User's Manual

R-IN32M3 Series

User's Manual

(CC-Link IE Field Intelligent device station)

- R-IN32M3-CL

All information of mention is things at the time of this document publication, and Renesas Electronics may change the product or specifications that are listed in this document without a notice. Please confirm the latest information such as shown by website of Renesas

Document number:R18UZ0015EJ0200
Issue date : Dec 25, 2014

Renesas Electronics
www.renesas.com

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
3. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics product.
5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below.

   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots etc.

   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; and safety equipment etc.

   Renesas Electronics products are neither intended nor authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems, surgical implantations etc.), or may cause serious property damages (nuclear reactor control systems, military equipment etc.). You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application for which it is not intended. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for which the product is not intended by Renesas Electronics.
6. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You should not use Renesas Electronics products or technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. When exporting the Renesas Electronics products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, who distributes, disposes of, or otherwise places the product with a third party, to notify such third party in advance of the contents and conditions set forth in this document, Renesas Electronics assumes no responsibility for any losses incurred by you or third parties as a result of unauthorized use of Renesas Electronics products.
11. This document may not be reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

(Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

Instructions for the use of product

In this section, the precautions are described for over whole of CMOS device.
Please refer to this manual about individual precaution.
When there is a mention unlike the text of this manual, a mention of the text takes first priority

1.Handling of Unused Pins

Handle unused pins in accord with the directions given under Handling of Unused Pins in the manual.

-The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of LSI, associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible. Unused pins should be handled as described under Handling of Unused Pins in the manual.

2.Processing at Power-on

The state of the product is undefined at the moment when power is supplied.

-The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the moment when power is supplied.
In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the moment when power is supplied until the reset process is completed.
In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the moment when power is supplied until the power reaches the level at which resetting has been specified.

3.Prohibition of Access to Reserved Addresses

Access to reserved addresses is prohibited.

-The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.

4.Clock Signals

After applying a reset, only release the reset line after the operating clock signal has become stable. When switching the clock signal during program execution, wait until the target clock signal has stabilized.

-When the clock signal is generated with an external resonator (or from an external oscillator) during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Moreover, when switching to a clock signal produced with an external resonator (or by an external oscillator) while program execution is in progress, wait until the target clock signal is stable.

How to use this manual

1. Purpose and target readers

This manual is intended for users who wish to understand the functions of " CC-Link IE Field Nework of intelligent device station" for designing application of it.

It is assumed that the reader of this manual has general knowledge in the fields of electrical engineering, logic circuits, and microcontrollers.

Particular attention should be paid to the precautionary notes when using the manual. These notes occur within the body of the text, at the end of each section, and in the Usage Notes section.

The revision history summarizes the locations of revisions and additions. It does not list all revisions. Refer to the text of the manual for details.

Related Documents

Literature may be preliminary versions. Note, however, that the following descriptions do not indicate "Preliminary". Some documents on cores were created when they were planned or still under development. So, they may be directed to specific customers. Last four digits of document number(described as ****) indicate version information of each document. Please download the latest document from our web site and refer to it.

The document related to CC-Link IE Field Network

Document name	Document number
R-IN32M3 Series Datasheet	R18DS0008EJ****
R-IN32M3-CL User's Manual	R18UZ0005EJ****
R-IN32M3 series User's Manual Peripheral function	R18UZ0007EJ****
R-IN32M3 Series Proguraming Manual (OS edition)	R18UZ0011EJ****
R-IN32M3 Series Proguraming Manual (Driver edition)	R18UZ0009EJ****
R-IN32M3 Series User's Manual CC-Link IE Intelligent device station	This manual

2. Notation of Numbers and Symbols

Weight in data notation: Left is high-order column, right is low-order column

Active low notation:

xxxZ (capital letter Z after pin name or signal name)

or xxx_N (capital letter _N after pin name or signal name)

or xxnx (pin name or signal name contains small letter n)

Note:

explanation of (Note) in the text

Caution:

Item deserving extra attention

Remark:

Supplementary explanation to the text

Numeric notation:

Binary ⋯ xxxx , xxxxB or n'bxxxx (n bits)

Decimal ⋯ xxxx

Hexadecimal ⋯ xxxxH or n'hxxxx (n bits)

Prefixes representing powers of 2 (address space, memory capacity):

K (kilo)⋯ $2^{10} = 1024$

M (mega)⋯ $2^{20} = 1024^2$

G (giga)⋯ $2^{30} = 1024^3$

Data Type:

Double word ⋯ 32 bits

Word ⋯ 16 bits

Byte ⋯ 8 bits

Contents

Contents of Figures

Contents of Tables

1. Basic Design Precautions

1.1 Component selection

Select components taking into consideration the information provided in the table below.

Table 1.1 Component Selection Check Sheet

No.	Item	Description	Check
1	MPU selection	Did you select an MPU that satisfies the following specifications? (1)Data width: 16 bits or higher (2)Address width: 17 bits or higher (3)Endian: Little endian (4)Timing indicated in Chapter 4	
2	RJ-45 connector selection	Is the connector an 8-pin ANSI/TIA/EIA-568-B shielded connector?	
3	Pulse transformer selection	Did you select an IEEE 802.3 1000BASE-T compatible component?	
4	PHY selection	Did you select a component that satisfies the following specifications? (1)IEEE 802.3 1000BASE-T full duplex compatible component (2)Component having an auto negotiation function (3)Component having a GMII interface (4)Component having an auto MDI/MDIX negotiation function (5)Component capable of operating at an MDC clock frequency of 7.812 MHz	
5	125-MHz crystal oscillator selection	Did you select a component having a frequency deviation within ±50 ppm?	
6	2.097152-MHz crystal oscillator selection	Did you select a component having a frequency deviation with ±50 ppm?	
7	PHY clock crystal oscillator selection	Did you select a PHY clock crystal oscillator in accordance with the required specifications of the PHY used? Frequency of crystal oscillator Total jitter of crystal oscillator	

1.2 Circuit design

Design the peripheral circuits of R-IN32M3-CL taking into consideration the information provided in the table below.

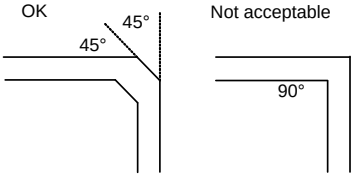
Table 1.2 Circuit Design Check Sheet

No.	Item	Description	Check
1	GMII wiring	Is a damping resistor installed for the GMII signal to supress overshooting/undershooting?	
2	PHY- RJ45 connctor connection	The signal lines between PHY and RJ45 connector mus be connected in + side and + side of each terminal, - side and – side of each terminal. Otherwise 1000BASE-T compliance test fails.	
3	Data signal	Are pull-up resistors installed for the data signals D15 to D00? (10-kΩ pull-up resistors are used in the circuit diagram examples.)	
4	PHY address	PHY address must be same as the port number of R-IN32M3-CL. PHY address 1 must be connected to MAC port 1. PHY address 2 must be connected to MAC port 2<R>	

1.3 Pattern design

Design the pattern wiring of the R-IN32M3-CL periphery taking into consideration the information provided in the table below.

Table 1.3 Pattern Design Check Sheet

No.	Item	Description	Check
1	2.097152-MHz crystal oscillator connected to R-IN32M3-CL	When connecting a 2.097152-MHz crystal oscillator to R-IN32M3-CL, place the oscillator near R-IN32M3-CL. Is the pattern length to the the CLK 2_097M pin shortest as possible? Is the pattern to the CLK 2_097M pin shielded by SG patterns?	
2	GMII wiring	Has the wiring layer and signal line thickness for the signal (GMII), which connects R-IN32M3-CL and PHY, been determined to achieve shortest pattern wiring and 50 Ω impedance?	
3	Signal pattern bending	When a pattern is bent, is it always bent at 45 degrees as shown below? 	
4	Power supply / GND pattern	Is the power supply / GND pattern wired using the thickest pattern possible?	

2. STATUS DISPLAY FUNCTIONS

2.1 LED Based Status Displays

For an intelligent device station developed using R-IN32M3-CL, mount the LEDs for indicating the status of port 1, port 2, and its host station as indicated in the table below.

For LED control, see Section 2.2.

Table 2.1 LED Status Display List

LED Name	Function	LED Status	
		LED On	LED Off
Status display of host station			
PW	Power supply status	Power supply ON	Power supply OFF
RUN	Operating status	Normal	Not normal
RD	Data reception status	Receiving data	Not receiving data
SD	Data transmission status	Sending data	Not sending data
ERR	Error status	Disconnected (excluding cases where cyclic communication has never been implemented)	Other than the left
D LINK	Data link status	Cyclic communication in progress	Cyclic communication stopped
User LED 1[Note1]	User LED 1[Note2]	—[Note1]	
User LED 2[Note1]	User LED 2[Note2]	—[Note1]	
L ERR.	Reception data error / line error status	Error frame reception (Turns on based on L ER. signal of each port.)	Normal frame reception (Turns off based on L ER. signal of each port)
Status display of port 1 (option)			
LINK	Port 1 link status[Note3]	Link up	Link down
L ER	Port 1 reception data error status[*3]	Error frame reception	Normal frame reception
Status display of port 2 (option)			
LINK	Port 2 link status[Note3]	Link up	Link down
L ER	Port 2 reception data error status[Note3]	Error frame reception	Normal frame reception

Note 1. The LED names are provisional names. Users can assign any name to these LEDs and use the LEDs to implement a desired function. For user LED control, see Section 2.2.2 "Controlling User LED 1 and User LED 2"

2. The mounting of this LED is optional.

3. The mounting of this LED is optional, bus recommended.

2.2 Controlling the LEDs

2.2.1 LED control overview

Some LEDs are controlled by hardware (R-IN32M3-CL, PHY, and power check circuits), and some LEDs are controlled by firmware.

The LEDs controlled by R-IN32M3-CL automatically turn on in accordance with the status of its host station. Wire the LED (LINK LED) controlled by PHY so that the LED turns on when the PHY link is up.

The LEDs controlled by firmware are controlled by LED on/off control functions. See Section 4.5.7 "LED control."

Table 2.2 LED Control List

LED Name	Function	R-IN32M3-CL Output Signal Name	Control Source	Output at Reset/Error[Note3]		
				Power ON Reset	System Reset	Internal WDT / External WDT / Host Station Error[Note1]
Status display of host station						
PW	Power supply status	—	Power supply check circuit	—	—	—
RUN	Operating status	RUNLEDL	Firmware, R-IN32M3-CL	Off	Off	Off
RD	Data reception status	RDLEDL	R-IN32M3-CL	Off	—	—
SD	Data transmission status	SDLEDL	R-IN32M3-CL	Off	—	—
ERR	Error status	ERRLEDL	Firmware, R-IN32M3-CL	Off	Off	On
D LINK	Data link status	DLINKLEDL	Firmware, R-IN32M3-CL	Off	Off	Off
User LED 1[Note2]	User LED 1[Note2]	USER1LEDL	Firmware, R-IN32M3-CL	Off	Off	Off
User LED 2[Note2]	User LED 2[Note2]	USER2LEDL	Firmware, R-IN32M3-CL	Off	Off	Off
L ERR.	Reception data error / line error status	—	Turns on according to LERR1LEDL and LERR2LEDL signal statuses (external AND logic required)[*3]	—	—	—
Status display of port 1 (option)						
LINK	Port 1 link status	—	PHY (Wire the LED so that it turns on when the PHY link is up.)	—	—	—
L ER	Port 1 reception data error status	LERR1LEDL	Firmware, R-IN32M3-CL	Off	Off	Off
Status display of port 2 (option)						
LINK	Port 2 link status	—	PHY (Wire the LED so that it turns on when the PHY link is up.)	—	—	—
L ER	Port 2 reception data error status	LERR2LEDL	Firmware, R-IN32M3-CL	Off	Off	Off

Note 1. An error that occurs when firmware calls the function gerR_IN32_ForceStop. For details of the function gerR_IN32_ForceStop, see Section 4.2.6 "Force Stop processing." and Section 4.5.5 "Own station status setup"

2. The LED names are provisional names. Vendors can assign any name to these LEDs and use the LEDs to implement a function. For user LED control, see Section 2.2.2 "Controlling User LED 1 and User LED 2"

3. For L.ERR LED control, see Section 2.2.3 "Controlling the L.ERR LED."

2.2.2 Controlling User LED 1 and User LED 2

R-IN32M3-CL provides two user LEDs, User LED 1 and User LED 2, where the user can assign desired functions.

You can control the on/off status of User LED 1 and User LED 2 using the functions gerR_IN32_SetUSER1LED and ger R_IN32_SetUSER2LED.

2.2.3 Controlling the L.ERR LED

For the L.ERR LED signal, set up the external AND logic for LERR1LEDL and LERR2LEDL in accordance with the figure below.

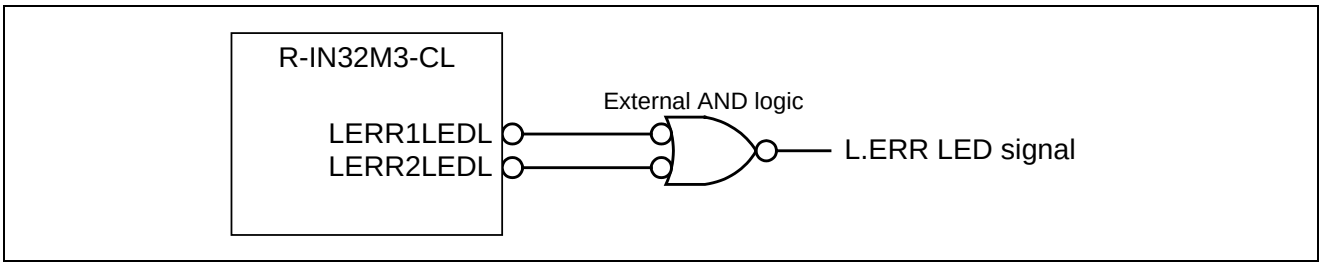


Figure 2.1 External AND Logic for Turning L.ERR On <R>

2.2.4 Enabling/Disabling LEDs

You can enable and disable the LEDs listed in the table below. Switch the LED enable/disable setting as necessary as shown in the following example:

Example:Disable the L ER.LEDs of port 1 and port 2 in a link down state since the LED light sometimes stays on when the link is down.

To disable an LED display, use the function gerR_IN32_DisableLED.

To enable an LED display, use the function gerR_IN32_EnableLED.

For LED display enable/disable functions, see Section 4.5.7 “LED control.”

Table 2.3 LEDs that Can Be Enabled/Disabled

LED Name	Function
Status display of host station	
RUN	Operating status
ERR	Error status
D LINK	Data link status
User LED 1	User LED 1
User LED 2	User LED 2
Status display of port 1	
L ER	Port 1 reception data error status
Status display of port 2	
L ER	Port 2 reception data error status

3. DATA COMMUNICATION METHOD

CC-Link IE Field Network has two types of communication methods: cyclic communication and transient communication.

3.1 Procedure for Cyclic Communication Process

Simply starting a R-IN32M3-CL driver interface function according to Section 4.2.1 "General flowchart" executes the cyclic communication process. The R-IN32M3-CL driver interface functions are used to read and write RX, RY, RWr, and RWw.

3.2 Procedure for Transient Communication Process

In transient communication, data is transmitted and received between its host station and another station on a one-to-one basis.

Transient communication involves Transient1 communication, which is required by the system, and Transient2 communication, which can be freely implemented by the vendor. Transient1 and Transient2 each involve requests and responses, and any station that receives a request or response sends TransientAck as a response in acknowledgement.

Table 3.1 Transient Communication List

Frame Type	Transmission/Reception	Master Station	Local Station	Intelligent Device Station
Transient1 request	Transmission	O	×	×
	Reception	×	O	O
Transient1 response	Transmission	×	O	O
	Reception	O	×	×
TransientAck	Transmission	O	O	O
	Reception	O	O	O
Transient2 request	Transmission	Δ	Δ	Δ
	Reception	Δ	Δ	Δ
Transient2 response	Transmission	Δ	Δ	Δ
	Reception	Δ	Δ	Δ

Remark ○ : Required, △ : Optional, × : Not required

To transmit and receive transient communication, firmware starts a R-IN32M3-CL driver interface function.

In transient communication, the request source sends a request frame, and the request destination sends a TransientAck frame and response frame. The following shows an image of transient communication.

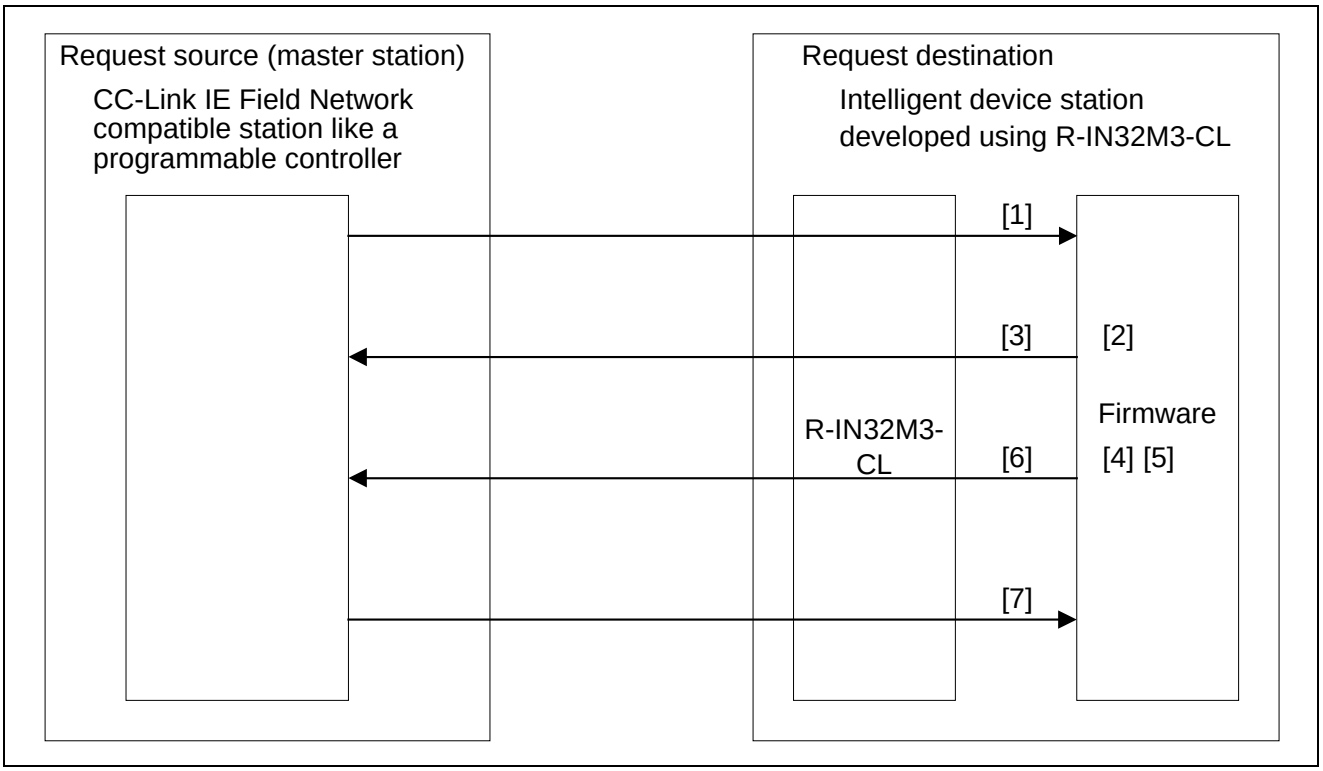


Figure 3.1 Transient1 Response Procedure (Request Source: Master Station) <R>

[1] The intelligent device station receives a Transient1 request frame. (See Section 4.2.15.)
[2] The intelligent device station creates a TransientAck frame. (See Section 4.2.29.)
[3] The intelligent device station sends the TransientAck frame. (See Section 4.2.16.)
[4] The intelligent device station analyzes the command of the Transient1 request frame. (See Sections 0 and 4.2.20.)
[5] The intelligent device station creates a Transient1 response frame in accordance with the command. (See Sections 4.2.23 and 4.2.24.)
[6] The intelligent device station sends the Transient1 response frame. (See Section 4.2.16.)
[7] The intelligent device station receives a TransientAck frame. (See Section 4.2.15.)

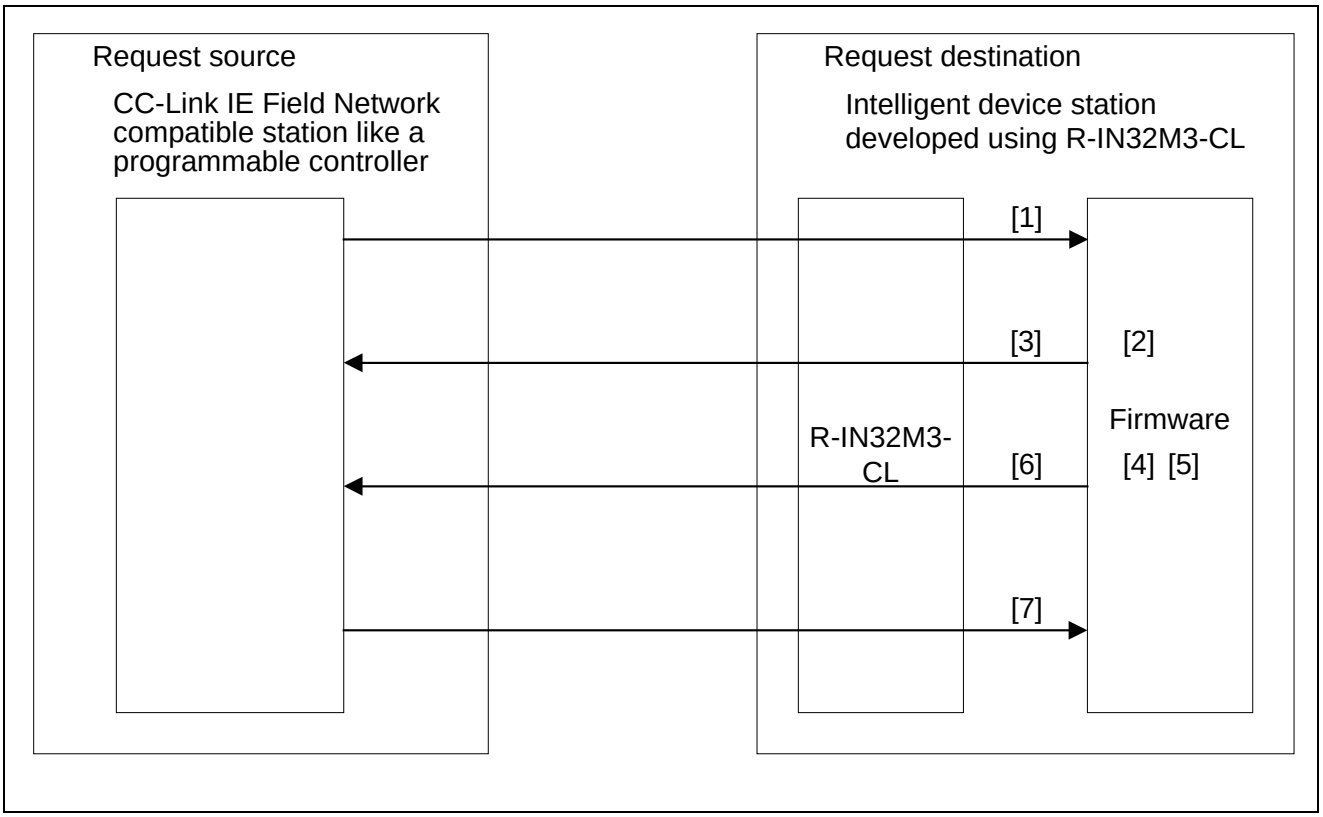


Figure 3.2 Transient2 Response Procedure <R>

[1] The intelligent device station receives a Transient2 request frame. (See Section 4.2.15.)
[2] The intelligent device station creates a TransientAck frame. (See Section 4.2.29.)
[3] The intelligent device station sends the TransientAck frame. (See Section 4.2.16.)
[4] The intelligent device station analyzes the command of the Transient2 request frame. (See Section 4.2.32.)
[5] The intelligent device station creates a Transient2 response frame in accordance with the command. (See Section 4.2.30.)
[6] The intelligent device station sends the Transient2 response frame. (See Section 4.2.16.)
[7] The intelligent device station receives a TransientAck frame. (See Section 4.2.15.)

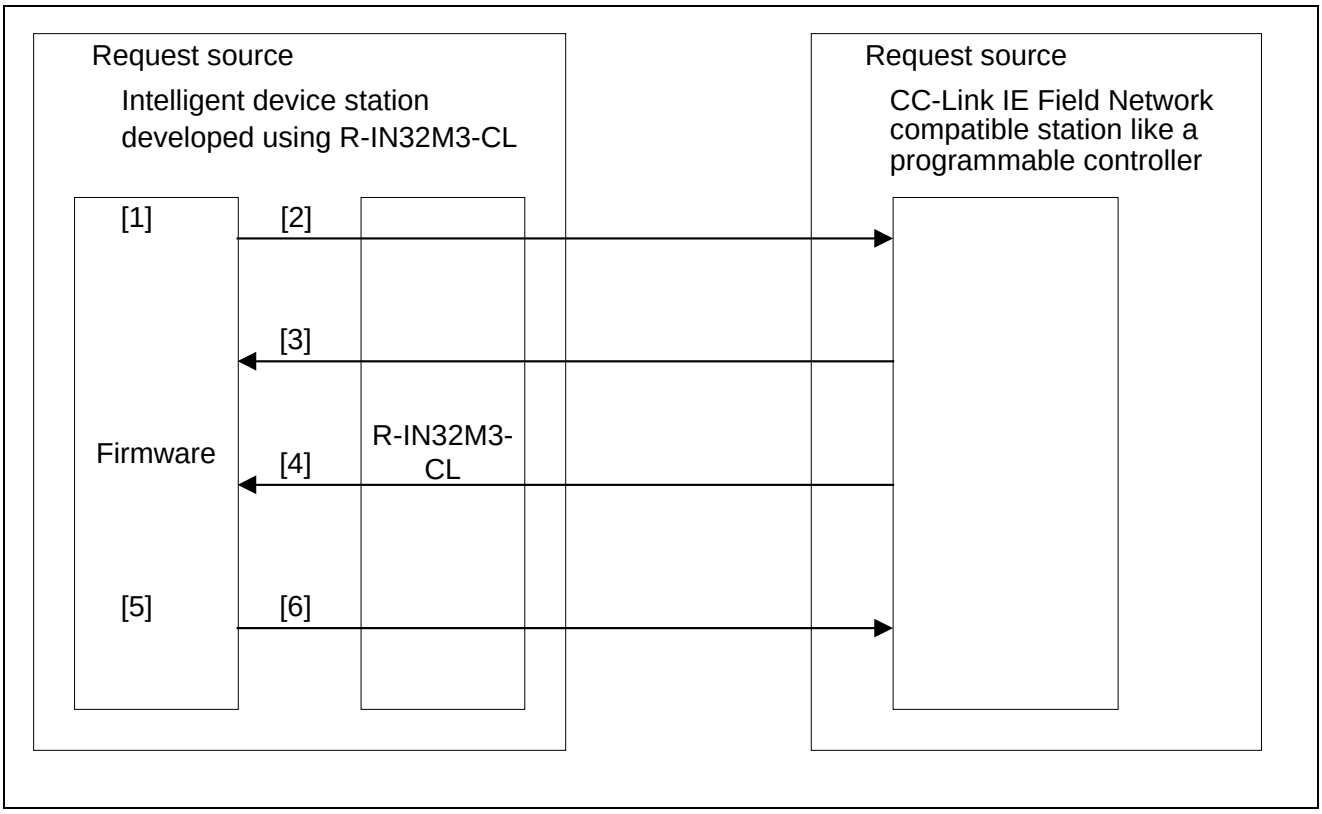


Figure 3.3 Transient2 Request Procedure

[1] The intelligent device station creates a Transient2 request frame. (See Section 4.2.30.)
[2] The intelligent device station sends the Transient2 request frame. (See Section 4.2.16.)
[3] The intelligent device station receives a TransientAck frame. (See Section 4.2.15.)
[4] The intelligent device station receives a Transient2 response frame. (See Sections 4.2.15 and 4.2.33.)
[5] The intelligent device station creates a TransientAck frame. (See Section 4.2.29.)
[6] The intelligent device station sends the TransientAck frame. (See Section 4.2.16.)

3.3 Frame Format Overview for Transient Communication

The frame format of transient communication complies with the Ethernet frames of IEEE802.3. An Ethernet frame has a frame size of 64 to 1518 bytes, from the MAC header to FCS.
This section describes the frames below which require format awareness in user programs.

- Transient1 frame
- TransientAck frame
- Transient2 frame

3.3.1 Transient1 frame format

The table below provides an overview of the Transient1 frame format.

Table 3.2 Transient1 Frame Format Overview

	Item	Size (Bytes)	Sub-Item	Size (Bytes)	Remarks
1	Transient frame common header	28	MAC header	14	
			CC-Link IE header	14	
2	Transient1 header	16		16	
3	Transient1 data area	20 to 1466	Extension header	20	
			Data	0 to 1446	
4	DCS	4		4	Data Check Sequence[Note]
5	FCS	4		4	Frame Check Sequence[Note]

Note Automatically calculated and added by R-IN32M3-CL

(1) Transient frame common header

The transient frame common header is a header used in common by the Transient1 frame, TransientAck frame, and Transient2 frame. The header comprises a MAC header, which is an Ethernet frame header, and a CC-Link IE header.

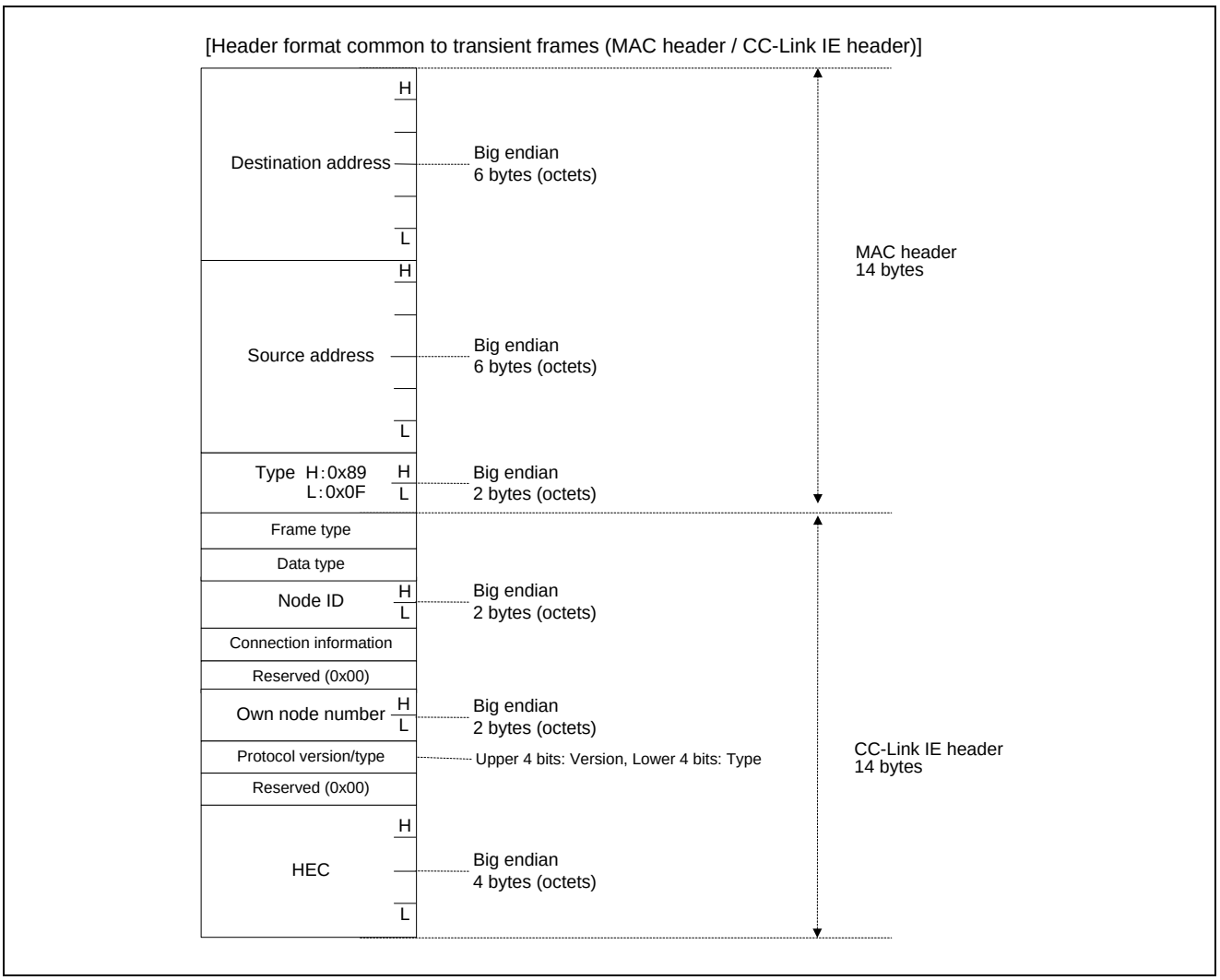


Figure 3.4 Transient Frame Common Header

Table 3.3 MAC Header Items

Item	Description	Value	Remarks
Destination address (octets 1 and 2)	Destination MAC address	0x0000 to 0xFFFF	0x0123 when the MAC address is "01:23:45:67:89:AB."
Destination address (octets 3 and 4)	Destination MAC address	0x0000 to 0xFFFF	0x4567 when the MAC address is "01:23:45:67:89:AB."
Destination address (octets 5 and 6)	Destination MAC address	0x0000 to 0xFFFF	0x89AB when the MAC address is "01:23:45:67:89:AB."
Source address (octets 1 and 2)	Source MAC address	0x0000 to 0xFFFF	0xF02E when the MAC address is "F0:2E:15:6C:77:9B."
Source address (octets 3 and 4)	Source MAC address	0x0000 to 0xFFFF	0x156C when the MAC address is "F0:2E:15:6C:77:9B."
Source address (octets 5 and 6)	Source MAC address	0x0000 to 0xFFFF	0x779B when the MAC address is "F0:2E:15:6C:77:9B."
Type	Upper layer packet type	Fixed value: 0x890F	

Note All items in this table are set using big endian.

Table 3.4 CC-Link IE Header Items

Item	Description	Value	Remarks
Frame type	Type of frame	See Table 3.5	
Data type	Type of data		
Node ID	Node identifier 0 to 255	Acquired by function gusR_IN32_GetNodeID.*1	This value is used in the CC-Link IE Field Network system.*3
Connection information	Transient identification information	Acquired by function gerR_IN32_GetSendTransientBuffer.*2	This information is for identifying the transient frame sent during a single token hold.
Reserved	Reserved	Fixed value: 0x00	
Own node number	Own node number	1 to 120	Note3
Protocol version	Protocol version	Fixed value: 0x00	
Protocol type	Protocol type	CC-Link IE Field Network: 0x1	
HEC	Header Error Control	Automatically calculated by R-IN32M3-CL.	

Note 1. See Section(2) of Section 4.5.11 "Transient transmission processing".
2. See Section(5) of Section 4.5.11 "Transient transmission processing".
3. Set using big endian.

Table 3.5 Frame Type and Data Type List

Frame Type	Value	Data Type	Value
Transient1 frame	0x22	Transient communication specific to CC-Link IE Field Network	0x07
TransientAck frame	0x23		
Transient2 frame	0x25	CC-Link compatible transient communication	0x04

(2) Transient1 header

Transient1 header is a header added to a Transient1 frame.

