ISO/IEC 14543-4-2

Edition 1.0 2008-05

INTERNATIONAL STANDARD

Information technology – Home electronic system (HES) architecture – Part 4-2: Communication layers – Transport, network and general parts of data link layer for network enhanced control devices of HES Class 1

ISO/IEC 14543-4-2:2008-05(en)

IEC Central Office
3, rue de Varembé
CH-1211 Geneva 20
Switzerland

Tel.: +41 22 919 02 11
info@iec.ch
www.iec.ch

About the IEC
The International Electrotechnical Commission (IEC) is the leading global organization that prepares and publishes International Standards for all electrical, electronic and related technologies.

About IEC publications
The technical content of IEC publications is kept under constant review by the IEC. Please make sure that you have the latest edition, a corrigendum or an amendment might have been published.

IEC publications search - webstore.iec.ch/advsearchform
The advanced search enables to find IEC publications by a variety of criteria (reference number, text, technical committee,…). It also gives information on projects, replaced and withdrawn publications.

IEC Just Published - webstore.iec.ch/justpublished
Stay up to date on all new IEC publications. Just Published details all new publications released. Available online and once a month by email.

IEC Customer Service Centre - webstore.iec.ch/csc
If you wish to give us your feedback on this publication or need further assistance, please contact the Customer Service Centre: sales@iec.ch.

Electropedia - www.electropedia.org
The world's leading online dictionary on electrotechnology, containing more than 22 000 terminological entries in English and French, with equivalent terms in 16 additional languages. Also known as the International Electrotechnical Vocabulary (IEV) online.

IEC Glossary - std.iec.ch/glossary
67 000 electrotechnical terminology entries in English and French extracted from the Terms and definitions clause of IEC publications issued between 2002 and 2015. Some entries have been collected from earlier publications of IEC TC 37, 77, 86 and CISPR.

ISO/IEC 14543-4-2

Edition 1.0 2008-05

INTERNATIONAL STANDARD

Information technology – Home electronic system (HES) architecture – Part 4-2: Communication layers – Transport, network and general parts of data link layer for network enhanced control devices of HES Class 1

INTERNATIONAL ELECTROTECHNICAL COMMISSION

ICS 35.240.67

ISBN 2-8318-9815-3

CONTENTS

INFORMATION TECHNOLOGY – HOME ELECTRONIC SYSTEM (HES) ARCHITECTURE –

Part 4-2: Communication layers – Transport, network and general parts of data link layer for network enhanced control devices of HES Class 1

FOREWORD

1) ISO (International Organization for Standardization) and IEC (International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards. Their preparation is entrusted to technical committees; any ISO and IEC member body interested in the subject dealt with may participate in this preparatory work. International governmental and non-governmental organizations liaising with ISO and IEC also participate in this preparation.

2) In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

3) The formal decisions or agreements of IEC and ISO on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested IEC and ISO member bodies.

4) IEC, ISO and ISO/IEC publications have the form of recommendations for international use and are accepted by IEC and ISO member bodies in that sense. While all reasonable efforts are made to ensure that the technical content of IEC, ISO and ISO/IEC publications is accurate, IEC or ISO cannot be held responsible for the way in which they are used or for any misinterpretation by any end user.

5) In order to promote international uniformity, IEC and ISO member bodies undertake to apply IEC, ISO and ISO/IEC publications transparently to the maximum extent possible in their national and regional publications. Any divergence between any ISO/IEC publication and the corresponding national or regional publication should be clearly indicated in the latter.

6) ISO and IEC provide no marking procedure to indicate their approval and cannot be rendered responsible for any equipment declared to be in conformity with an ISO/IEC publication.

7) All users should ensure that they have the latest edition of this publication.

8) No liability shall attach to IEC or ISO or its directors, employees, servants or agents including individual experts and members of their technical committees and IEC or ISO member bodies for any personal injury, property damage or other damage of any nature whatsoever, whether direct or indirect, or for costs (including legal fees) and expenses arising out of the publication of, use of, or reliance upon, this ISO/IEC publication or any other IEC, ISO or ISO/IEC publications.

9) Attention is drawn to the normative references cited in this publication. Use of the referenced publications is indispensable for the correct application of this publication.

10) Attention is drawn to the possibility that some of the elements of this International Standard may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

International Standard ISO/IEC 14543-4-2 was prepared by subcommittee 25: Interconnection of information technology equipment, of ISO/IEC joint technical committee 1: Information technology.

The list of all currently available parts of the ISO/IEC 14543 series, under the general title *Information technology – Home electronic system (HES) architecture*, can be found on the IEC web site.

This International Standard has been approved by vote of the member bodies, and the voting results may be obtained from the address given on the second title page.

This publication has been drafted in accordance with the ISO/IEC Directives, Part 2.

INTRODUCTION

This part of ISO/IEC 14543 specifies the media independent requirements for the data link layer and the requirements for the network layer and the transport layer for Home Electronic System. This standard stipulates the communication stack for providing the services specified in ISO/IEC 14543-4-1. It can be used as the communication stack on the physical layers as specified in ECHONET [1] Specifications. This part of ISO/IEC 14543 is based on ECHONET[1] specifications.

ISO/IEC 14543 *Information technology – Home Electronic System (HES) architecture*, currently consists of 13 parts:

Part 2-1: *Introduction and device modularity*
Part 3-1: *Communication layers – Application layer for network based control of HES Class 1*
Part 3-2: *Communication layers – Transport, network and general parts of data link layer for network based control of HES Class 1*
Part 3-3: *User process for network based control of HES Class 1*
Part 3-4: *System management – Management procedures for network based control of HES Class 1*
Part 3-5: *Media and media dependent layers – Power line for network based control of HES Class 1*
Part 3-6: *Media and media dependent layers – Twisted pair for network based control of HES Class 1*
Part 3-7: *Media and media dependent layers – Radio frequency for network based control of HES Class 1*
Part 4: *Home and building automation in a mixed-use building (technical report)*
Part 4-1: *Communication layers – Application layer for network enhanced control devices of HES Class 1*
Part 4-2: *Communication layers – Transport, network and general parts of data link layer for network enhanced control devices of HES Class 1 (this standard)*
Part 5-1: *Intelligent grouping and resource sharing for HES Class 2 and Class 3 – Core protocol (under consideration)*
Part 5-2: *Intelligent grouping and resource sharing for HES Class 2 and Class 3 – Device certification (under consideration)*

Additional parts are under preparation.

[1] Echonet™ is the trade name of a product supplied by ECHONET Consortium. This information is given for the convenience of users of this document and does not constitute an endorsement by IEC or ISO of the product named. Equivalent products may be used if they can be shown lo lead to the same results.

INFORMATION TECHNOLOGY – HOME ELECTRONIC SYSTEM (HES) ARCHITECTURE –

Part 4-2: Communication layers – Transport, network and general parts of data link layer for network enhanced control devices of HES Class 1

1 Scope

This part of ISO/IEC 14543 specifies the services and protocol in a manner independent of the physical layer for the data link layer and for the network layer and the transport layer for usage in network enhanced home electronic systems Class 1.

ISO/IEC 14543-4 is designed to enable the use of power line and wireless protocols as transmission media. Slow transmission speeds discourage large data transfers, and it is desirable to reduce the mounting load on simple devices. In light of this situation, this part of ISO/IEC 14543 specifies the frame format for the communications middleware block to minimize message size while fulfilling the requirements of the communications layer structure.

This part of ISO/IEC 14543 specifies the protocol difference absorption processing block and a part of the communications processing block. Figure 1 shows the relationship between the protocol of ISO/IEC 14543-4 and HES reference model based on ISO/IEC 7498.

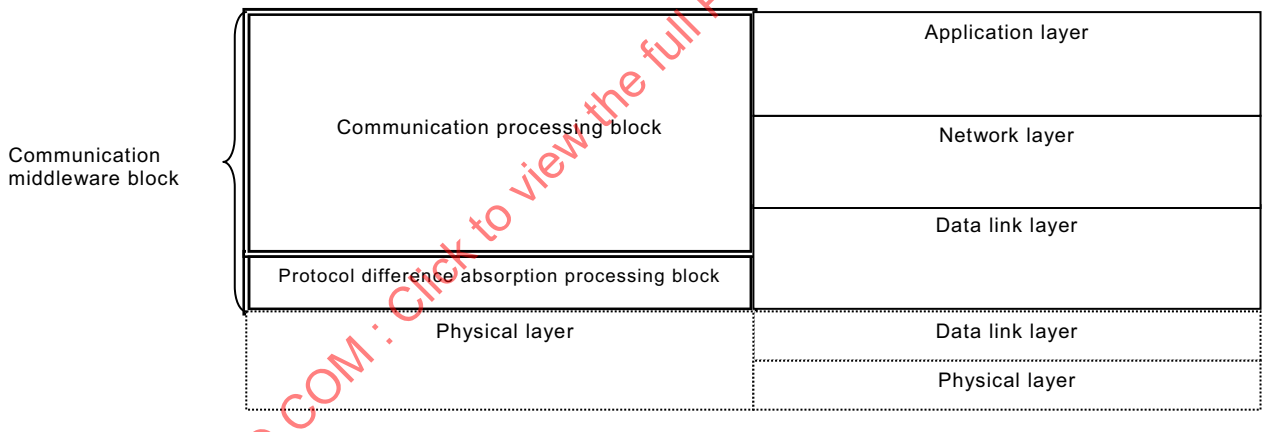


Figure 1 – Relationship between the protocol of ISO/IEC 14543-4 and OSI reference model

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 7498 (all parts), *Information technology – Open systems interconnection – Basic reference model*

ISO/IEC 14543-2-1, *Information technology – Home electronic system (HES) architecture – Part 2-1: Introduction and device modularity*

ISO/IEC 24767-2, *Information technology – Home network security – Part 2: Internal security services* (under consideration)

3 Terms, definitions and abbreviations

3.1 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 14543-2-1 and the following apply.

