Part V
ECHONET Common Lower-layer Communication Interface Specification
Revision record

- Version 1.0  
  March 18th  2000  released  Open to consortium members
  July  2000  Open to the public
- Version 1.01  
  May 23rd  2001  Open to consortium members
  Version 1.0 addendum & corrigendum
- Version 2.00  
  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:

<table>
<thead>
<tr>
<th>Revised entry</th>
<th>Revision/addition</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 4.2.1</td>
<td>Descriptions were changed because the power line A and power line B methods were integrated into a single method.</td>
</tr>
<tr>
<td>2 4.2.10</td>
<td>Descriptions were changed because the power line A and power line B methods were integrated into a single method.</td>
</tr>
<tr>
<td>3 4.2.15</td>
<td>Descriptions were changed because the power line A and power line B methods were integrated into a single method.</td>
</tr>
</tbody>
</table>

- 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:

<table>
<thead>
<tr>
<th>Revised entries in the table of contents</th>
<th>Revision/addition</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2.1</td>
<td>- The following interfaces were added in accordance with the revision to the state transition stipulated in Part 2: &quot;Request for lower-layer communication software mounting information&quot;, &quot;request for complete initialization&quot;, &quot;request for communication stop&quot;, and &quot;request for complete stop&quot;</td>
</tr>
<tr>
<td>2 2.1</td>
<td>- The request named &quot;request for reset&quot; was renamed to &quot;request for warm start&quot; in accordance with the revision to the state transition stipulated in Part 2.</td>
</tr>
<tr>
<td>3 2.2</td>
<td>- The detailed interface descriptions were changed in accordance with the revision to the state transition stipulated in Part 2.</td>
</tr>
</tbody>
</table>
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.
### Revised entry | Revision/addition
--- | ---
4 3.1 | 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”
5 3.1 | 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.
6 3.2 | The detailed interface descriptions were changed in accordance with the revision to the state transition stipulated in Part 2.
7 4.1 | The following APIs were added in accordance with the revision to the state transition stipulated in Part 2: “ClInitAll”, “ClcStop”, “ClcHalt”, “ClcLowInitAll”, “ClcLowStop”, and “ClcLowHalt”
8 4.1 | In accordance with the revision to the state transition stipulated in Part 2, “ClcReset” and “ClcLowReset” were renamed to “ClcStart” and “ClcLowStart”, respectively.
9 4.2 | Detailed API descriptions were changed in accordance with the revision to the state transition stipulated in Part 2.
Contents

Chapter 1 Overview .....................................................................................................................1-1
  1.1 Basic Concept .............................................................................................................1-1
  1.2 Positioning on Communication Layers ......................................................................1-2

Chapter 2 ECHONET Common Lower-layer Communication Interface Function Specification 2-1
  2.1 List of ECHONET Common Lower-layer Communication Interface Functions ..... 2-1
  2.2 ECHONET Common Lower-Layer Communication Interface Function Detailed Specification ...............................................................................................................2-2

Chapter 3 Level 1 ECHONET Common Lower-layer Communication Interface Specification 3-1
  3.1 List of Level 1 ECHONET Common Lower-Layer Communication Interface Services ................................................................................................................................3-1
  3.2 Level 1 ECHONET Common Lower-layer Communication Interface Detailed Specification ...............................................................................................................3-3

Chapter 4 Level 2 ECHONET Common Lower-layer Communication Interface Specification 4-1
  4.1 List of Level 2 ECHONET Common Lower-Layer Communication Interface Functions for C Language ........................................................................................................... 4-2
  4.2 C Language-oriented Level 2 ECHONET Common Lower-layer Communication Interface Detailed Specification .................................................................................4-4
    4.2.1 ClcGetDevID .............................................................................................................4-5
    4.2.2 ClcInit .................................................................................................................. 4-6
    4.2.3 ClcRequestRun ........................................................................................................4-7
    4.2.4 ClcSetTrouble ............................................................................................................ 4-8
    4.2.5 ClcStart ................................................................................................................. 4-9
    4.2.6 ClcSuspend .............................................................................................................. 4-10
    4.2.7 ClcWakeUp ............................................................................................................. 4-11
    4.2.8 ClcGetProData ........................................................................................................4-12
    4.2.9 ClcGetStatus ............................................................................................................4-13
    4.2.10 ClcInitAll .................................................................................................................4-14
    4.2.11 ClcStop ....................................................................................................................4-15
    4.2.12 ClcHalt ....................................................................................................................4-16
    4.2.13 ClcLowInit ..............................................................................................................4-17
    4.2.14 ClcLowRequestRun ................................................................................................4-19
    4.2.15 ClcLowStart ............................................................................................................4-20
    4.2.16 ClcLowSuspend ......................................................................................................4-21
    4.2.17 ClcLowWakeUp ....................................................................................................4-22
    4.2.18 ClcGetLowProData ...............................................................................................4-23
    4.2.19 ClcGetLowStatus ..................................................................................................4-25
    4.2.20 ClcSendData ............................................................................................................4-27
    4.2.21 ClcGetSendResult .................................................................................................4-29
    4.2.22 ClcSendCancel .......................................................................................................4-30
    4.2.23 ClcReceiveData.....................................................................................................4-31
    4.2.24 ClcGetNodeID ......................................................................................................4-32
    4.2.25 ClcSetNodeID .......................................................................................................4-33
    4.2.26 ClcLowInitAll .......................................................................................................4-34
    4.2.27 ClcLowStop ......................................................................................................... 4-36
4.2.28 ClcLowHalt............................................................................................................. 4-37
Chapter 1 Overview