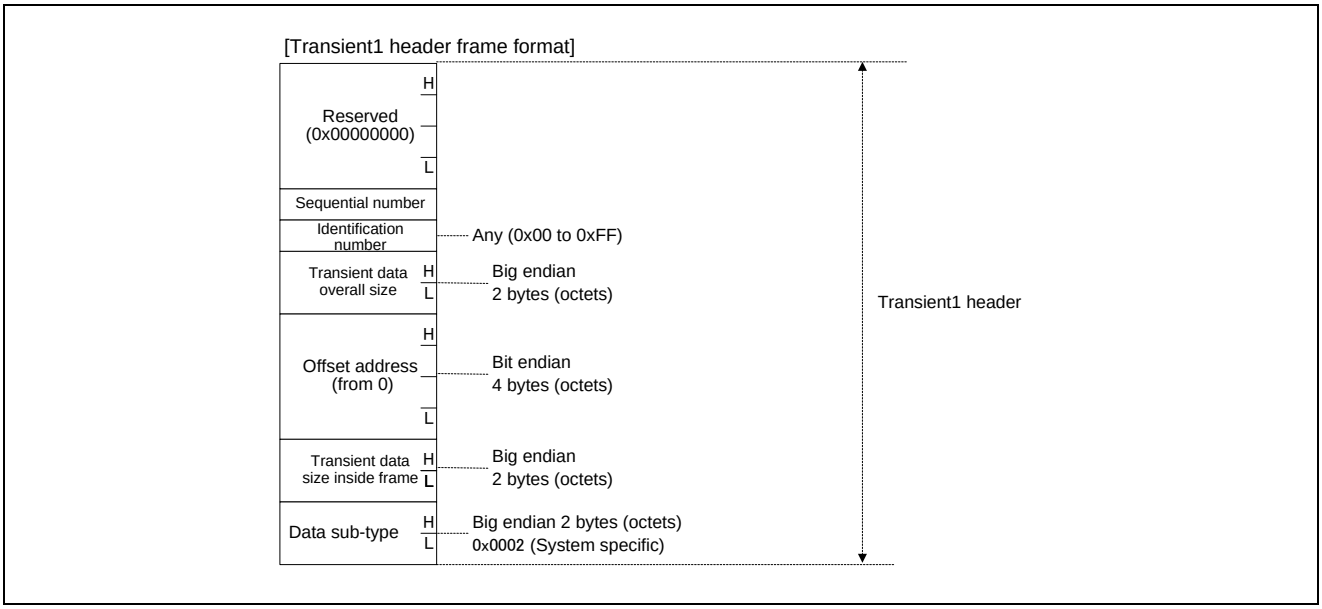


Figure 3.5 Transient1 Header

Table 3.6 Transient1 Header Items

Item	Description	Value	Remarks
Reserved	Reserved	Fixed value: 0x00000000	
Sequential number	Sequential number of Transient1 frame	Bit 7:Final frame (1b), non-final frame (0b) Bits 6 to 0:0x00 to 0x7F	
Identification number	Identification number of transient data	0x00 to 0xFF	Set the same identification number for divided frames.Note
Transient data overall size	Number of bytes from command to data	20 and up	The overall size of the transient data prior to division*1
Offset address	Offset address from start (command) of transient data in general	0 and up	Fixed to 0 when not dividedNote
Transient data size inside frame	Size of Transient1 data area	12 to 1466	Size of transient data after divisionNote
Data sub-type	–	System specific: 0x0002	Note

Note Set in big endian format.

The following shows the relationship between the sequential number and identification number of transient data.

The examples shows the first transient data not divided, the second transient data divided into three sections, and the third transient data divided in half.

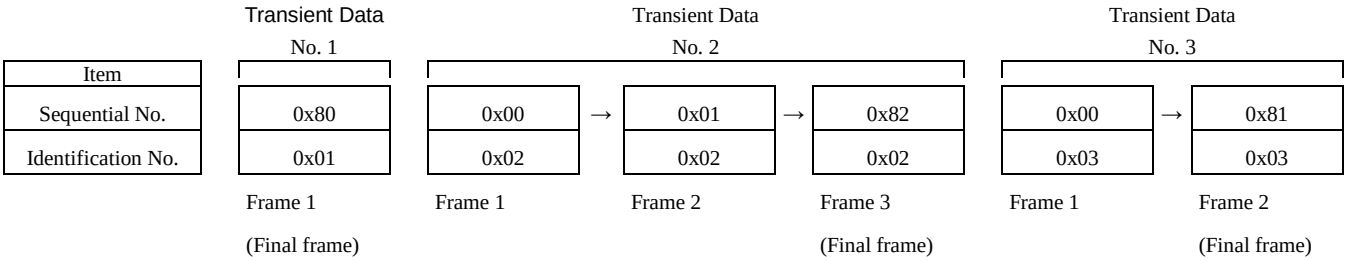


Figure 3.6 Transient1 Header: Relationship between Sequential No. and Identification No. of Transient Data

(3) Transient1 data area

The Transient1 data area is a data area for Transient1 frames, and comprises an extension header and data.

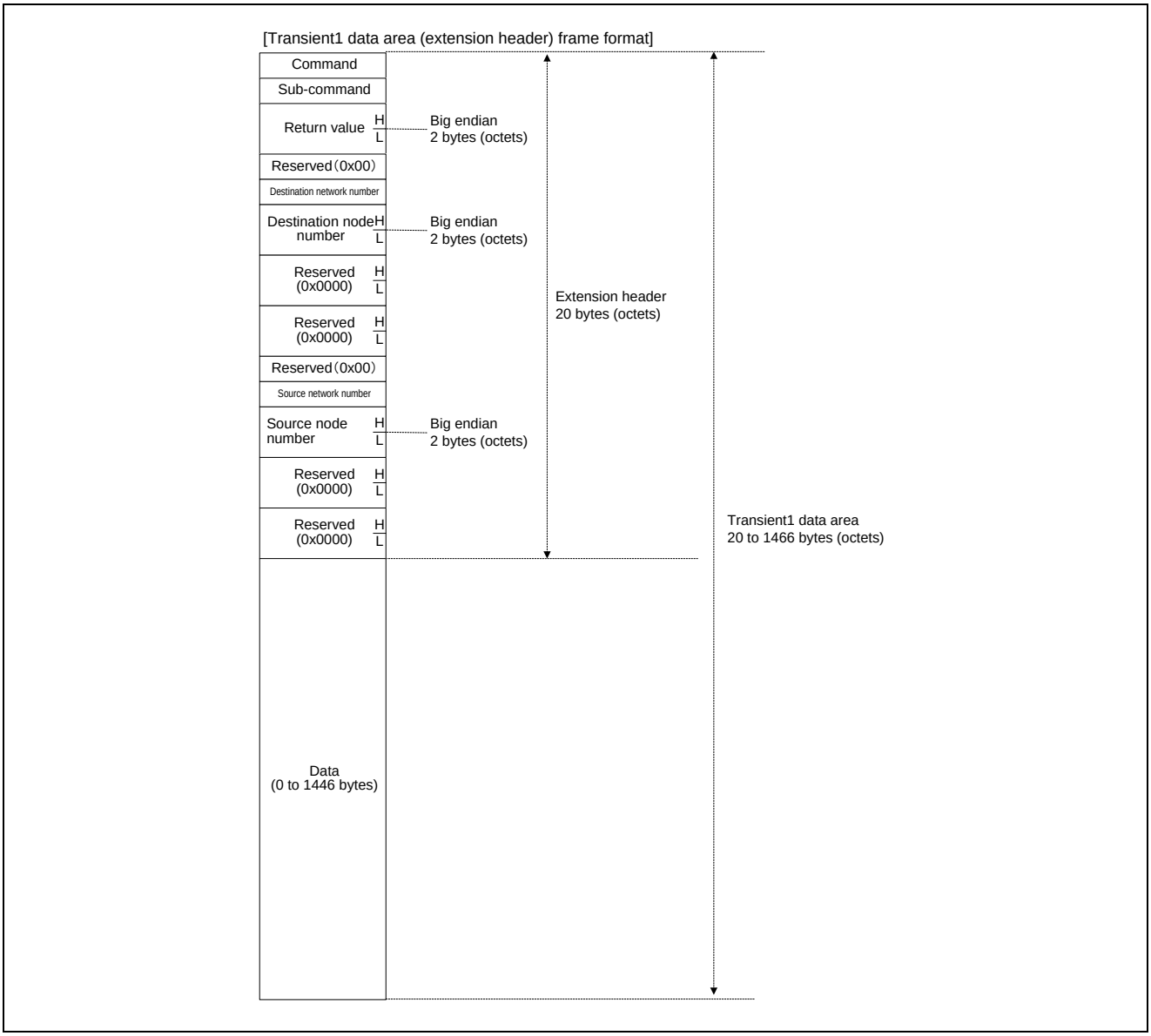


Figure 3.7 Transient1 Data Area

Table 3.7 Extension Header Items

Item	Description	Value	Remarks
Command	Command	See Table 3.8	
Sub-command	Sub-command		
Return value	Return value in response to request	Request: 0x0000 (Fixed) Response:0x0000 (Normal) 0x0001 to 0xFFFF (Abnormal)	Note
Reserved	Reserved	Fixed value: 0	
Destination network number	Destination network number	Broadcast: 0 Destination network: 1 to 239	
Destination node number	Destination node number	1 to 120 Master station: 0x007D Broadcast: 0xFFFF	Note
Reserved	Reserved	Fixed value:0	
Reserved	Reserved	Fixed value:0	
Reserved	Reserved	Fixed value:0	
Source network number	Network number of transmission source	1 to 239	
Source node number	Node number of transmission source	1 to 120	Note
Reserved	Reserved	Fixed value:0	
Reserved	Reserved	Fixed value:0	

Note Set in big endian format.

Table 3.8 Transient1 Command List

Command	Sub-Command	Command Type	Transmission Direction	Remarks
0x01	0x00	Distribute Node Information request	Master station → Slave station	Response not required
0x03	0x00	Get Statistical Information request	Master station → Slave station	
0x03	0x80	Get Statistical Information response	Master station ← Slave station	
0x04	0x00	Get Detailed Node Information request	Master station → Slave station	
0x04	0x80	Get Detailed Node Information response	Master station ← Slave station	

(a) Distribute node information

The Distribute Node Information command is used by the intelligent device station to find a destination MAC address from a destination node number.

The intelligent device station sends a TransientAck frame in response to a Distribute Node Information request frame received from the master station. Transmission of a Transient1 response frame (response to a Distribute Node Information request) is not required.

If the number of sets of distributed node information is 60 or more, the frame is divided into two frames since the frame size exceeds 1518 bytes, which is the maximum number of an Ethernet frame. In such a case, the Transient1 reception data needs to be assembled by firmware.

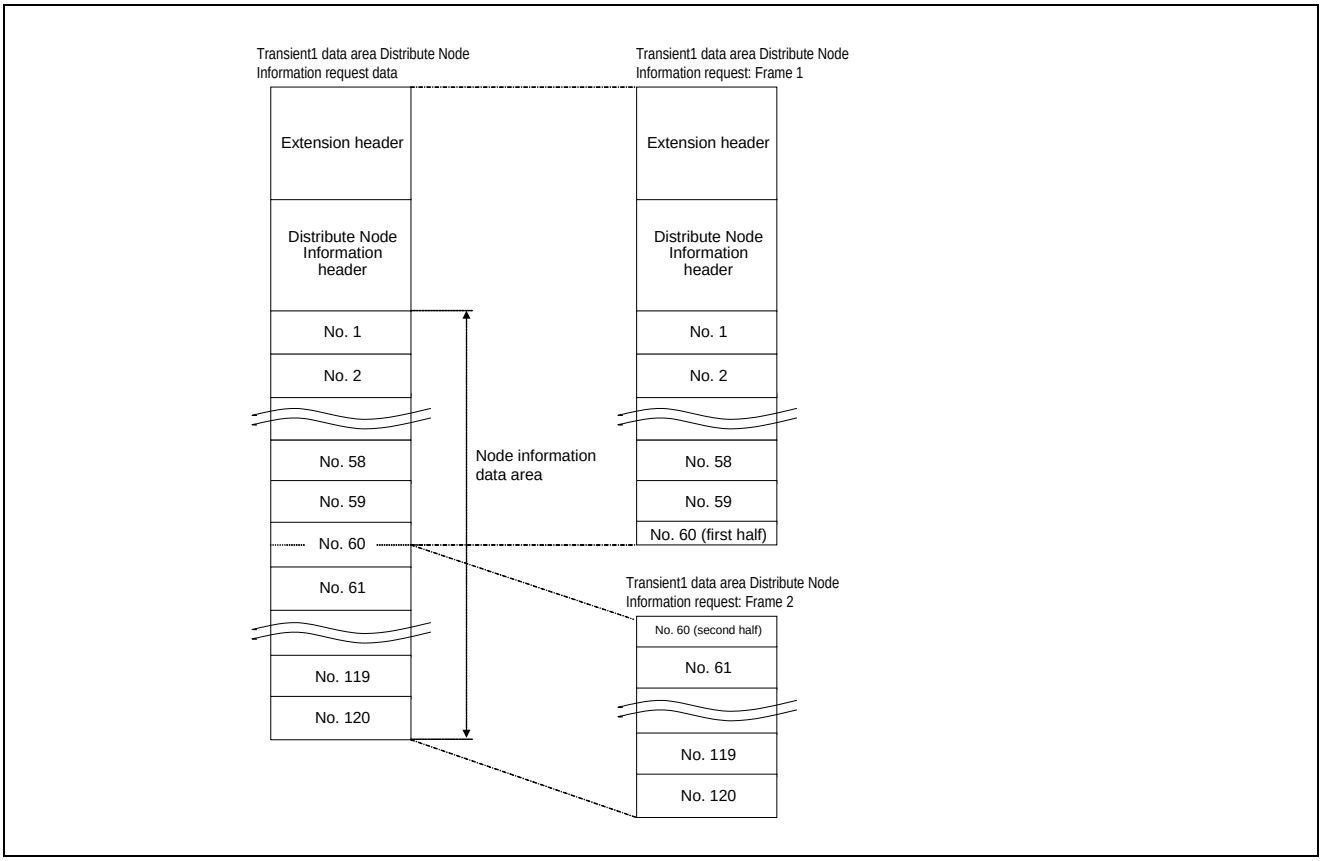


Figure 3.8 Transient1 Data Area: Frames When Distribute Node Information Request Is Divided

For the frame format, see Table 3.9, Table 3.10, and Table 3.11 in accordance with the table below.

Table 3.9 Frame Format for Distribute Node Information Request

Number of Distributions	Reference
Less than 60	“Figure 3.9 Transient1 Data Area: Distribute Node Information Request”
60 or more	“Figure 3.10 Transient1 Data Area: Distribute Node Information Request - Frame 1” “Figure 3.11 Transient1 Data Area: Distribute Node Information Request - Frame 2”

For details on assembling the Transient1 reception data, see Section 4.2.18 "Start Making Received Transient1 Data processing" and Section 4.2.19 "Make Received Transient1 Data processing."

The following shows the frame format for a Distribute Node Information request involving less than 60 distributions.

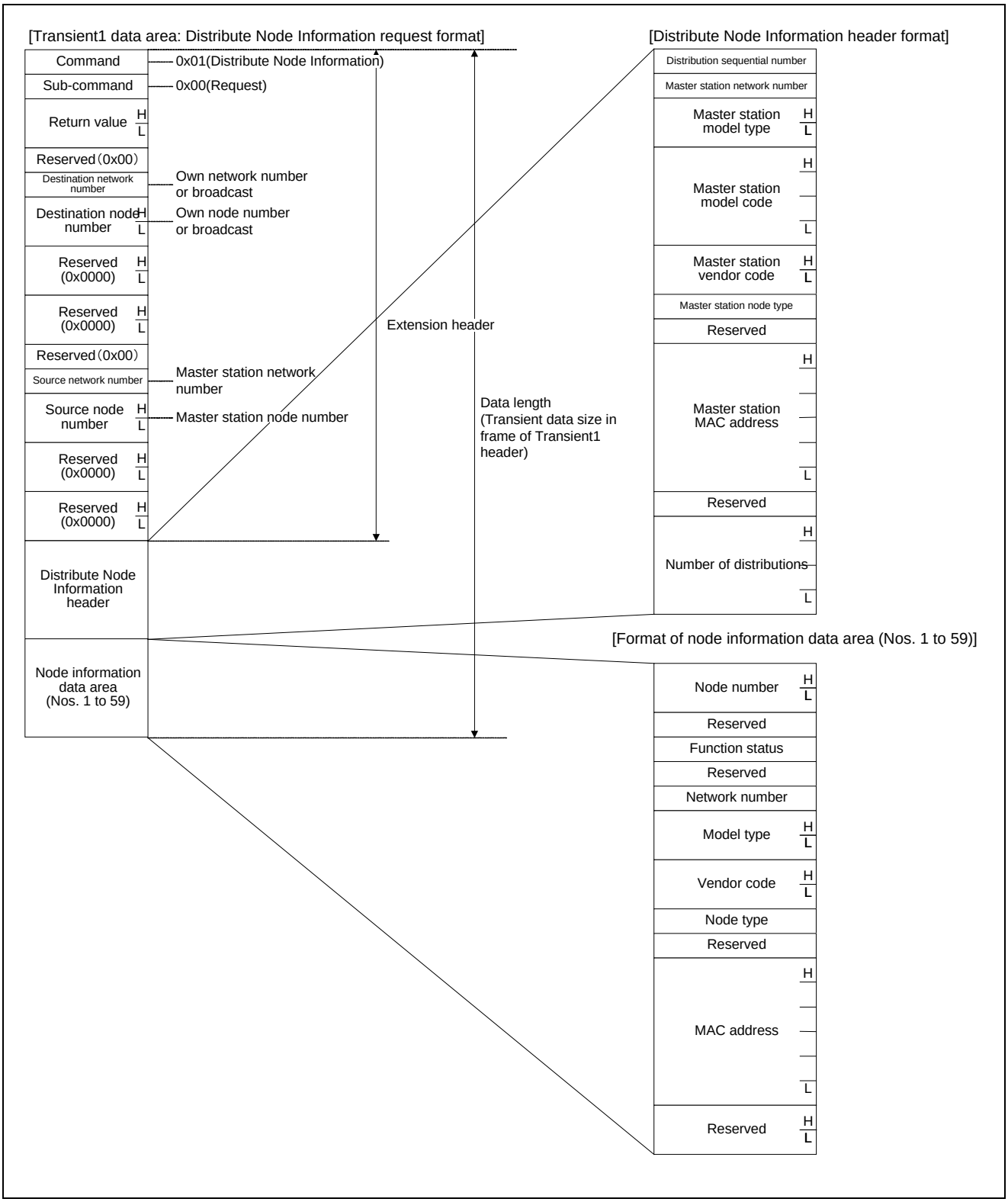


Figure 3.9 Transient1 Data Area: Distribute Node Information Request

The following shows the frame format (frame 1) for a Distribute Node Information request involving 60 or more distributions.

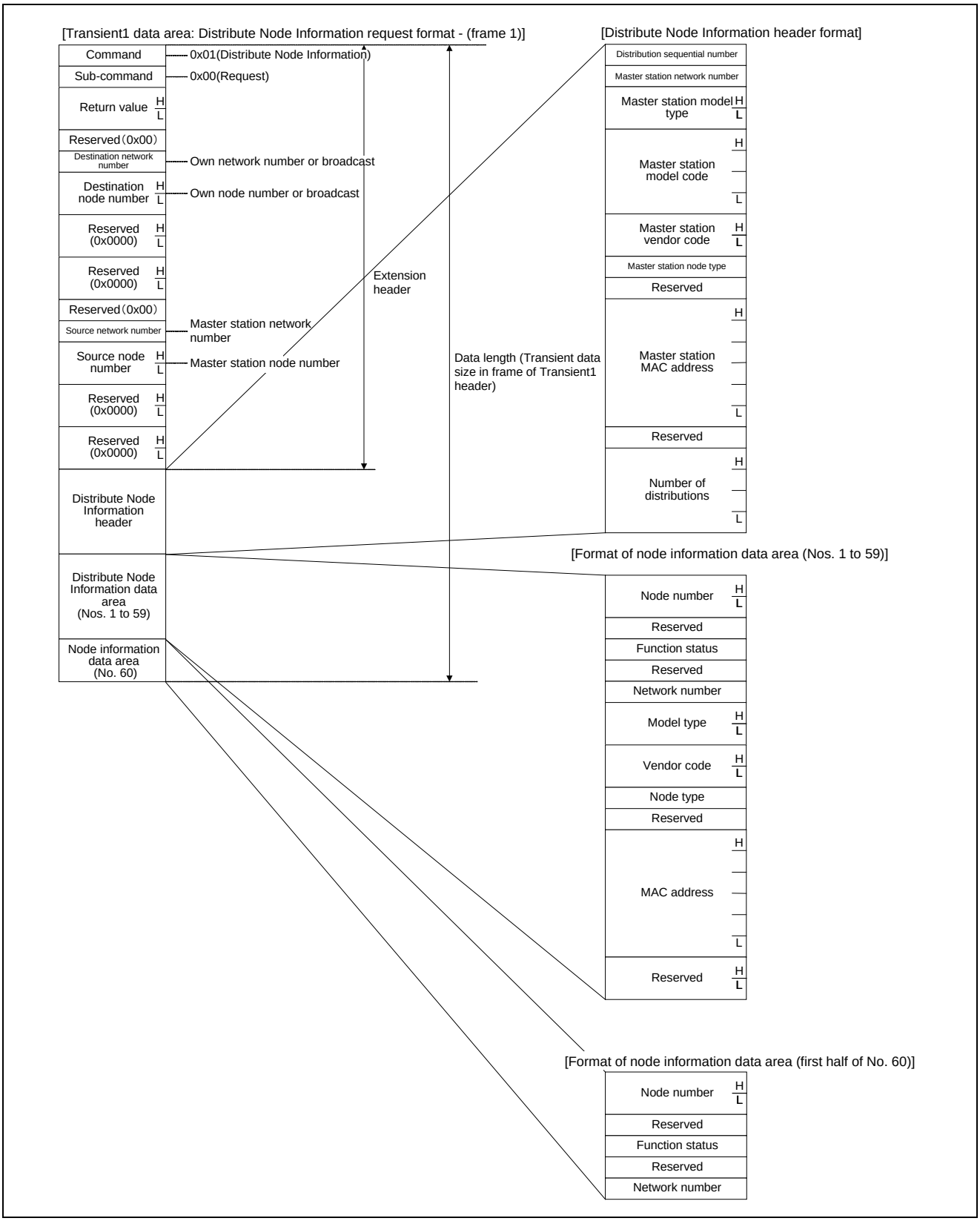


Figure 3.10 Transient1 Data Area: Distribute Node Information Request - Frame 1

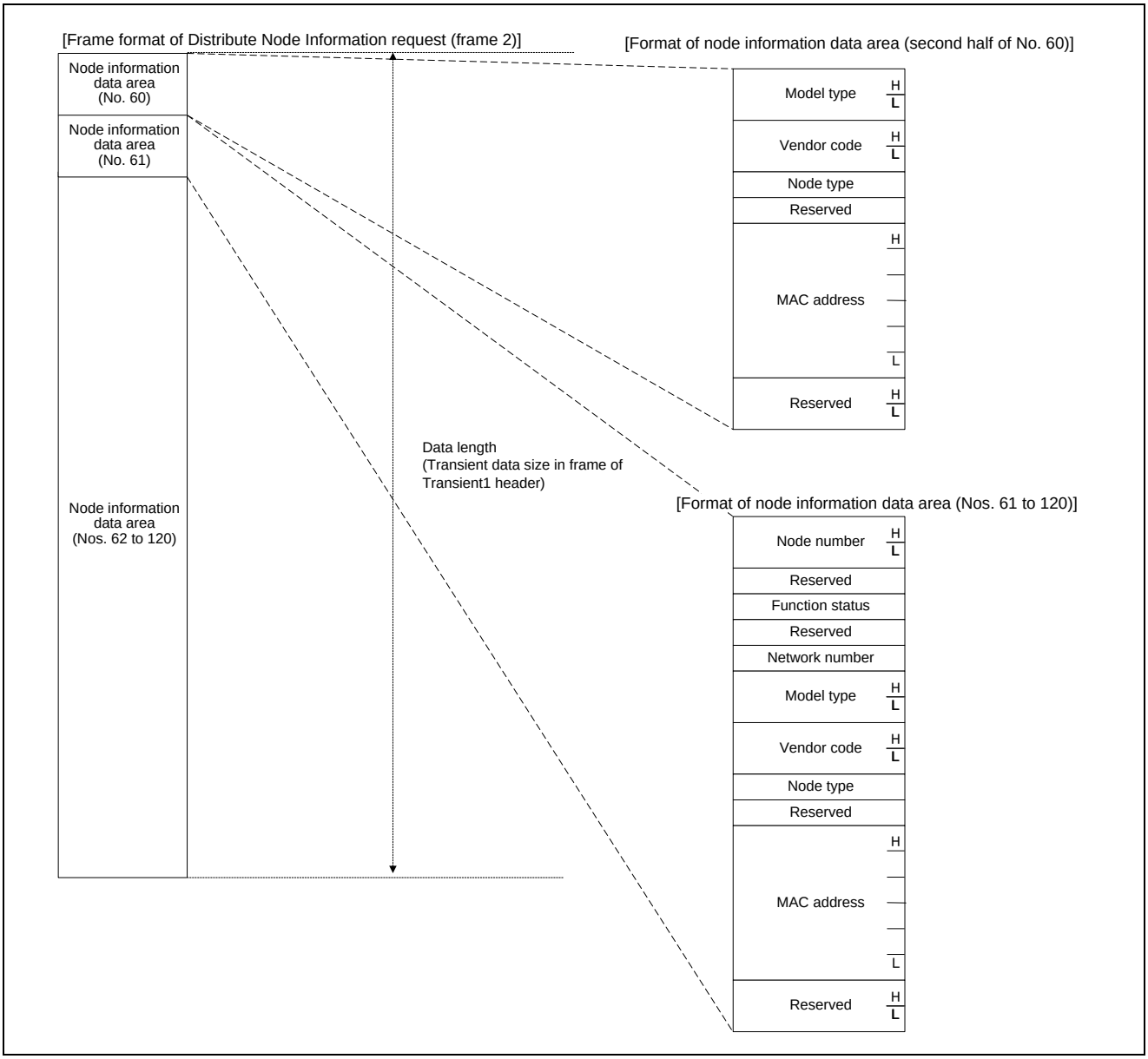


Figure 3.11 Transient1 Data Area: Distribute Node Information Request - Frame 2

Table 3.10 Distribute Node Information Header Items

Item	Description	Value	Remarks
Distribution sequential number	Distribution sequential number	1 to 7	When the distribution sequential numbers are the same, the node information is the same. Discard it.
Master station network number	Network number of master station	1 to 239	
Master station model type	Model type of master station	0x0001 to 0xFFFF	Model type managed by CC-Link Partner Association
Master station model code	Model code of master station	0x00000000 to 0xFFFFFFFF	Model code of network that is unique within the same vendor code
Master station vendor code	Vendor code of master station	0x0000 to 0xFFFF	Vendor code (vendor ID) managed by CC-Link Partner Association
Master station node type	Node type of master station	See Table 3.12.	
Reserved	Reserved	Fixed value: 0	
Master station MAC address	MAC address of master station	6-byte MAC address	
Reserved	Reserved	Fixed value: 0	
No. of distributions	No. of distributions of node information	1 to 120	

Table 3.11 Node Information Data Area Items

Item	Description	Value	Remarks
Node number	Node number	1 to 120	
Reserved	Reserved	Fixed value: 0	
Function status	Transient reception function status	Yes: 0x01 No: 0x00	
Reserved	Reserved	Fixed value: 0	
Network number	Network number of node	1 to 239	
Model type	Model type of node	0x0001to 0xFFFF	Model type managed by CC-Link Partner Association
Model code	Model code of node	0x00000000 to 0xFFFFFFFF	Model code of network that is unique within the same vendor code
Vendor code	Vendor code	0x0000 to 0xFFFF	Vendor code (vendor ID) managed by CC-Link Partner Association
Node type	Node type	See Table 3.12.	
Reserved	Reserved	Fixed value: 0	
MAC address	MAC address	6-byte MAC address	
Reserved	Reserved	Fixed value: 0	

Table 3.12 Node Type List

Node Type	Description	Remarks
0x30	Master station	−
0x31	Reserved	−
0x32	Local station	−
0x33	Intelligent device station	−
0x34	Remote device station	−
0x35	Remote I/O station	−

(b) Get statistical information

The Get Statistical Information command is used by the master station to collect the statistical information of a slave station. The master station sends the request to a slave station, and the slave station sends a response to the master station.

The intelligent device station (slave station) sends a TransientAck frame and a Transient1 response frame (Get Statistical Information response) in response to the Transient1 request frame (Get Statistical Information request) sent by the master station.

In response to the response frame, the master station sends a TransientAck frame. Reception processing is therefore required.

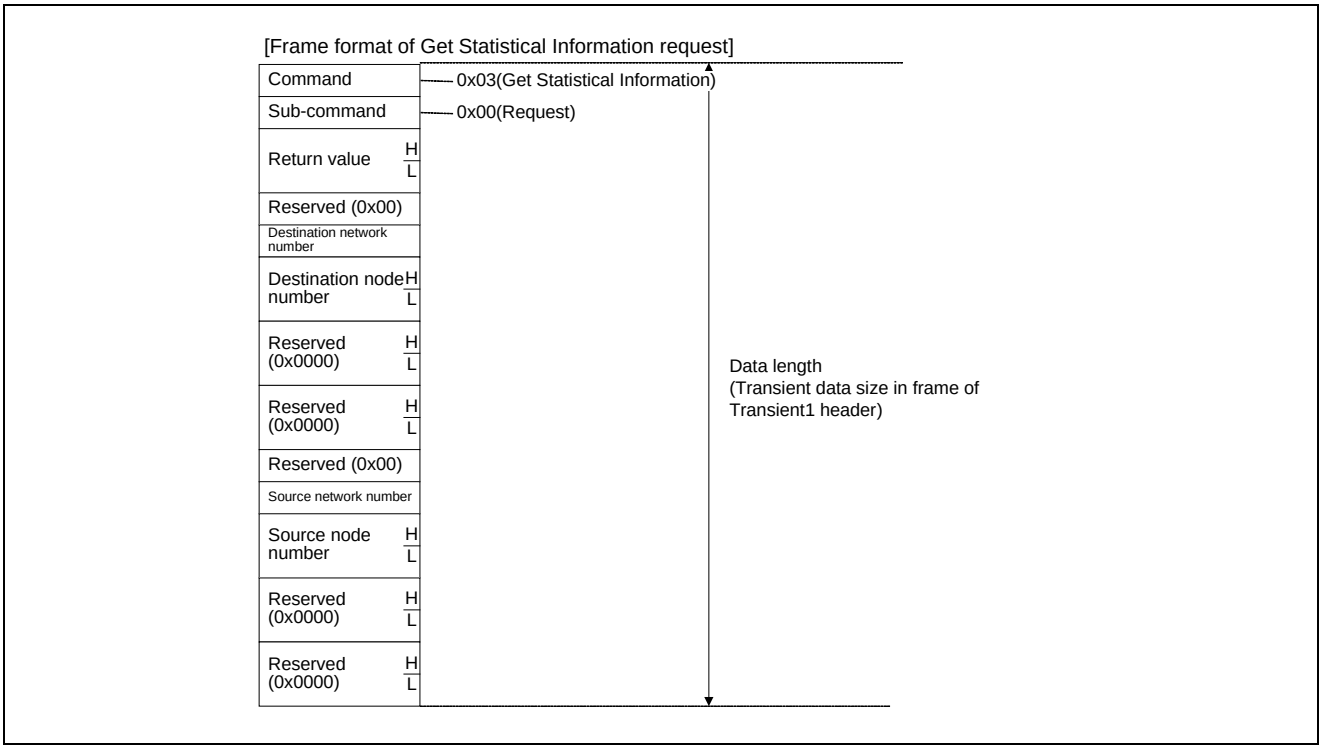


Figure 3.12 Transient1 Data Area: Get Statistical Information Request

For each item in the figure above, see “Table 3.7 Extension Header Items” in Section (3) “Transient1 data area” in Section 3.3.1 “Transient1 frame format."

The following shows the frame format of the Get Statistical Information response.

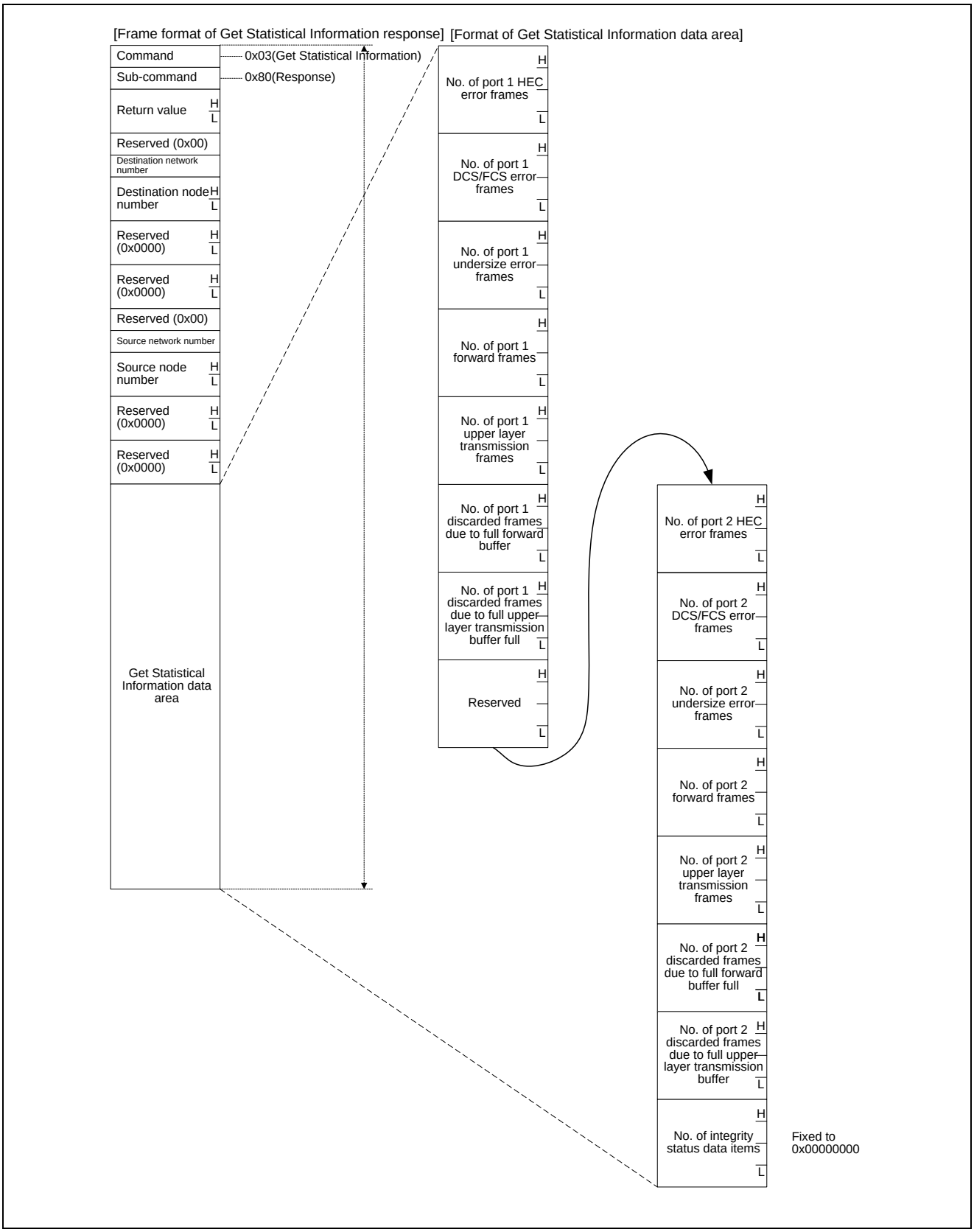


Figure 3.13 Transient1 Data Area: Get Statistical Information Response

Each of the items of the Get Statistical Information data area shown in the table below is acquired by the function gerR_IN32_GetMIB. For item details, see Section (6) of Section 4.5.6 "Host station status acquisition."