3.1.1
application data (ADATA)
data region for messages exchanged by communications middleware

NOTE The maximum size is 256 bytes.

3.1.2
application data counter (ADC)
indicates the size of the ADATA region

NOTE The size is variable in 1-byte increments.

3.1.3
application object (AOJ)
model of information to be disclosed to the network from information owned by the communications processing block or an access procedure model

NOTE The information or control target owned by each device is specified as a property and the operating method (setting, browsing) for this is specified as a service. AOJs are used when class or instance is not considered.

3.1.4
application programming interface (API)
assembly of interface functions for middleware

NOTE API makes it easy to operate middleware for designers.

3.1.5
application property value data (APD)
is a data value related to the application property code (APC), such as a status notification or specific setting and control by an application service code (ASC)

NOTE Detailed specifications are provided for the size, code value, etc. of the APD for each APC.

3.1.6
communications middleware block
this middleware is arranged from data link layer to application layer and performs communications processing according to the ISO/IEC 14543-4 protocol

NOTE The major features of ISO/IEC 14543-4 are implemented by communications middleware.

3.1.7
communications processing block
one processing block for the communications middleware; this block performs communication protocol processing to facilitate remote device control/monitoring processing for application software, stores information for the above and controls various information on the self-device as well as other device statuses

3.1.8
DA data
node address of the destination of messages between lower-layer communications software

3.1.9
data link address (DLA)
address permitting unique identification of a node in a home network

NOTE This is a logical address that is defined separately from the Node address native to lower-layer communications software; it consists of a NetID and NodeID.

3.1.10
data link data
data that is composed of DHD, SDLA, DDLA, ADC and ADATA

3.1.11
data link frame
frame that is composed of DDC, DHD, SDLA, DDLA, ADC and ADATA

3.1.12
data link data counter (DDC)
specifies the order of split messages, indicates the end split of messages and stipulates split-transmission message identifiers

3.1.13
data link header (DHD)
four kinds of data are included:

- the first data is the message format for the ADATA/PADATA section;
- the second specifies secure message or plain message;
- the third specifies whether DDLA is a broadcast address or an individual address;
- and the fourth constitutes a routing hop counter

3.1.14
data link router
node used to connect subnets

NOTE It connects the subnets of different lower-layer communications protocols (for different protocols, regardless of transmission media type) or divides the same protocol into subnets. The lower-layer communications protocol is connected seamlessly on the system using routing processing based on data link addresses as a function.

3.1.15
data link split frame
messages exchanged between protocol difference absorption processing blocks are called data link split frames

3.1.16
hardware address
address defined based on a medium-specific addressing scheme, such as an ISO/IEC 8802-3 address; this is a unique value for a node among the same kind of transmission medium

3.1.17
NetID
SUBNET identifier that is also a component of a data link address

3.1.18
node
communication node conforming to ISO/IEC 14543-4

NOTE In ISO/IEC 14543-4, this is a communications function to be uniquely identified by a Data Link Address. There is no distinction between the application functions of nodes. The term “node” is used to describe the function of one communication terminal.

3.1.19
node address
address to implement layer-2 communication in transmission media

NOTE In ISO/IEC 14543-4, this does not signify an Ethernet MAC [2] address.

3.1.20
NodeID
identifier used to identify a node uniquely within the SUBNET

NOTE This is a logical address converted from the Node address native to the lower-layer communications software. This is also a component of the data link address.

3.1.21
protocol difference absorption processing block
one processing block of the communications middleware

NOTE This block is intended to absorb differences of multiple protocols, including power lines and low-power wireless, to configure a single network. The block performs address translation, communication type conversion, data division and data assembly.

3.1.22
SA data
node address of the source of messages between lower-layer communications software

3.1.23
SUBNET
group of nodes using the same lower-layer communications protocol

NOTE Each subnet has a NetID; different subnets can be connected by a data link router.

3.2 Abbreviations

ADATA	Application Data
ADC	Application Data Counter
AOJ	Application Object
APD	Application Property Value Date
API	Application Programming Interface
DDLA	Destination Data Link Address
DDC	Data Link Data Counter
DHD	Data Link Header
DLA	Data Link Address
DSDATA	Data Link Split Data
PADATA	Plain Application Data
SDLA	Source Data Link Address

4 Conformance

Products that claim conformance to this International Standard shall provide communication layers with a frame format in conformance with clause 5 and 6.4 of this standard.

---

2 Ethernet™ is the trade name of a product supplied by Xerox Corporation. This information is given for the convenience of users of this document and does not constitute an endorsement by ISO or IEC of the product named. Equivalent products may be used if they can be shown to lead to the same results. For technical specifications of the MAC address see ISO/IEC 8802-3.

5 Frame format of communication layers

Figure 2 shows the frame format of the communication layers. In this document (ISO/IEC 14543-4-2) DDC, DHD, SDLA and DDLA are described. ADC and ADATA are described in ISO/IEC 14543-4-1. When property value data of 2 bytes or larger comprises application property value data (APD), the most significant value in the sequence is stored first. Bit 7 is the most significant bit and bit 0 is the least significant bit of the octet.

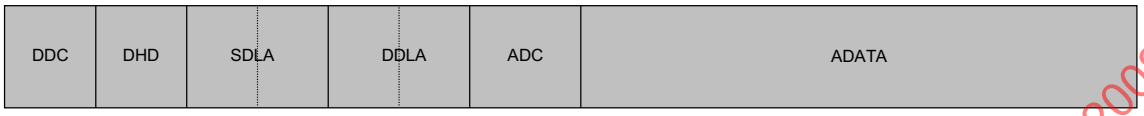


Figure 2 – Data link frame format of communication layers

6 Requirements for the physical layer and independent data link layer

6.1 Requirements for the physical layer

When two or more home network systems reside close together, the communications between home network systems must not mix. Therefore, each transmission media shall specify the principles for identifying each home network system, such as including in each communication message a house code that can identify each home network system.

The physical layer shall have the following functions.

- A function to maintain the uniqueness of the nodes' node addresses within the subnet
- A function to serve as a container for data link frame format shown in Figure 2
- An intra-subnet communication function
- A function to allow the individual nodes to retain their own profiles and report them to the network layer. The items of the profile are as follows:
  - Node address length
  - Node address mask pattern

    In the special case of a NULL value for a Node address, a special conversion rule is applied.
  - Node address
  - Maximum message length
  - Lower-layer communications software ID (+ Transmission medium ID)
  - Broadcast function ID

    A flag which shows broadcast function or no broadcast function.
  - Transmission rate
- A function to allow the individual nodes to retain their own statuses and report them to the network layer. The compulsory status items are as follows:
  - Stop

    A state in which no lower-layer communications software operations are performed.
  - Initialization

    A state in which the lower-layer communications software is initialized.
  - Normal operation

    A state in which data is transmitted to or received from a transmission medium as the primary function of the lower-layer communications software.

– Error stop

A state in which operation is stopped by the occurrence of an error.

– Suspension

A state in which operation is paused by an instruction from the communications middleware.

6.2 Functions of the data link layer

ISO/IEC 14543-4 was designed to enable the use of power line and wireless protocols as transmission media. Slow transmission speeds discourage large data transfers, and it is desirable to reduce the mounting load on simple devices. In light of this situation, this part of ISO/IEC 14543 specifies the frame format for the communications middleware block to minimize message size while fulfilling the requirements of the communications layer structure.

6.3 Possible media and their impact on layer-2

The data link layer is defined for the following media:

- Power lines
- Specific low electric power radio
- Twisted pair cable
- IrDA_Control [3]
- LonTalk® [4]
- Bluetooth™ [5]
- Ethernet

6.4 Data link layer services

6.4.1 Data link header

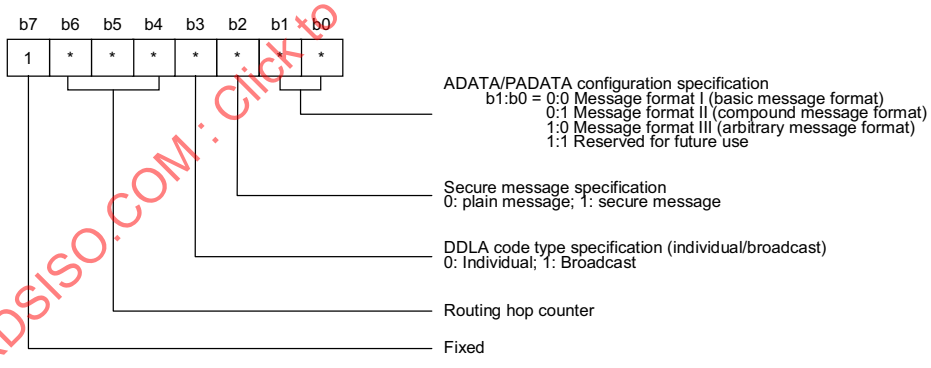


Figure 3 – Configuration of DHD

The combination of b1 and b0 specifies the message format for the ADATA/PADATA section. When b1:b0 = 0:0, it indicates message format I (basic message format), which allows one

---

3 IrDA™ is the trade name of products supplied by the Infrared Data Association. This information is given for the convenience of users of this document and does not constitute an endorsement by ISO or IEC of the products named. Equivalent products may be used if they can be shown to lead to the same results.