1.1 Basic Concept

The ECHONET Common Lower-layer Communication Interface Specification in Part 5 is 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 specification 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 specification provides 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 specification are based on the concept of levels 1 and 2 of the basic API.
1.2 Positioning on Communication Layers

The interface specification described in this Section is 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.

![Diagram showing the positioning of the Common Lower-layer Communication Interface](image)
Chapter 2 ECHONET Common Lower-layer Communication Interface Function Specification

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. The Protocol Difference Absorption Processing Block shall be provided with these functions. The interface function specification is 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
2.2 ECHONET Common Lower-Layer Communication Interface Function Detailed Specification

This section explains ECHONET common lower-layer communication interface functions supported by the protocol difference absorption processing block. For state transitions of the protocol difference absorption processing block and 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 into 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 communication stop state to normal operation state.

(4) Fault notice
   Notifies Protocol Difference Absorption Processing Block of fault (error) status of high-order layer from 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 into 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 normal operation state to suspension state.

(7) Request for operation restart
   Requests that the protocol difference absorption processing block and lower-layer communication software exit suspension state and enter 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 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 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 communication stop state.

(20) Request for complete stop
Requests that the protocol difference absorption processing block place the lower-layer communication software in stop state.

(22) Stop notice
Protocol Difference Absorption Processing Block notifies the ECHONET communication processing block that the lower-layer communication software has switched to stop state.
Chapter 3 Level 1 ECHONET Common Lower-layer Communication Interface Specification

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 communication processing block and protocol difference absorption processing block. For mounting in compliance with the level 1 ECHONET common lower-layer communication interface specification, 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)

<table>
<thead>
<tr>
<th>No.</th>
<th>Service name</th>
<th>Function outline</th>
<th>Mounting specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Request for lower-layer communication software mounting information</td>
<td>Requests the number of mounted (accessible) lower-layer communication software programs and their types.</td>
<td>Required</td>
</tr>
<tr>
<td>2</td>
<td>Request for initialization</td>
<td>Requests that a specified protocol difference absorption processing block and lower-layer communication software effect initialization by performing a cold start.</td>
<td>Required</td>
</tr>
<tr>
<td>3</td>
<td>Request for operation start</td>
<td>Requests that a specified protocol difference absorption processing block and lower-layer communication software start running.</td>
<td>Required</td>
</tr>
<tr>
<td>4</td>
<td>Fault notice</td>
<td>Notifies Protocol Difference Absorption Processing Block of fault (error) status of high-order layer from the ECHONET communications processing block.</td>
<td>Optional</td>
</tr>
<tr>
<td>5</td>
<td>Request for warm start</td>
<td>Requests that a specified protocol difference absorption processing block and lower-layer communication software effect initialization by performing a warm start.</td>
<td>Required</td>
</tr>
<tr>
<td>6</td>
<td>Request for suspension</td>
<td>Requests that a specified protocol difference absorption processing block and lower-layer communication software suspend operation.</td>
<td>Optional</td>
</tr>
<tr>
<td>7</td>
<td>Request for operation restart</td>
<td>Requests that a specified protocol difference absorption processing block and lower-layer communication software resume operation.</td>
<td>Optional</td>
</tr>
<tr>
<td>8</td>
<td>Request for protocol difference absorption processing block profile data acquisition</td>
<td>Obtains static information for Protocol Difference Absorption Processing Block.</td>
<td>Required</td>
</tr>
<tr>
<td>9</td>
<td>Request for lower-layer communication software profile data acquisition</td>
<td>Obtains static information for Lower-layer Communication Software.</td>
<td>Required</td>
</tr>
</tbody>
</table>
Table 3.1 List of Level 1 ECHONET Common Lower-Layer Communication Interface Services (2/2)