Table 3.13 Get Statistical Information Data Items

Item	Description	Value	Remarks
No. of port 1 HEC error frames	No. of HEC error frames of port 1	0 to 4294967295	Counts the number of HEC errors in received frames.
No. of port 1 DCS/FCS error frames	No. of DCS/FCS error frames of port 1	0 to 4294967295	Counts the number of DCS/FCS errors in received frames.
No. of port 1 undersize error frames	No. of undersize error frames of port 1	0 to 4294967295	Counts the number of cases in which the received frame size is less than 28 bytes.
No. of port 1 forward frames	No. of forward frames of port 1	0 to 4294967295	Counts the number of forwarded frames.
No. of port 1 upper layer transmission frames	No. of upper layer transmission frames of port 1	0 to 4294967295	Counts the number of frames transmitted to the upper layer.
No. of port 1 discarded frames due to full forward buffer	No. of port 1 discarded frames due to full forward buffer	0 to 4294967295	Counts the number of frames discarded due to a full forward buffer.
No. of port 1 discarded frames due to full upper layer transmission buffer	No. of port 1 discarded frames due to full upper layer transmission buffer	0 to 4294967295	Counts the number of frames discarded due to a full upper layer transmission buffer.
Reserved	Reserved	Fixed value: 0	
No. of port 2 HEC error frames	No. of HEC error frames of port 2	0 to 4294967295	Counts the number of HEC errors in received frames.
No. of port 2 DCS/FCS error frames	No. of DCS/FCS error frames of port 2	0 to 4294967295	Counts the number of DCS/FCS errors in received frames.
No. of port 2 undersize error frames	No. of undersize error frames of port 2	0 to 4294967295	Counts the number of cases in which the received frame size is less than 28 bytes.
No. of port 2 forward frames	No. of forwad frames of port 2	0 to 4294967295	Counts the number of forwarded frames.
No. of port 2 upper layer transmission frames	No. of upper layer transmission frames of port 2	0 to 4294967295	Counts the number of frames transmitted to the upper layer.
No. of port 2 discarded frames due to full forward buffer	No. of port 2 discarded frames due to full forward buffer	0 to 4294967295	Counts the number of frames discarded due to a full forward buffer.
No. of port 2 discarded frames due to full upper layer transmission buffer	No. of port 2 discarded frames due to full upper layer transmission buffer	0 to 4294967295	Counts the number of frames discarded due to a full upper layer transmission buffer.
No. of integrity status data items	No. of integrity status data items	Fixed value: 0x00000000	

(c) Get Detailed Node Information

The Get Detailed Node Information command is used by the master station to collect the detailed node information of a slave station. The master station sends the request to a slave station, and the slave station sends a response to the master station.

The intelligent device station (slave station) sends a TransientAck frame and Transient1 response frame (Get Detailed Node Information response) in response to a Transient1 request frame (Get Detailed Node Information request) sent by the master station.

In response to the response frame, the master station sends a TransientAck frame. Reception processing is therefore required.

The following shows the frame format of a Get Detailed Node Information request.

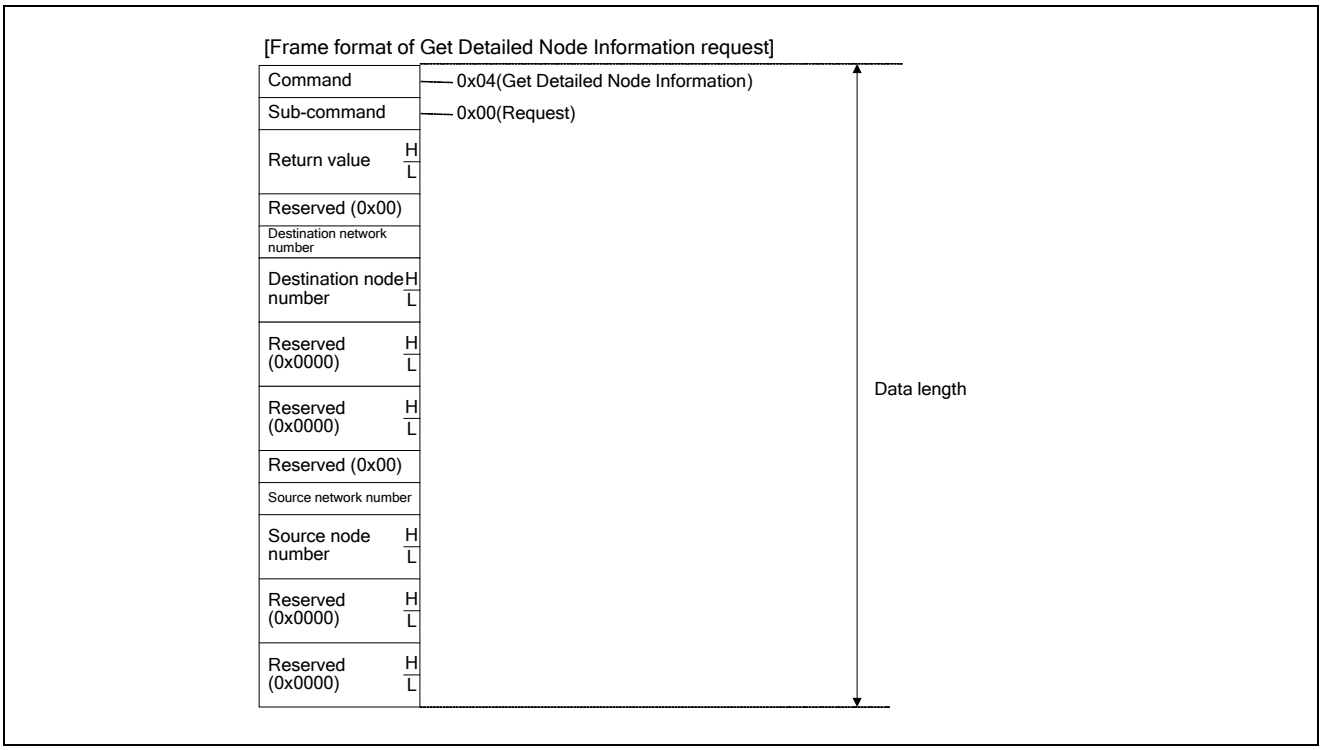


Figure 3.14 Transient1 Data Area: Get Detailed Node Information Request

For the details of each item in the figure above, see "Table 3.7Extension Header Items" in Section (3) "Transient1 data area" in Section 3.3.1 "Transient1 frame format."

The following shows the frame format of the Get Detailed Node Information response.

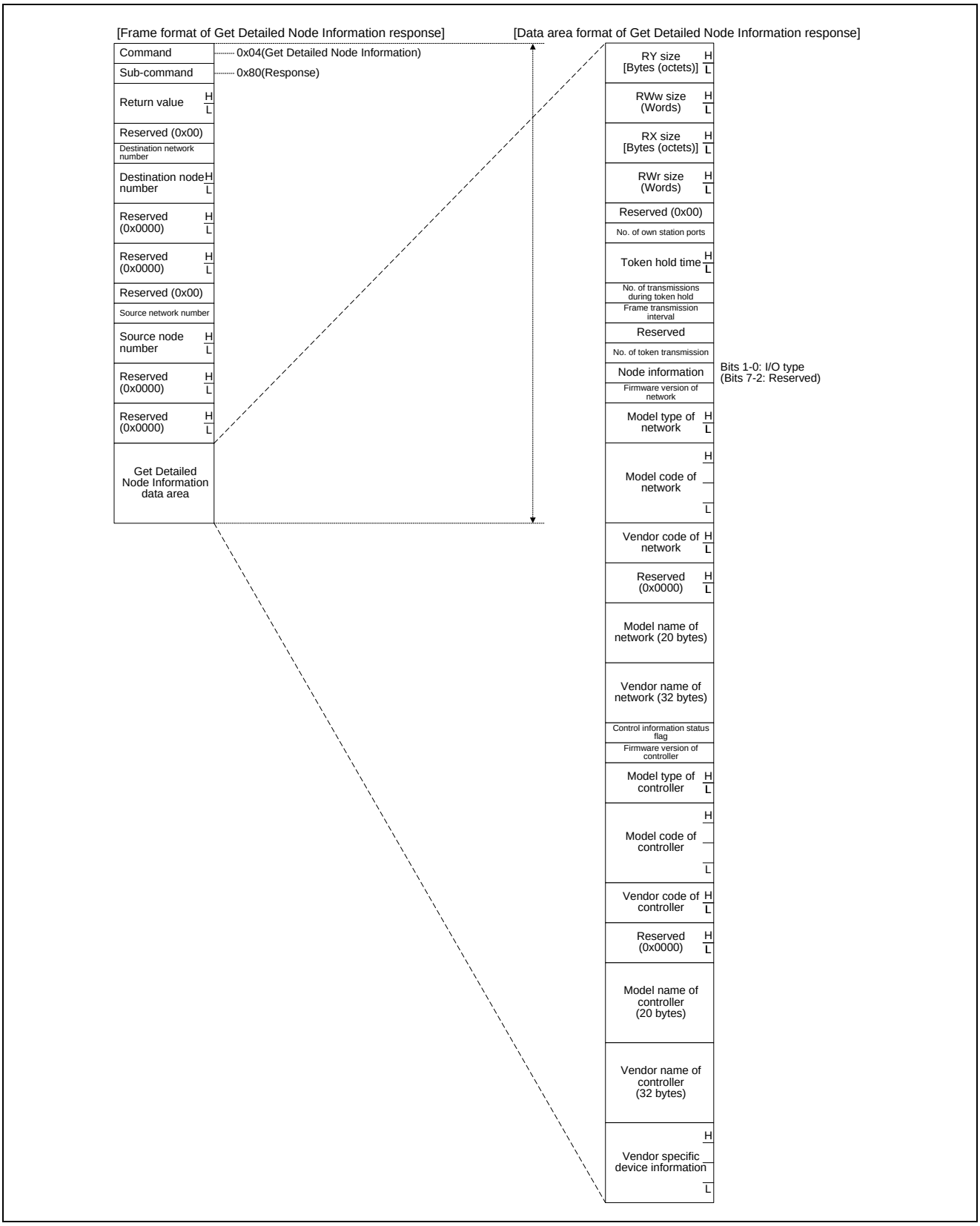


Figure 3.15 Transient1 Data Area: Get Detailed Node Information Response

Table 3.14 Data Area Items of Get Detailed Node Information Response (1/2)

Item	Description	Value	Remarks
RY size (Bytes (octets))	RY size of host station	Minimum value: 0 Maximum value: 256	"RY size" acquired by the function gerR_IN32_GetUnitInformation.Note
RWw size (Words)	RWw size of host station	Minimum value: 0 Maximum value: 1024	"RWw size" acquired by the function gerR_IN32_GetUnitInformation.Note
RX size (Bytes (octets))	RX size of host station	Minimum value: 0 Maximum value: 256	"RX size" acquired by the function gerR_IN32_GetUnitInformation.Note
RWr size (Words)	RWr size of host station	Minimum value: 0 Maximum value: 1024	"RWr size" acquired by the function gerR_IN32_GetUnitInformation.Note
Reserved	Reserved	Fixed value: 0	
No. of host station ports	No. of ports of host station	Fixed value: 2	"No. of ports of host station" acquired by the function gerR_IN32_GetUnitInformation.Note
Token hold time	Maximum value (µs) of token hold time of host station	1 to 32767	"Token hold time" acquired by the function gerR_IN32_GetUnitInformation.Note
No. of transmissions during token hold	Number of frame transmissions other than token frame transmission during token hold	1 to 255	"No. of transmissions during token hold" acquired by the function gerR_IN32_GetUnitInformation.Note
Frame transmission interval	Frame interval after token frame reception to MyStatus frame transmission	1 to 255	"Frame transmission interval" acquired by the function gerR_IN32_GetUnitInformation.Note
Reserved	Reserved	Fixed value: 0	
No. of token transmission	No. of repeated transmissions of token frame transmitted during token hold	1 to 255	"No. of token transmissions" acquired by the function gerR_IN32_GetUnitInformation.Note
Node information (I/O type)	I/O type	Front/Back mixed: 0x00 Input: 0x01 Output: 0x02 Mixed: 0x03	"Node information (I/O type)" acquired by the function gerR_IN32_GetUnitInformation.Note
Firmware version of network	Firmware version of network	0 to 255	"Firmware version of network" acquired by the function gerR_IN32_GetUnitInformation.Note
Model type of network	Model type of network	0x0001 to 0xFFFF	"Model type of network" acquired by the function gerR_IN32_GetUnitInformation.Note Model type managed by CC-Link Partner Association.
Model code of network	Model code of network	0x00000000 to 0xFFFFFFFF	"Model code of network" acquired by the function gerR_IN32_GetUnitInformation.Note Model code of network that is unique within the same vendor code.

Note See Section(2) of Section 4.5.1 “Initial setup”

Table 3.14 Data Area Items of Get Detailed Node Information Response (2/2)

Item	Description	Value	Remarks
Vendor code of network	Vendor code of network	0x0000 to 0xFFFF	"Vendor code of network" acquired by the function gerR_IN32_GetUnitInformation.Note Vendor code (vendor ID) managed by CC-Link Partner Association.
Reserved	Reserved	Fixed value: 0	
Model name of network	Model name of network	Model name (20 bytes)	"Model name of network" acquired by the function gerR_IN32_GetUnitInformation.Note
Vendor name of network	Vendor name of network	Vendor name (32 bytes)	"Vendor name of network" acquired by the function gerR_IN32_GetUnitInformation.Note
Controller information status flag	Controller information (from “Firmware version of controller” to “Vendor specific device information of controller”) status flag	Disable: 0 Enable: 1	"Controller information status flag" acquired by the function gerR_IN32_GetUnitInformation.Note
Firmware version of controller	Firmware version of controller	0 to 255	"Firmware version of controller" acquired by the function gerR_IN32_GetUnitInformation.Note
Model type of controller	Model type of controller	0x0001 to 0xFFFF	"Model type of controller" acquired by the function gerR_IN32_GetUnitInformation.Note Model type managed by CC-Link Partner Association.
Model code of controller	Model code of controller	0x00000000 to 0xFFFFFFFF	"Model code of controller" acquired by the function gerR_IN32_GetUnitInformation.Note Model code of controller that is unique within the same vendor code.
Vendor code of controller	Vendor code of controller	0x0000 to 0xFFFF	"Vendor code of controller" acquired by the function gerR_IN32_GetUnitInformation.Note Vendor code (vendor ID) managed by CC-Link Partner Association.
Reserved	Reserved	Fixed value: 0	
Model name of controller	Model name of controller	Model name (20 bytes)	"Model name of controller" acquired by the function gerR_IN32_GetUnitInformation.Note
Vendor name of controller	Vendor name of controller	Vendor name (32 bytes)	"Vendor name of controller" acquired by the function gerR_IN32_GetUnitInformation.Note
Vendor specific device information of controller	Vendor specific device information of controller	0x00000000 to 0xFFFFFFFF	"Vendor specific device information of controller" acquired by the function gerR_IN32_GetUnitInformation.Note

Note See Section(2) of Section 4.5.1 “Initial setup”

3.3.2 TransientAck frame format

The following table provides an overview of the format of a TransientAck frame.

Table 3.15 Overview of TransientAck Frame Format

	Item	Size (Bytes)	Sub-Item	Size (Bytes)	Remarks
1	Transient frame common header	28	MAC header	14	
			CC-Link IE header	14	
2	No. of Ack data items	4		4	Fixed value: 0x00000001
3	Ack data area	8		8	
4	Padding	16		16	Automatically added by R-IN32M3-CL.
5	DCS	4		4	Data Check Sequence*1
6	FCS	4		4	Frame Check Sequence*1

Note Automatically calculated and added by R-IN32M3-CL

(1) Transient frame common header

See Section (1) "Transient frame common header" of Section 3.3.1 "Transient1 frame format."

(2) TransientAck data area

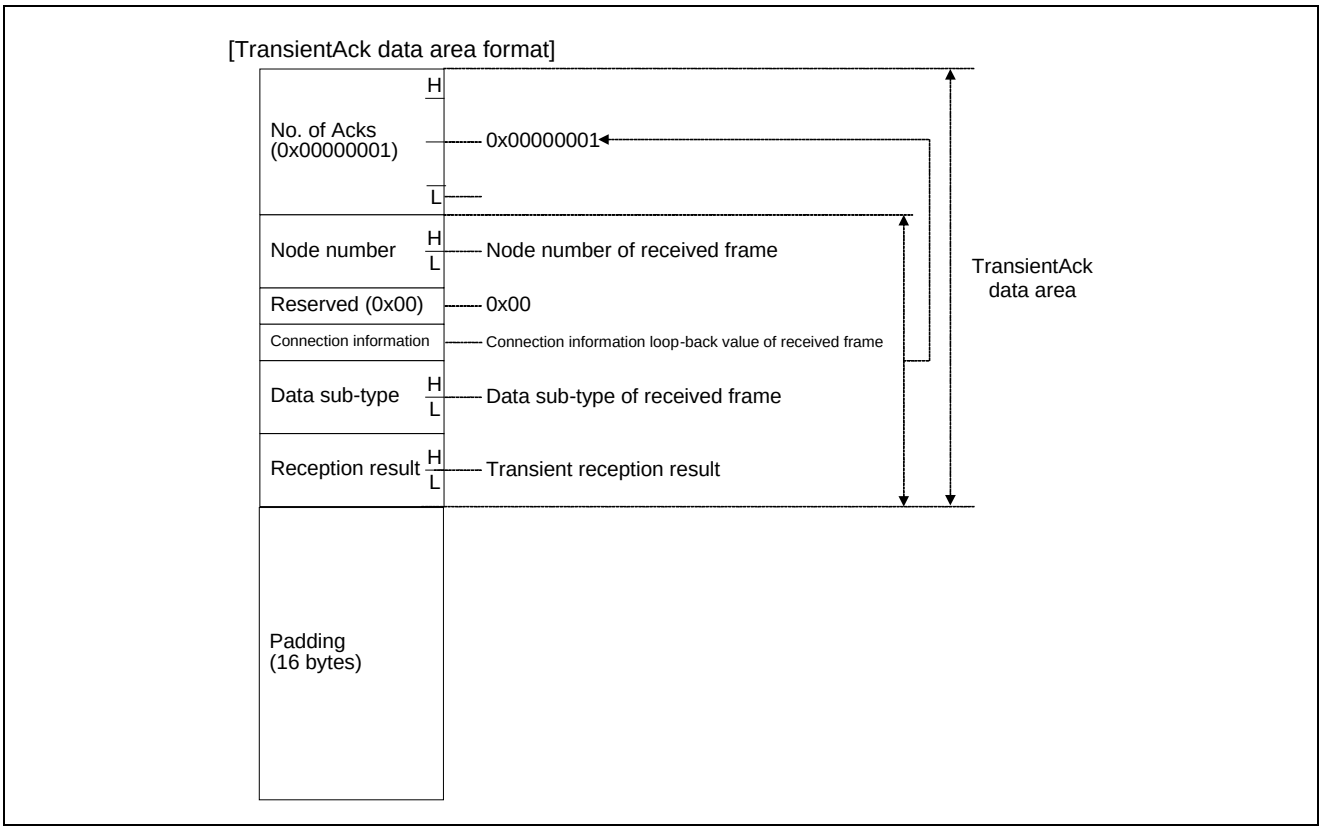


Figure 3.16 TransientAck Data Area

Table 3.16 TransientAck Data Items

Item	Description	Value	Remarks
No. of Acks	No. of Acks from node number to reception result	Fixed value: 0x00000001	
Node number	Node number of TransientAck frame transmission destination	Source node number of received Transient1 or Transient2	
Reserved	Reserved	Fixed value: 0x00	
Connection information	Connection information loopback value (Connection Information) of Ack transmission target frame	Connection information of received Transient1 or Transient2	
Data sub-type	Data sub-type of received Transient1 frame	Transient1: 0x0002 Transient2: 0x0000	Transient2 does not require specification of a data sub-type. Fixed value: 0x0000.
Reception result	Reception result (RET) of Transient1 frame or Transient2 frame	Normal: 0x0000 Abnormal: Other than 0x0000	
Padding	Padding (16 bytes)	–	Automatically padded by R-IN32M3-CL to satisfy the minimum Ethernet frame size of 64 bytes.

3.3.3 Transient2 frame format

The following table provides an overview of the format of a Transient2 frame.

Table 3.17 Overview of Transient2 Frame Format

	Item	Size (Bytes)	Sub-Item	Size (Bytes)	Remarks
1	Transient frame common header	28	MAC header	14	
			CC-Link IE header	14	
2	Transient2 header	26	—	26	From L to APS in "Figure 3.17 Transient2 Header."
3	Transient2 data area	0 to 968	—	0 to 968	
4	DCS	4	—	4	Data Check Sequence[Note]
5	FCS	4	—	4	Frame Check Sequence[Note]

Note Automatically calculated and added by R-IN32M3-CL

(1) Transient frame common header

See Section (1) "Transient frame common header" of Section 3.3.1 "Transient1 frame format."

(2) Transient2 header

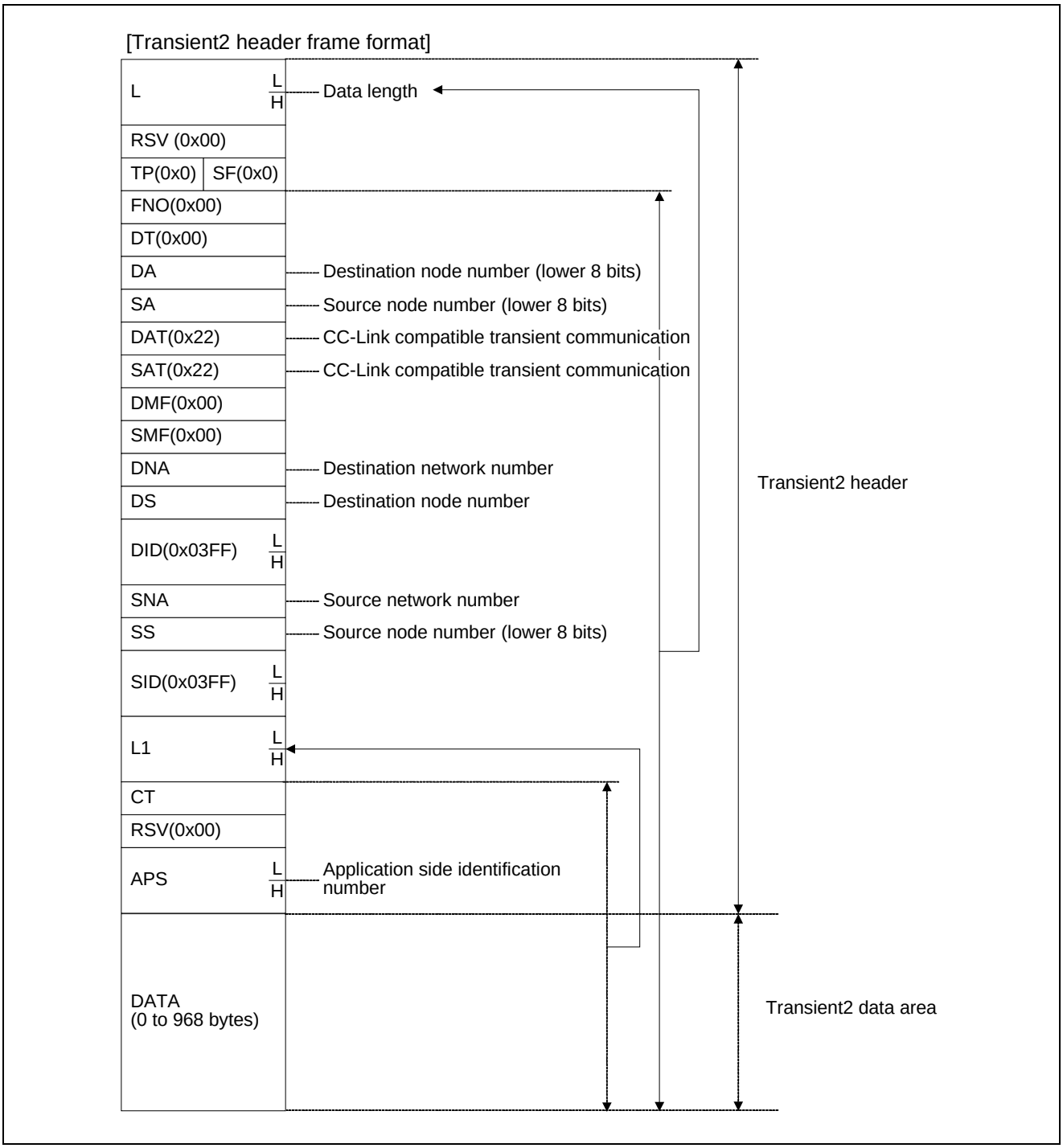


Figure 3.17 Transient2 Header

Table 3.18 Transient2 Header Items

Item	Item Name	Description	Value	Remarks
L	Length	Length (bytes) from FNO to DATA	22 to 990	
RSV	ReSerVe	Reserved	Fixed value: 0x00	
TP/SF	TyPe/SequenceFlag	Not used (type/sequence number)	Fixed value: 0x00	
FNO	Frame Sequence	Not used (divided frame number)	Fixed value: 0x00	
DT	Data frameType	Not used (data frame type)	Fixed value: 0x00	
DA	Destination Address	Destination node number	1 to 120, master station: 0x7D	Same value as DS
SA	Source Address	Source node number	0 to 120	
DAT	Destination Application Type	Application type	Fixed value: 0x22	CC-Link compatible transient communication
SAT	Source Application Type	Application type	Fixed value: 0x22	CC-Link compatible transient communication
DMF	Destination Module Flag	Not used (target destination module flag)	Fixed value: 0x00	
SMF	Source Module Flag	Not used (startup source module flag)	Fixed value: 0x00	
DNA	Destination Network Address	Destination network number	1 to 239	
DS	Destination Station no	Destination node number	1 to 120, master station: 0x7D	Same value as DA
DID	DestinationID	Destination identification number	Fixed value: 0x03FF	
SNA	Source Network Address	Startup source network number	1 to 239	
SS	Source Station no	Startup source node number	1 to 120	
SID	SourceID	Transmission source identification number	Fixed value: 0x03FF	
L1	Length1	Length (bytes) from CT to DATA	4 to 972	
CT	Command Type	Command type	Note	
RSV	ReSerVe	Reserved	Fixed value: 0x00	
APS	Application Sequence	Application number	0 to 255	Startup source identification number. Set any value.

Note See Table 3.19 "Transient2 Command Types"

Table 3.19 Transient2 Command Types

CT	Command type
0x04	Get Memory Access Information request
0x84	Get Memory Access Information response
0x08	RUN request
0x88	RUN response
0x09	STOP request
0x89	STOP response
0x10	Get Memory request
0x90	Get Memory response
0x12	Set Memory request
0x92	Set Memory response
0x60 to 0x7F	Vendor specific

(3) Transient2 data area

(a) Return code

The return code (RSTS) format is common to the Transient2 response frame. 0x0000 indicates normal.

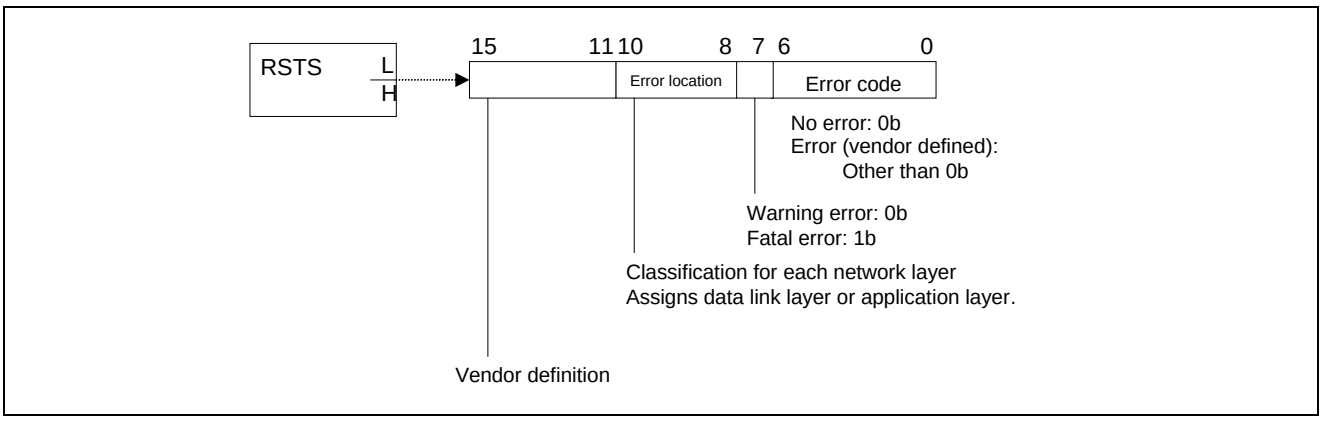


Figure 3.18 Return Code (RSTS)

(b) Get memory access information

・Request

The Get Memory Access Information acquires the memory access information of the master station.

CT=0x04

Data area not required (0 bytes)

・Response

CT=0x84

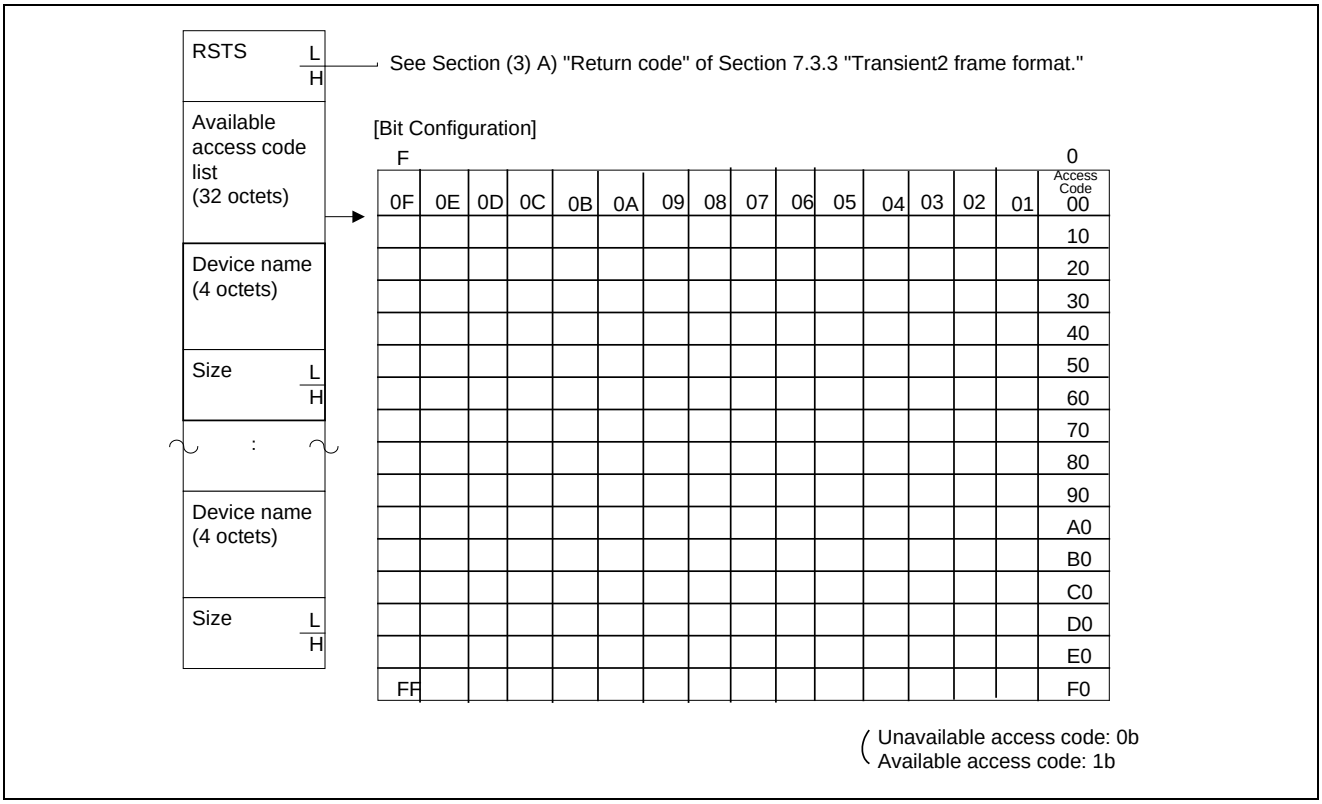


Figure 3.19 Data Area of Get Memory Access Information Response

7	6	5	4	3	2	1	0
Hold	Status	Link	Counter	Timer	Word data	Bit data	
						Output	Input
Type					Function		

Figure 3.20 Access Code

Table 3.20 Access Code List (Example: Mitsubishi Product)

Device Description	Name	Access Code	Device Number Radix	Device Number and Access Range	Type	
					Bit	Word
Input	X	0x01	Hexadecimal	The device number and access range differ for each PLC.	O	
Output	Y	0x02	Hexadecimal		O	
Internal relay	M	0x03	Decimal		O	
Latch relay	L	0x83	Decimal		O	
Link relay	B	0x23	Hexadecimal		O	
Timer (contact)	T	0x09	Decimal		O	
Timer (coil)	T	0x0A	Decimal		O	
Timer (present value)	T	0x0C	Decimal			O
Accumulation timer (contact)	ST	0x89	Decimal		O	
Accumulation timer (coil)	ST	0x8A	Decimal		O	
Accumulation timer (present value)	ST	0x8C	Decimal			O
Counter (contact)	C	0x11	Decimal		O	
Counter (coil)	C	0x12	Decimal		O	
Counter (present value)	C	0x14	Decimal			O
Data register	D	0x04	Decimal			O
Link register	W	0x24	Hexadecimal			O
File register	R	0x84	Decimal			O
Link special relay	SB	0x63	Hexadecimal		O	
Link special register	SW	0x64	Hexadecimal			O
Special relay	SM	0x43	Decimal		O	
Special register	SD	0x44	Decimal			O

(c) RUN

・Request

The RUN request sets the master station to a RUN state.

CT=0x08

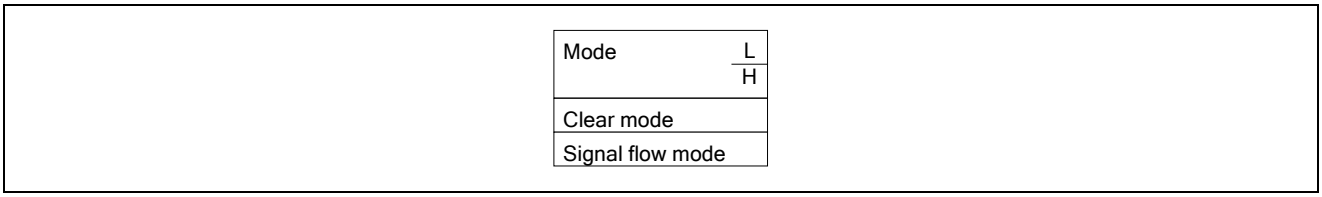


Figure 3.21 RUN Request Data Area

Table 3.21 RUN Request Setting List

Item	Setting	Value
Mode	Normal RUN	0x0003
	Forced RUN	0x0001
Clear mode	Clear all	0x02
	Clear all areas other than latch range	0x01
	Do not clear device	0x00
Signal flow mode	Fixed value	0x00

・Response

The RUN response sends the result of the RUN request.