4 LonTalk® is the trade name of a product supplied by Echelon Corporation. This information is given for the convenience of users of this document and does not constitute an endorsement by ISO or IEC of the product named. Equivalent products may be used if they can be shown to lead to the same results.

5 Bluetooth™ is the trade name of a product supplied by Bluetooth SIG, Inc. This information is given for the convenience of users of this document and does not constitute an endorsement by ISO or IEC of the product named. Equivalent products may be used if they can be shown to lead to the same results.

message to operate on one property of one object. When b1:b0 = 0:1, it indicates message format II (compound message format), which allows one message to operate on two or more properties of one object. When b1:b0 = 1:0, it indicates message format III (arbitrary message format), whose ADATA/PADATA section is in an arbitrary format.

Bit b2 indicates whether or not the ADATA section is encrypted. When b2 = 1, it means that the ADATA section is encrypted. When b2 = 0, it means that the ADATA section is not encrypted. Detailed information about encrypted and other secure messages is given in ISO/IEC 24767-2.

Bit b3 specifies whether the DDLA (Destination Data Link Address) shown in Figure 2 is a broadcast address or an individual address. When b3 = 1, it indicates that a broadcast address is stipulated by the DDLA code. When b3 = 0, it indicates that an individual address is stipulated by the DDLA code. Broadcast address codes are discussed in 6.4.2.2.

Bits b4, b5 and b6 constitute a routing hop counter, which can be manipulated only by data link routers. When a message received at one subnet of a data link router is forwarded to another subnet, the counter is incremented. For every transmission from an ordinary node, a hop count of 0 is used. The relationship between b4, b5 and b6 and the hop count is shown in Table 1 below. The number of hops can be set to a value between 0 and 7.

Table 1 – Number of hop counts

b6	b5	b4	Hop count (Router passes)
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

6.4.2 Data link address

6.4.2.1 General

The data link address was introduced to conceal differences in lower-layer transmission media from the communications processing block and the application software. The basic requirement for the address is that it uniquely identifies a node. The data link address is a logical address defined separately from the node address unique to each given transmission medium.

A data link address consists of (1) an address (hereafter referred to as a NodeID) determined based on an address (hereafter referred to as a node address) that enables communication in Layer-2 of the transmission medium and (2) an address (hereafter referred to as a NetID) that specifies the subnet.

Specifically, it consists of a NetID and a NodeID that uniquely correspond to the node address, as shown in Figure 4. The NodeID is logically assigned so as to be unique within the subnet.

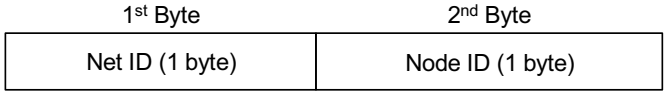


Figure 4 – Configuration of SDLA and DDLA for individual address

The NetID signifies a subnet identifier. The NetID of each node is set based on subnet information held in data link routers. Until the NetID of a node is set by a data link router, the NetID is set to 0x00, indicating NetID not specified, and the node can communicate only within the subnet to which the node belongs.

Table 2 – NetID codes

	NetID (HEX)	Meaning	Remarks
1	0x00	NetID not specified	
2	0x01 to 0x8F	NetID assigned codes	
3	0x90 to 0xFF	Codes available to user (manually assigned codes)	Used, for example, when a system manager for an apartment complex or building is present.

The NodeID is an identifier used to uniquely identify a node within a subnet. The NodeID is converted from a node address such that it is unique within the subnet. Each type of lower-layer communications software has its own conversion specifications.

6.4.2.2 Source/destination data link address (SDLA/DDLA)

This subclause provides detailed specifications for the source data link address (SDLA) and destination data link address (DDLA).

Figure 4 shows the configuration of the source data link address (SDLA) and the destination data link address (DDLA) prevailing when an individual address is stipulated by setting b3 of DHD to 0. When b3 of DHD is set to 1 to specify a broadcast, the destination data link address (DDLA) becomes a code indicating a broadcast message for specific data link address groups (including a general broadcast). In this case, the SDLA configuration is as shown in Figure 4, the DDLA configuration is as shown in Figure 5 and Table 3 and the broadcast target requirement code is as shown in Figure 6 and Figure 7.

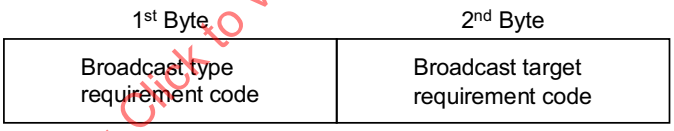


Figure 5 – DDLA (broadcast-stipulated) address configuration

Table 3 – DDLA (broadcast-stipulated) address configuration

Broadcast type requirement code	Broadcast target requirement code	Remarks
0x00	Specifies the node groups to be targeted for a broadcast within all subnets. For node group selection, see Figure 6 and Figure 7.	An intra-domain broadcast. In all subnets within a domain, a broadcast is sent to the nodes stipulated by the broadcast target requirement code.
0x01	Specifies the node groups to be targeted for a broadcast within own subnet. For node group selection, see Figure 6 and Figure 7.	An intra-own-subnet broadcast. Within own subnet, a broadcast is sent to the nodes stipulated by the broadcast target requirement code.
0x02	All nodes within the subnet having the NetID code stipulated by the broadcast target requirement code are targeted.	A general broadcast within a specified subnet. A broadcast is sent to all nodes within the subnet stipulated by the broadcast target requirement code.
0x03	Specifies the group multicast number to be targeted for a multicast within all subnets.	An intra-domain group multicast. In all subnets within a domain, a group multicast is sent to the nodes stipulated by the multicast target requirement code. This code is optional. However, if the intra-domain group multicast function is installed, the intra-own-subnet group multicast function must also be installed.
0x04	Specifies the group multicast number to be targeted for a multicast within own subnet.	An intra-own-subnet group multicast. Within own subnet, a group multicast is sent to the nodes stipulated by the multicast target requirement code. This code is optional. However, if the intra-own-subnet group multicast function is installed, the intra-domain group multicast function must also be installed.
0x05 – 0x7F	Reserved for future use.	
0x80 – 0xFF	Open to user.	Used when a system manager will manage the system in a collective housing unit or small office building.

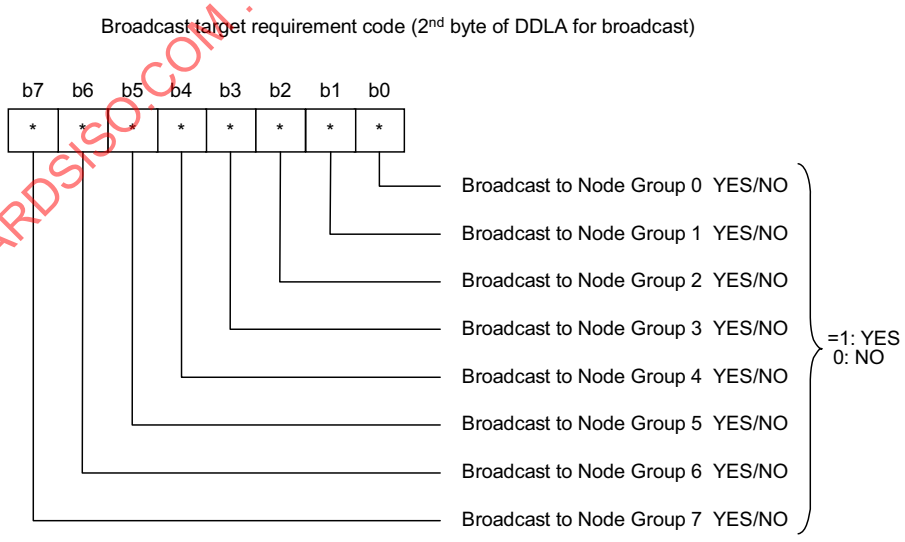


Figure 6 – Broadcast target requirement code

Note that the Node IDs of the nodes belonging to Node Groups 0 to 7 are as indicated below. For example, a node whose ID is 0xA2 belongs to Group 2.

	0	8	4	C	2	A	6	E	1	9	5	D	3	B	7	F	
0																	Group 0
8																	
4																	Group 1
C																	
2																	Group 2
A																	
6																	Group 3
E																	
1																	Group 4
9																	
5																	Group 5
D																	
3																	Group 6
B																	
7																	Group 7
F																	

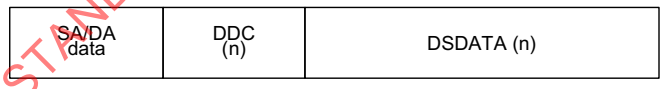
Figure 7 – Node group requirement bit specifications

6.4.3 Application data counter

Indicates the size of the application data region (ADATA region) shown in Figure 2. The size is variable in 1-byte increments. The acceptable ADATA region size ranges from 6 bytes to 256 bytes (0x06 to 0xFF; 0x00 = 256). The lower limit is 6 bytes, which indicates that a message consists of at least 6 bytes.

6.4.4 Data link split frames

In these specifications, messages exchanged between protocol difference absorption processing sections are called data link split frames. The composition of these messages allows differences in message size to be absorbed so that the processing at the communications processing block is independent of the lower-layer communications software.



N : Number of split frames (max. 18)
DDC (n) : Data Link Data Counter (1 byte)
DSDATA (1)-(n) : Data Link Data from DHD–ADATA (Data Link Frame) split into n parts. Max. 262 bytes.
SA/DA data : DA (recipient's Node address data) when sending message
SA (source's Node address data) when receiving message
When sending, DA data is created from the DDLA value in the Data Link Frame. Includes broadcast specification data.