<table>
<thead>
<tr>
<th>No.</th>
<th>Service name</th>
<th>Function outline</th>
<th>Mounting specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Request for protocol difference absorption processing block status data acquisition</td>
<td>Obtains dynamic status (processing fault, etc.) of Protocol Difference Absorption Processing Block.</td>
<td>Optional</td>
</tr>
<tr>
<td>11</td>
<td>Request for lower-layer communication software status data acquisition</td>
<td>Obtains dynamic status (processing fault, address redundancy, etc.) of Lower-layer Communication Software.</td>
<td>Required</td>
</tr>
<tr>
<td>12</td>
<td>Request for data transmission</td>
<td>Requests data transmission from Protocol Difference Absorption Processing Block.</td>
<td>Required</td>
</tr>
<tr>
<td>13</td>
<td>Transmission result acquisition</td>
<td>Requests data transmission result from Protocol Difference Absorption Processing Block.</td>
<td>Optional</td>
</tr>
<tr>
<td>14</td>
<td>Request for transmission stop</td>
<td>Requests that the protocol difference absorption processing block stop a data transmission.</td>
<td>Optional</td>
</tr>
<tr>
<td>15</td>
<td>Request for received data</td>
<td>Requests received data from Protocol Difference Absorption Processing Block.</td>
<td>Required</td>
</tr>
<tr>
<td>16</td>
<td>Request for node ID acquisition</td>
<td>Makes a request for acquiring a node ID retained by a protocol difference absorption processing block.</td>
<td>Required</td>
</tr>
<tr>
<td>17</td>
<td>Request for node ID setup</td>
<td>Sets NodeID for Protocol Difference Absorption Processing Block.</td>
<td>Optional</td>
</tr>
<tr>
<td>18</td>
<td>Request for complete initialization</td>
<td>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.</td>
<td>Optional</td>
</tr>
<tr>
<td>19</td>
<td>Request for communication stop</td>
<td>Requests that a specified protocol difference absorption processing block and lower-layer communication software switch into communication stop state.</td>
<td>Optional</td>
</tr>
<tr>
<td>20</td>
<td>Request for complete stop</td>
<td>Requests that a specified protocol difference absorption processing block and lower-layer communication software switch into stop state.</td>
<td>Optional</td>
</tr>
<tr>
<td>21</td>
<td>Stop notice</td>
<td>Protocol Difference Absorption Processing Block notifies the ECHONET communication processing block that the lower-layer communication software has switched to stop state.</td>
<td>Optional</td>
</tr>
</tbody>
</table>
3.2 Level 1 ECHONET Common Lower-layer Communication Interface Detailed Specification

Input/output data are 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 communication processing block. More specifically, the term "input" denotes the transfer of data from the ECHONET communication 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 communication processing block. When these data transfer operations can be performed, the level 1 ECHONET common lower-layer communication interface specification is 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 radio, etc.). Table 3.2 shows the data specifications.

<table>
<thead>
<tr>
<th>Direction</th>
<th>Data name</th>
<th>Contents and condition</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>–</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Output</td>
<td>device_num</td>
<td>- Shows the number of mounted lower-layer communication software programs.</td>
<td>Optional</td>
</tr>
</tbody>
</table>
| Output     | device_id  | - Indicates the type of the lower-layer communication software.  
- The power line lower-layer communication software, specific low-power radio 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 specified lower-layer communication software effect initialization by performing a cold start and switch into 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

<table>
<thead>
<tr>
<th>Direction</th>
<th>Data name</th>
<th>Contents and condition</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>device_id</td>
<td>- Specifies lower-layer communication software to be initialized.</td>
<td>Required</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Options to be provided for initializing all lower-layer communication software programs simultaneously and initializing a specific lower-layer communication software program.</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>p_init</td>
<td>- Specifies the initialization parameters.</td>
<td>Required</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 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.</td>
<td></td>
</tr>
<tr>
<td>Output</td>
<td>Return Value</td>
<td>TRUE: Successful initialization, FALSE: Failed initialization.</td>
<td>Optional</td>
</tr>
</tbody>
</table>