CT=0x88

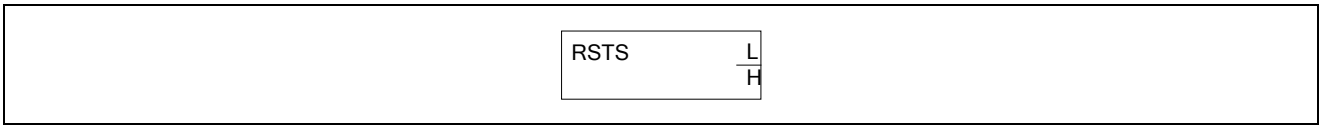


Figure 3.22 RUN Response Data Area

For RSTS, see Section (3) (a) "Return code" of Section 3.3.3 "Transient2 frame format."

(d) STOP

・Request

The STOP request sets the master station to a STOP state.

CT=0x09

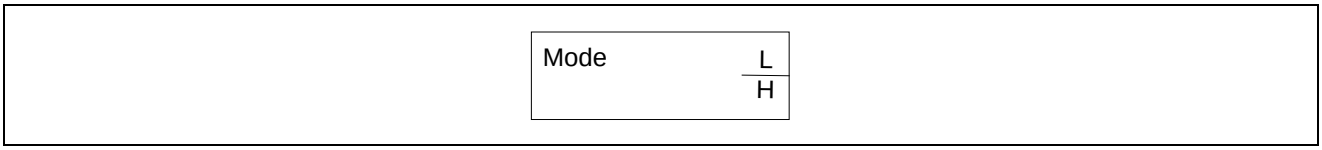


Figure 3.23 STOP Request Data Area

Table 3.22 STOP Request Setting List

Item	Setting	Value
Mode	Normal STOP	0x0003
	Forced STOP	0x0001

・Response

The STOP response sends the result of the STOP request.

CT=0x89

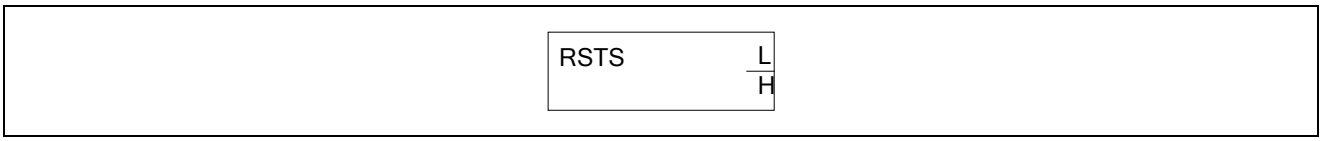


Figure 3.24 STOP Response Data Area

For RSTS, see Section (3) (a) "Return code" of Section 3.3.3 "Transient2 frame format."

(e) Get memory

・Request

CT=0x10

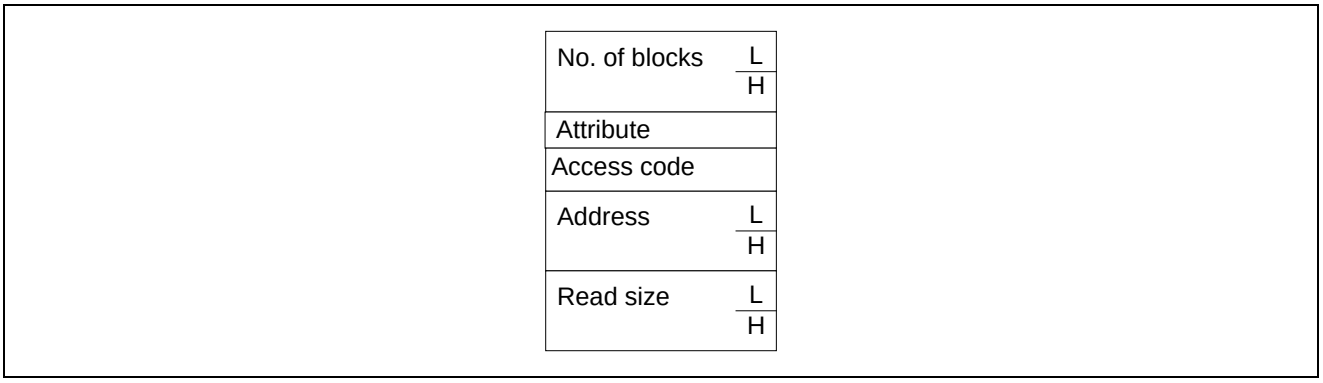


Figure 3.25 Get Memory Request

Table 3.23 Get Memory Request

Item	Setting	Value
No. of blocks	No. of blocks from Attribute to Read Size.	Fixed to 0x0001
Attribute	External information (unit: words)*1	Fixed to 5
Access code	Access code*2	
Address	Address of device	0 to 7664
Read size	Unit: Words	1 to 480

Note 1. See Figure 3.26 "Attributes."

2. See Table 3.20 "Access Code List (Example: Mitsubishi Product)."

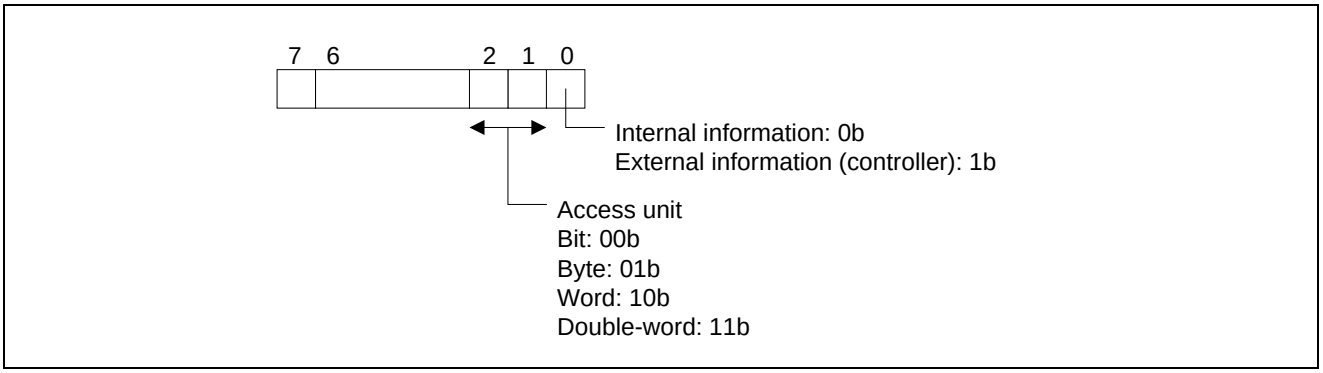


Figure 3.26 Attributes

・Response

The Get Memory response sends the read results of the Get Memory request.

CT=0x90

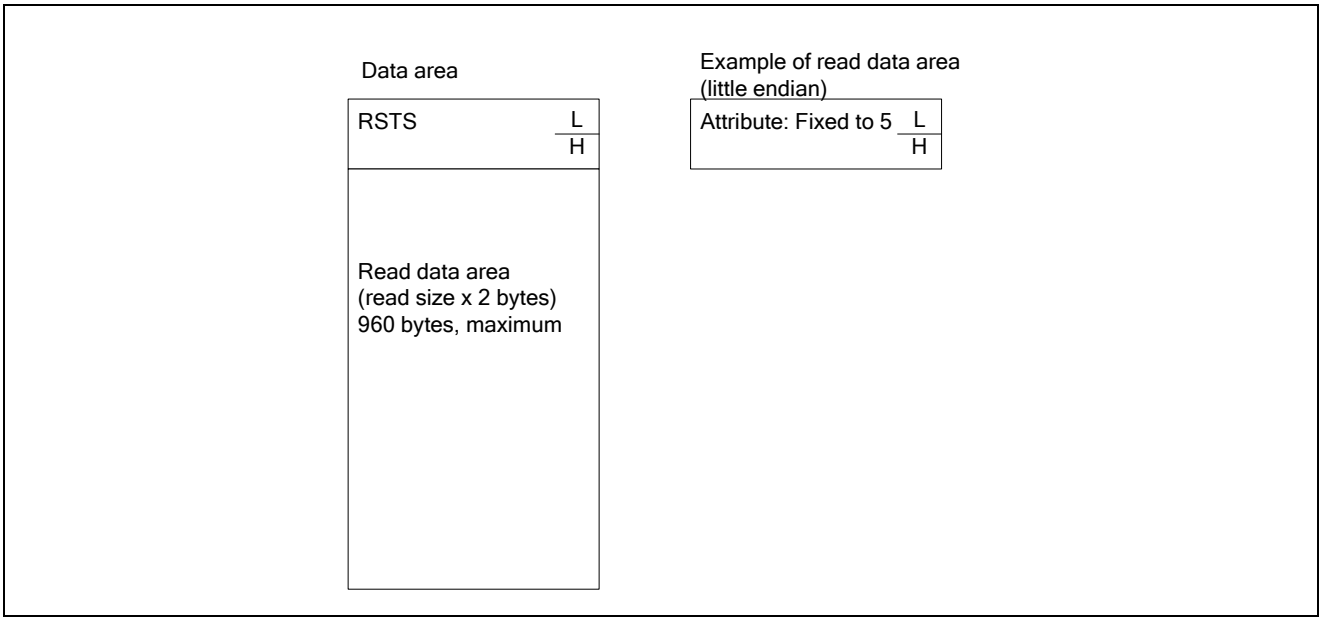


Figure 3.27 Get Memory Response

For RSTS, see Section (3) (a) "Return code" of Section 3.3.3 "Transient2 frame format."

(f) Set memory

・Request

CT=0x12

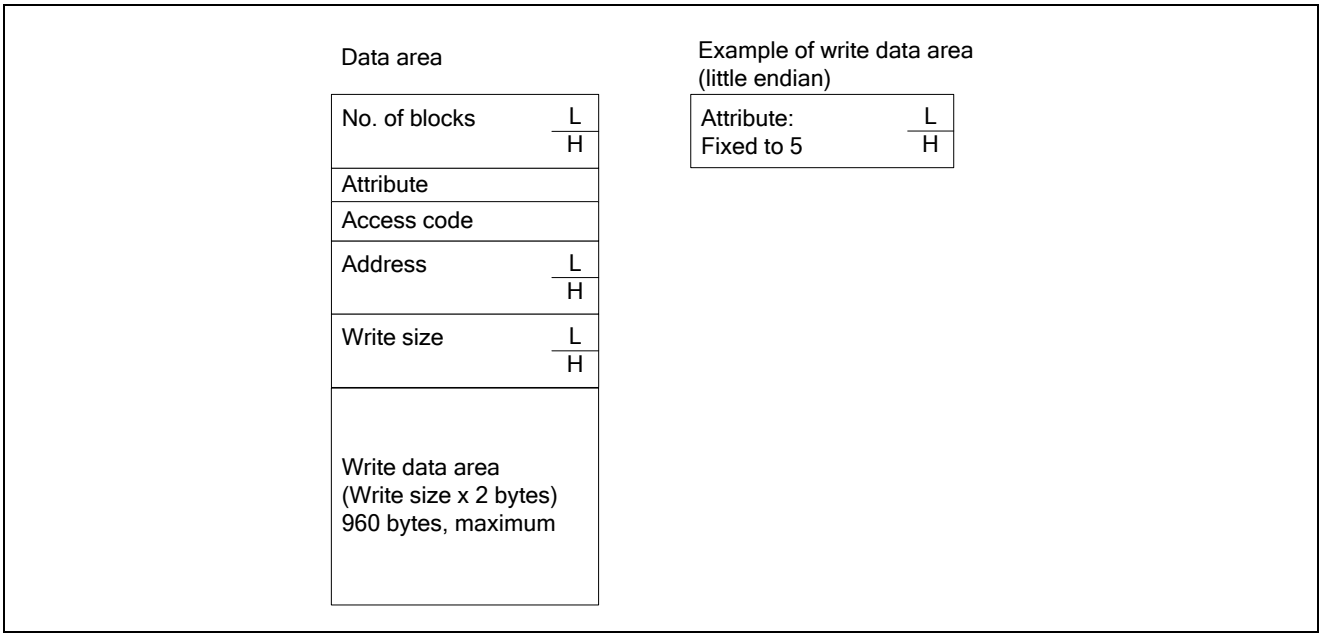


Figure 3.28 Set Memory Request

Table 3.24 Set Memory Request Setting List

Item	Setting	Value
No. of blocks	No. of blocks from Attribute to Write Size.	Fixed to 0x0001
Attribute	External information (unit: words)Note1	Fixed to 5
Access code	Access codeNote2	
Address	Address of device	0 to 7664
Write size	Unit: Words	1 to 480

Note 1. See Figure 3.26 "Attributes."

2. See Table 3.20 "Access Code List (Example: Mitsubishi Product)."

・Response

CT=0x92

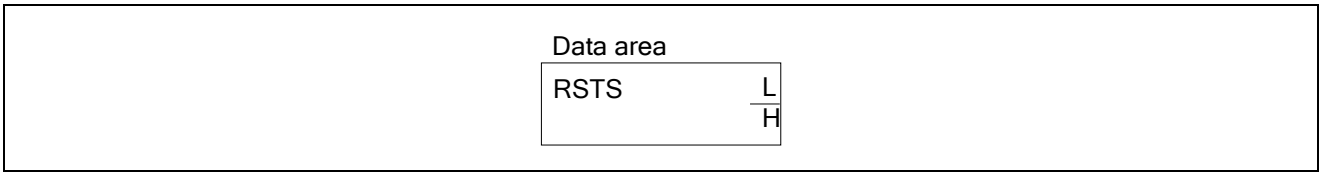


Figure 3.29 Set Memory Response

For RSTS, see Section (3) (a) "Return code" of Section 3.3.3 "Transient2 frame format."

4. DEVELOPING FIRMWARE

4.1 Development Procedure

This section describes the procedure for developing firmware that uses the sample code.
The sample code comprises the program elements described in Table 4.1.
While customization of the R-IN32M3-CL driver main unit is not required, other program elements need to be customized in accordance with the hardware of the device (target) to be developed.

Table 4.1 List of Program Elements Included in Sample Code

Program Element Name	Description	Customization by Vendor
User program	A program for mounting device functions. The sample code describes only the section related to communication. Create a new program while referring to Section 4.2.	◎
R-IN32M3-CL driver main unit	Comprises functions that are called from the user program. The user program can call the R-IN32M3-CL driver interface functions described in Sections 4.2 and 4.5, enabling use of R-IN32M3-CL functions. Customization of the R-IN32M3-CL driver main unit is not required.	X
Target-dependent function group for R-IN32M3-CL driver	A function group that requires customization in accordance with the hardware developed by the vendor. For target dependent function group details, see Section 4.6.	O
Call-back function group for R-IN32M3-CL driver	A function group called from the driver when an event occurs, requiring customization by the vendor. For call-back function group details, see Section 4.7.	O

Remark ◎ : Required (must be produced newly)
○ : Required
× : Not required

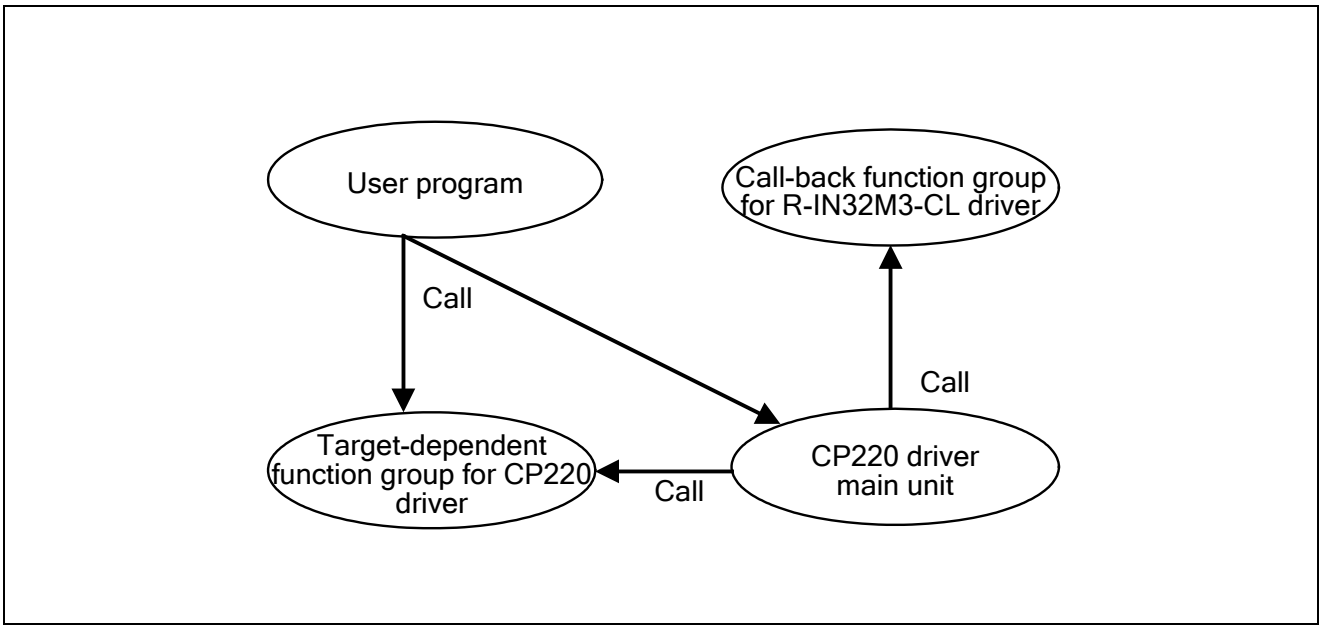


Figure 4.1 Configuration of Firmware

Point	
Be sure to implement the processing described in Section 4.6.2 "Creating a target-dependent function group for the R-IN32M3-CL driver."	

The following describes the procedure for developing firmware. (See Figure 4.2.)

Step1: Creating the user program
The vendor creates the user program while referring to Section 4.2.1, "General flowchart."

Step 2: Customizing the target-dependent function group for the R-IN32M3-CL driver
The vendor customizes the target-dependent function group for the R-IN32M3-CL driver in accordance with the hardware of the device to be developed. For details, see Section 4.6, “Customizing the Target-Dependent Function Group for the R-IN32M3-CL Driver.”

Step 3: Customizing the call-back function group for the R-IN32M3-CL driver
The vendor customizes the call-back function group for the R-IN32M3-CL driver in accordance with the functions to be achieved by the device to be developed. For details, see Section 4.7, “Customizing the Call-Back Function Group for the R-IN32M3-CL Driver.”

Step 4: Creating the R-IN32M3-CL library
Once the R-IN32M3-CL driver main unit and target-dependent function group for the R-IN32M3-CL driver are compiled, execute the librarian and create a library file.

Step 5: Connecting the user program and library file
Connect the user program, the customized call-back function group for the R-IN32M3-CL driver, and the library file, and create a load module file.

Step 6: Load the load module file to the device (target) to be developed.

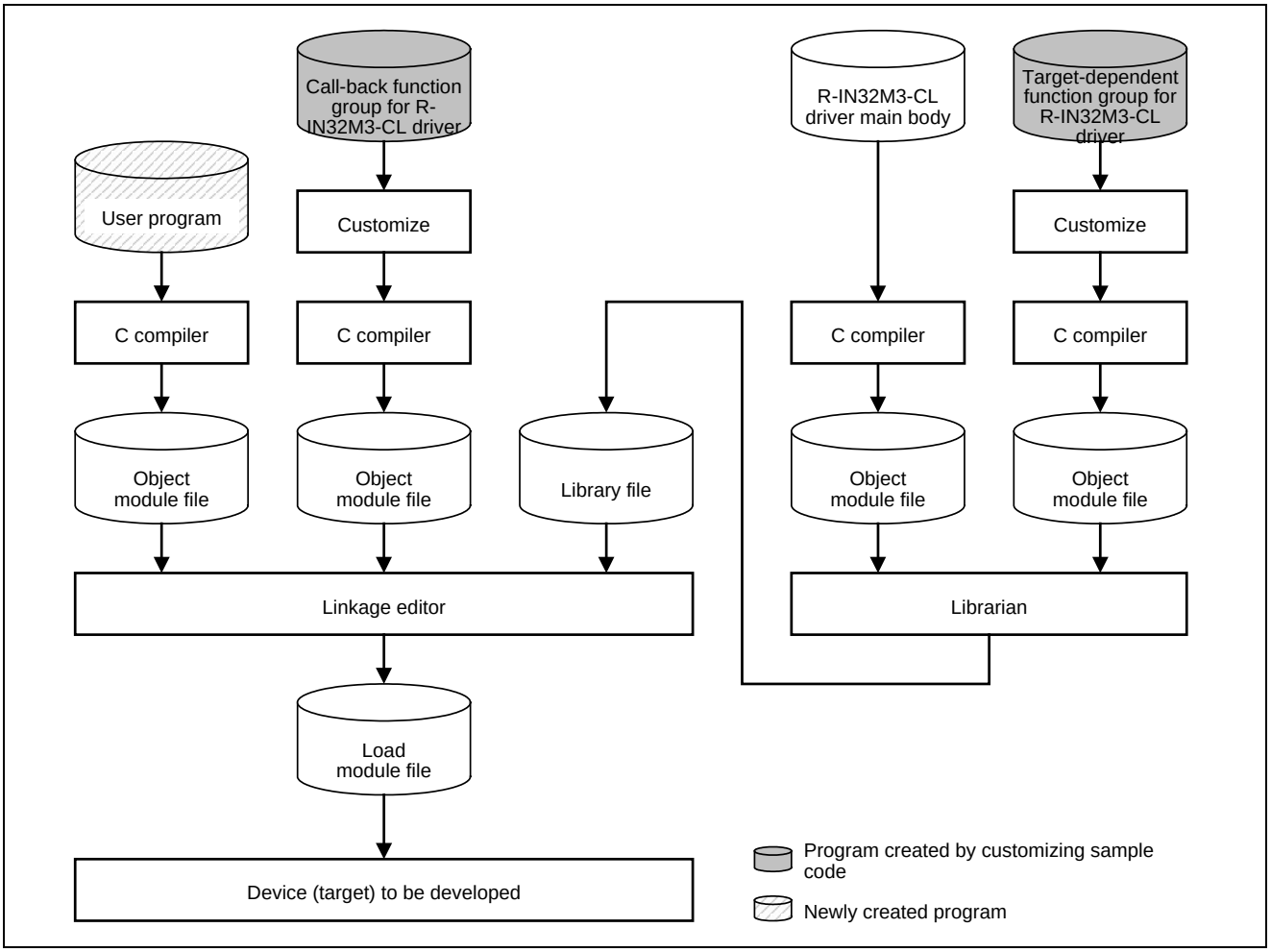


Figure 4.2 Firmware Development Procedure

4.2 Sample Flowcharts of User Program

This section presents sample flowcharts of a user program that uses driver interface functions.
Write your user program based on the sample flowcharts.

Table 4.2 List of Sample Flowcharts(1/2)

No.	Function	Refer to	Implementation by Vendor	Remarks
1	General flow	Section 4.2.1	◎	
2	Initialization processing	Section 4.2.2	◎	
3	Start Communication processing	Section 4.2.3	◎	
4	Check PHY processing	Section 4.2.4	△	Necessity of implementation varies according to PHY.
5	Change PHY Setting processing	Section 4.2.5	△	
6	Force Stop processing	Section 4.2.6	△	
7	Stop Cyclic Communication processing	Section 4.2.7	△	
8	Event processing	Section 4.2.8	◎	
9	Receive MyStatus from Master Station and Cyclic Data processing	Section 4.2.9	◎	
10	Send MyStatus processing	Section 4.2.10	◎	
11	Send Cyclic Data processing	Section 4.2.11	◎	
12	Update Communication Status processing	Section 4.2.12	◎	
13	Update Cyclic Communication Status processing	Section 4.2.13	△	
14	Get MIB Information processing	Section 4.2.14	△	
15	Receive Transient1, Transient2, and TransientAck processing	Section 4.2.15	◎	
16	Create Transient2 Request Frame processing	Section 0	O	
17	Send Transient1, Transient2, and TransientAck processing	Section 4.2.16	◎	
18	Received Transient1 Data processing	Section 0	◎	
19	Start Making Received Transient1 Data processing	Section 4.2.18	◎	
20	Make Received Transient1 Data processing	Section 4.2.19	◎	
21	Received Node Information Distribution Frame processing	Section 4.2.20	O	
22	Check Node Information Distribution Frame processing	Section 4.2.21	O	
23	Received Statistical Information Request Frame processing	Section 4.2.22	◎	
24	Create Get Statistical Information Response Frame processing	Section 4.2.23	◎	

Remark ◎ : Required
○ : Required with Transient2 implementation
△ : Optional

Table 4.2 List of Sample Flowcharts(2/2)

No.	Function	Refer to	Implementation by Vendor	Remarks
25	Received Detailed Node Information Request Frame processing	Section 4.2.24	◎	
26	Create Get Detailed Node Information Response Frame processing	Section 4.2.25	◎	
27	Received Transient2 Data processing	Section 4.2.26	O	
28	Check Received Transient2 Data processing	Section 4.2.27	O	
29	Received TransientAck Data processing	Section 4.2.28	◎	
30	Create TransientAck Frame processing	Section 4.2.29	◎	
31	Create Transient2 Response Frame processing	Section 4.2.30	O	
32	Create Transient2 Get Memory Request Frame processing	Section 4.2.31	O	
33	Received Transient2 Set Memory Request processing	Section 4.2.32	O	
34	Received Transient2 Set Memory Response processing	Section 4.2.33	O	
35	Hardware test (IEEE 802.3ab compliance test)	Section 4.2.34	◎	
36	Hardware test (loop-back communication test)	Section 4.2.35	△	Implementation recommended.

Remark ◎ : Required
O : Required with Transient2 implementation
△ : Optional

4.2.1 General flowchart

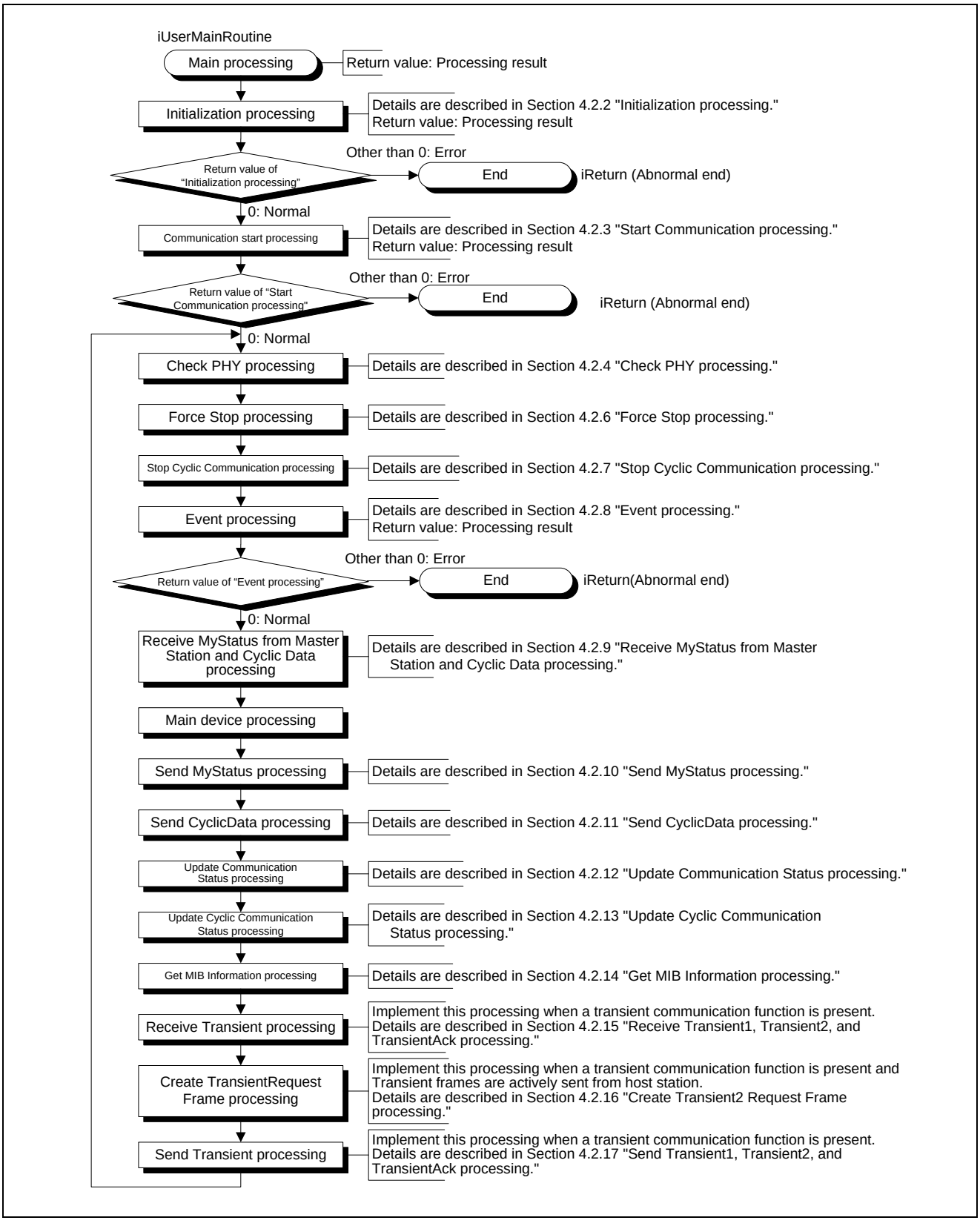


Figure 4.3 General Flowchart

4.2.2 Initialization processing

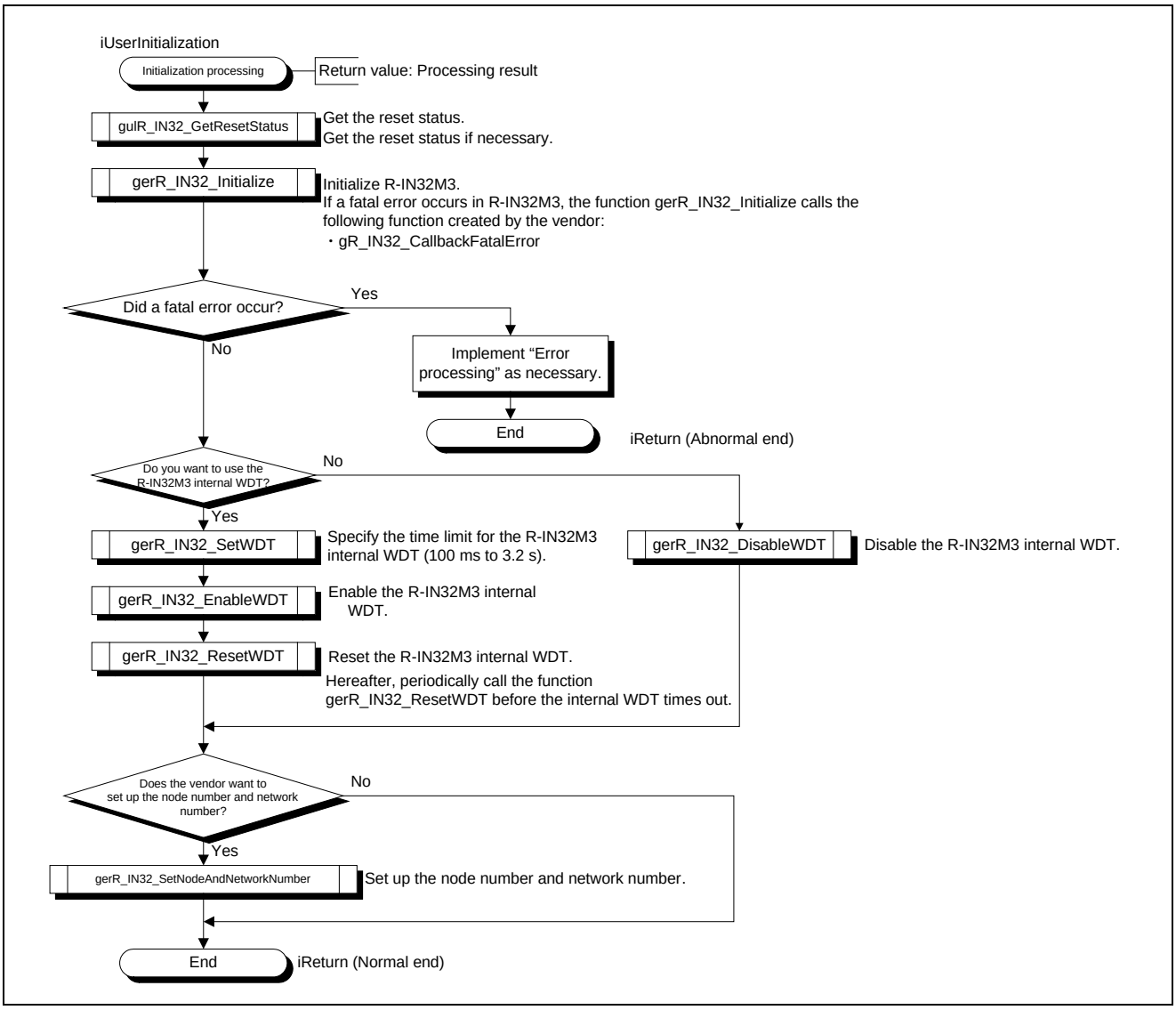


Figure 4.4 Initialization Processing Flowchart

4.2.3 Start Communication processing

Start Communication processing provides instructions for starting R-IN32M3-CL communication.

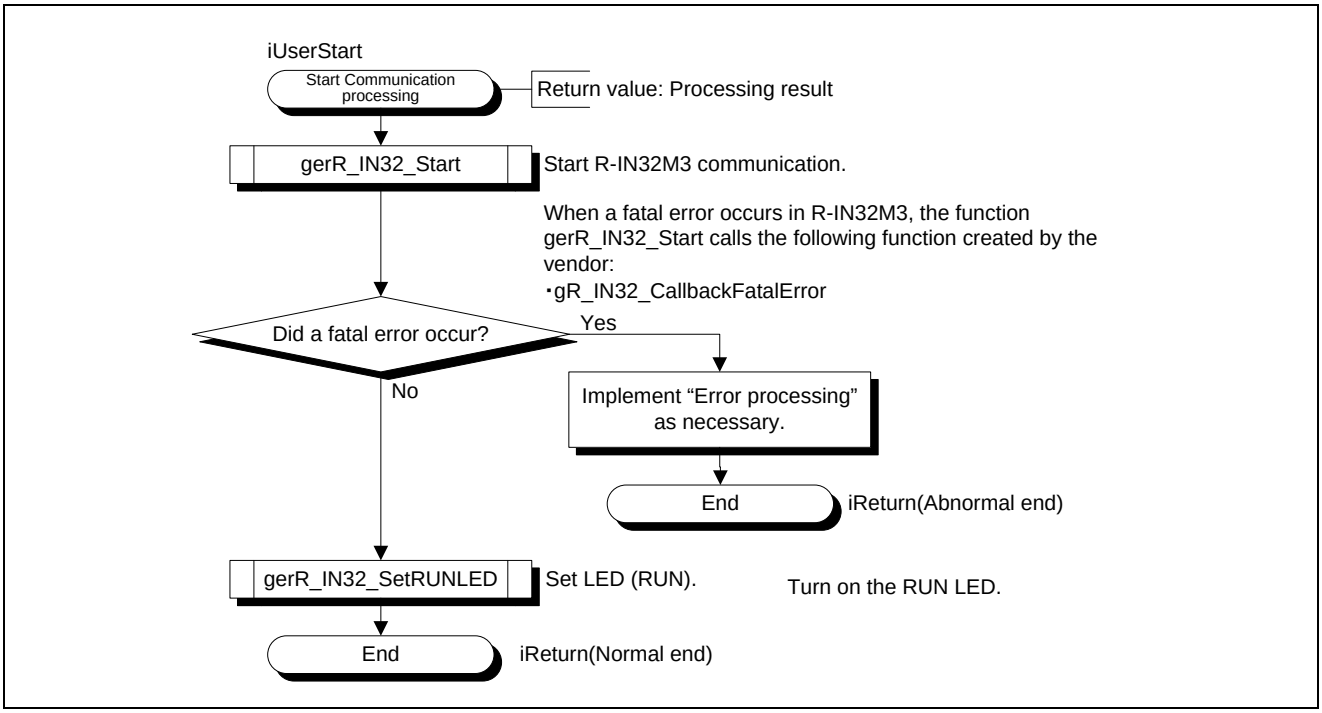


Figure 4.5 Start Communication Processing Flowchart

4.2.4 Check PHY processing

The intelligent device station requires 1-Gbps/fullduplex linkup. Check PHY processing checks if PHY is linked under settings other than 1-Gbps/full duplex.