Figure 8 – Format for protocol difference absorption processing section

Relationship with upper-layer messages

The data link split frame described above consists of a data link frame that has been split into messages no larger than the maximum processable size for lower-layer communications software.

Each one also contains header codes (DDC) for frame splitting, assembly and routing as well as address data for the source and destination.

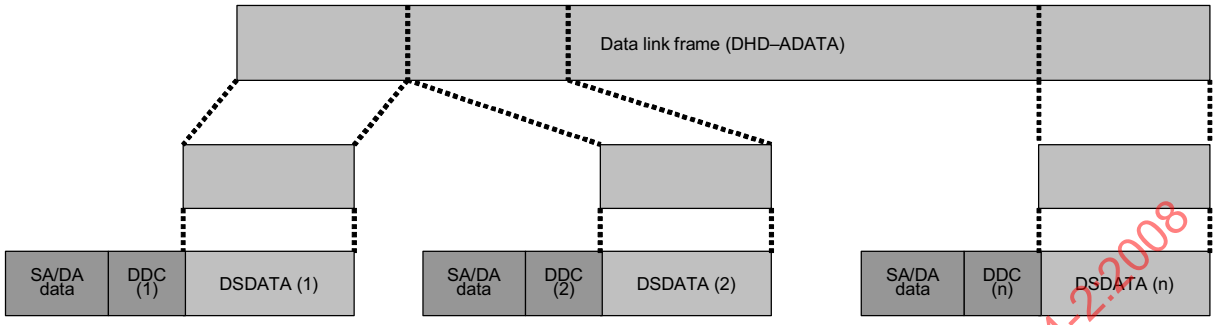


Figure 9 – Relationship with upper-layer messages

6.4.5 Data link data counter

Messages may be split into a maximum of eight components and b0–b2 show the order of the split messages (starts at b0 = b1 = b2 = 0; ends at a maximum of b0 = b1 = b2 = 1). The split message identifier requirement bit (b4, b5, b6) is also specified for cases in which a message from the communications processing block is sent repeatedly to the same node and in which all of the repeated messages require splitting. However, the method for setting this value will not be specified here. Therefore, in the receiving-side protocol difference absorption processing block, messages with the same node address for the source and the same values for b4–b6 are assembled based on the data contained in the b0–b2 split counter.

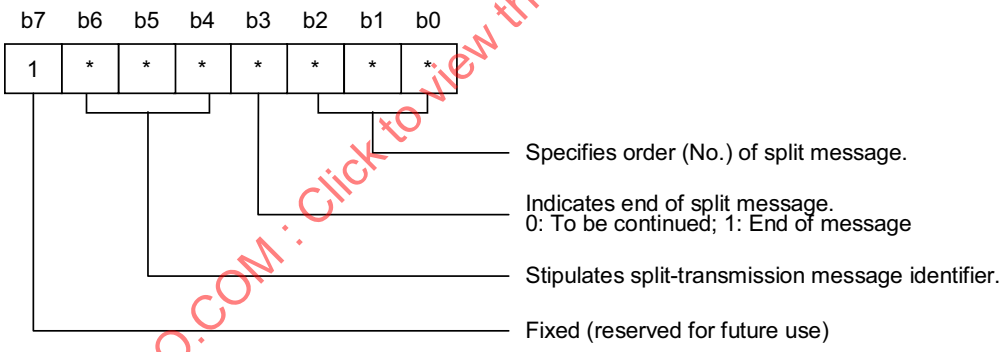


Figure 10 – Configuration of DDC

When a message is not split, values shall be set as follows: b2:b1:b0 = 0:0:0 and b3 = 1.

When the intended destination is the same, it is recommended that split messages having the same identifier be sent without any intervening messages.

6.5 Protocol difference absorption processing block

6.5.1 Overview

This subclause describes the processing specifications that are to be specified by the communications processing block in the protocol difference absorption processing block. Note that the processes are used simply to describe basic processing in the communications processing block and are not intended as specifications for actual software structure.

- Message receipt/assembly processing
- Message splitting/transmission processing

- Address conversion processing
- Conversion by communications type processing
- Lower-layer common lower-layer communications interface processing

6.5.2 Message receipt/assembly processing

A message is received from the lower-layer communications software block via an individual lower-layer communications interface. Depending on the message header (DDC) in the protocol difference absorption processing block, two types of processing are possible:

- when received message is a complete (not split) message;
- when received message is a split message.

When the received message is complete (not split), message data (DSDATA) in the protocol difference absorption processing block is handed off to common lower-layer communications interface processing as data to be forwarded to the communications processing block and processing is terminated.

When the received message is a split message, message assembly processing is performed in the communications processing block using the received message, the node address of the source, the message identification stipulator within the DDC and the split message number, all of which were received from the lower-layer communications software. The received message is held until the message is properly assembled. Once assembly is complete, the message is handed off to common lower-layer communications interface processing as data to be forwarded to the communications processing block and processing is terminated.

Such items as wait-time for the next message (required for assembly) and the number of messages that can be processed simultaneously are not specified. Also, split message receiving functions are not required.

Node address data passed on from the lower-layer communications software as an individual lower-layer communications interface is used only for message assembly and do not require address conversion processing.

6.5.3 Message splitting/transmission processing

When the send message length is smaller than the transmission buffer size (splitting not required), a DDC stipulating no splitting is created, the send message data and node address data for the intended recipient are handed off to the lower-layer communications software block via the individual lower-layer communications interface and processing is terminated.

When the send message length is larger than the transmission buffer size (splitting required), the message data is split into pieces of an appropriate size smaller than the transmission buffer size. These pieces, designated DSDATA(1)–DSDATA(n), are then handed off in order to the lower-layer communications software block via the individual lower-layer communications interface, together with a DDC for each (DDC[1]–DDC[n]) and the node address of the intended recipient, starting with the first message. Once all messages have been handed off, processing is terminated.

The actual size and number of split messages and the decision on whether or not to incorporate splitting functions, are implementation issues and therefore are not specified.

6.5.4 Address conversion processing

Two types of processing are possible depending on the address data (consists of a data link header code and a NodeID code) received together with the send message from common lower-layer communications interface processing

- when the address stipulates broadcast, processing is passed to communications type conversion processing,

- when the address stipulates individual, address conversion processing is performed for the specified NodeID and node address for each lower-layer communications protocol, the address is designated as the intended recipient address and processing is passed to message splitting/transmission processing.

NodeID and node address conversion processing specifications are dependent on each lower-layer communications protocol. Each lower-layer communications protocol is specified for each transmission media.

The following are examples of address transmission. For some transmission media, NodeID is equal to node address, so in the case, address conversion is not needed. For some of the other transmission media, the value of the lowest byte of the node address is the same as NodeID.

The number of available node addresses varies from one lower-layer communications protocol to another. Furthermore, a special node address use is defined by some lower-layer communications protocols. To comply with two or more lower-layer communications software programs, the NodeID and node address shall be selected while giving due consideration to the aforementioned matter.

6.5.5 Communications type conversion processing

6.5.5.1 Overview

Broadcast addresses are converted based on the broadcast stipulated intended recipient data received from address conversion processing (this consists of the data link header code and the code for b2 of the DDLA). Here, one of two types of processing is performed, based on whether or not broadcast is stipulated in the lower-layer communications protocol.

6.5.5.2 When broadcast is stipulated in the lower-layer communications protocol

The intended recipient requirement data is converted in accordance with the lower-layer communications protocol broadcast requirement specifications. The broadcast requirement data, intended recipient address and the send message are handed off to message splitting/transmission processing and processing is terminated.

6.5.5.3 When broadcast is not stipulated in the lower-layer communications protocol

All transmission recipient node addresses are extracted and the node address data (extracted in order) and the send message are handed off to message splitting/transmission processing until transmission of the stipulated message to all node addresses has been completed. When the requests intended for all node addresses have been handed off to message splitting/transmission processing, processing is terminated.

Specifications for establishing broadcast address data using b2 of the DDLA are described below for each lower-layer communications protocol.

6.5.6 Common lower-layer communications interface processing

Provides a common lower-layer communications interface to the communications processing block. settings and control request data (send messages, etc.) are received from the communications processing block via the common lower-layer communications interface. If the data consists of send message data, it is handed off to address conversion processing; if it consists of lower-layer communications block settings or data request data, it is handed off to the lower-layer communications software block via the individual lower-layer communications interface. The individual lower-layer communications interface is shown in Annex A.

In contrast, when received message data is received from message receipt/assembly processing, or when settings/data response data is received from the lower-layer communications software block via the individual lower-layer communications interface, it is notified to the communications processing block in the format specified in the common lower-layer communications interface.

7 Requirements for the network layer

7.1 Overview

This clause presents the specifications for communications processing in the communications middleware. Note that the processes are used simply to describe basic processing in the communications processing block and are not intended as specifications for actual software structure.

- Received message judgment processing
- Routing processing
- Send message assembly and management processing

7.2 Received message determination processing

7.2.1 Overview

Processing shall be performed to confirm the recipients of messages received from the common lower-layer communications interface. The received message identification processing specifications are divided into those for data link routers and those for nodes without the data link router function. Each of the two types of processing is defined below. The home DLA value, the home subnet NetID value, etc. shall be retained as properties of the node profile class of the profile object.

7.2.2 Received message identification processing specifications for nodes without the data link router function

If none of cases 1 to 9 listed below applies, the received message shall be discarded and the processing shall be terminated.