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

Requests that 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

<table>
<thead>
<tr>
<th>Direction</th>
<th>Data name</th>
<th>Contents and condition</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>device_id</td>
<td>- Specifies lower-layer communication software that should start running.</td>
<td>Optional</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 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.</td>
<td></td>
</tr>
<tr>
<td>Output</td>
<td>Return Value</td>
<td>TRUE: Successful operation start, FALSE: Failed operation start.</td>
<td>Optional</td>
</tr>
</tbody>
</table>
(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>
<thead>
<tr>
<th>Direction</th>
<th>Data name</th>
<th>Contents and condition</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>trouble_no</td>
<td>Reports a trouble number indicating an abnormal state.</td>
<td>Required</td>
</tr>
<tr>
<td>Output</td>
<td>Return Value</td>
<td>TRUE: Fault notice acceptable, FALSE: Fault notice not acceptable</td>
<td>Optional</td>
</tr>
</tbody>
</table>

(5) Request for warm start (mandatory function for mounting)
Requests that specified lower-layer communication software and associated protocol difference absorption processing block effect initialization by performing a warm start and then switch into 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>
<thead>
<tr>
<th>Direction</th>
<th>Data name</th>
<th>Contents and condition</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>device_id</td>
<td>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.</td>
<td>Required</td>
</tr>
<tr>
<td>Output</td>
<td>Return Value</td>
<td>TRUE: Warm start request accepted; FALSE: Request denied.</td>
<td>Optional</td>
</tr>
</tbody>
</table>

(6) Request for suspension
Requests that 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>
<thead>
<tr>
<th>Direction</th>
<th>Data name</th>
<th>Contents and condition</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>device_id</td>
<td>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.</td>
<td>Required</td>
</tr>
<tr>
<td>Output</td>
<td>Return Value</td>
<td>TRUE: Suspension acceptable, FALSE: Suspension not acceptable</td>
<td>Optional</td>
</tr>
</tbody>
</table>
(7) Request for operation restart

Requests that 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

<table>
<thead>
<tr>
<th>Direction</th>
<th>Data name</th>
<th>Contents and condition</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>device_id</td>
<td>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.</td>
<td>Required</td>
</tr>
<tr>
<td>Output</td>
<td>Return Value</td>
<td>TRUE: Successful restart, FALSE: Restart disable (including failure)</td>
<td>Optional</td>
</tr>
</tbody>
</table>

(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

<table>
<thead>
<tr>
<th>Direction</th>
<th>Data name</th>
<th>Contents and condition</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>device_idinfo</td>
<td>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.</td>
<td>Required</td>
</tr>
<tr>
<td>Output</td>
<td>version_No</td>
<td>Presents version information for the protocol difference absorption processing block.</td>
<td>Optional</td>
</tr>
<tr>
<td>Output</td>
<td>company_name</td>
<td>Shows the manufacturer code.</td>
<td>Optional</td>
</tr>
<tr>
<td>Output</td>
<td>rwlen</td>
<td>Presents buffer size information.</td>
<td>Optional</td>
</tr>
<tr>
<td>Output</td>
<td>Return Value</td>
<td>TRUE: Normal, FALSE: Error</td>
<td>Optional</td>
</tr>
</tbody>
</table>
(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

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

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

Asks 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

<table>
<thead>
<tr>
<th>Direction</th>
<th>Data name</th>
<th>Contents and condition</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>device_id</td>
<td>- Specifies lower-layer communication software associated with the protocol difference absorption processing block to be targeted for acquisition.</td>
<td>Required</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Options to be provided for specifying all lower-layer communication software programs and for specifying a specific lower-layer communication software program.</td>
<td></td>
</tr>
<tr>
<td>Output</td>
<td>state</td>
<td>- Presents state transition information.</td>
<td>Required</td>
</tr>
<tr>
<td>Output</td>
<td>trouble_no</td>
<td>- Fault information for Protocol Difference Absorption Processing Block.</td>
<td>Optional</td>
</tr>
<tr>
<td>Output</td>
<td>upper_trouble</td>
<td>- Information recognized as fault in high-order layer</td>
<td>Optional</td>
</tr>
<tr>
<td>Output</td>
<td>Low_trouble</td>
<td>- Information recognized as fault in low-order layer</td>
<td>Optional</td>
</tr>
<tr>
<td>Output</td>
<td>Return Value</td>
<td>TRUE: Normal, FALSE: Error</td>
<td>Optional</td>
</tr>
</tbody>
</table>
(11) Request for lower-layer communication software status data acquisition (mandatory function for mounting)