If PHY is linked under settings other than 1-Gbps/full duplex, Check PHY processing changes the PHY setup.

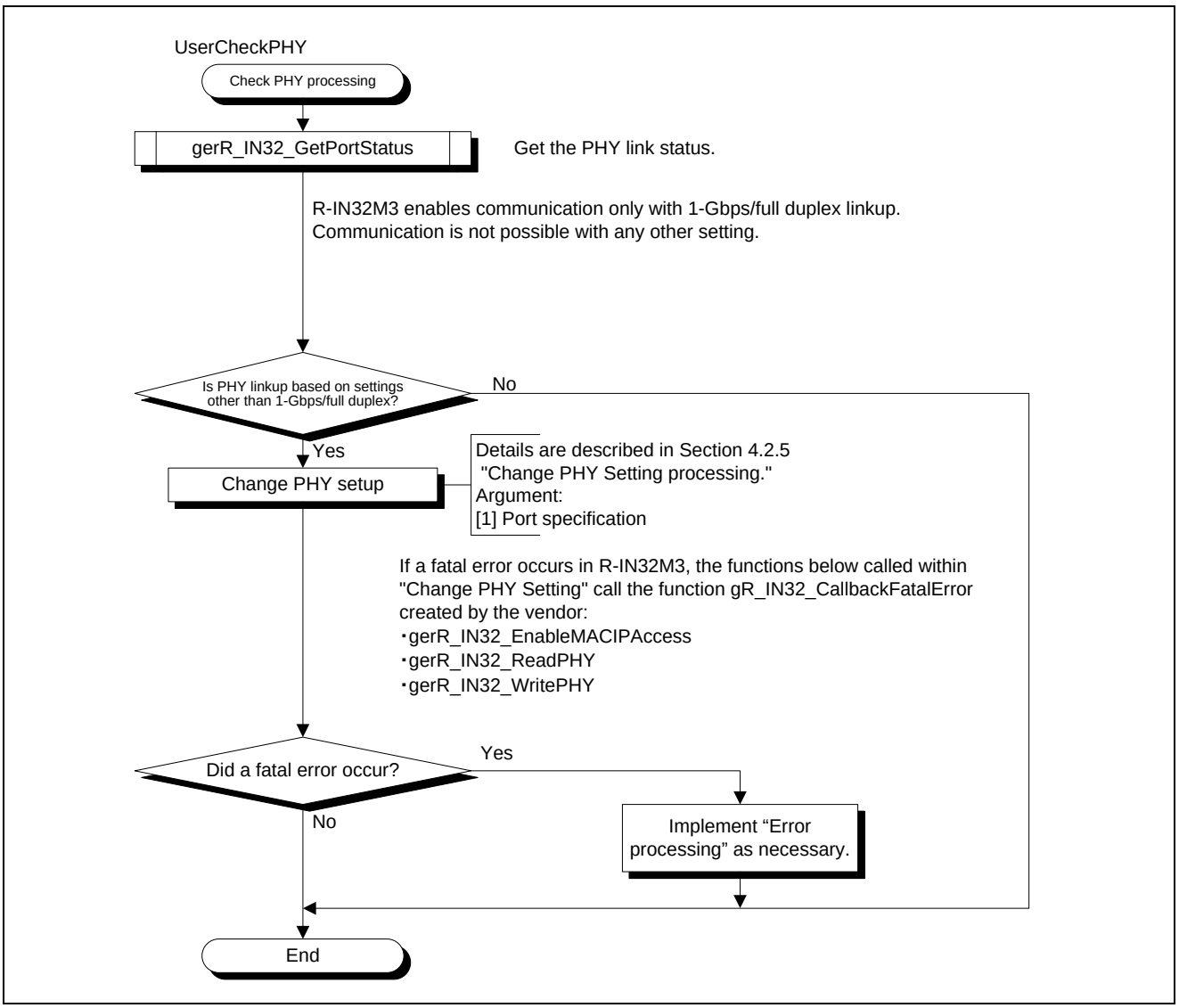


Figure 4.6 Check PHY Processing Flowchart

Point	
Implement the above processing on both port 1 and port 2. Implementation is not required if the PHY used permits linkup fixed to 1-Gbps/full duplex according to hardware settings.	

4.2.5 Change PHY Setting processing

The Change PHY Setting processing sets PHY so that it only permits linkup under 1-Gbps/full duplex settings.

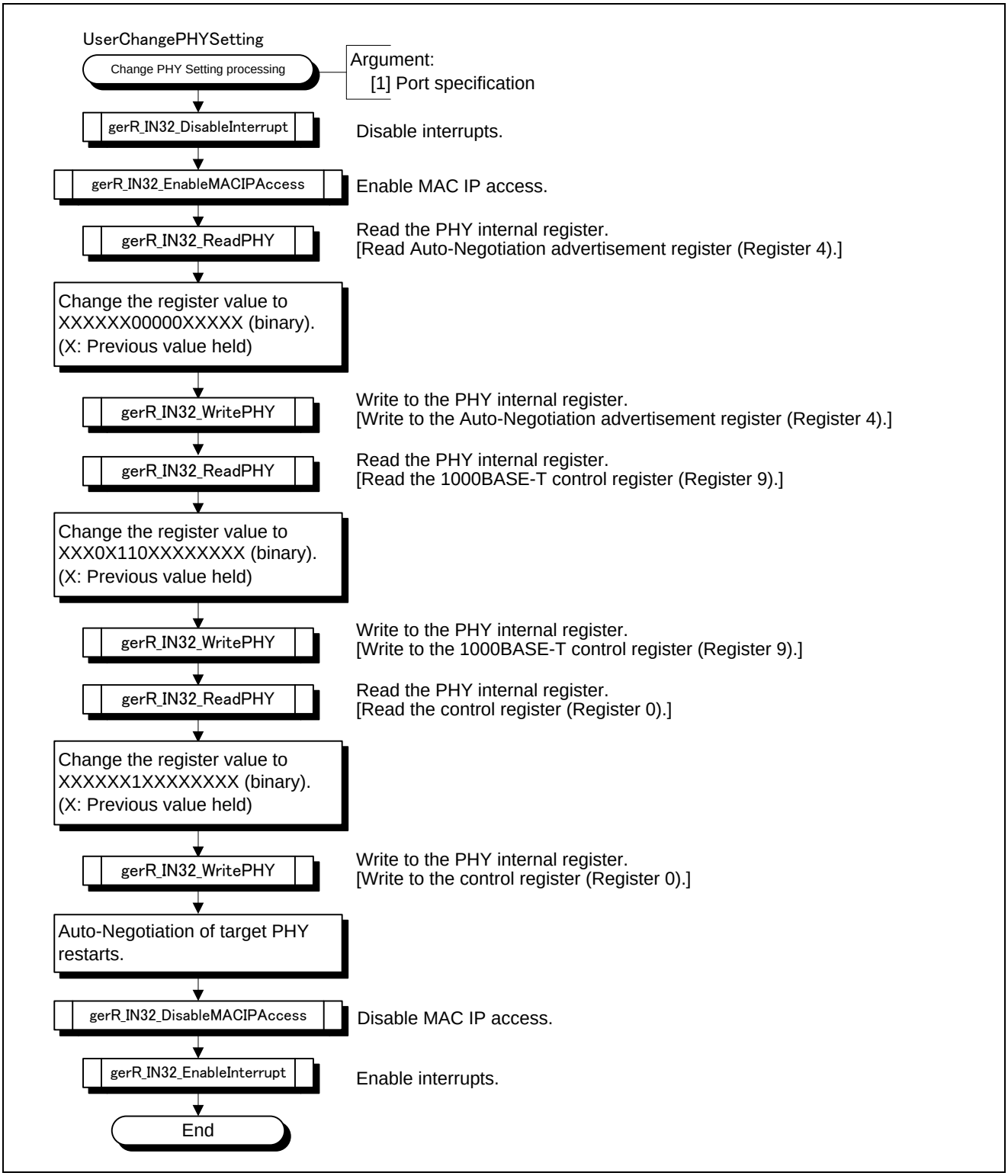


Figure 4.7 Change PHY Setting Flowchart

4.2.6 Force Stop processing

Force Stop processing allows you to forcibly stop the intelligent device station for device-side reasons. (This processing is optional.)

When the intelligent device station is forcibly stopped, R-IN32M3-CL changes to bypass mode. (Communication frames are neither transmitted nor received. Received frames are forwarded as is to the other port.)

To clear a forced stop, power ON reset or system reset is required.

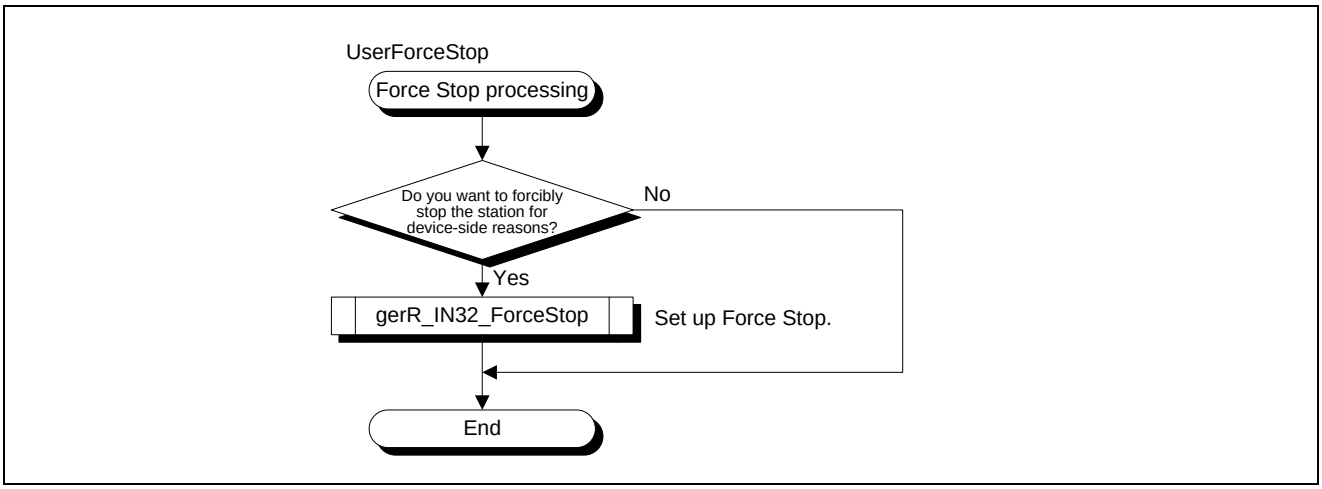


Figure 4.8 Force Stop Processing Flowchart

4.2.7 Stop Cyclic Communication processing

Stop Cyclic Communication processing allows you to control the stopping and restarting of cyclic communication for device-side reasons. (This processing is optional.)

Even if you stop cyclic communication, token passing continues.

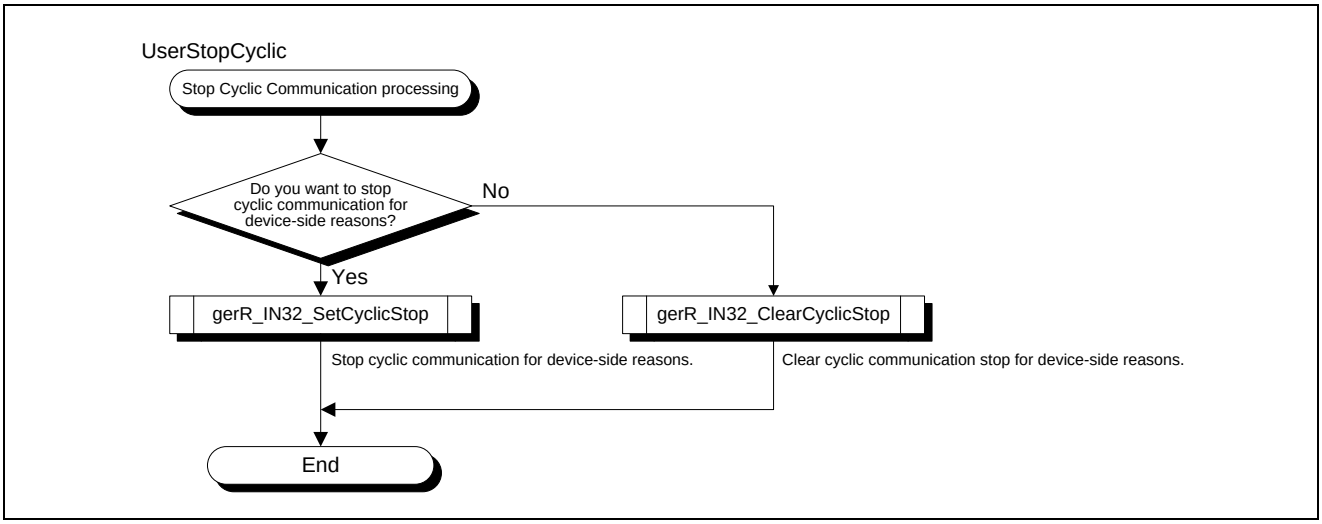


Figure 4.9 Stop Cyclic Communication Processing Flowchart

4.2.8 Event processing

Event processing detects MPU interrupts (R-IN32M3-CL events), processes the event and updates MIB information.

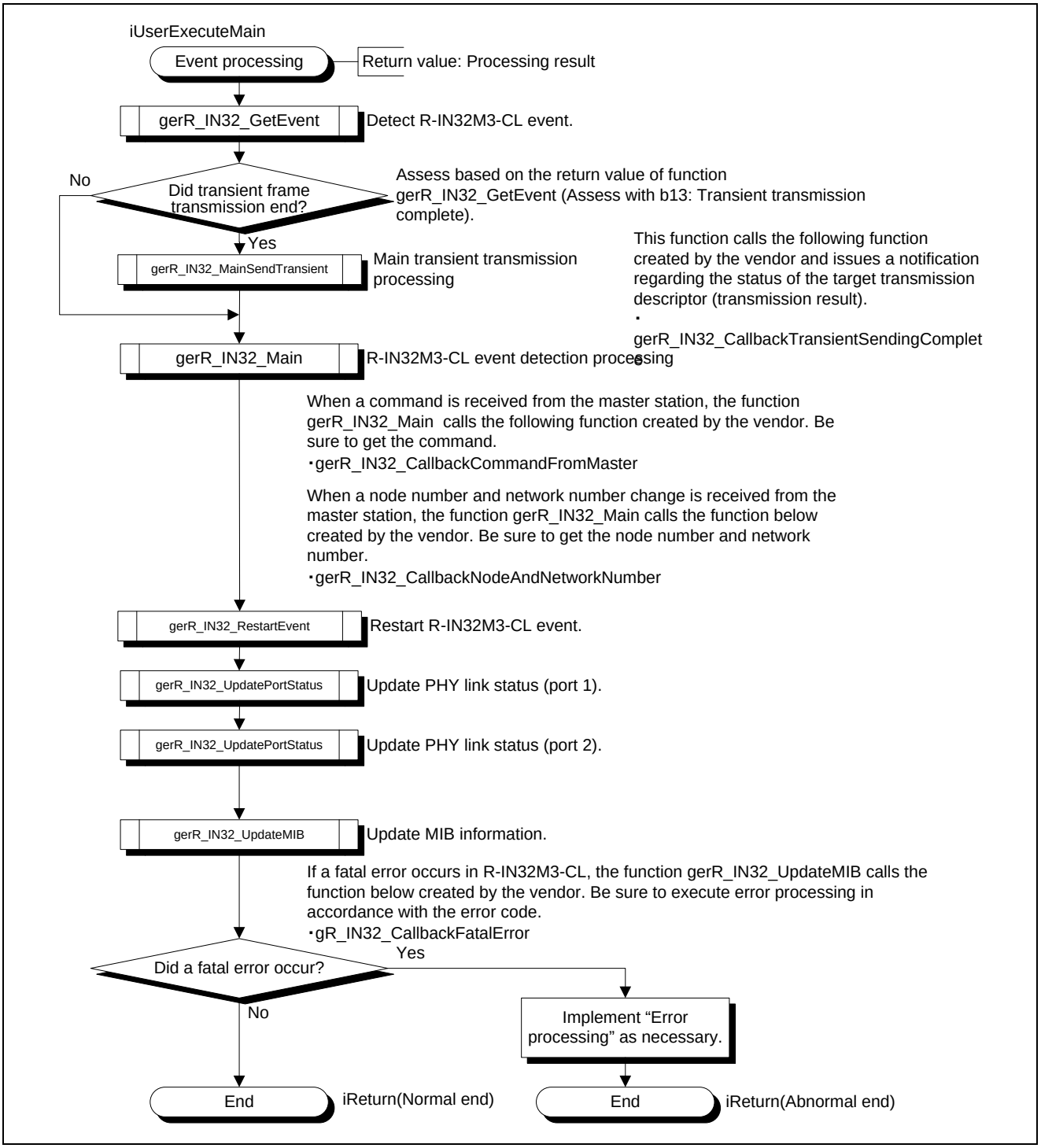


Figure 4.10 Event Processing Flowchart

4.2.9 Receive MyStatus from Master Station and Cyclic Data processing

Receive MyStatus from Master Station and Cyclic Data processing acquires the status of the master station from the received MyStatus frame and acquires cyclic data (RY and RWw) from the received cyclic frame. When the master station is stopped or an error has occurred, perform “Hold/Clear Device processing” (hold or clear the data (RY or RWw) received up to that time according to device specifications).

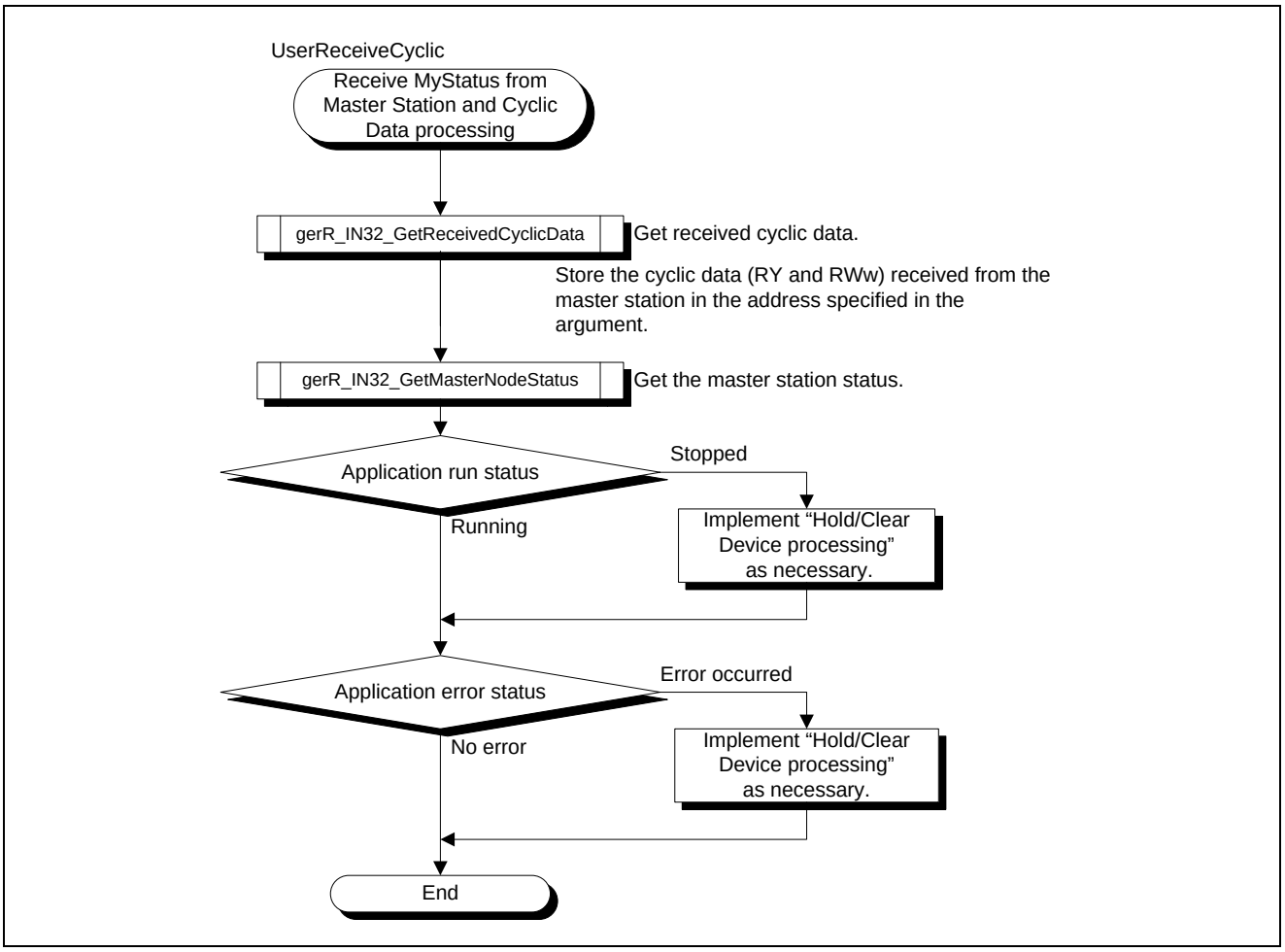


Figure 4.11 Receive MyStatus from Master Station and Cyclic Data Processing Flowchart

4.2.10 Send MyStatus processing

Send MyStatus processing sets its own station status in R-IN32M3-CL and sets the MyStatus transmission data.

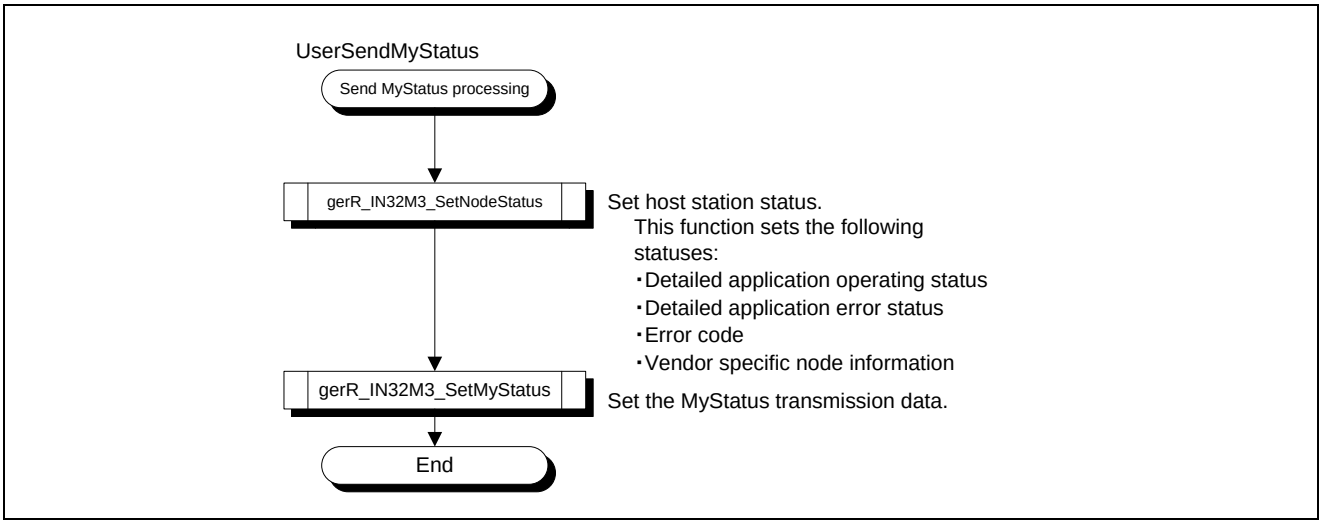


Figure 4.12 Send MyStatus Processing Flowchart

4.2.11 Send Cyclic Data processing

Send Cyclic Data processing sends cyclic transmission data (RX and RWr).

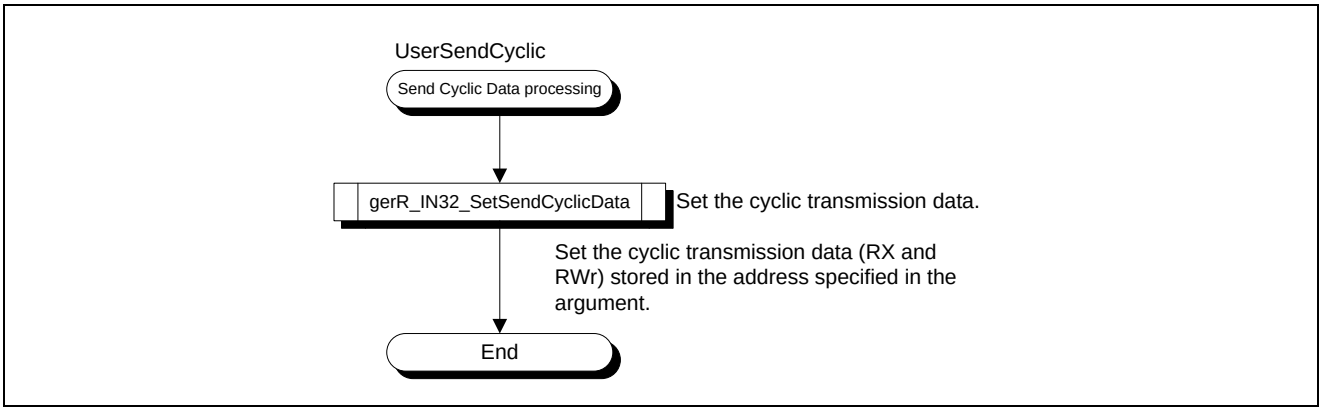


Figure 4.13 Send Cyclic Data Processing Flowchart

4.2.12 Update Communication Status processing

Update Communication Status processing acquires the R-IN32M3-CL data link status, and controls the on/off status of the D LINK LED and ERR LED. When there is no data link (when the data link is disconnected), perform “Hold/Clear Device processing" (hold or clear the data (RY and RWw) received up to that time according to device specifications).

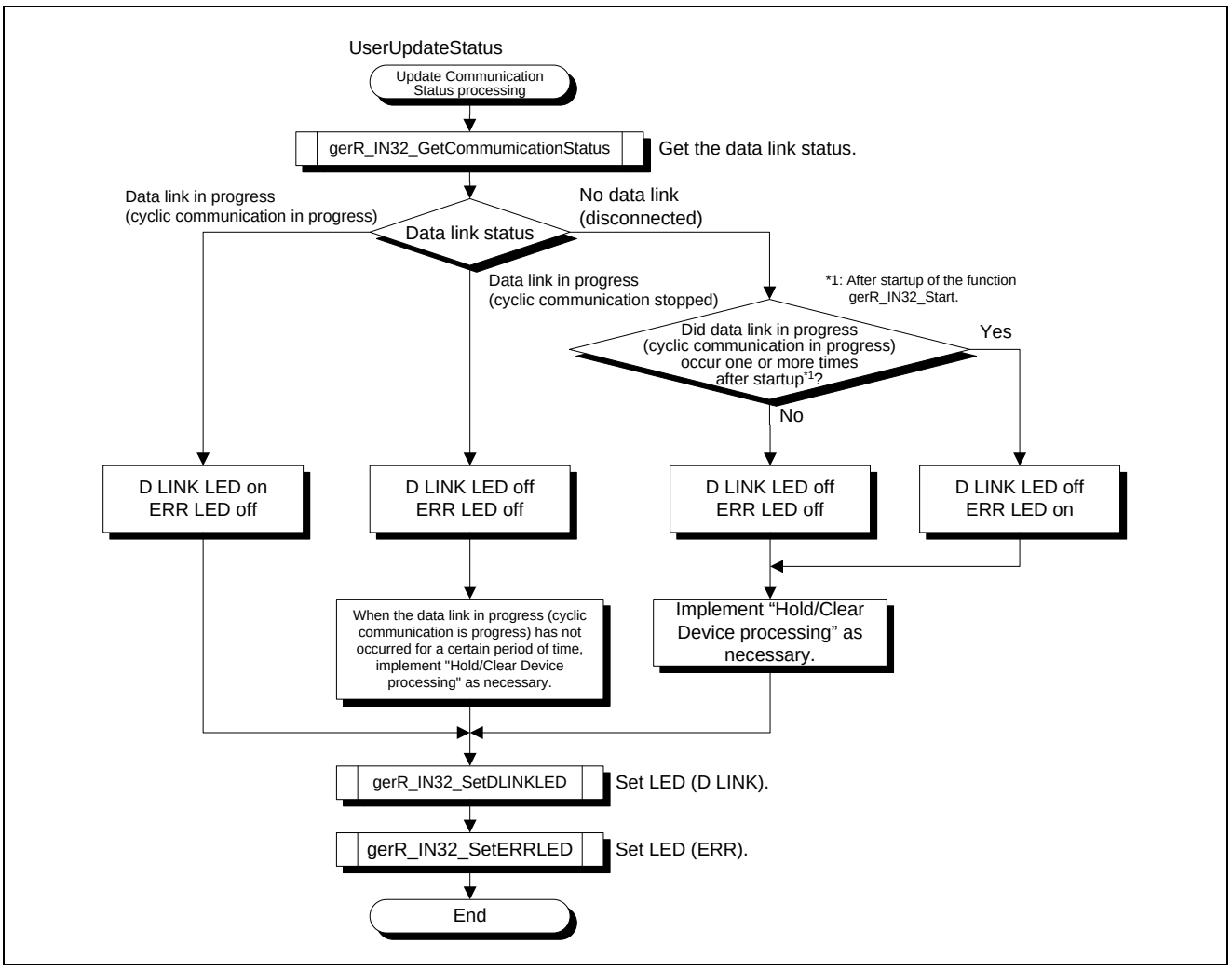


Figure 4.14 Update Communication Status Processing Flowchart

4.2.13 Update Cyclic Communication Status processing

Update Cyclic Communication Status processing allows you to acquire the cyclic communication size specified by the master station and the cyclic communication status.

Cyclic communication is processed by R-IN32M3-CL. As a result, the user program does not need to acquire the cyclic communication size or cyclic communication status. (This processing is optional.)

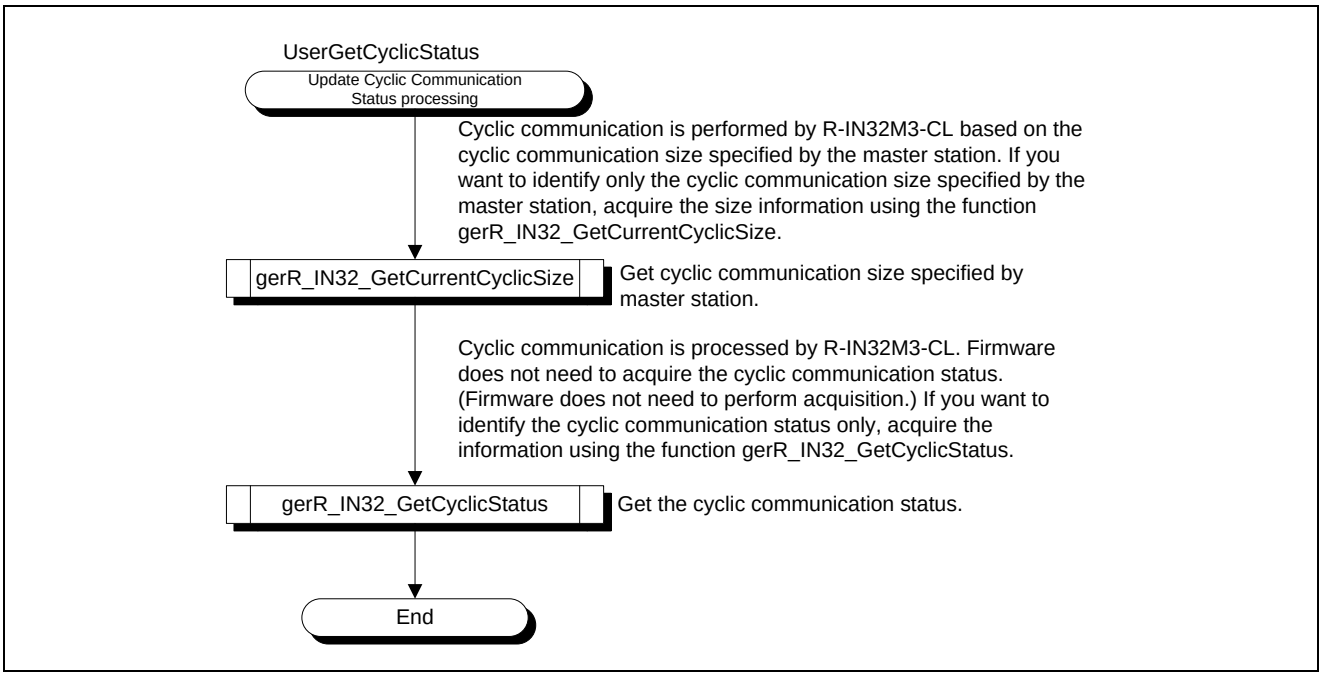


Figure 4.15 Update Cyclic Communication Status Processing Flowchart

4.2.14 Get MIB Information processing

Get MIB Information processing acquires and/or clears MIB information.

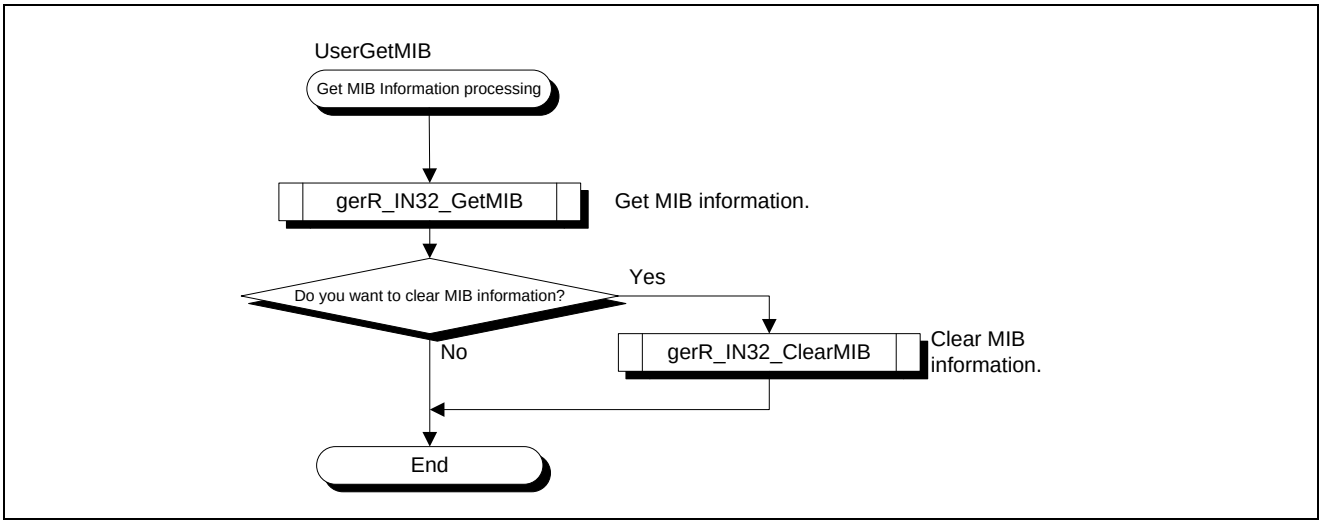


Figure 4.16 Get MIB Information Processing Flowchart

Precaution

MIB information is non-disclosed information. Regard the information as deployed by the vendor only.