Case 1: b3 of the DHD is 0 (individual message), the value of the hop counter in the DHD is 0, the NetID in the SDLA is 0x00 or the same as the NetID in the home DLA and the DDLA matches the home DLA.

Case 2: b3 of the DHD is 1 (broadcast message), the value of the hop counter in the DHD is 0, the NetID in the SDLA is 0x00 or the same as the NetID in the home DLA, the broadcasting type code is 0x00 or 0x01 and the broadcasting target code matches the home node group.

Case 3: b3 of the DHD is 1 (broadcast message), the value of the hop counter in the DHD is 0, the NetID in the SDLA is 0x00 or the same as the NetID in the home DLA, the broadcasting type code is 0x02 and the broadcasting target code points to the home NetID.

Case 4: b3 of the DHD is 1 (broadcast message), the value of the hop counter in the DHD is 0, the NetID in the SDLA is 0x00 or the same as the NetID in the home DLA, the broadcasting type code is 0x03 and the broadcasting target code matches the home group multicast number. An intra-domain group multicast is an optional function, so nodes that do not hold this function shall discard the received message and the processing shall be terminated.

Case 5: b3 of the DHD is 1 (broadcast message), the value of the hop counter in the DHD is 0, the NetID in the SDLA is 0x00 or the same as the NetID in the home DLA, the broadcasting type code is 0x04 and the broadcasting target code matches the home group multicast number. An intra-own-subnet group multicast is an optional function, so nodes that do not hold this function shall discard the received message and the processing shall be terminated.

Case 6: b3 of the DHD is 0 (individual message), the value of the hop counter in the DHD is 1-7, the NetID in the SDLA is not the same as the NetID in the home DLA and the DDLA matches the home DLA.

Case 7: b3 of the DHD is 1 (broadcast message), the value of the hop counter in the DHD is 1-7, the NetID in the SDLA is not the same as the NetID in the home DLA, the broadcasting type code is 0x00 and the broadcasting target code is 0xFF or matches the home node group.

Case 8: b3 of the DHD is 1 (broadcast message), the value of the hop counter in the DHD is 1-7, the NetID in the SDLA is not the same as the NetID in the home DLA, the broadcasting type code is 0x02 and the broadcasting target code points to the home NetID.

Case 9: The value of the hop counter in the DHD is 0 and the NetID in the SDLA is 0x00.

Even if one of the cases listed above applies, the received message shall be discarded and the processing shall be terminated if:

Case 10: A node that does not have the secure communication function receives a message for which b2 of the DHD is 1 (secure message).

Case 11: A node that does not have the composite message processing function receives a message for which b1 and b0 of the DHD are 0 and 1, respectively (composite message).

Case 12: A message is received for which b7 of the DHD is 0.

Case 13: A message is received for which b1 and b0 of the DHD are 1 and 0, respectively or a message is received for which b1 and b0 of the DHD are both 1.

7.2.3 Specifications for the received message identification processing for data link routers

In the case of a data link router, the recipient of the received message shall be confirmed (based on the DHD and DDLA data) and if one of the cases listed below applies, the processing of the received message shall be handed only to the routing processing part. With regard to cases 1 and 2, however, one of the sets of processing specifications shall be implemented.

Case 1: The DDLA code (designation of type) in the DHD specifies individual transmission (b3 = 0), the DDLA does not match the home DLA and the NetID in the DDLA does not match the NetID in the home DLA.

Case 2: The DDLA code (designation of type) in the DHD specifies individual transmission (b3 = 0) and the DDLA does not match the home DLA.

Case 3: The DDLA code (designation of type) in the DHD specifies broadcasting (b3 = 1) and the DDLA specifies subnet-specific intra-subnet simultaneous broadcasting for a subnet other than the home subnet.

Case 4: The DDLA code (designation of type) in the DHD specifies broadcasting (b3 = 1), the broadcasting type requirement code in the DDLA specifies intra-domain group multicast and the broadcasting target code in the DDLA does not match the home group multicast number.

If the following applies, the received message processing shall be handed to the routing processing and object processing parts:

Case 5: The DDLA code (designation of type) in the DHD specifies broadcasting (b3 = 1), the DDLA broadcasting type code specifies intra-domain broadcasting and the broadcasting target code points to a node group that includes the home NodeID.

Case 6: The DDLA code (designation of type) in the DHD specifies broadcasting (b3 = 1), the broadcasting type requirement code in the DDLA specifies intra-domain group multicast and the broadcasting target code in the DDLA matches the home group multicast number.

If one of the cases listed below applies, the processing shall be handed only to the object processing part.

Case 7: The DDLA code (designation of type) in the DHD specifies individual transmission (b3 = 0) and the DDLA matches the home DLA.

Case 8: The DDLA code (designation of type) in the DHD specifies broadcasting (b3 = 1), the DDLA broadcasting type code specifies intra-subnet broadcasting and the broadcasting target code points to a node group that includes the home NodeID.

Case 9: The DDLA code (designation of type) in the DHD specifies broadcasting (b3 = 1), the broadcasting type requirement code in the DDLA specifies intra-own-subnet group multicast and the broadcasting target code in the DDLA matches the home group multicast number.

If none of the cases listed above applies, the received message shall be discarded and the processing shall be terminated.

7.3 Routing processing

7.3.1 Overview

The routing processing specifications are divided into specifications for data link routers and specifications for nodes without the data link router function. The routing processing uses data held as properties of the node profile class and router profile class in the profile object.

7.3.2 Routing processing for nodes without the data link router function

There are two types of routing processing specifications for nodes without the data link router function: the simple type and the advanced type. Either type of specifications may be implemented and the processing to be performed differs accordingly.

Simple type routing processing specifications: All messages to other subnets are handed to the default router. Specifically, the default router NodeID data is specified as the data to specify the destination location within the subnet, the message to be sent is handed together with the processing to the protocol difference absorption processing section through the common lower-layer communications interface and the processing at the communications processing block is terminated.

Advanced type routing processing specifications: Only the messages that have been identified as deliverable messages shall be transmitted. The NodeID data of the appropriate router is specified, the processing is handed to the protocol difference absorption processing section through the common lower-layer communications interface and the processing at the communications processing block is terminated. Appropriate router shall mean the router located on the transmission path as determined based on the transmission path identified using the data on all routers. The messages that have been identified as undeliverable messages shall be discarded instead of being transmitted. In this case, the processing at the communications processing block shall be terminated after the message is discarded. Implementation of the advanced type specifications requires that the data on all routers be acquired from the router in advance.

7.3.3 Routing processing for data link routers

The normal operation of a data link router shall be defined as the state in which the data link router can perform the processing to respond to a write or read request for an object it holds and can perform the following processing:

- received message routing processing;
- default router processing.

The received message routing processing is performed when a data link router is requested to perform message routing and the data link router is located on the direct routing path, whereby the received message is sent to a node located in an adjacent subnet other than the adjacent subnet that received the message. An adjacent subnet shall mean a subnet whose location is such that a data link router can exchange messages directly with that subnet (see Figure 11).

The default router processing is performed when a data link router is requested to perform message routing and the data link router is not located on the direct routing path, whereby the received message is sent to a router located in the subnet that received the message and on the direct routing path for the received message.

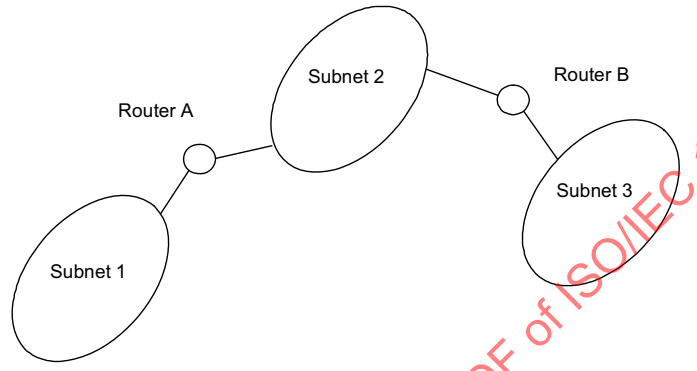


Figure 11 – Subnet connections

When a data link router is requested to perform message routing, it shall perform the received message routing processing if any one of the following conditions is satisfied:

- The DDLA code type in the DHD is "individual" and the path to the subnet specified by the NetID in the DDLA is located on the side of the adjacent subnet that is different from the adjacent subnet that received the message.
- The DDLA code type in the DHD is "broadcast," the DDLA broadcasting type code is 0x02 and the path to the subnet specified by the NetID specified by the broadcasting target code is located on the side of the adjacent subnet that is different from the adjacent subnet that received the message.

If neither of these conditions is satisfied, it shall attempt to perform the default router processing. If the processing cannot be completed successfully, the received message shall be discarded. Each of the two types of processing is explained below.

a) Received message routing processing

When a data link router receives a message, it shall check the data link frame of the message and if one of the following conditions is satisfied, perform the received message routing processing:

- b3 of the DHD is 0 (individual message) and the DDLA is different from its own DDLA (individual message processing);
- b3 of the DHD is 1 (broadcast message) and the destination NetID (broadcasting type code) is 0x00 (intra-domain broadcasting processing);
- b3 of the DHD is 1 (broadcast message), the destination NetID (broadcasting type code) is 0x02 and the destination NodeID (broadcasting target code) is different from its own NetID (subnet-specific broadcasting processing for a subnet different from the one it belongs to).