Asks 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**

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

(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**

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

<p>| Table 3.14 Input/Output Data List for Transmission Result Acquisition Service |
|-------------------------------------------------|---------------------------------|-------------------------------------------------|</p>
<table>
<thead>
<tr>
<th>Direction</th>
<th>Data name</th>
<th>Contents and condition</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>device_id</td>
<td>Specifies lower-layer communication software to be targeted for transmission result acquisition.</td>
<td>Required</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Individual lower-layer communication software programs must be recognizable.</td>
<td></td>
</tr>
<tr>
<td>Output</td>
<td>result</td>
<td>Information on transmitting status, normal termination of transmission, termination due to transmission error, or transmission stopping status.</td>
<td>Required</td>
</tr>
<tr>
<td>Output</td>
<td>Return Value</td>
<td>TRUE: Normal, FALSE: Error</td>
<td>Optional</td>
</tr>
</tbody>
</table>

(14) Request for transmission stop

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

<p>| Table 3.15 Input/Output Data List for Transmission Stop Request Service |
|-------------------------------------------------|---------------------------------|---------|</p>
<table>
<thead>
<tr>
<th>Direction</th>
<th>Data name</th>
<th>Contents and condition</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>device_id</td>
<td>Specifies lower-layer communication software to be targeted for transmission stop.</td>
<td>Required</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Individual lower-layer communication software programs must be recognizable.</td>
<td></td>
</tr>
<tr>
<td>Output</td>
<td>Return Value</td>
<td>TRUE: Stop success, FALSE: Stop failure (already transmitted)</td>
<td>Optional</td>
</tr>
</tbody>
</table>
(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**

<table>
<thead>
<tr>
<th>Direction</th>
<th>Data name</th>
<th>Contents and condition</th>
<th>Remarks</th>
</tr>
</thead>
</table>
| 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 communication 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**

<table>
<thead>
<tr>
<th>Direction</th>
<th>Data name</th>
<th>Contents and condition</th>
<th>Remarks</th>
</tr>
</thead>
</table>
| 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

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

(18) Request for complete initialization

Requests that 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

<table>
<thead>
<tr>
<th>Direction</th>
<th>Data name</th>
<th>Contents and condition</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>software_id</td>
<td>- Specifies lower-layer communication software to be targeted for complete initialization.</td>
<td>Required</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Options to be provided for completely initializing all lower-layer communication software programs simultaneously and completely initializing a specific lower-layer communication software program.</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>p_init</td>
<td>- Specifies initialization parameters.</td>
<td>Required</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 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.</td>
<td></td>
</tr>
<tr>
<td>Output</td>
<td>Return Value</td>
<td>TRUE: Initialization successful; FALSE: Initialization not successful.</td>
<td>Optional</td>
</tr>
</tbody>
</table>
(19) Request for communication stop

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

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

<table>
<thead>
<tr>
<th>Direction</th>
<th>Data name</th>
<th>Contents and condition</th>
<th>Remarks</th>
</tr>
</thead>
</table>
| 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 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

<table>
<thead>
<tr>
<th>Direction</th>
<th>Data name</th>
<th>Contents and condition</th>
<th>Remarks</th>
</tr>
</thead>
</table>
| 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 communication 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

<table>
<thead>
<tr>
<th>Direction</th>
<th>Data name</th>
<th>Contents and condition</th>
<th>Implementation Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output</td>
<td>software_id</td>
<td>• Indicates lower layer communication software that has switched to stop state.</td>
<td>Required</td>
</tr>
<tr>
<td>Input</td>
<td>Return Value</td>
<td>TRUE: notice received, FALSE: notice cannot be received</td>
<td>Optional</td>
</tr>
</tbody>
</table>
Chapter 4 Level 2 ECHONET Common Lower-layer Communication Interface Specification