(1) List of MIB Information of Ring Control Area

Table 4.3 List of MIB Information of Ring Control Area

No.	MIB Information	Description
1	No. of HEC error frames	Counts the number of HEC errors in received frames.
2	No. of DCS/FCS error frames	Counts the number of DCS/FCS errors in received frames.
3	No. of undersize error frames	Counts the number of received error frames with a size less than 28 bytes.
4	No. of forwarded frames	Counts the number of forwarded frames.
5	No. of upper layer transmission frames	Counts the number of frames transmitted to upper layers.
6	No. of discarded frames due to full forward buffer	Counts the number of frames discarded due to a full forward buffer.
7	No. of discarded frames due to full upper layer transmission buffer	Counts the number of frames discarded due to a full upper layer transmission buffer.

(2) List of MIB Information of MAC IP Area

Table 4.4 List of MIB Information of MAC IP Area

No.	MIB Information	Description
1	No. of received frames	Counts all frame receptions, including error frames. Error frames: FCS error, undersized, oversized frames
2	No. of sent frames	Counts the number of sent frames.
3	No. of received undersized frames	Counts the number of received frames with a size less than 64 bytes.
4	No. of received oversized frames	Counts the number of received frames with a size exceeding 1,518 bytes.
5	No. of received frame FCS errors	Counts the number of received frames with an FCS error.
6	No. of received frame fragment errors	Counts the number of received frames with fragment errors. Fragment error: A frame with less than 64 bytes and an FCS error
7	No. of frames detected within minimum IFG	Counts the number of frames detected within the minimum inter-frame gap (IFG).
8	No of received frames with SFD or less	Counts the number of received frames that ended at a field up to SFD and were not recognized as a valid frame.
9	No. of reception code errors	Counts the number of GMII reception data errors detected (RECV_*_ERR=1*[1]). Counts a RECV_*_ERR*[1] that occurred multiple times in an idle state (RECV_*_DV=1*[1]) as one error. *1. The asterisk (“*”) indicates a wild character. (A: Port 1, B: Port 2)
10	No. of received invalid carrier errors	Counts the number of invalid carriers that occurred in an idle state. Counts multiple invalid carriers that occurred in an idle state as one error.
11	No. of received invalid carrier extension errors	Counts the number of invalid carrier extensions that occurred in an idle state. Counts multiple invalid carrier extensions that occurred in an idle state as one error.

(3) List of Other MIB Information

Table 4.5 List of Other MIB Information

No.	MIB Information	Description
1	No. of link downs (port 1)	Counts the number of link downs of port 1.
2	No. of link downs (port 2)	Counts the number of link downs of port 2.
3	No. of master watchdog timer errors	Counts the number of timeouts of the watchdog timer.
4	No. of received cyclic frames	Counts the number of cyclic frames received by R-IN32M3-CL.
5	No. of received transient frames	Counts the number of transient frames received by R-IN32M3-CL.
6	No. of received transient frames discarded	Counts the number of received transient frames discarded by R-IN32M3-CL.

4.2.15 Receive Transient1, Transient2, and TransientAck processing

Receive Transient1, Transient2, and TransientAck processing receives Transient1, Transient2, and TransientAck frames and processes the data.

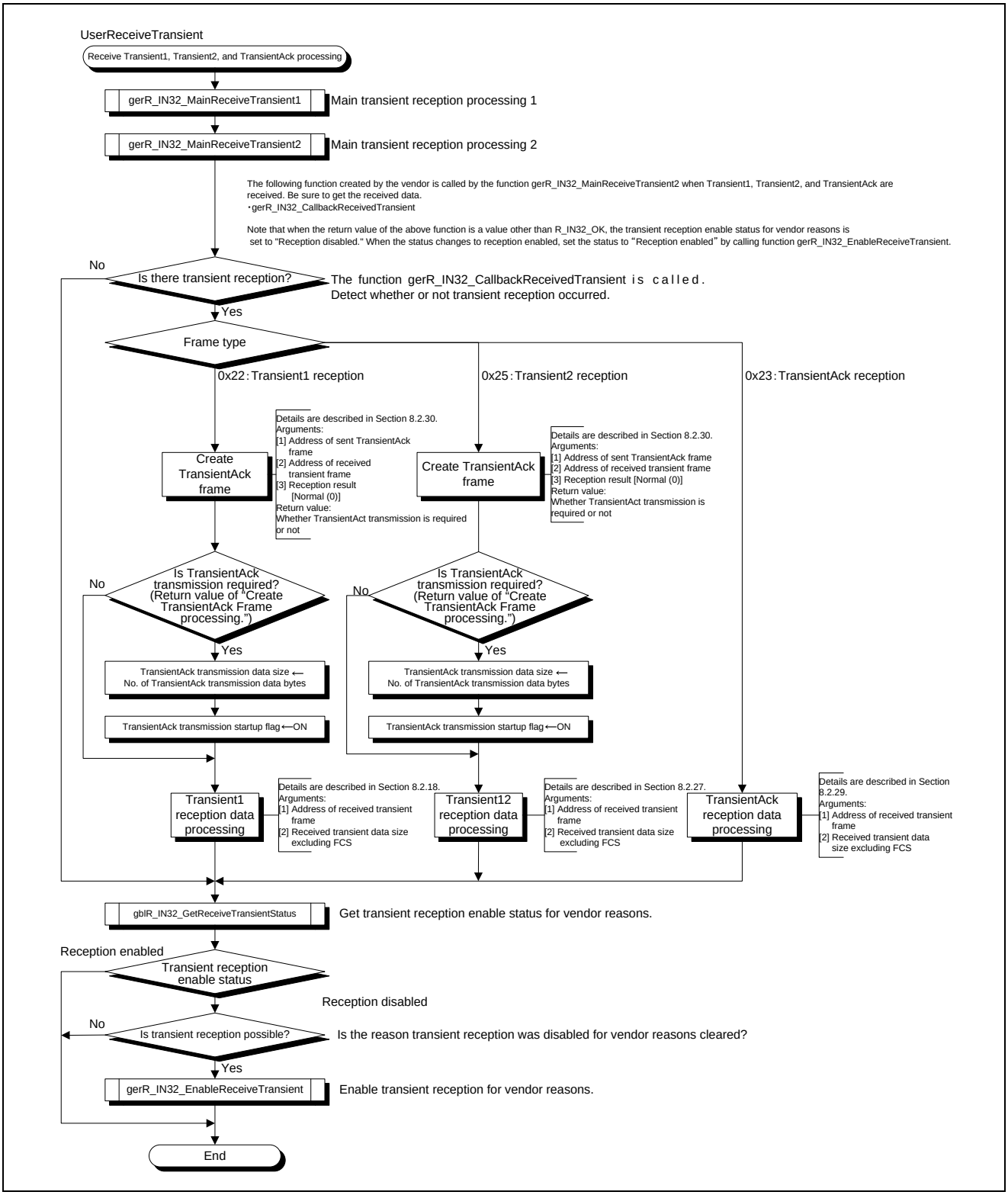
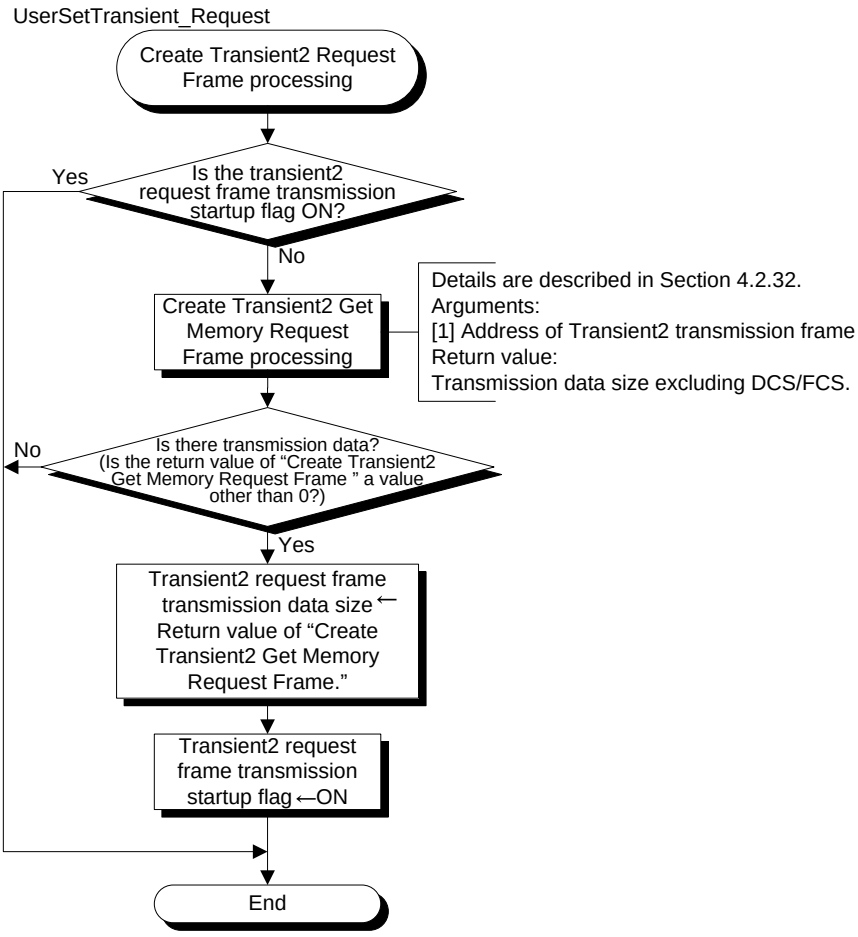


Figure 4.17 Receive Transient1, Transient2, and TransientAck Processing Flowchart

Create Transient2 Request Frame processing

Create Transient2 Request Frame processing creates a Transient2 request frame for the Get Memory command.



*This flowchart uses the "Transient2 Get Memory request" as an example. Implement processing as necessary.

Figure 4.18 Create Transient2 Request Frame Processing Flowchart

4.2.16 Send Transient1, Transient2, and TransientAck processing

Send Transient1, Transient2, and TransientAck processing sends Transient1, Transient2, and TransientAck frames.

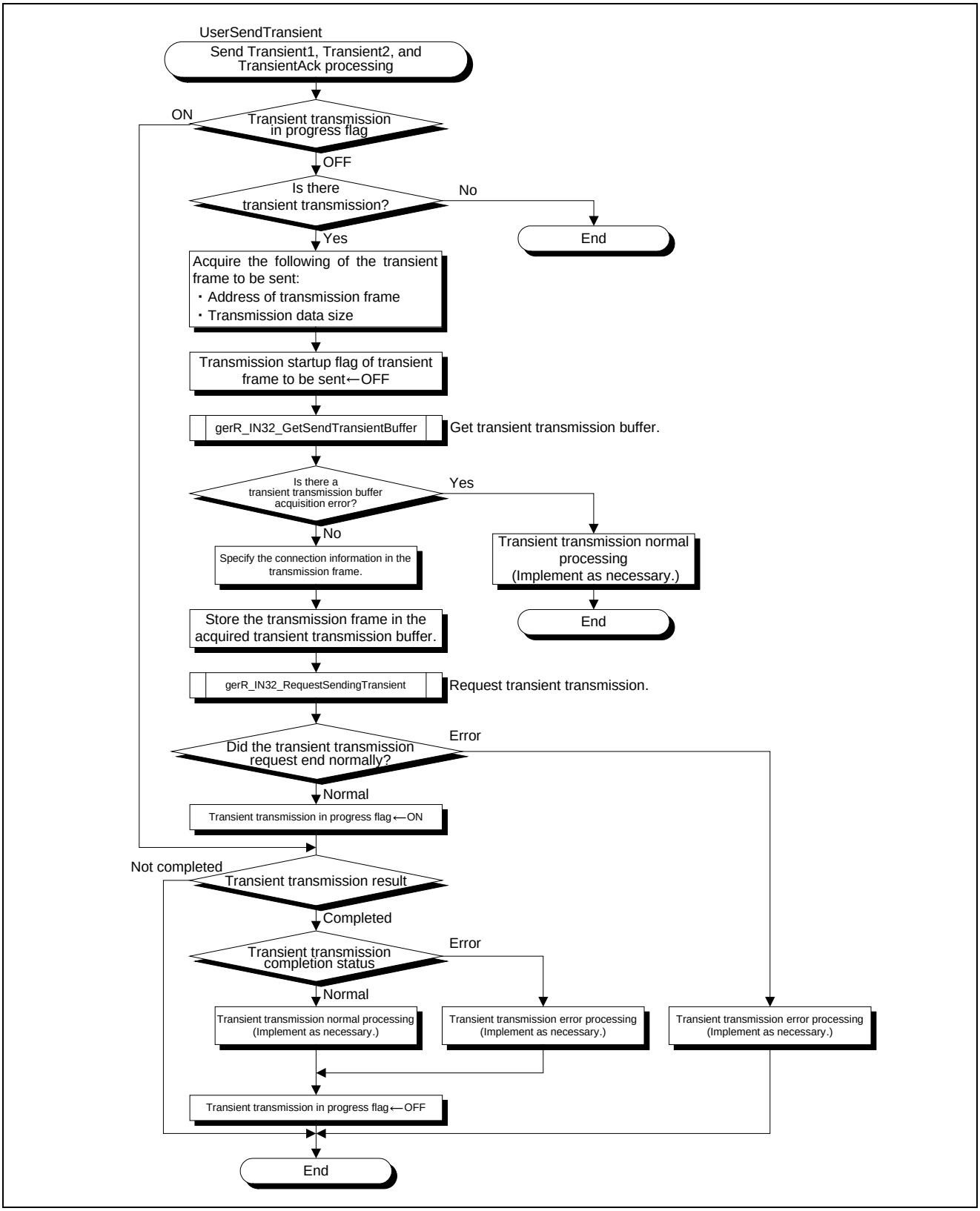


Figure 4.19 Send Transient1, Transient2, and TransientAck Processing Flowchart

4.2.17 Received Transient1 Data processing

Received Transient1 Data processing analyzes a received Transient1 frame and performs processing in accordance with the analysis result. In addition, the processing assembles data when a Transient1 frame is received in fragments.

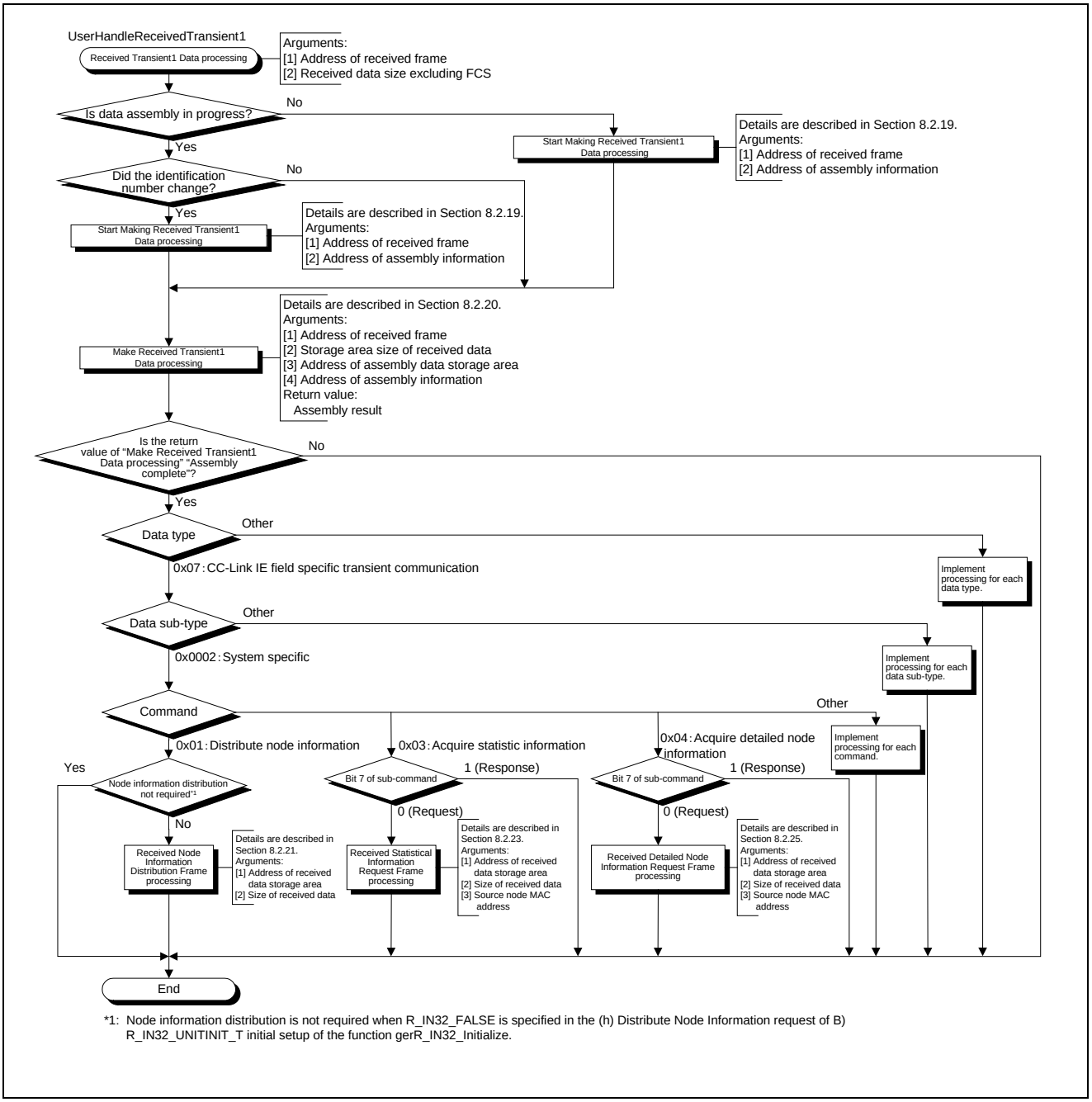


Figure 4.20 Received Transient1 Data Processing Flowchart

4.2.18 Start Making Received Transient1 Data processing

Start Making Received Transient1 Data processing starts the data assembly process of the Transient1 frame.

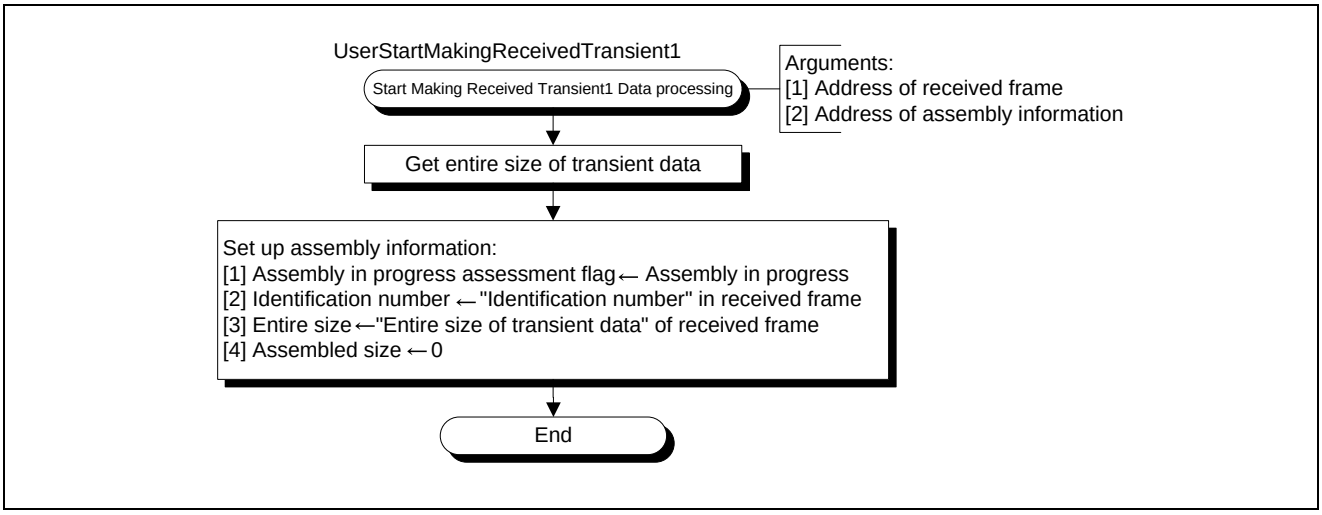


Figure 4.21 Start Making Received Transient1 Data Processing Flowchart

4.2.19 Make Received Transient1 Data processing

Make Received Transient1 Data processing assembles the data of the Transient1 frame.

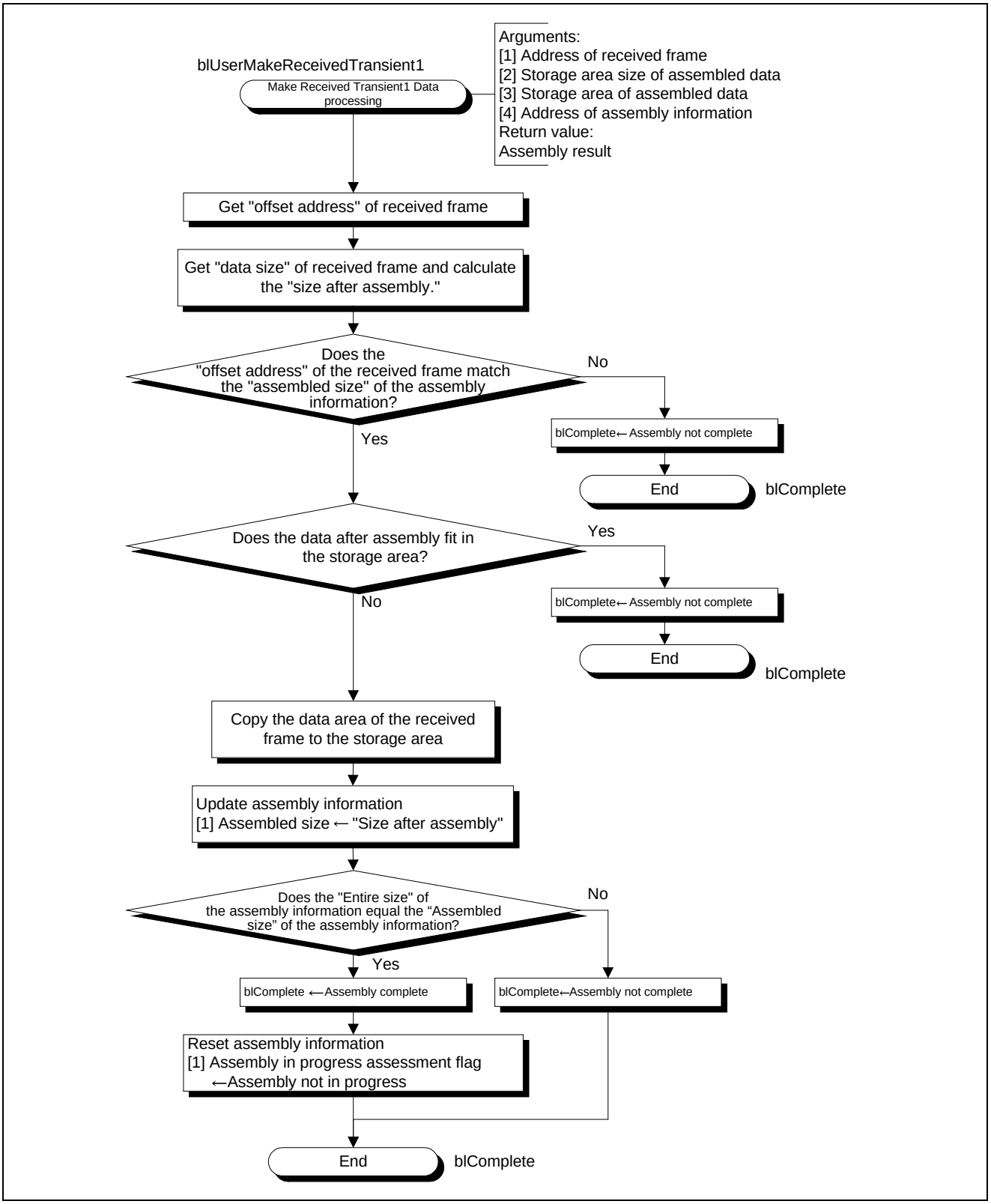


Figure 4.22 Make Received Transient1 Data Processing Flowchart

4.2.20 Received Node Information Distribution Frame processing

Received Node Information Distribution Frame processing receives a node information distribution frame and registers the information of each node.

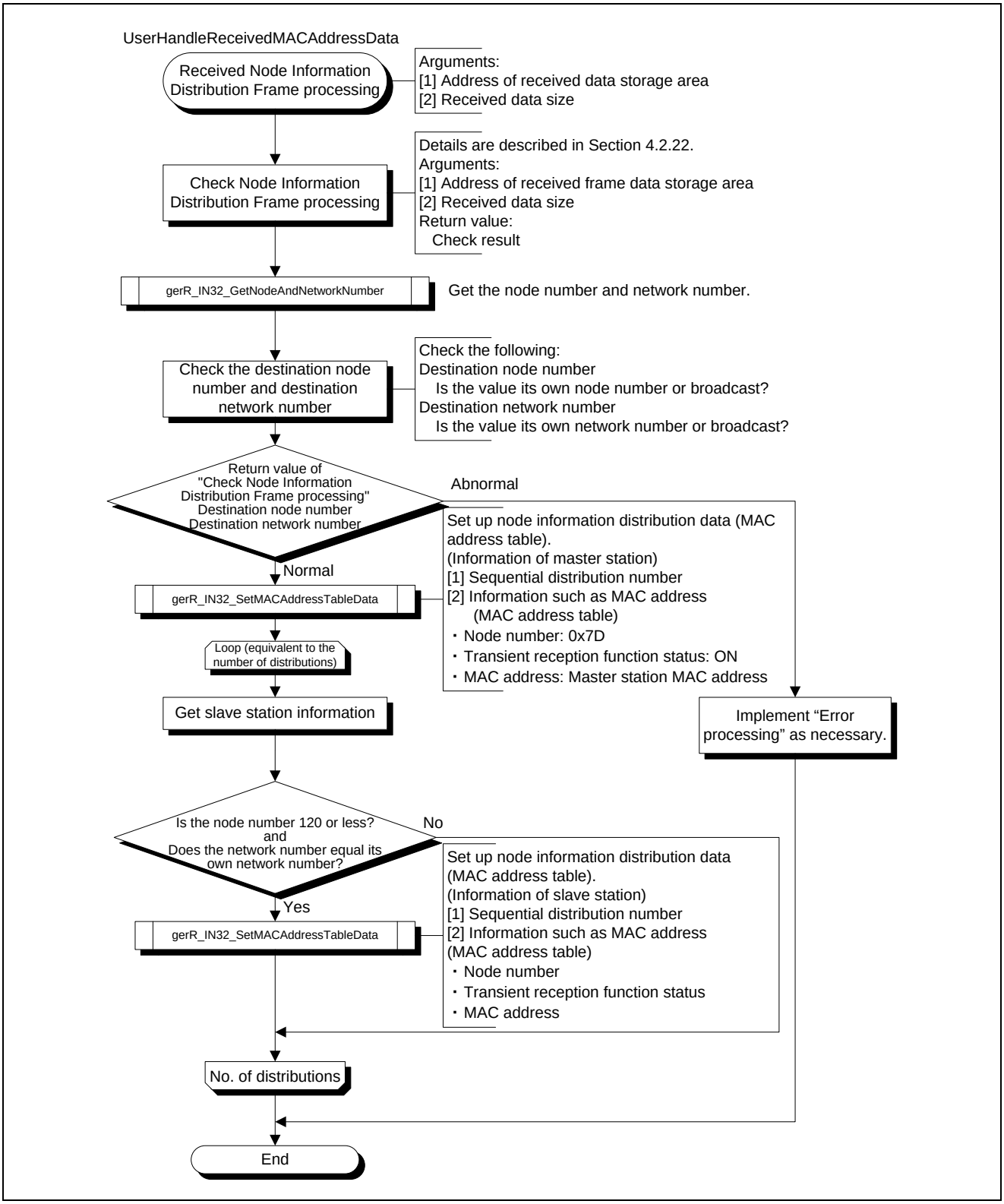


Figure 4.23 Received Node Information Distribution Frame Processing Flowchart

4.2.21 Check Node Information Distribution Frame processing

Check Node Information Distribution Frame processing checks the data inside the node information distribution frame.

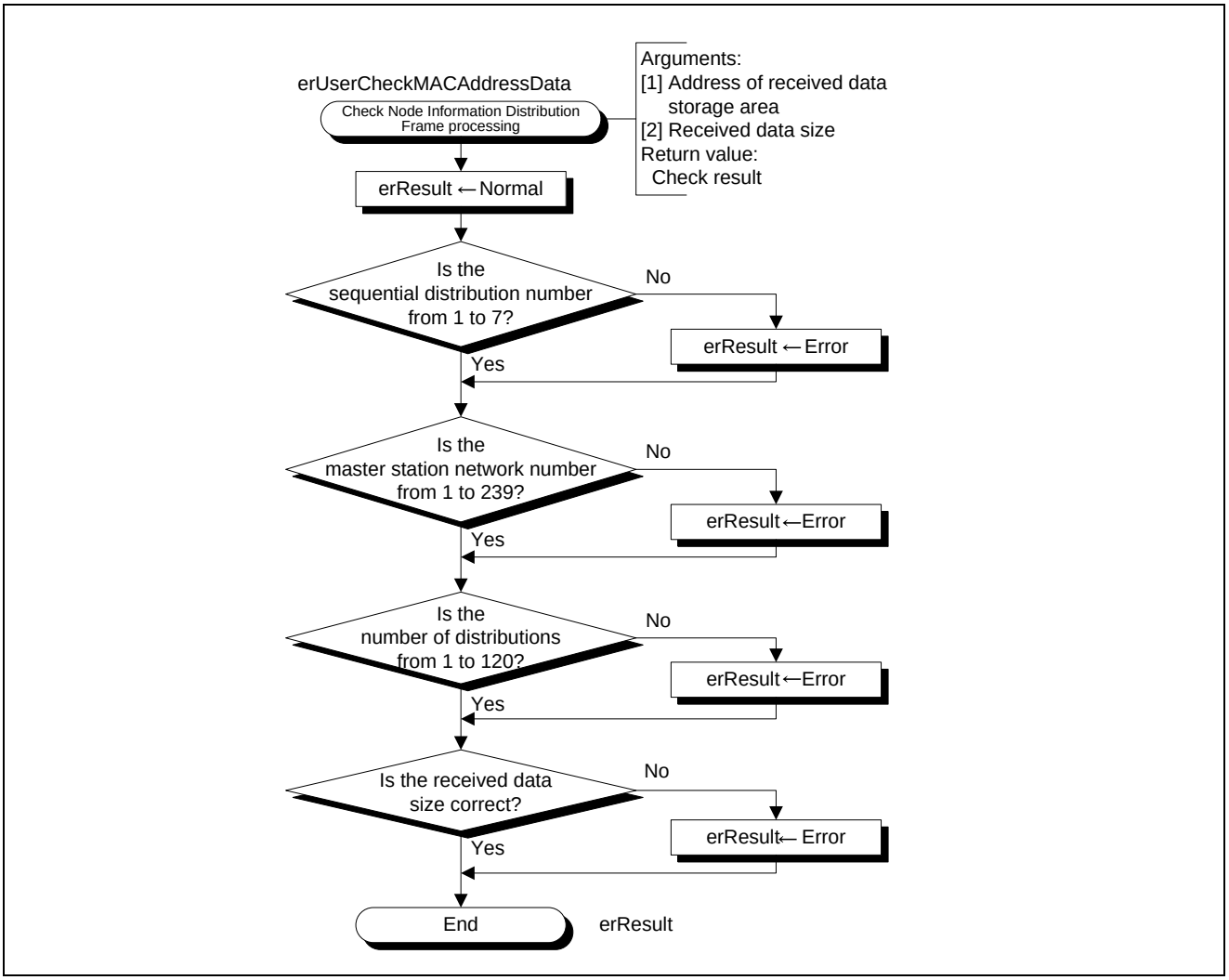


Figure 4.24 Check Node Information Distribution Frame Processing Flowchart

4.2.22 Received Statistical Information Request Frame processing

Received Statistical Information Request Frame processing is executed when a Get Statistical Information request frame is received.

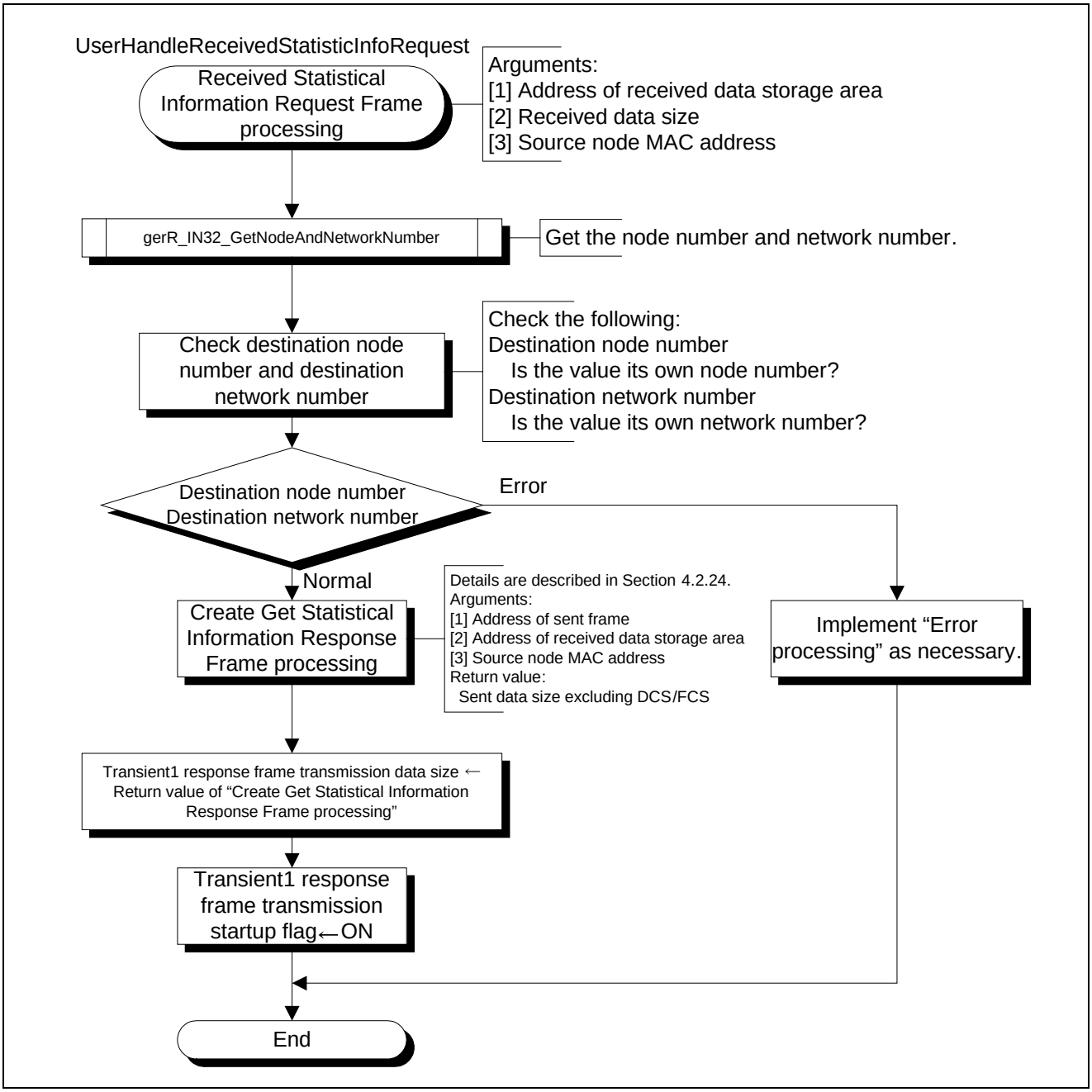


Figure 4.25 Received Statistical Information Request Frame Processing Flowchart

4.2.23 Create Get Statistical Information Response Frame processing

Create Get Statistical Information Response Frame processing creates a Get Statistical Information response frame.