The received message routing processing shall include the following:

1) A check to confirm that it is justifiable to perform routing processing

The received message routing processing shall be terminated if:

- the value of the hop counter in the DHD is 7 (maximum hop value error);
- the DDLA code type in the DHD is “individual” and the NetID in the DDLA is not included in the data link router’s own data on all routers (destination unknown);
- the DDLA code type in the DHD is “individual” and the NetID in the DDLA is included in the data link router’s own data on all routers, but the path thereto is not included (path unknown);
- the DDLA code type in the DHD is “broadcast,” the DDLA broadcasting type code is 0x02 and the NetID specified by the broadcasting target code is not included in the data link router’s own data on all routers (destination unknown); or
- the DDLA code type in the DHD is “broadcast,” the DDLA broadcasting type code is 0x02, the NetID specified by the broadcasting target code is included in the data link router’s own data on all routers, but the path thereto is not included (path unknown).

2) Adding 1 to the hop counter value

The DHD hop counter value shall be increased by 1.

3) Determination of the destination adjacent subnet

Based on the data on all routers, the adjacent subnet located on the path to the subnet that has the destination data link router or node shall be determined.

4) Transmission request

The lower-layer communications software shall be requested to send the processed data link frame as a data link message and the received message routing processing shall be terminated.

- If individual transmission is specified and the destination node is located in the selected adjacent subnet, the lower-layer communications software shall be requested to send the data link frame to that node.
- If subnet-specific broadcasting is specified and the selected adjacent subnet is the destination subnet for the subnet-specific broadcast, the lower-layer communications software shall be requested to broadcast the data link frame.
- If individual transmission or subnet-specific broadcasting is specified and there is a data link router that is located on the routing path to the destination subnet for the subnet-specific broadcast or the destination node located in the selected adjacent subnet, the lower-layer communications software shall be requested to send the data link frame to that data link router.
- If intra-domain simultaneous broadcast is specified, the lower-layer communications software shall be requested to broadcast the data link frame to all adjacent subnets with the exception of the adjacent subnet that received the message.

b) Default router processing

If a data link router receives a message, it shall check the data link frame of the message and if one of the following conditions is satisfied, perform the default router processing:

- b3 of the DHD is 0 (individual message) and the DDLA is different from its own DDLA (individual message processing);
- b3 of the DHD is 1 (broadcast message), the destination NetID (broadcasting type code) is 0x02 and the destination NodeID (broadcasting target code) is different from its own NetID (subnet-specific broadcasting processing for a subnet different from the one it belongs to).

The default router processing shall include the following:

1) A check to confirm that it is justifiable to perform default router processing

The default router processing shall be terminated if:

- the DDLA code type in the DHD is “individual” and the NetID in the DDLA is not included in the Data Link Router’s own data on all Routers (destination unknown);
- the DDLA code type in the DHD is “individual” and the NetID in the DDLA is included in the data link router’s own data on all routers, but the path thereto is not included (path unknown);
- the DDLA code type in the DHD is “broadcast,” the DDLA broadcasting type code is 0x02 and the NetID specified by the broadcasting target code is not included in the data link router’s own data on all routers (destination unknown); or
- the DDLA code type in the DHD is “broadcast,” the DDLA broadcasting type code is 0x02 and the NetID specified by the broadcasting target code is included in the data link router’s own data on all routers, but the path thereto is not included (path unknown).

2) Transmission request

The lower-layer communications software shall be requested to send the data link frame as a data link message and the processing shall be terminated. The destination subnet shall be the adjacent subnet that received the message.

- If individual transmission or subnet-specific broadcasting is specified and there is a data link router that is located on the routing path to the destination subnet for the subnet-specific broadcast or the destination node located in the destination adjacent subnet, the lower-layer communications software shall be requested to send the data link frame to that data link router.

7.4 Send message creation/management processing

When the data necessary to create a data link message is received from startup processing or object processing, the data required for a data link message, such as self-DLA, data link header (DHD) and application data counter (ADC), is added to create the message and processing is then handed off to routing processing.

8 Requirements for the transport layer

None.

Annex A
(informative)

API functions

A.1 API function for application layer

No specific API function is required.

A.2 API functions for individual lower-layer communications interface

A.2.1 General

This clause provides the physical layer interface as APIs for individual lower-layer communications interface. The requirements set forth presume that the API process is implemented in the lower-layer communication software in the physical layer (the protocol difference absorption processing block calls a lower-layer communication software process). It describes the API function for physical layer (lower-layer communication layer) for C (ANSI) language .The detail APIs are specified because the specifications are intended to secure interchangeability of the lower-layer communication layer viewpoint of the communication middleware software developer. The API functions to be specified for C language are based on the following assumptions. This does not mean that the setting and use of functions other than those specified in this clause are prohibited.

- An 8-bit to 32-bit C-language-compatible microcomputer
- An operating system such as Windows or µITRON

The API functions for other languages will be specified in the future.

A.2.2 List of individual low-layer communication interface functions

The following 29 functions are listed in Table A.1 as individual lower-layer communication interfaces for physical layer interface functions.

Table A.1 – List of individual low-layer communication interface functions

No.	API name	API function name	Function
1	Request for lower-layer communication software type	LowGetDevID	Requests type and ID of lower-layer communication software.
2	Request for initialization	LowInit	Requests initialization of lower-layer communication software.
3	Request for operation start	LowRequestRun	Requests that lower-layer communication software start operation.
4	Fault notice	LowSetTrouble	Notifies lower-layer communication software of fault (error) status of high-order layer from Protocol Difference Absorption Processing Block.
5	Request for warm start	LowStart	Requests that lower-layer communication software perform a warm start process.
6	Request for suspension	LowSuspend	Requests that lower-layer communication software suspend operation.
7	Request for operation restart	LowWakeup	Requests that lower-layer communication software restart operation.
8	Request for profile data acquisition	LowGetProData	Gets profile data (static information) of lower-layer communication software.

No.	API name	API function name	Function
9	Request for status data acquisition	LowGetStatus	Gets dynamic status (processing fault, address redundancy, etc.) of lower-layer communication software.
10	Request for data transmission	LowSendData	Requests that lower-layer communication software send data.
11	Transmission result acquisition	LowGetSendResult	Requests data transmission result from lower-layer communication software.
12	Request for transmission stop	LowSendCancel	Requests that lower-layer communication software stop data transmission.
13	Request for received data	LowReceiveData	Requests that lower-layer communication software exchange received data.
14	Address information acquisition	LowGetAddress	Gets address information, such as Node addresses or house codes, recognized by lower-layer communication software.
15	Request for address information setup	LowSetAddress	Sets such address information as Node addresses and house codes for lower-layer communication software.
16	Request for physical address translation	LowReqToMac	Requests translation of NodeID into corresponding Node address.
17	Request for NodeID translation	LowReqToID	Requests translation of Node address into corresponding NodeID.
18	Request for broadcast destination address acquisition	LowReqBcastID	Requests target NodeID for broadcast.
19	Request for complete initialization	LowInitAll	Requests that lower-layer communication software effect initialization and acquire house code information again.
20	Request for communication stop	LowStop	Requests that lower-layer communication software stop communications.
21	Request for complete stop	LowHalt	Requests that lower-layer communication software stop completely.
22	Reception of stop notification	LowReceiveStop	Requests the lower-layer communication software to provide a stop notification
23	Lower-layer communication software address table data size acquisition	LowGetAddressTable DataSize	Acquires the number of lower-layer address table data pairs held (lower-layer communication software).
24	Lower-layer communication software address table data acquisition	LowGetAddressTable Data	Acquires lower-layer address table data held (lower-layer communication software).
25	Master router notification	LowSetMasterRouter Flag	Notifies the lower-layer communication software as to whether or not the home node is a master router.
26	Hardware address data acquisition	LowGetHardwareAdd ress	Acquires the hardware address data for the home node (lower-layer communication software).
27	Node address list acquisition	LowGetDLAList	Acquires the Node address list held (lower-layer communication software).
28	Master router data acquisition	LowGetMasterRouterI nfo	Acquires the master router data held (lower-layer communication software).
29	Hardware address conversion request	LowReqToHardwareA ddress	Requests the hardware address that corresponds to the Node address data delivered (lower-layer communication software).

A.2.3 Individual lower-layer communication interface detail specifications

Individual Lower-Layer Communication Interface Detail Specifications is described below. “Lower-layer communication software ID “ in the Function is power line = 0x11 to 0x1F, Specific low electric power radio= 0x31 to 0x3F, Twisted Pair Cable= 0x41 to 0x4F, IrDA_Control = 0x51 to 0x5F, LonTalk® = 0x61 to 0x6F, Bluetooth™ = 0x71 to 0x7F, Ethernet™ = 0x81 to 0x8F, 0x90 to 0xFF:for future reserved.

A.2.3.1 LowGetDevID

(1) Name

Lower-layer communication software type request function

(2) Function

Requests lower-layer communication software ID indicating lower-layer communication software type.

(3) Syntax

```
BOOL    LowGetDevID (
            unsigned char *device_id      /* [OUT] Lower-layer communication software ID
                                             */
        )
```

(4) Explanation

device_id : Lower-layer communication software ID

(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 (LowInit) and operation start request function (LowRequestRun).

A.2.3.2 LowInit

(1) Name

Initialization request function

(2) Function

Requests that lower-layer communication software effect initialization (by performing a cold start) and acquire Node address again. Upon receipt of this request, the lower-layer communication software performs a cold start to switch to the communication stop state and then sets the initialization parameters for itself.