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 specification 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>
<thead>
<tr>
<th>No.</th>
<th>Function name</th>
<th>Name</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ClcGetDevID</td>
<td>Lower-layer Communication Software mounting information request function</td>
<td>Required</td>
</tr>
<tr>
<td>2</td>
<td>ClcInit</td>
<td>Initialization request function</td>
<td>Required</td>
</tr>
<tr>
<td>3</td>
<td>ClcRequestRun</td>
<td>Operation start request function</td>
<td>Required</td>
</tr>
<tr>
<td>4</td>
<td>ClcSetTrouble</td>
<td>High-order layer fault notice function</td>
<td>Required</td>
</tr>
<tr>
<td>5</td>
<td>ClcStart</td>
<td>Warm start request function</td>
<td>Required</td>
</tr>
<tr>
<td>6</td>
<td>ClcSuspend</td>
<td>Operation suspension request function</td>
<td>Required</td>
</tr>
<tr>
<td>7</td>
<td>ClcWakeUp</td>
<td>Operation restart request function</td>
<td>Required</td>
</tr>
<tr>
<td>8</td>
<td>ClcGetProData</td>
<td>Protocol difference absorption processing block profile data acquisition request function</td>
<td>Required</td>
</tr>
<tr>
<td>9</td>
<td>ClcGetStatus</td>
<td>Protocol difference absorption processing block status data acquisition request function</td>
<td>Optional</td>
</tr>
<tr>
<td>10</td>
<td>ClcInitAll</td>
<td>Complete initialization request function</td>
<td>Optional</td>
</tr>
<tr>
<td>11</td>
<td>ClcStop</td>
<td>Communication stop request function</td>
<td>Optional</td>
</tr>
<tr>
<td>12</td>
<td>ClcHalt</td>
<td>Complete stop request function</td>
<td>Optional</td>
</tr>
<tr>
<td>13</td>
<td>ClcLowInit</td>
<td>Lower-layer communication software initialization request function</td>
<td>Optional</td>
</tr>
<tr>
<td>14</td>
<td>ClcLowRequestRun</td>
<td>Lower-layer communication software operation start request function</td>
<td>Optional</td>
</tr>
<tr>
<td>15</td>
<td>ClcLowStart</td>
<td>Lower-layer communication software warm start request function</td>
<td>Optional</td>
</tr>
<tr>
<td>16</td>
<td>ClcLowSuspend</td>
<td>Lower-layer communication software suspension request function</td>
<td>Optional</td>
</tr>
<tr>
<td>17</td>
<td>ClcLowWakeUp</td>
<td>Lower-layer communication software operation restart request function</td>
<td>Optional</td>
</tr>
<tr>
<td>18</td>
<td>ClcGetLowProData</td>
<td>Lower-layer communication software profile data acquisition request function</td>
<td>Required</td>
</tr>
<tr>
<td>19</td>
<td>ClcGetLowStatus</td>
<td>Lower-layer communication software status data acquisition request function</td>
<td>Required</td>
</tr>
<tr>
<td>20</td>
<td>ClcSendData</td>
<td>Data transmission request function</td>
<td>Required</td>
</tr>
<tr>
<td>21</td>
<td>ClcGetSendResult</td>
<td>Transmission result request function</td>
<td>Optional</td>
</tr>
<tr>
<td>22</td>
<td>ClcSendCancel</td>
<td>Transmission stop request function</td>
<td>Optional</td>
</tr>
<tr>
<td>No.</td>
<td>Function name</td>
<td>Name</td>
<td>Remarks</td>
</tr>
<tr>
<td>-----</td>
<td>------------------</td>
<td>-------------------------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>23</td>
<td>ClcReceiveData</td>
<td>Received data request function</td>
<td>Required</td>
</tr>
<tr>
<td>24</td>
<td>ClcGetNodeID</td>
<td>Node ID acquisition request function</td>
<td>Required</td>
</tr>
<tr>
<td>25</td>
<td>ClcSetNodeID</td>
<td>Node ID setup request function</td>
<td>Required</td>
</tr>
<tr>
<td>26</td>
<td>ClcLowInitAll</td>
<td>Lower-layer communication software complete initialization request function</td>
<td>Optional</td>
</tr>
<tr>
<td>27</td>
<td>ClcLowStop</td>
<td>Lower-layer communication software communication stop request function</td>
<td>Optional</td>
</tr>
<tr>
<td>28</td>
<td>ClcLowHalt</td>
<td>Lower-layer communication software complete stop request function</td>
<td>Optional</td>
</tr>
</tbody>
</table>
4.2 C Language-oriented Level 2 ECHONET Common Lower-layer Communication Interface Detailed Specification

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
```c
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: 0x11～0x1F
- Specific low-power radio: 0x31～0x3F
- Extended HBS: 0x41～0x4F
- IrDA_Control: 0x51～0x5F
- LonTalk®: 0x61～0x6F

(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 communication stop state, and is initialized in accordance with the initialization parameter.

(3) Syntax
```c
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 struct {
    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 specification */
        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 start request function

(2) Function
  Requests that all existing lower-layer communication software programs and
  associated protocol difference absorption processing block start running. Upon receipt
  of this request, the protocol difference absorption processing block places all
  lower-layer communication software programs in normal operation state.