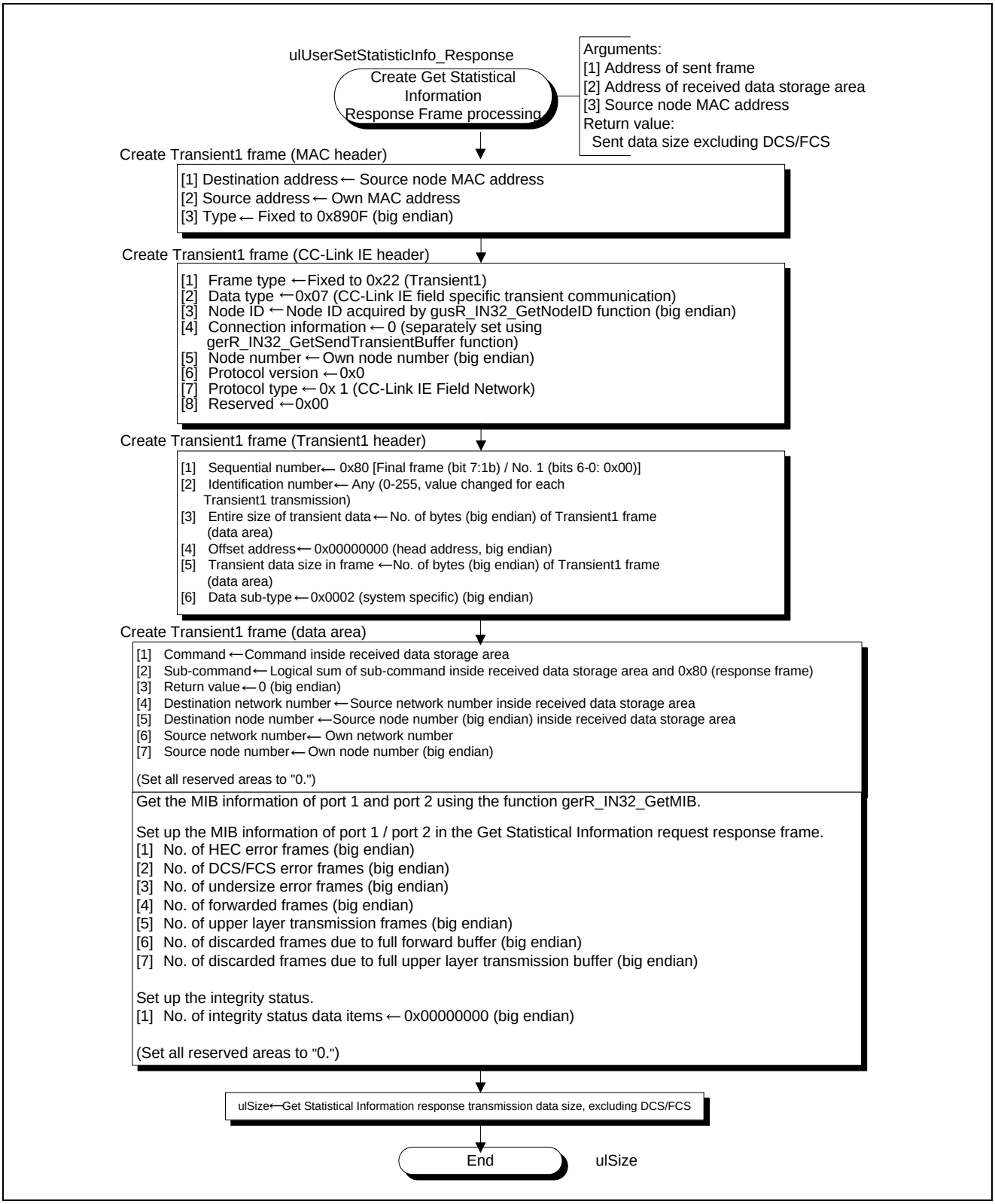


Figure 4.26 Create Get Statistical Information Response Frame Flowchart

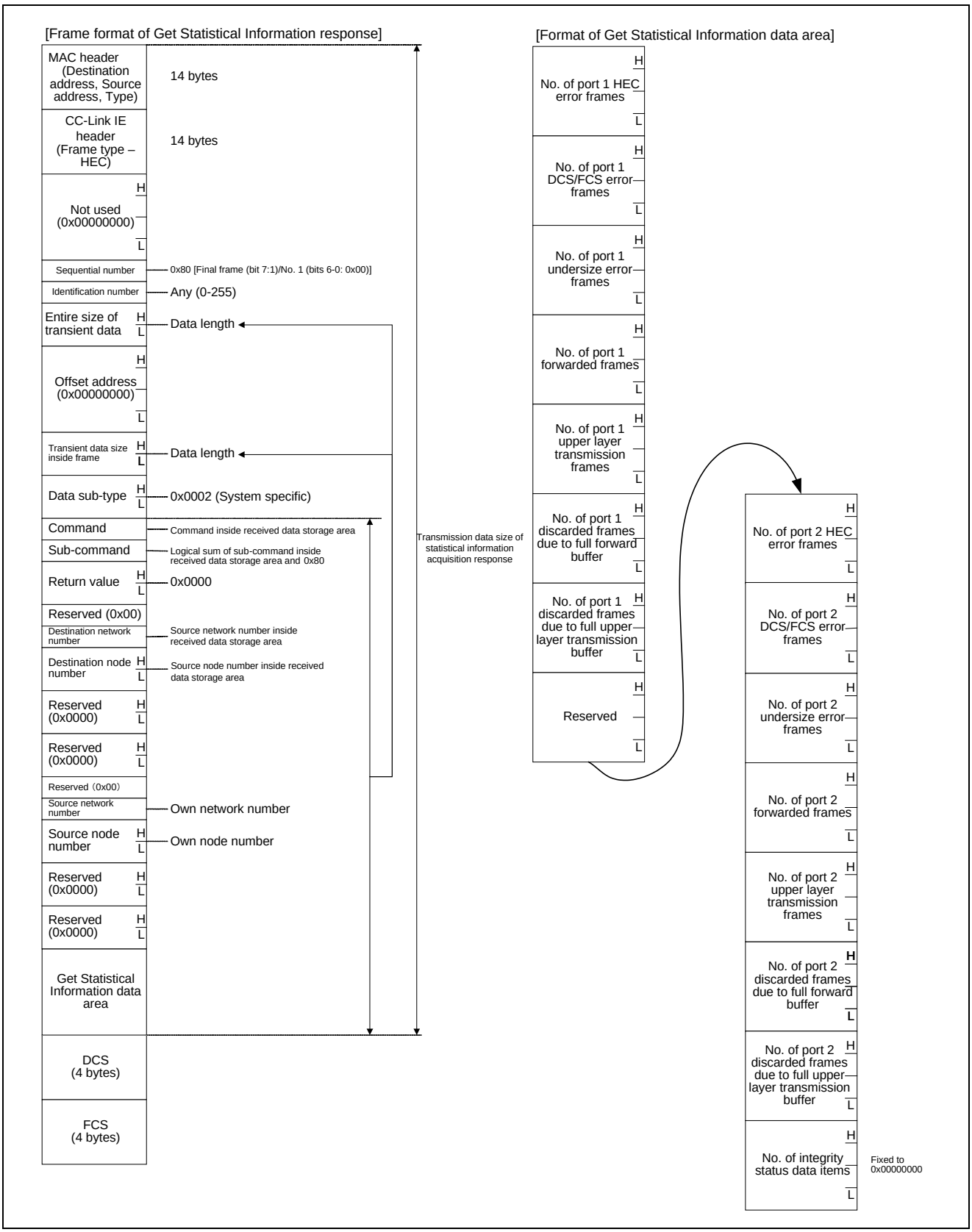


Figure 4.27 Frame Format of Get Statistical Information Response

4.2.24 Received Detailed Node Information Request Frame processing

Received Detailed Node Information Request Frame processing is executed when a Get Detailed Node Information request frame is received.

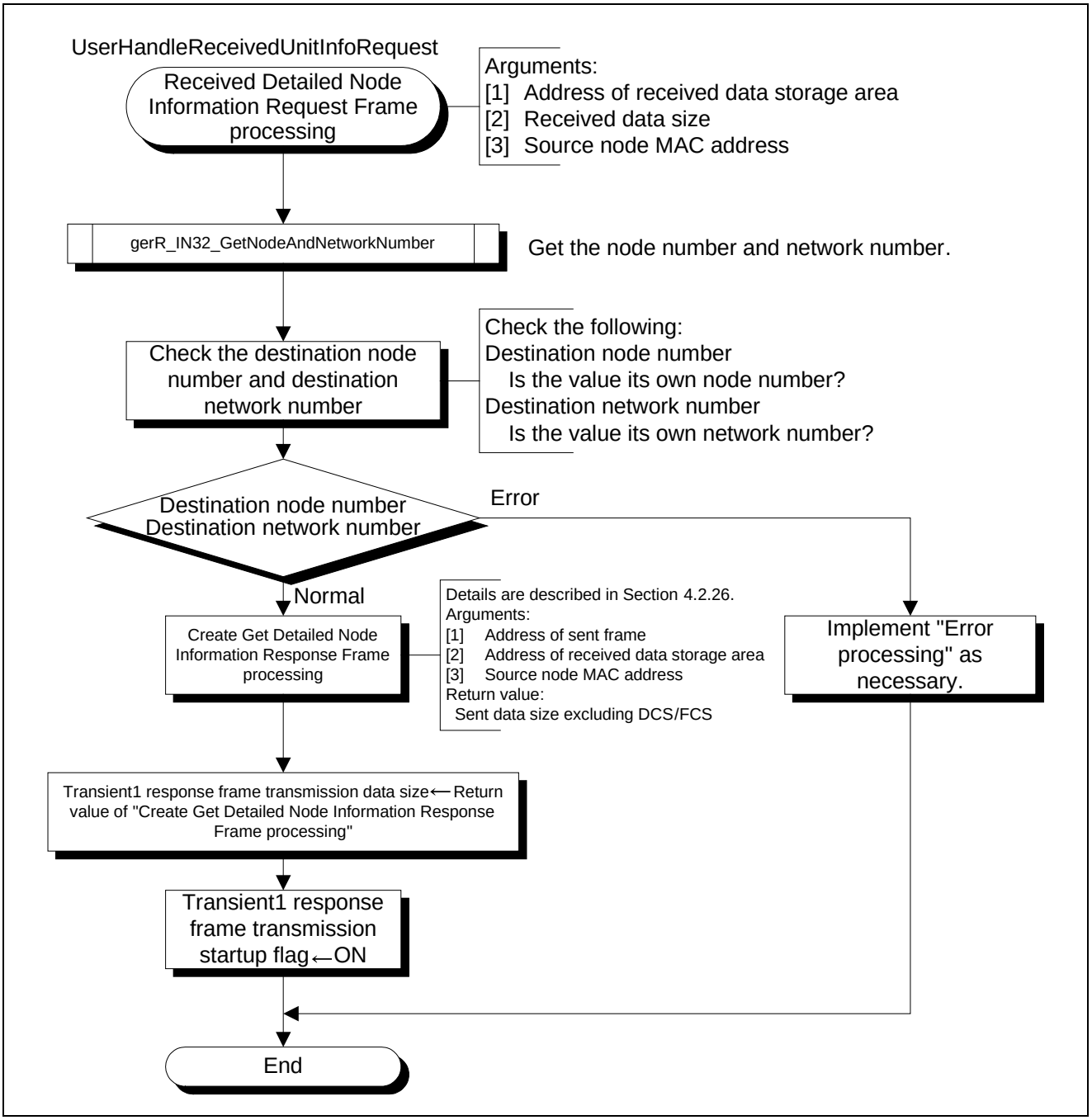


Figure 4.28 Received Detailed Node Information Request Frame Processing Flowchart

4.2.25 Create Get Detailed Node Information Response Frame processing

Create Get Detailed Node Information Response Frame processing creates a Get Detailed Node Information response frame.

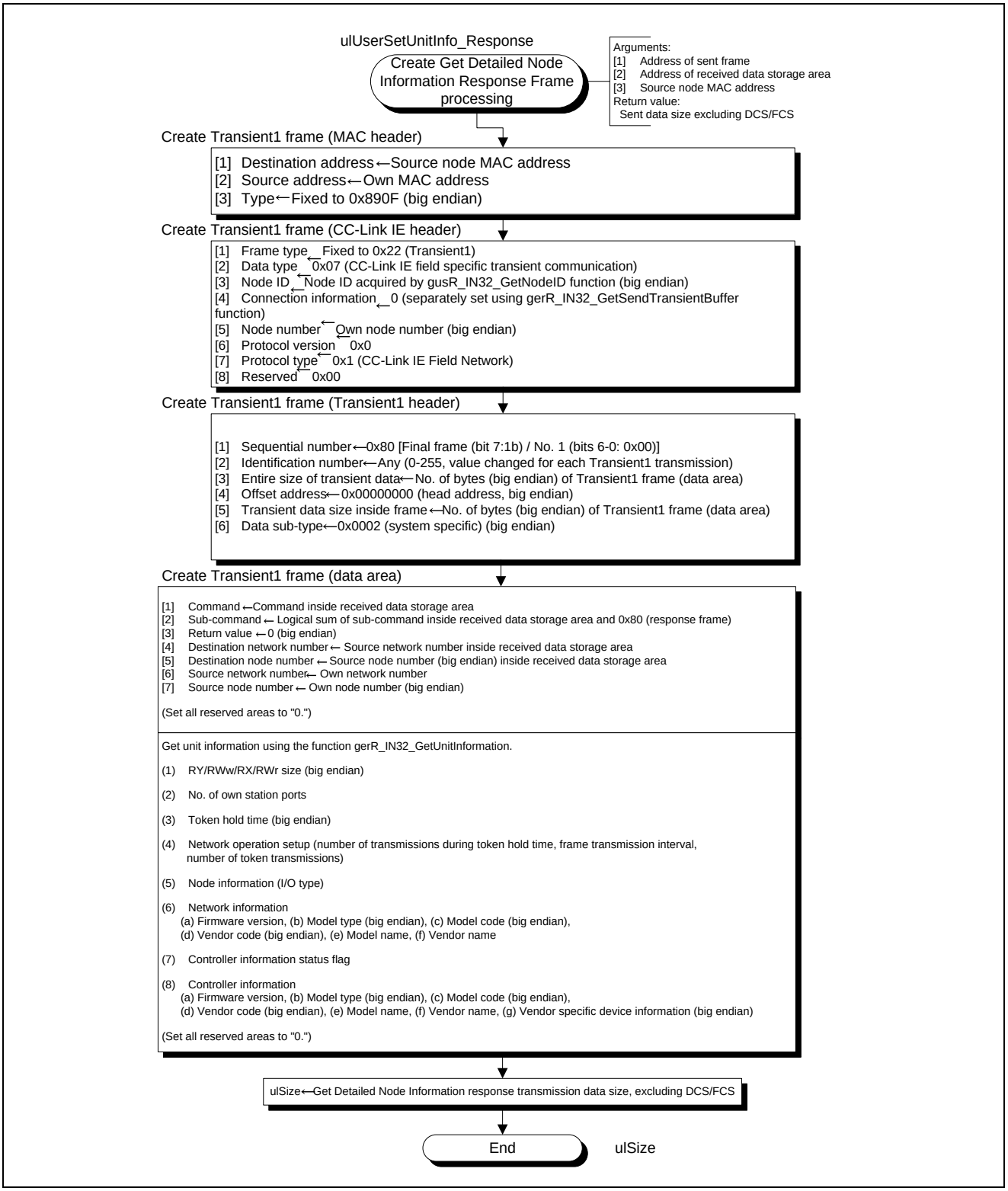


Figure 4.29 Create Get Detailed Node Information Response Frame Processing Flowchart

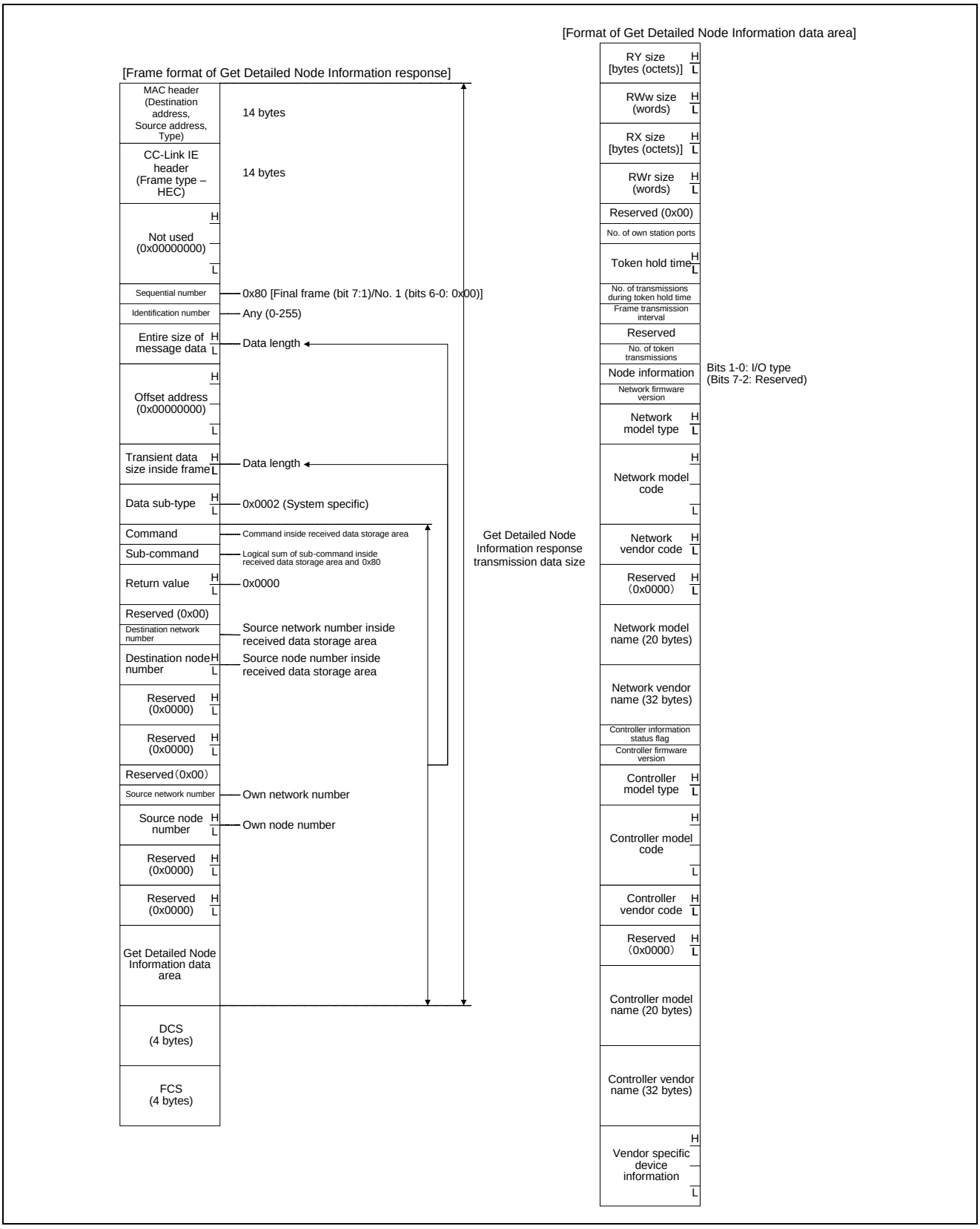


Figure 4.30 Frame Format of Get Detailed Node Information Response

4.2.26 Received Transient2 Data processing

Received Transient2 Data processing analyzes a received Transient2 frame and creates or receives a response frame in accordance with the analysis results.

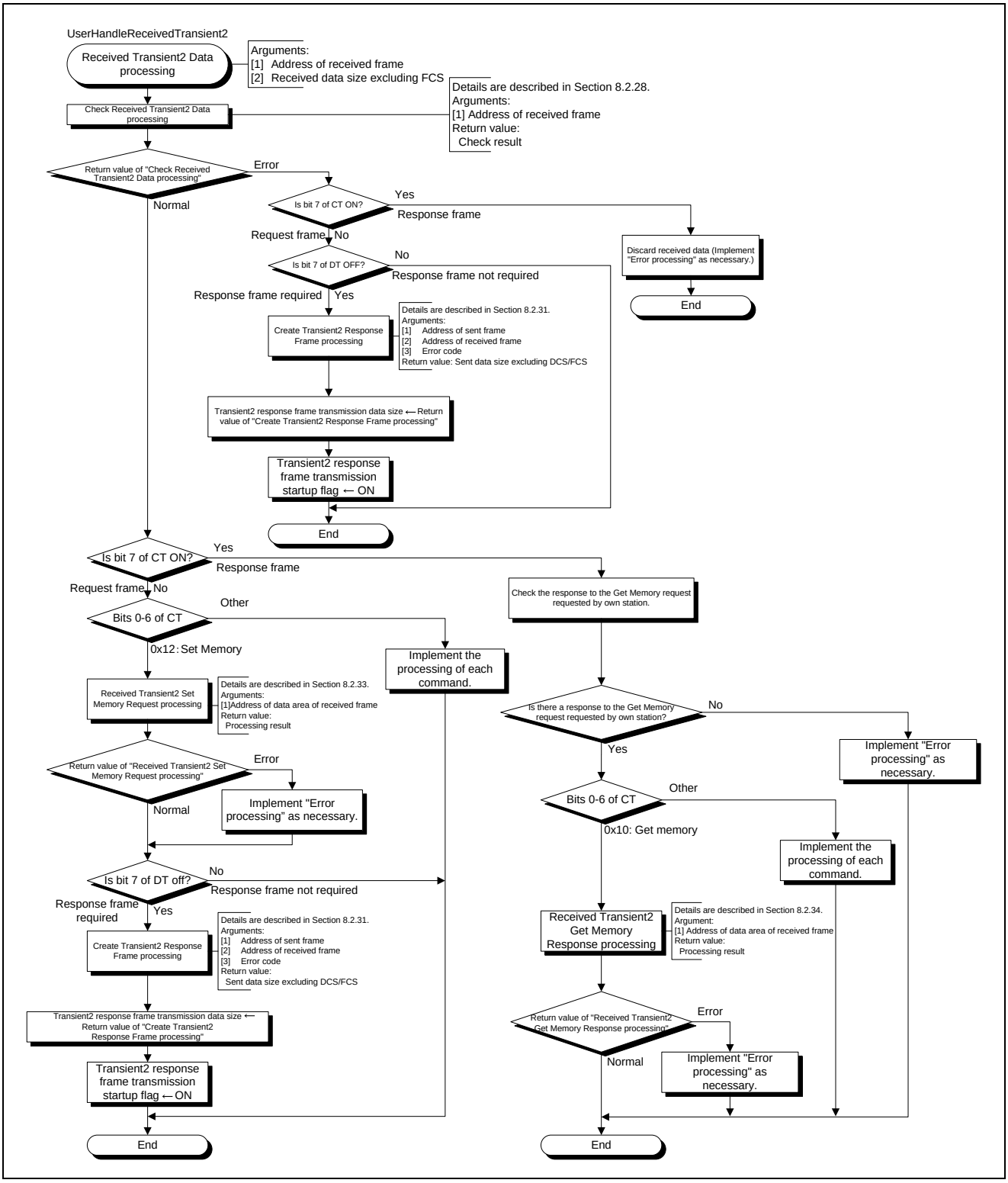


Figure 4.31 Received Transient2 Data Processing Flowchart

4.2.27 Check Received Transient2 Data processing

Check Received Transient2 Data processing checks if the received Transient2 frame is addressed to its own station, and checks the destination node number [DA (Destination Address) and DS (Destination Station No.)] and destination network number [DNA (Destination Network Address).

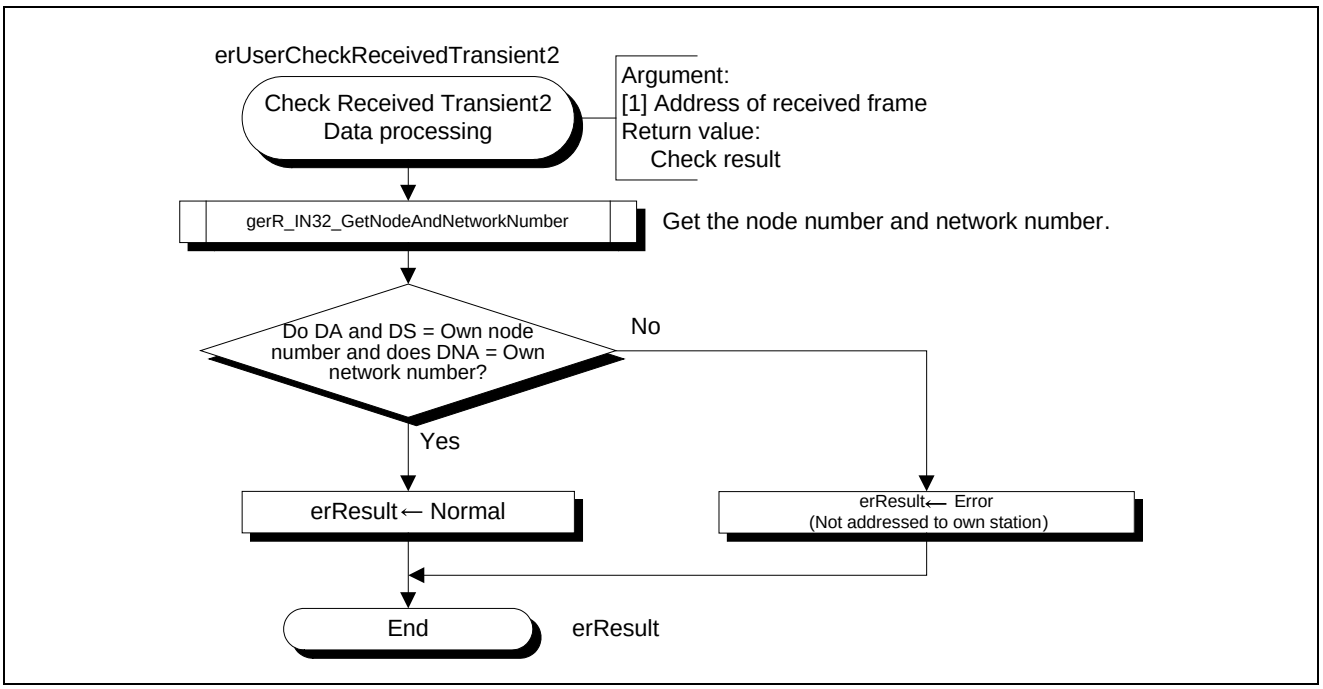


Figure 4.32 Check Received Transient2 Data Processing Flowchart

4.2.28 Received TransientAck Data processing

Received TransientAck Data processing processes a received TransientAck frame.

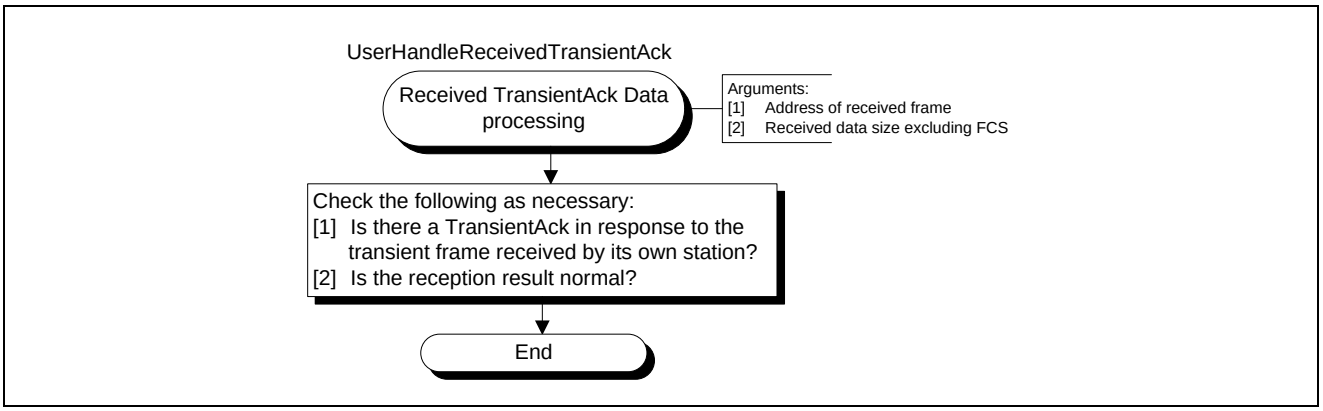


Figure 4.33 Received TransientAck Data Processing Flowchart

4.2.29 Create TransientAck Frame processing

Create TransientAck Frame processing creates a TransientAck frame.

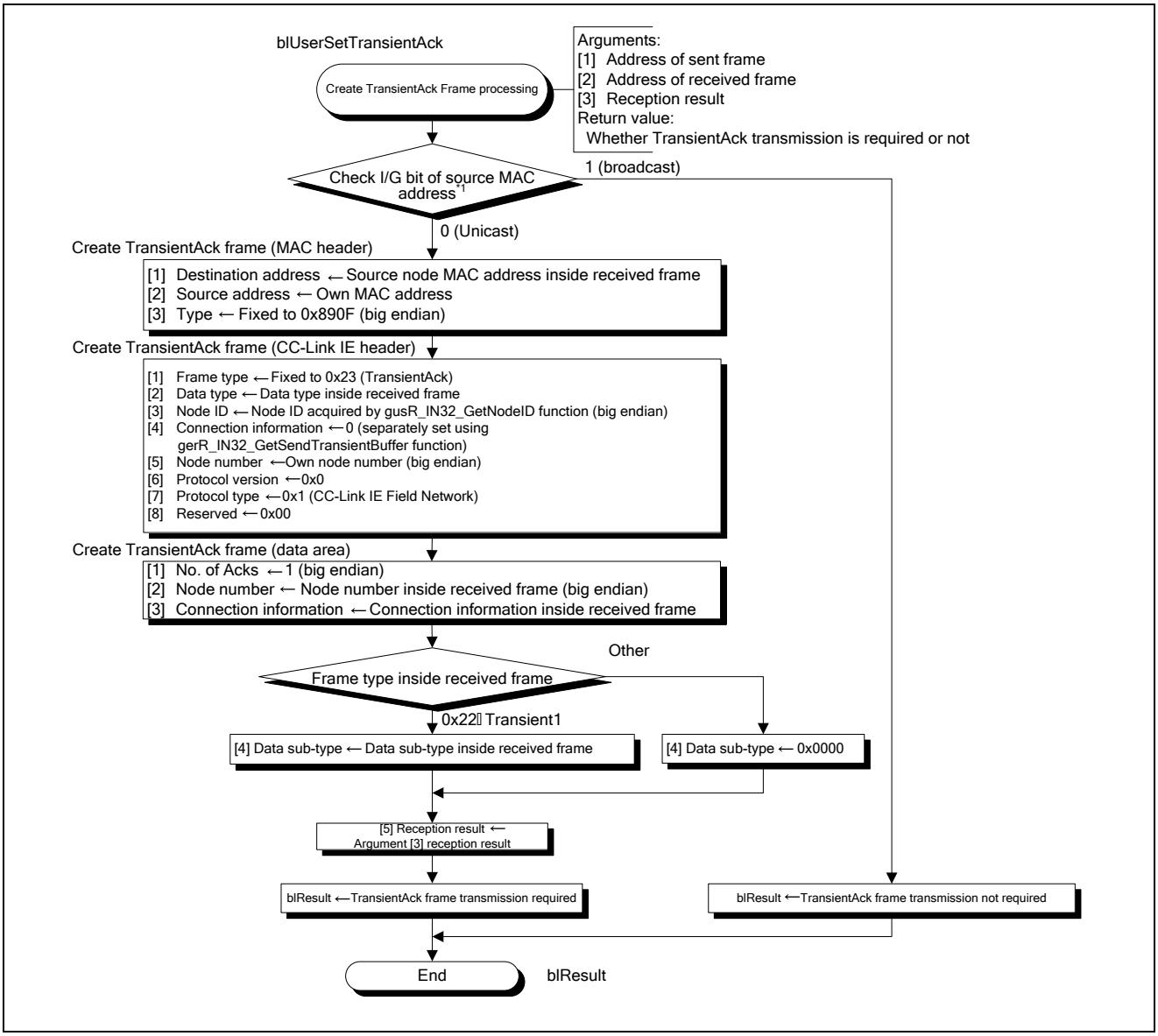


Figure 4.34 Create TransientAck Frame Processing Flowchart

*1: The I/G bit refers to the lowest bit of the first byte (octet) of the MAC address.