(3) Syntax

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

(4) Explanation

device_id : Lower-layer communication software ID
*init_data : Pointer to initialization parameter of the common specification item
*low_init : Pointer to initialization parameter, which differs for each lower-layer

communication software. Parameter contents are specified for each discrete lower-layer communication software program.

(5) Return value

0: Failed initialization

1: Successful initialization

(6) Structure

```
typedef strut {
    short           sfholdtime,    /*  Maximum holding time for transmission data */
    short           rfholdtime,    /*  Maximum holding time for received data */
    unsigned char   low_mode,      /*  Operation mode specifications */
                                   0x00  Normal operation mode
                                   0x01  Test/maintenance mode
                                         (Details are not stipulated.) */
    short           node_len,      /*  Node address length */
    unsigned char   node_ad[6],    /*  Node address */
} LOW_INIT_DATA
```

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

When node_len is set to NULL, node_ad[6] is not significant. (When node_len is NULL, the Node address is not set.)

(7) Notes/restrictions

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

A.2.3.3 LowRequestRun

(1) Name

Operation start request function

(2) Function

Requests that lower-layer communication software start operation. Upon receiving this request, the lower-layer communication software starts operation.

(3) Syntax

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

(4) Explanation

device_id : Lower-layer communication software ID

(5) Return value

0: Failure to start

1: Successful start

(6) Structure

None

(7) Notes/restrictions

If the lower-layer communication software is not in the communication stop state, this

function returns “Failure to start”.

A.2.3.4 LowSetTrouble

(1) Name

Fault notice function

(2) Function

Notifies the lower-layer communication software of the fault (error) status of a high-order layer from the Protocol Difference Absorption Processing Block.

(3) Syntax

```
BOOL    LowSetTrouble (
            unsigned char device_id,     /* [IN] Lower-layer communication software ID */
            char htrouble_no             /* [IN] Higher-layer trouble number */
)
```

(4) Explanation

device_id : Lower-layer communication software ID
htrouble_no : Trouble No.
1 Trouble removed
2 Application software error
3 Communications Processing Block error
4 Protocol Difference Absorption Processing Block error

(5) Return value

0: Failed notice

1: Successful notice

(6) Structure

None

(7) Notes/restrictions

While an abnormality is reported, the lower-layer communication software performs the following operations:

- Data reception process
  Refrains from performing data reception or discards received data.
- Data transmission request from Protocol Difference Absorption Processing Block
  Causes an error to be returned.

A.2.3.5 LowStart

(1) Name

Warm start request function

(2) Function

Requests that lower-layer communication software effect initialization (by performing a warm start) while retaining the Node address. Upon receipt of this request, the lower-layer communication software performs a warm start and switches to the communication stop state.

(3) Syntax

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

(4) Explanation

device_id : Lower-layer communication software ID

(5) Return value

0: Failed request

1: Successful request

(6) Structure

None

(7) Notes/restrictions

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

When this request is received, the following warm start process is performed:

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

A.2.3.6 LowSuspend

(1) Name

Suspension request function

(2) Function

Requests that lower-layer communication software suspend operation. Upon receipt of this request, the lower-layer communication software switches to the suspension state.

(3) Syntax

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

(4) Explanation

device_id : Lower-layer communication software ID

(5) Return value

0: Failed suspension

1: Successful suspension

(6) Structure

None

(7) Notes/restrictions

If the lower-layer communication software is in a state other than normal operation, this

function returns “Failed suspension”.

If the lower-layer communication software is in the midst of data transmission when this request is received, it terminates a series of transmission processes and switches to the suspension state. If it is in the midst of data reception, on the other hand, it discards the received data and terminates the process.

The following operations are performed in the suspension state:

- Data reception

  No data is to be received.

- Data transmission request from Communication Processing Block

  An error is returned.

A.2.3.7 LowWakeup

(1) Name

Operation restart request function

(2) Function

Requests that lower-layer communication software exit the suspension state. Upon receipt of this request, the lower-layer communication software switches to the normal operation state.

(3) Syntax

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

(4) Explanation

device_id : Lower-layer communication software ID

(5) Return value

0: Failure to restart

1: Successful restart

(6) Structure

None

(7) Notes/restrictions

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

A.2.3.8 LowGetProData

(1) Name

Profile data acquisition request function

(2) Function

Acquires profile data for lower-layer communication software and a special process function address retained by the lower-layer communication software. Profile data requested by this function consists of property value information for the lower-layer

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

(3) Syntax

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

(4) Explanation

device_id : Lower-layer communication software ID

*pro_data : Pointer to profile data structure of lower-layer communication software.

**chmacfunc : Pointer to address of function for translating a NodeID to the Node address specific to the lower-layer communication software is returned. If the lower-layer communication software has a NodeID equal to the Node address or effects simple linear translation, NULL is returned. Specifications for the function arguments to be delivered are as follows:

node_id: [in] NodeID before translation
node: [out] Node address after translation

This function returns Node address size (in bytes).

**chnodefunc : Pointer to address of function for translating the Node address specific to that lower-layer communication software to a NodeID is returned. If the lower-layer communication software has a NodeID equal to the Node address or effects simple linear translation, NULL is returned. The specification for the function argument to be delivered is as follows:

node: [out] Node address before translation

As the return value, this function returns the NodeID derived from translation.

**broadfunc : Pointer to address of broadcast destination acquisition function is returned.
If lower-layer communication software has broadcast capability, NULL is returned.
Specifications for function arguments to be delivered are as follows:

bcast : [in] Broadcast target designation code for intra-domain or intra-local-subnet broadcast designation.

map[32] : [out] Returns array for broadcast destination NodeID bitmap. The relationship between broadcast destination NodeIDs 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;           /*  Lower-layer communication software ID
        unsigned char   ver[3];         /*  Lower-layer communication software version No.
                                            */
        unsigned char   maker[3];       /*  Manufacturer code */
        short           node_len;       /*  Node address length */
        unsigned char   node_ad[6];     /*  Node address */
        unsigned char   node_mask[6];   /*  Node address mask value */
        short           house_len;      /*  House code length */
        short           *housecode;     /*  Pointer to house code information */
        short           slen;           /*  Transmittable data length */
        short           rlen;           /*  Receivable data length */
        short           broad;          /*  Existence/non-existence of broadcast
                                            function (0: Non-existence, 1: Existence) */
        short           baud;           /*  Transmission rate */
} LOW_PRO_DATA
```

(7) Notes/restrictions

None

A.2.3.9 LowGetStatus

(1) Name

Status data acquisition request function

(2) Function

Requests the lower-layer communication software to provide the lower-layer communication software status data. The status data that can be obtained by this function is dynamic information such as that indicating a change to an abnormal state or the current processing state.

(3) Syntax

```
/*[IN] lower-layer communication software ID */
/*[OUT] status of the lower-layer communication software */
```

(4) Explanation

device_id : Lower-layer communication software ID

*status : Pointer to status data structure is returned.

(5) Return value

0: Failed acquisition

1: Successful acquisition

(6) Structure

```
typedef strut {
```

```
    char    upper_trouble;   /* High-order layer fault code (0–127)
                                No fault or removal of trouble (0) */
    char    low_trouble;     /* Lower-layer communication software block fault
                                code (0–127)
                                No fault or removal of trouble (0) */
    char    low_mode;        /* Operation mode code
                                Normal operation (0)
                                Test mode, such as maintenance (1)
                                Monitoring mode (2) */
    short   state;           /* Lower-layer communication software block status
                                LOW_STS_STOP  : 0 Stop status
                                LOW_STS_INI   : 1 Initializing status
                                LOW_STS_RUN   : 2 Normal processing status
                                LOW_STS_ESTOP : 3 Error stop status */
                                LOW_STS_RST   : 4 warm start state
                                LOW_STS_CSTOP : 5 communication stop state
                                LOW_STS_SPD   : 6 suspend status
} LOW_STATUS;
```

(7) Notes/restrictions

None

A.2.3.10 LowSendData

(1) Name

Data transmission request function

(2) Function

Requests that lower-layer communication software transmit data.

(3) Syntax

```
short   LowSendData (
        unsigned char device_id,        /* [IN] Lower-layer communication software
                                           type ID */
        const unsigned char *buf,       /* [IN] Pointer to transmission data */
        short snd_sz,                   /* [IN] Transmission data size */
        const unsigned char *da,        /* [IN] Physical address of transmission
                                           destination */
        unsigned char broad,            /* [IN] Broadcast specification */
)
```

(4) Explanation

device_id : Lower-layer communication software identification information.

*buf : Specifies pointer to data to be transmitted. The data to be delivered here is one of the data exchanged between protocol difference absorption processing blocks .

snd_sz : Specifies transmission data size.

*da : Specifies pointer to Node address of transmission destination within local subnet. If “broad” specifies a simultaneous broadcast within the domain or a broadcast within the local subnet, this parameter is not used and the lower-layer communication software performs a simultaneous broadcast.

broad : Specifies broadcast.

0x00 : Specifies no broadcast or a simultaneous broadcast within a

specified subnet.
0xFF : Specifies a broadcast within the domain or within the local subnet.

(5) Return value

LOW_BUFFER_FULL(0)	:	Buffer full error
LOW_NO_ERROR(1)	:	Transmission accepted
LOW_BUFFER_SIZE_ERROR(2)	:	Buffer size error
LOW_STATE_ERROR(3)	:	Internal error in lower-layer communication software

(6) Structure

None

(7) Notes/restrictions

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

A.2.3.11 LowGetSendResult

(1) Name

Transmission result acquisition request function

(2) Function

Requests the results of the last transmission of a message that was made by the lower-layer communication software in response to a request by the message transmission function.

(3) Syntax

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

/*[OUT] transmission result */

(4) Explanation

device_id : Lower-layer communication software identification information.

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

(5) Return value

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

(6) Structure

None

(7) Notes/restrictions

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

Note that “result” is meaningful only when the return value is normal (NO_ERROR).

A.2.3.12 LowSendCancel

(1) Name

Transmission stop request function

(2) Function

Requests that the lower-layer communication software cancel data transmission being performed in accordance with the data transmission function

(3) Syntax

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

(4) Explanation

device_id : Lower-layer communication software identification information.

(5) Return value

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

(6) Structure

None

(7) Notes/restrictions

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

Upon receipt of this request, the lower-layer communication software discards all data retained in the transmitting buffer.

A.2.3.13 LowReceiveData

(1) Name

Received-data request function

(2) Function

Requests received data retained by lower-layer communication software.

(3) Syntax