(3) Syntax
  BOOL   ClcRequestRun (void)

(4) Explanation
  None

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

(6) Structure
  None

(7) Notes/restrictions
  Only lower-layer communication software programs in communication stop state will
  be placed in normal operation state by this function. This function returns "Successful
  start" when at least one lower-layer communication software program changes its
  status successfully.
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
```c
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 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
```c
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
```c
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 struct {
    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 communication stop state, and is initialized in accordance with the initialization parameter.

(3) Syntax
```c
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
```c
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 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 specification */
    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 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 communication stop state all lower-layer communication software programs
that can be placed in 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 into 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 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 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 into 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 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 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 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 0x11〜0x1F
            Specific low-power radio 0x31〜0x3F
            Extended HBS 0x41〜0x4F
            IrDA_Control 0x51〜0x5F
            LonTalk® 0x61〜0x6F

*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 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   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 specification */
                      0x00 Normal operation mode
                      0x01 Test/maintenance mode (details not specified)
} CLC_INIT_DATA

typedef struct {
    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 specification */
                     0x00 Normal operation mode
                     0x01 Test/maintenance mode (details not specified)
    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 targeted lower-layer communication software is in 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 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 normal operation state.

(3) **Syntax**

```c
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: 0x11 ~ 0x1F
  - Specific low-power radio: 0x31 ~ 0x3F
  - Extended HBS: 0x41 ~ 0x4F
  - IrDA_Control: 0x51 ~ 0x5F
  - LonTalk®: 0x61 ~ 0x6F

(5) **Return value**

- **0**: Failed start
- **1**: Successful start

(6) **Structure**

None

(7) **Notes/restrictions**

If specified lower-layer communication software is not in 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 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 communication stop state.

(3) Syntax

```c
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 0x11~0x1F
- Specific low-power radio 0x31~0x3F
- Extended HBS 0x41~0x4F
- IrDA_Control 0x51~0x5F
- LonTalk® 0x61~0x6F

(5) Return value

0: Failed request
1: Successful request

(6) Structure

None

(7) Notes/restrictions

If targeted lower-layer communication software is in 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 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 suspension state.

(3) Syntax
```c
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: 0x11~0x1F
  - Specific low-power radio: 0x31~0x3F
  - Extended HBS: 0x41~0x4F
  - IrDA_Control: 0x51~0x5F
  - LonTalk®: 0x61~0x6F

(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 into 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 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 specified lower-layer communication software and associated protocol difference absorption processing block exit suspension state. Upon receipt of this request, the protocol difference absorption processing block places the specified lower-layer communication software in 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: 0x11 ~ 0x1F
     - Specific low-power radio: 0x31 ~ 0x3F
     - Extended HBS: 0x41 ~ 0x4F
     - IrDA_Control: 0x51 ~ 0x5F
     - LonTalk®: 0x61 ~ 0x6F

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 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 lower-layer communication software profile class, such as the
software development manufacturer name and version number.

(3) Syntax

```c
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 0x11~0x1F
    Specific low-power radio 0x31~0x3F
    Extended HBS 0x41~0x4F
    IrDA_Control 0x51~0x5F
    LonTalk® 0x61~0x6F

*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 specification 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 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 radio: 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 house_len; /* House code length */
    short *housecode; /* Pointer to house code information */
    short mac_len; /* MAC address length */
    unsigned char mac_ad[7]; /* MAC address */
    unsigned char mac_mask[7]; /* MAC address mask value */
    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

```c
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 0x11〜0x1F
- Specific low-power radio 0x31〜0x3F
- Extended HBS 0x41〜0x4F
- IrDA_Control 0x51〜0x5F
- LonTalk® 0x61〜0x6F

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

(5) Return value
0: Failed acquisition
1: Successful acquisition
(6) Structure
typedef struct {
    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 specified lower-layer communication software. The protocol difference absorption processing block associated with the specified lower-layer communication software creates a data in compliance with lower-layer communication software specifications and requests that the lower-layer communication software transmit the data.

(3) **Syntax**

```c
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 specification */
);
```

(4) **Explanation**

- **device_id**: Identification information for lower-layer communication software to be used for data transmission.
  - Power line: 0x11~0x1F
  - Specific low-power radio: 0x31~0x3F
  - Extended HBS: 0x41~0x4F
  - IrDA_Control: 0x51~0x5F
  - LonTalk®: 0x61~0x6F

- **buf**: Specifies pointer to ECHONET data to be transmitted. The ECHONET data to be passed here is one of the data exchanged between ECHONET communication 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

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLC_BUFFER_FULL</td>
<td>0 Buffer full error</td>
</tr>
<tr>
<td>CLC_NO_ERROR</td>
<td>1 Transmission accepted</td>
</tr>
<tr>
<td>CLC_BUFFER_SIZE_ERROR</td>
<td>2 Buffer size error</td>
</tr>
<tr>
<td>CLC_STATE_ERROR</td>
<td>3 Internal error of Lower-layer Communication Software</td>
</tr>
<tr>
<td>CLC_ADAPTER_ERROR</td>
<td>4 Device adapter processing failed</td>
</tr>
</tbody>
</table>

(6) Structure

None

(7) Notes/restrictions

If the specified lower-layer communication software is not in 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 result 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 result.

