE

4.3 Initial Setting Information Specification

This section describes the initialization parameter specifications provided in the area indicated by the argument initialization parameter pointer "*low_init" of the "Request for initialization: LowInit" (see Remark below) for each of the following six types of Lower-layer Communication Software:

(1) Communication software for Power line A system
(2) Communication software for Power line B system
(3) Communication software for Specific Low-power Radio
(4) Communication software for Extended HBS
(5) Communication software for IrDA Control
(6) Communication software for LON

Remark: Syntax of the LowInit function

```
BOOL    LowInit (
            short     device_id,              /*[IN] Lower-layer Communication Software type ID */
            LOW_INIT_DATA  *init_data,        /*[IN] Pointer to initialization parameter  (1) */
            void      *low_init               /*[IN] Pointer to initialization parameter  (2) */ )
```

4.3.1 Power line A initialization parameter specification

```
typedef struct {
        short   sbuf_len;       /* Transmitting buffer size */
        short   *sbuf;          /* Pointer to transmitting buffer */
        short   rbuf_len;       /* Receiving buffer size */
        short   *rbuf           /* Pointer to receiving buffer */
} PLCA_INIT_DATA
```

4.3.2 Power line B initialization parameter specification

```
typedef struct {
} PLCB_INIT_DATA
```

4.3.3 Specific low-power radio initialization parameter specification

```
typedef struct {
} RF_INIT_DATA
```

4.3.4 Extended HBS initialization parameter specification

```
typedef struct {
        short   sbuf_len;       /* Transmitting buffer size */
        short   *sbuf;          /* Pointer to transmitting buffer */
        short   rbuf_len;       /* Receiving buffer size */
        short   *rbuf           /* Pointer to receiving buffer */
} HBS_INIT_DATA
```

4.3.5 IrDA Control initialization parameter specification

```
typedef struct {
        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   mac_table_len;  /* MAC address translation table size */
        short   *mac_table      /* Pointer to MAC address translation table */
} IRDA_INIT_DATA
```

4.3.6 LON initialization parameter specification

```
typedef struct {
        short   sbuf_len;       /* Transmitting buffer size */
        short   *sbuf;          /* Pointer to transmitting buffer */
        short   rbuf_len;       /* Receiving buffer size */
        short   *rbuf           /* Pointer to receiving buffer */
} LON_INIT_DATA
```