```
short    LowReceiveData (
         unsigned char device_id,     /*  [IN] Lower-layer communication software ID
                                          */
         unsigned char *buf,          /*  [IN] Pointer to receiving buffer */
         short buf_sz                 /*  [IN] Receiving buffer size */
         short *rcv_cz                /*  [OUT] Received data size */
         unsigned char *sa            /*  [OUT] Transmission source Node address */
)
```

(4) Explanation

device_id : Lower-layer communication software identification information.

*buf : Specifies pointer (1st byte: DDC) to receiving buffer.
buf_sz : Specifies receiving buffer size.
rcv_sz : Returns actual received data size.
sa : Returns transmission source Node address.

(5) Return value

LOW_NO_RECEIVE(0) : No received data
LOW_NO_ERROR(1) : Normal (with received data)
LOW_BUFFER_SIZE_ERROR(2) : Buffer size error
LOW_INTERNAL_ERROR(3) : Internal error in lower-layer communication software

(6) Structure

None

(7) Notes/restrictions

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

A.2.3.14 LowGetAddress

(1) Name

Address data acquisition request function

(2) Function

Requests address information retained by lower-layer communication software.

(3) Syntax

```
BOOL    LowGetAddress (
            unsigned char device_id,        /*  [IN] Lower-layer communication software ID */
            short node_len,                 /*  [OUT] Node address length */
            unsigned char node_ad[7],       /*  [OUT] Node address */
            unsigned char node_mask[7],     /*  [OUT] Node address mask value */
            short *housecode_len,           /*  [OUT] Pointer to house code information size */
            unsigned char *housecode;       /*  [OUT] Pointer to house code information */
        )
```

(4) Explanation

device_id : Lower-layer communication software identification information.
node_ad : Returns Node address size.
node_len : Returns Node address.
housecode_len : Pointer to house code information size is returned. The value “0x00” indicates that no house code information is needed.
housecode : Pointer to house code information is returned.

(5) Return value

0: Failed address acquisition

1: Successful address acquisition

(6) Structure

None

(7) Notes/restrictions

None

A.2.3.15 LowSetAddress

(1) Name

Address information setup request information

(2) Function

Sets the address information for lower-layer communication software.

(3) Syntax

```
short    LowSetAddress (
             unsigned char device_id,        /*  [IN] Lower-layer communication software ID
                                                 */
             short node_len,                 /*  [IN] Node address length */
             unsigned char node_ad[7],       /*  [IN] Node address */
             unsigned char node_mask[7],     /*  [IN] Node address mask value */
             short housecode_len,            /*  [IN] House code information size */
             unsigned char *housecode,       /*  [IN] Pointer to house code information */
)
```

(4) Explanation

device_id : Lower-layer communication software identification information.

node_ad : Specifies Node address size. The value “0x00” indicates that Node address setup is not requested.

node_len : Sets Node address.

housecode_len : Specifies house code information size. The value “0x00” indicates that house code setup is not requested.

*housecode : Specifies pointer to the house code information.

(5) Return value

LOW_NO_CHANGE(0)	:	Unchangeable with software
LOW_NO_ERROR(1)	:	Normal
LOW_INTERNAL_ERROR(3)	:	Internal error in lower-layer communication software

(6) Structure

None

(7) Notes/restrictions

None

A.2.3.16 LowReqToMac

(1) Name

Physical address translation request function

(2) Function

Requests lower-layer communication software to furnish the Node address corresponding to a delivered NodeID.

(3) Syntax

```
BOOL    LowReqToMac (
            unsigned char device_id,    /*  [IN] Lower-layer communication software ID
                                           */
            unsigned char node_id,      /*  [IN] NodeID to be translated */
            unsigned char *node,        /*  [OUT] Pointer to Node address derived from
                                           translation */
            short *node_len             /*  [OUT] Pointer to Node address size derived
                                           from translation */
        )
```

(4) Explanation

device_id	:	Lower-layer communication software identification information.
node_id	:	Sets NodeID to be translated.
*node	:	Pointer to Node address derived from translation is returned.
*node_len	:	Pointer to Node address size derived from translation is returned.

(5) Return value

0: Failed translation

1: Successful translation

(6) Structure

None

(7) Notes/restrictions

None

A.2.3.17 LowReqToID

(1) Name

NodeID conversion request function

(2) Function

Requests the NodeID that corresponds to the Node address delivered (lower-layer communication software).

(3) Syntax

```
BOOL    LowReqToID (
            unsigned char device_id,    /*[IN]  Lower-layer communication software ID */
            short node_len              /*[IN]  Node address length to be translated */
            unsigned char *node,        /*[IN]  Node address to be translated */
            unsigned char *node_id,     /*[OUT] NodeID derived from translation */
        )
```

(4) Explanation

device_id	:	Lower-layer communication software identification information.
node_len	:	Node address length to be translated
node	:	Specifies Node address to be translated.
*node_id	:	Pointer to NodeID derived from translation is returned.

(5) Return value

0: Failed translation

1: Successful translation

(6) Structure

None

(7) Notes/restrictions

None

A.2.3.18 LowReqBcastID

(1) Name

Broadcast destination acquisition request function

(2) Function

Extracts target NodeID from the DDLA intra-domain or intra-local-subnet broadcast target designation code delivered to lower-layer communication software.

(3) Syntax

```
BOOL    LowReqBcastID (
            unsigned char device_id,        /*  [IN] Lower-layer communication software ID
                                                */
            unsigned char bcast,            /*  [IN] Broadcast target designation code */
            short    *map_len               /*  [OUT] Address length for transmitting
                                                destination node */
            unsigned char map               /*  [OUT] Address information for transmitting
                                                destination node */
    )
```

(4) Explanation

device_id : Lower-layer communication software identification information.

bcast : Broadcast target designation code to be targeted (broadcast target designation code in DDLA 2nd byte position for intra-domain or intra-local-subnet broadcast designation).

map_len : Address length to bit map indicating translated NodeID

map : Returns address for bitmap indicating NodeID derived from translation. The relationship between broadcast destination NodeIDs and bits is shown below:

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

(5) Return value

0: Failed translation

1: Successful translation

(6) Structure

None

(7) Notes/restrictions

This function is not needed when the lower-layer communication software has broadcast capability.

A.2.3.19 LowInitAll

(1) Name

Complete initialization request function

(2) Function

Requests that lower-layer communication software effect initialization (by performing a cold start) and acquire house code information and Node address again. Upon receipt of this request, the lower-layer communication software performs a cold start to switch to the communication stop state and then sets the initialization parameters for itself.

(3) Syntax

```
BOOL    LowInitAll (
        unsigned char device_id,            /*  [IN] Lower-layer communication
                                                software ID */
        LOW_INIT_DATA *lowinit_data         /*  [IN] Pointer to initialization parameter
                                                (1) */
        void *low_init                      /*  [IN] Pointer to initialization parameter
                                                (2) */
)
```

(4) Explanation

device_id : Lower-layer communication software identification information.
*lowinit_data : Pointer to initialization parameter for lower-layer communication software common specification items.
*low_init : Pointer to initialization parameter, which varies with lower-layer communication software. The parameter is variously stipulated for all lower-layer communication software programs.

(5) Return value

0: Failed initialization

1: Successful initialization

(6) Structure

```
typedef strut {
    short           sfholdtime,     /*  Information on maximum holding time for data
                                        transmitted by lower-layer communication software
                                        */
    short           rfholdtime,     /*  Information on maximum holding time for data
                                        received by lower-layer communication software */
    unsigned char   low_mode,       /*  Operation mode selection */
    short           node_len,       /*  Node address length */
    unsigned char   node_ad[7]      /*  Node address */
} LOW_INIT_DATA
```

* Except for node_ad[7], NULL is to be set particularly when there is no initialization data.

* When node_len is set to NULL, node_ad[7] has no significance. (When node_len is NULL, there will be no Node address setting.)

(7) Notes/restrictions

If the 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.

A.2.3.20 LowStop

(1) Name

Communication stop request function

(2) Function

Requests the lower-layer communication software to stop the communication processing. Upon reception of this request, the lower-layer communication software will shift to the communication stop state.

(3) Syntax

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

(4) Explanation

device_id : Lower-layer communication software identification information.

(5) Return value

0: Failure to stop communications

1: Successful communication stop

(6) Structure

None

(7) Notes/restrictions

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

If the lower-layer communication software is in the midst of data transmission when this request is received, it terminates a series of transmission processes and switches to the suspension state. If it is in the midst of data reception, on the other hand, it discards the received data and terminates the process.

The following operations are performed in the suspension state:

- Data reception

  No data is to be received.

- Data transmission request from Communication Processing Block

  An error is returned.