(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 0x11—0x1F
    Specific low-power radio 0x31—0x3F
    Extended HBS 0x41—0x4F
    IrDA_Control 0x51—0x5F
    LonTalk® 0x61—0x6F
*result : Pointer to the transmission result.
    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 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 specified lower-layer communication software cancel ECHONET data transmission being performed in accordance with 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: 0x11~0x1F
- Specific low-power radio: 0x31~0x3F
- Extended HBS: 0x41~0x4F
- IrDA_Control: 0x51~0x5F
- LonTalk®: 0x61~0x6F

(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 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.
\[ \begin{align*}
\text{Power line} & : 0x11 \sim 0x1F \\
\text{Specific low-power radio} & : 0x31 \sim 0x3F \\
\text{Extended HBS} & : 0x41 \sim 0x4F \\
\text{IrDA\_Control} & : 0x51 \sim 0x5F \\
\text{LonTalk\(^\circ\)} & : 0x61 \sim 0x6F
\end{align*} \]
*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 local subnet. If received data was transmitted from a remote subnet, router node ID is returned.

(5) Return value
\begin{align*}
\text{CLC\_NO\_RECEIVE} & : 0 \quad \text{No received data} \\
\text{CLC\_NO\_ERROR} & : 1 \quad \text{Normal (with received data)} \\
\text{CLC\_BUFFER\_SIZE\_ERROR} & : 2 \quad \text{Buffer size error} \\
\text{CLC\_INTERNAL\_ERROR} & : 3 \quad \text{Internal error in Lower-layer Communication Software}
\end{align*}

(6) Structure
None

(7) Notes/restrictions
If the specified lower-layer communication software is not in 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 protocol difference absorption processing block associated with specified lower-layer communication software.

(3) Syntax
```c
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: 0x11～0x1F
  - Specific low-power radio: 0x31～0x3F
  - Extended HBS: 0x41～0x4F
  - IrDA_Control: 0x51～0x5F
  - LonTalk®: 0x61～0x6F
- **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".

© 2000 ( 2002 ) ECHONET CONSORTIUM ALL RIGHT RESERVED
4.2.25 ClcSetNodeID

(1) Name
Node ID setup request function

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

(3) Syntax
```c
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: 0x11～0x1F
  - Specific low-power radio: 0x31～0x3F
  - Extended HBS: 0x41～0x4F
  - IrDA_Control: 0x51～0x5F
  - LonTalk®: 0x61～0x6F

- `node_id`: NodeID code

(5) Return value
- CLC_NO_CHANGE: 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 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 communication stop state, and is initialized in
accordance with the initialization parameter.

(3) Syntax

```c
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 0x11〜0x1F
- Specific low-power radio 0x31〜0x3F
- Extended HBS 0x41〜0x4F
- IrDA_Control 0x51〜0x5F
- LonTalk® 0x61〜0x6F

*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 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 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 specification */
        0x00 Normal operation mode
        0x01 Test/maintenance mode (details not specified)
} CLC_INIT_DATA

typedef struct {
    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 specification */
    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 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 communication stop state.

(3) Syntax
```c
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: 0x11→0x1F
- Specific low-power radio: 0x31→0x3F
- Extended HBS: 0x41→0x4F
- IrDA_Control: 0x51→0x5F
- LonTalk®: 0x61→0x6F

(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 into 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 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 complete stop request function

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

3. **Syntax**
   ```c
   BOOL ClcLowHalt (unsigned char device_id /* [IN] Lower-layer communication software ID */)
   ```

4. **Explanation**
   - **device_id**: Identification information for lower-layer communication software to be stopped completely.
     - Power line 0x11—0x1F
     - Specific low-power radio 0x31—0x3F
     - Extended HBS 0x41—0x4F
     - IrDA_Control 0x51—0x5F
     - LonTalk® 0x61—0x6F

5. **Return value**
   - 0: Failed complete stop
   - 1: Successful complete 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 into 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 communication stop state:
     - Data reception
       - No data is to be received.
     - Transmission request from ECHONET communication control processing block
An error is returned.