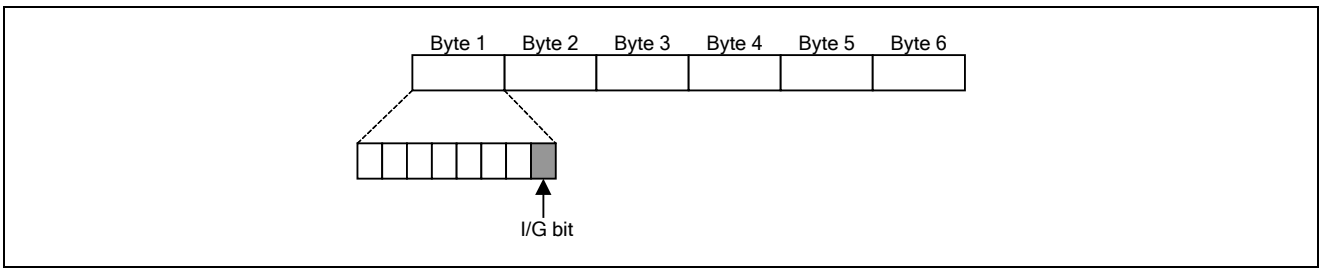


Figure 4.35 I/G Bit

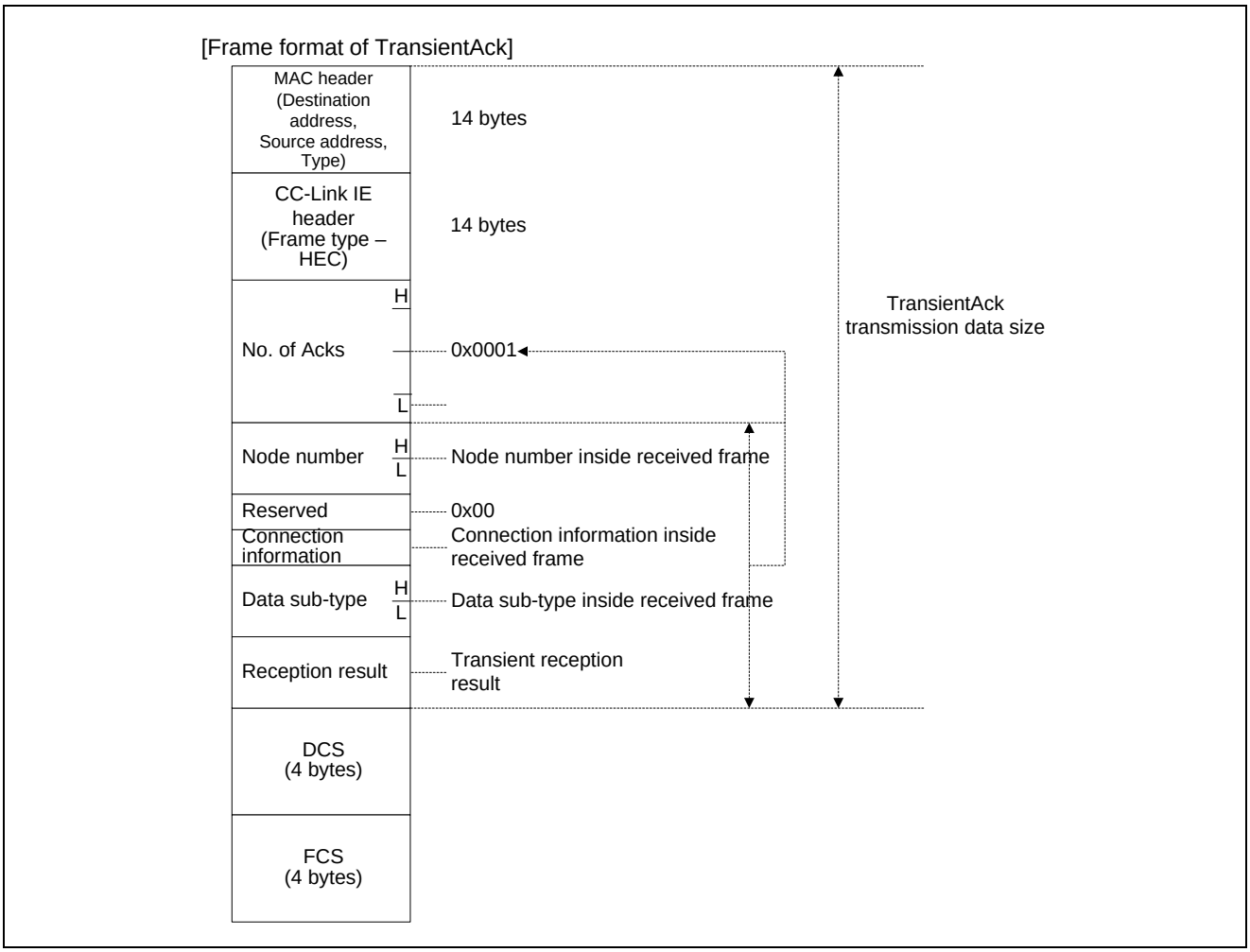


Figure 4.36 Frame Format of TransientAck

4.2.30 Create Transient2 Response Frame processing

Create Transient2 Response Frame processing creates a Transient2 response frame.

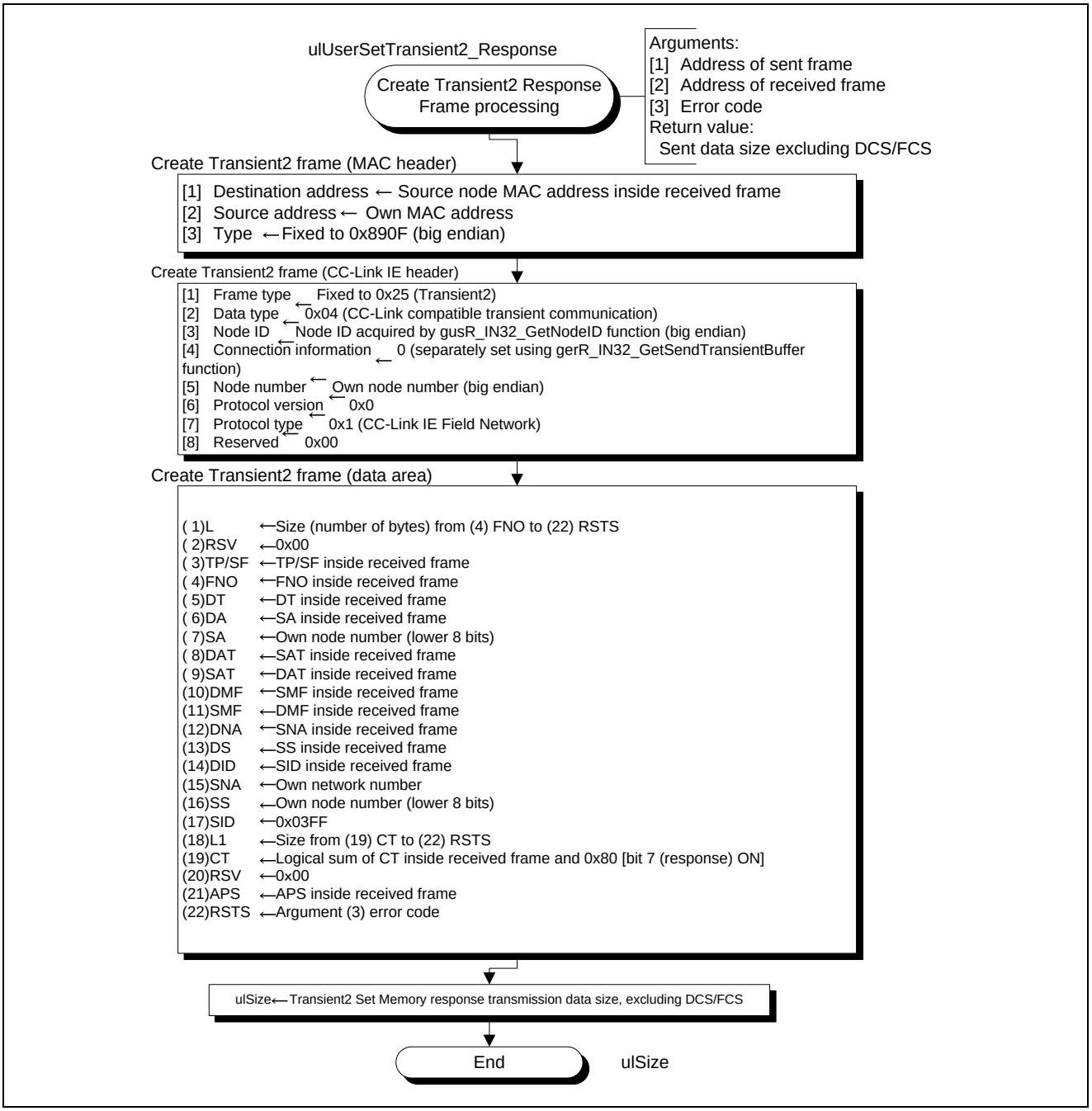


Figure 4.37 Create Transient2 Response Frame Processing Flowchart

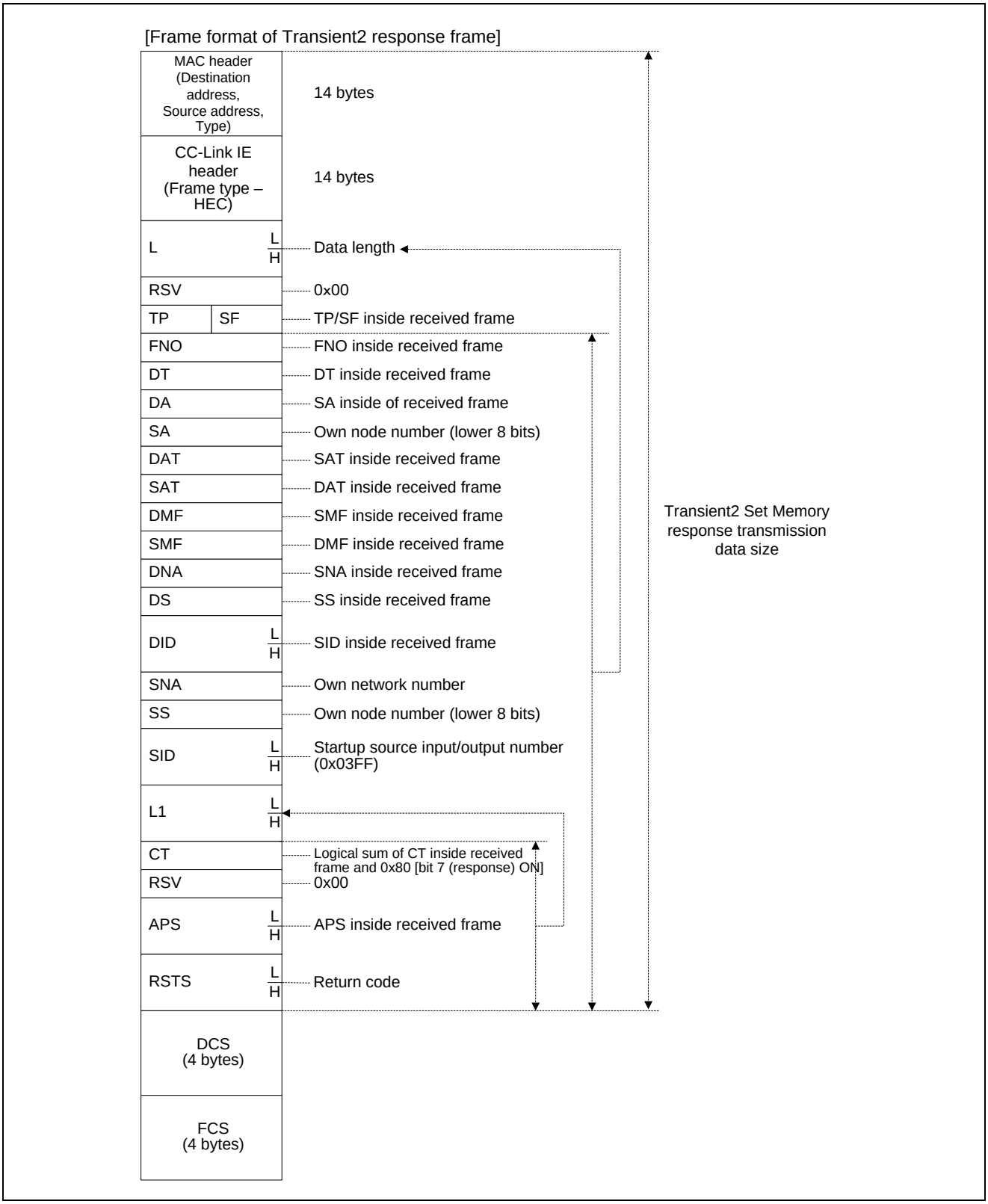


Figure 4.38 Frame Format of Transient2 Response

4.2.31 Create Transient2 Get Memory Request Frame processing

Create Transient2 Get Memory Request Frame processing creates a Transient2 Get Memory request frame.

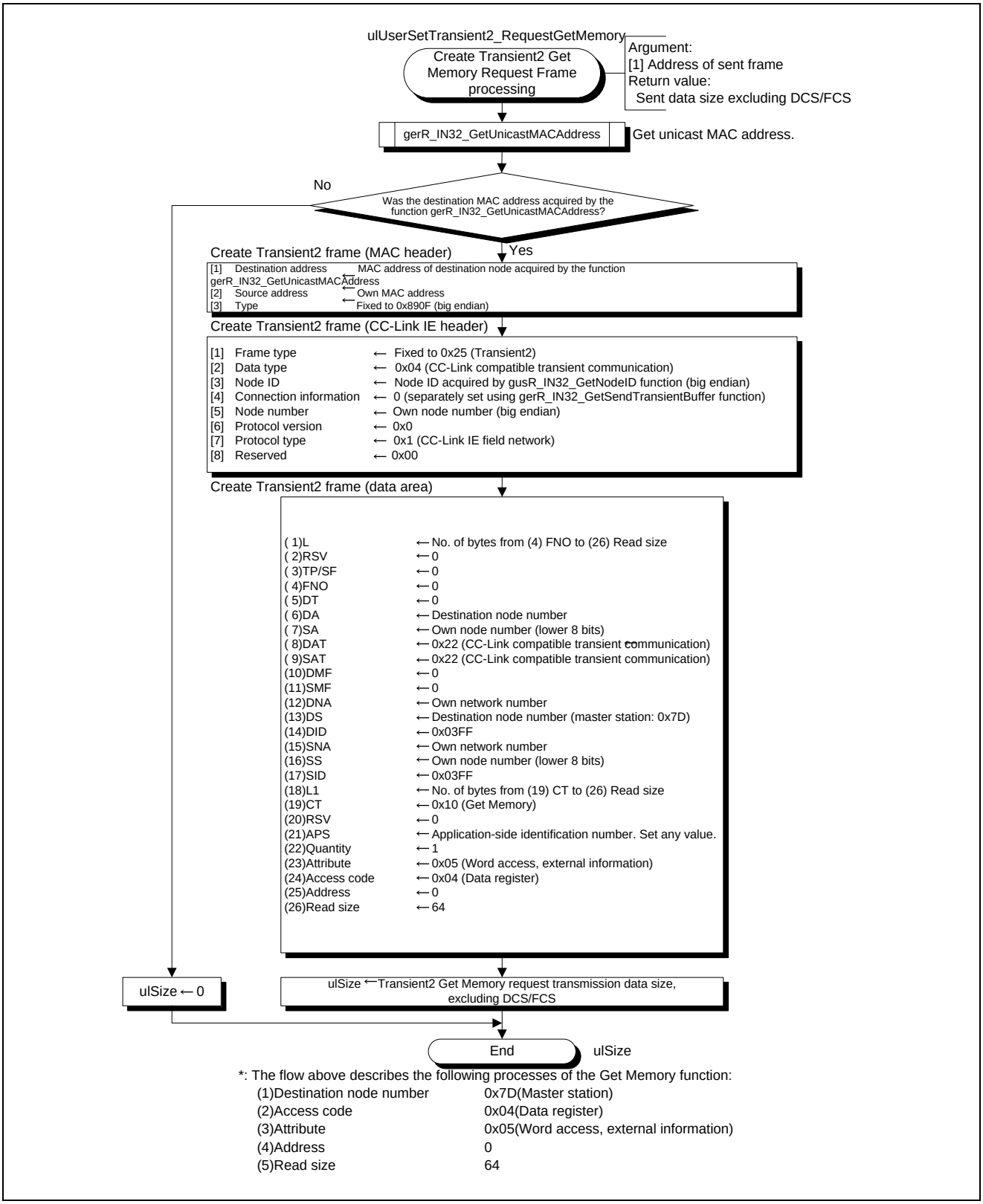


Figure 4.39 Create Transient2 Get Memory Request Frame Processing Flowchart

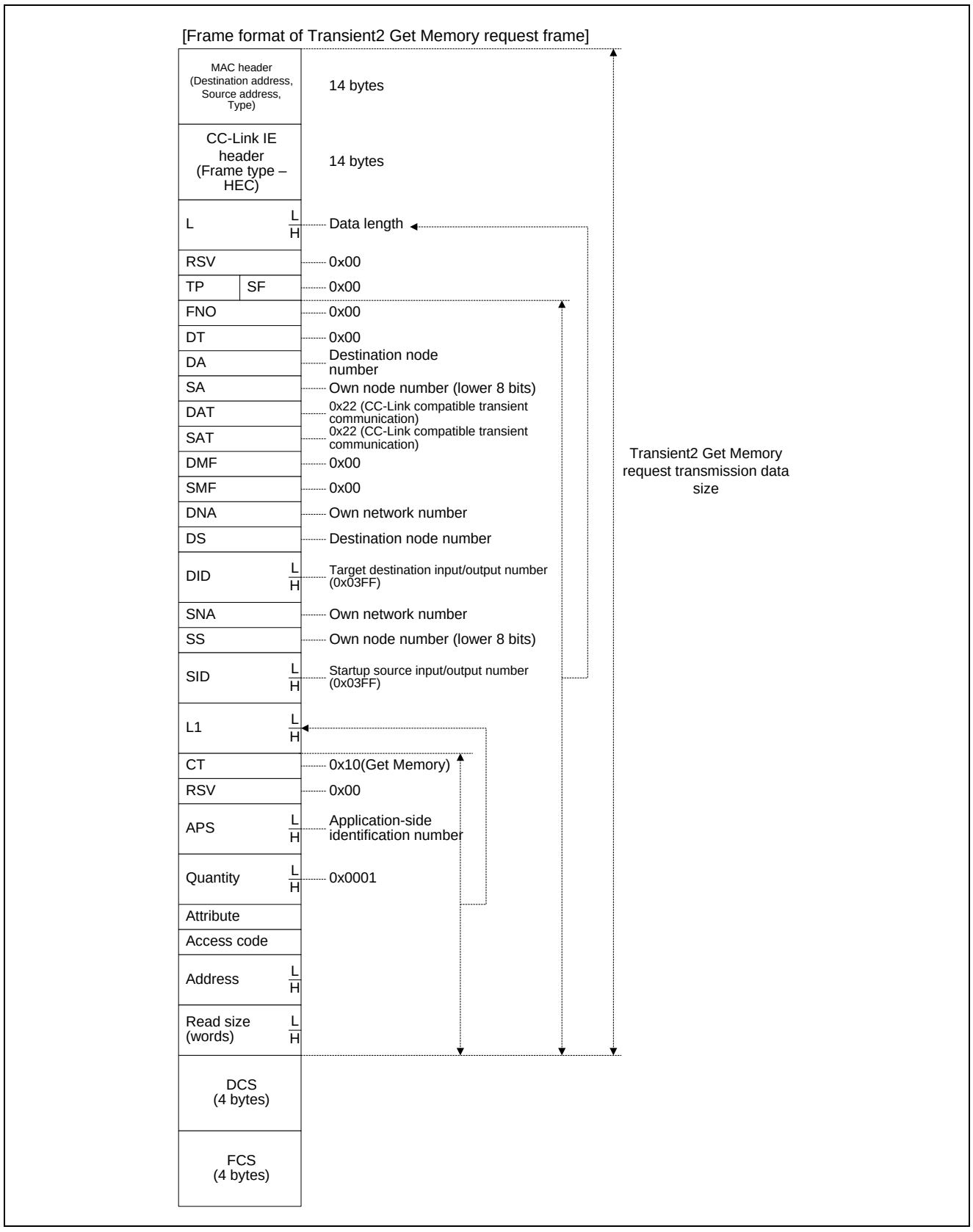


Figure 4.40 Frame Format of Transient2 Get Memory Request

4.2.32 Received Transient2 Set Memory Request processing

Received Transient2 Set Memory Request processing is performed when a Transient2 Set Memory request frame is received.

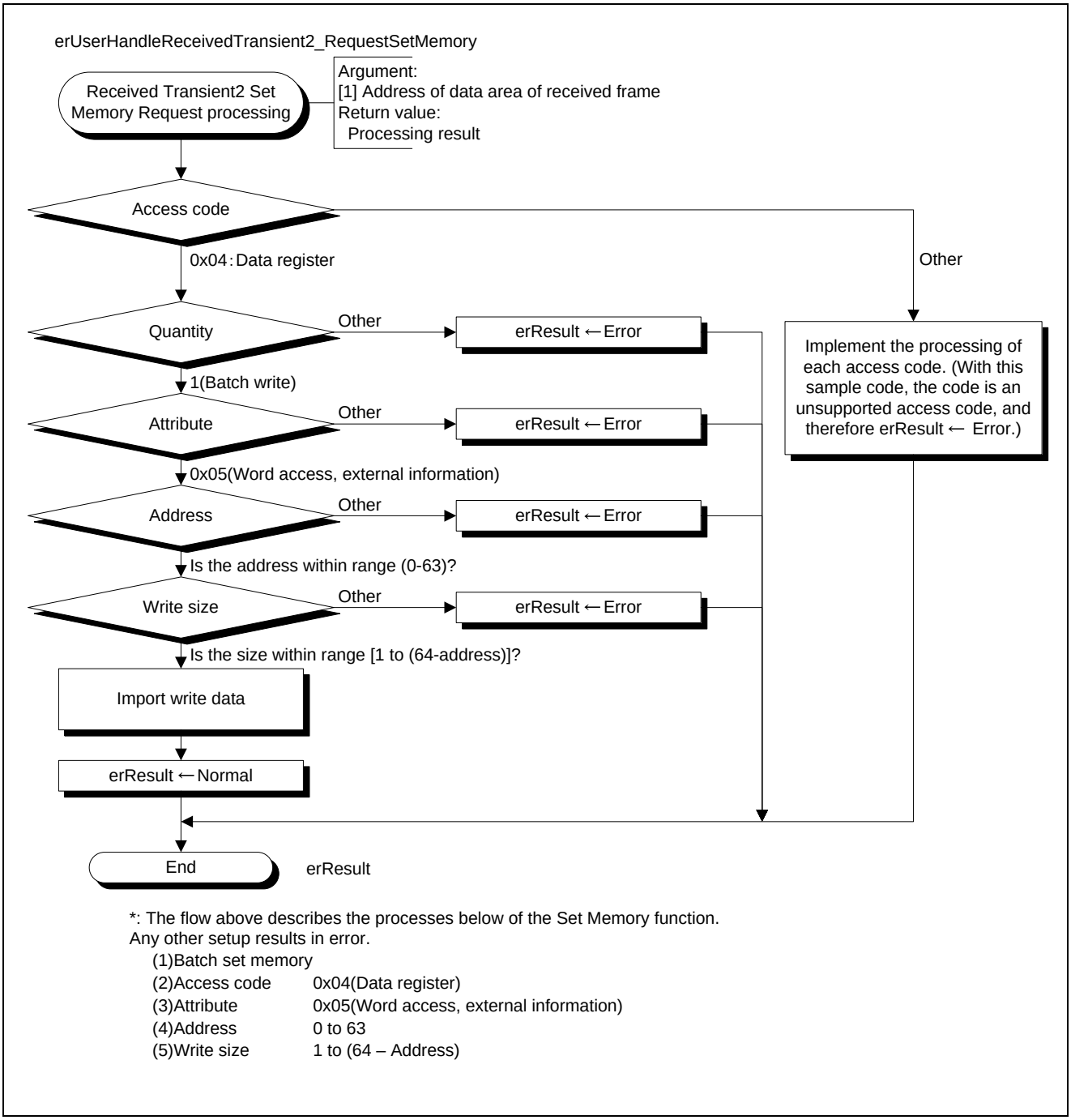


Figure 4.41 Received Transient2 Set Memory Request Processing Flowchart

4.2.33 Received Transient2 Get Memory Response processing

Received Transient2 Get Memory Response processing is performed when a Transient2 Get Memory response frame is received.

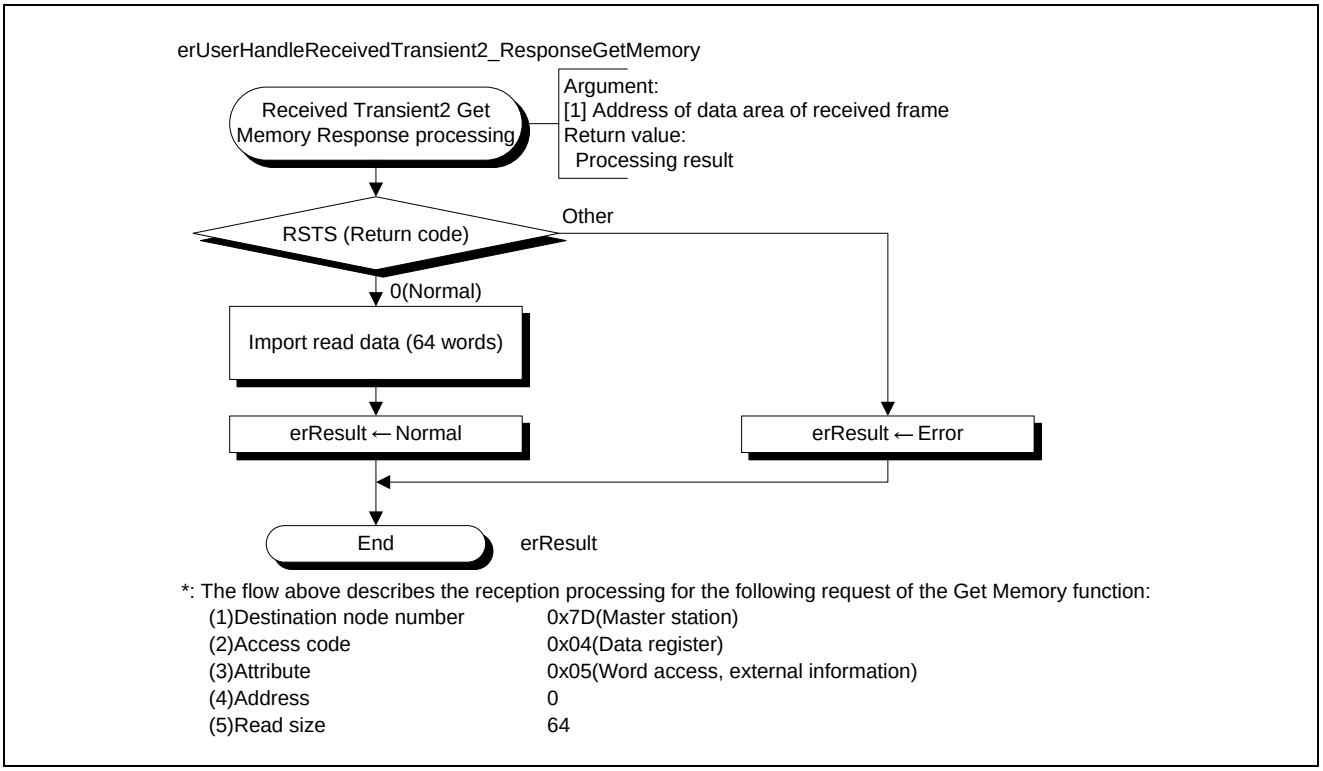


Figure 4.42 Received Transient2 Get Memory Response Processing Flowchart

4.2.34 Hardware test (IEEE 802.3ab compliance test)

The hardware test performs an IEEE 802.3ab compliance test.

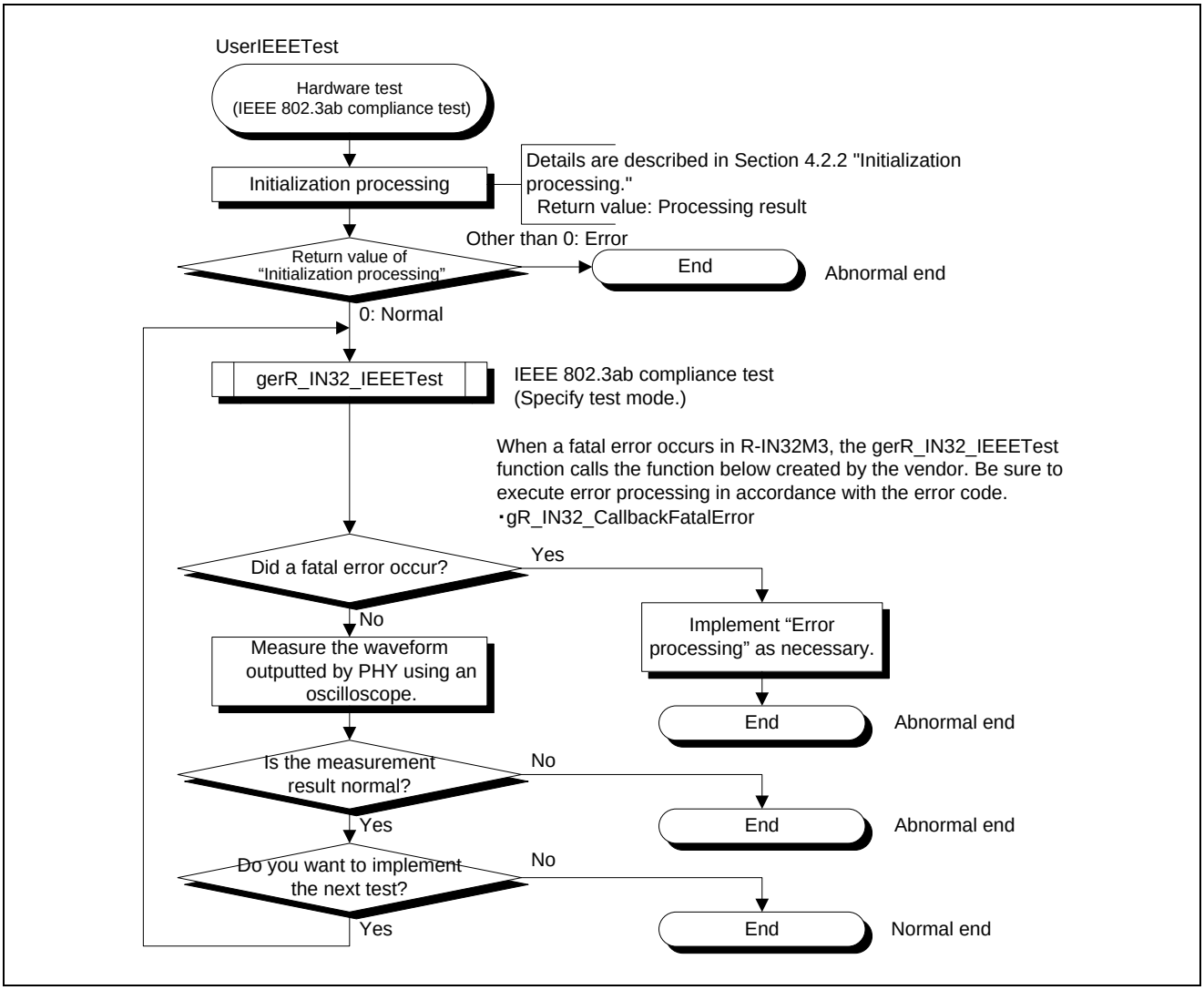


Figure 4.43 Hardware Test (IEEE 802.3ab Compliance Test) Flowchart

Precaution

To implement the tests described in the CC-Link IE Field Network Intelligent Device Station Conformance Test Specification (BAP-C0401-037), this function needs to be implemented.

4.2.35 Hardware test (loop-back communication test)

The loop-back communication test involves the test items below. Implement the test in accordance with the precautions of each test item.

Table 4.6 Test Item Precautions

No.	Test Item	Precaution
1	Internal loop-back communication test	When the internal loop-back communication test is implemented, the PHY link is disconnected. It takes 3 or more seconds for the PHY link to go up again. Be sure to execute reset processing so that WDT does not time out. (When you want to use an internal R-IN32M3-CL WDT, start the function gerR_IN32_ResetWDT.)
2	External loop-back communication test	Connect port 1 and port 2 using an Ethernet cable.

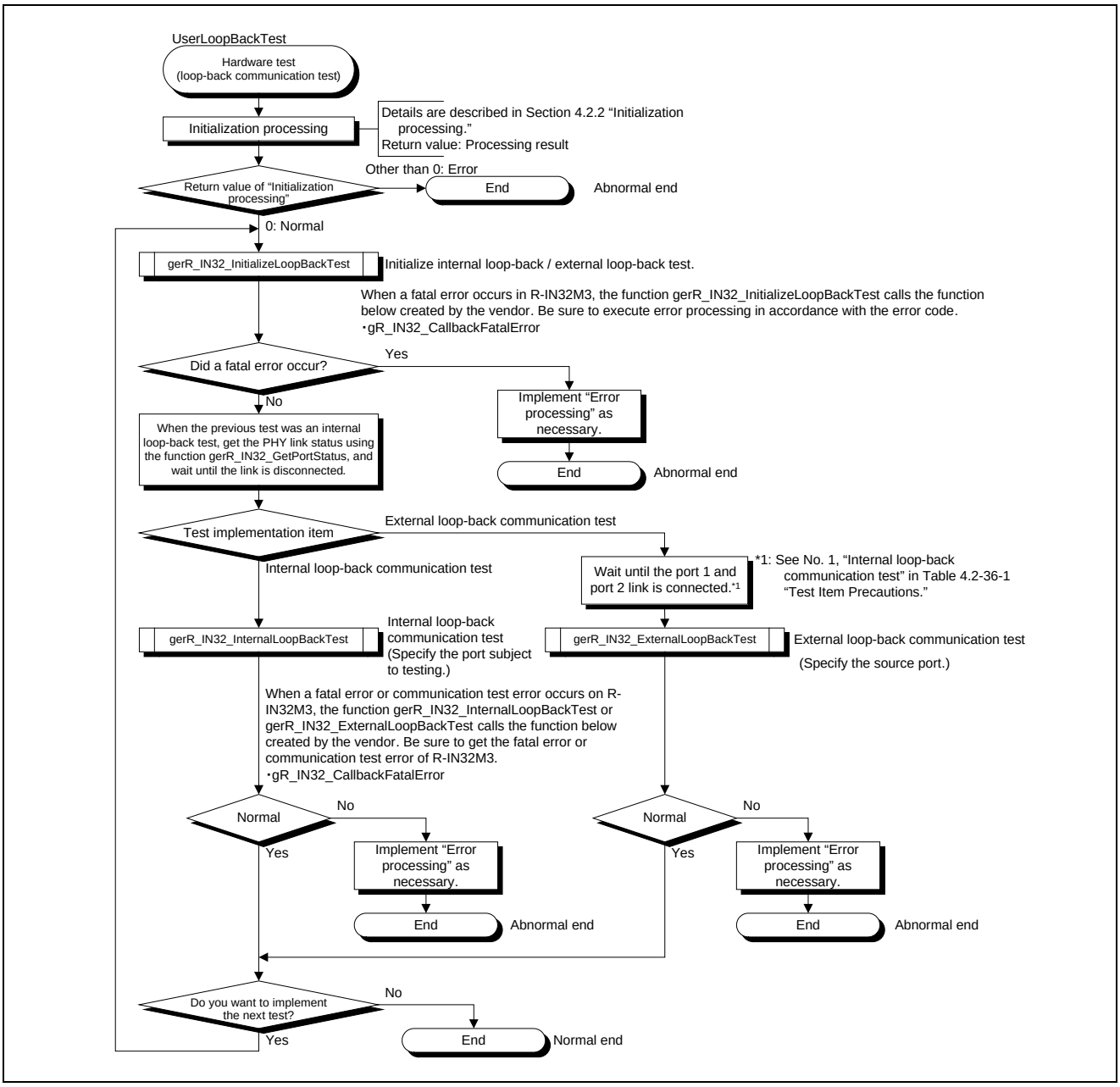


Figure 4.44 Hardware Test (Loop-back Communication Test) Flowchart

You can troubleshoot ports that you suspect to have failed based on the hardware test (communication test) result.

Table 4.7 Troubleshooting Based on Hardware Test

Target Port Resulting in R_IN32_ERR by Internal Loop-Back Communication Test (gerR_IN32_InternalLoopBackTest)	Source Port Resulting in R_IN32_ERR by External Loop-Back Communication Test (gerR_IN32_ExternalLoopBackTest)	Port Suspected of Failure
Port 1	Port 1	Port 1 XMIT
	Port 2	Port 1 RECV
Port 2	Port 1	Port 2 RECV
	Port 2	Port 2 XMIT

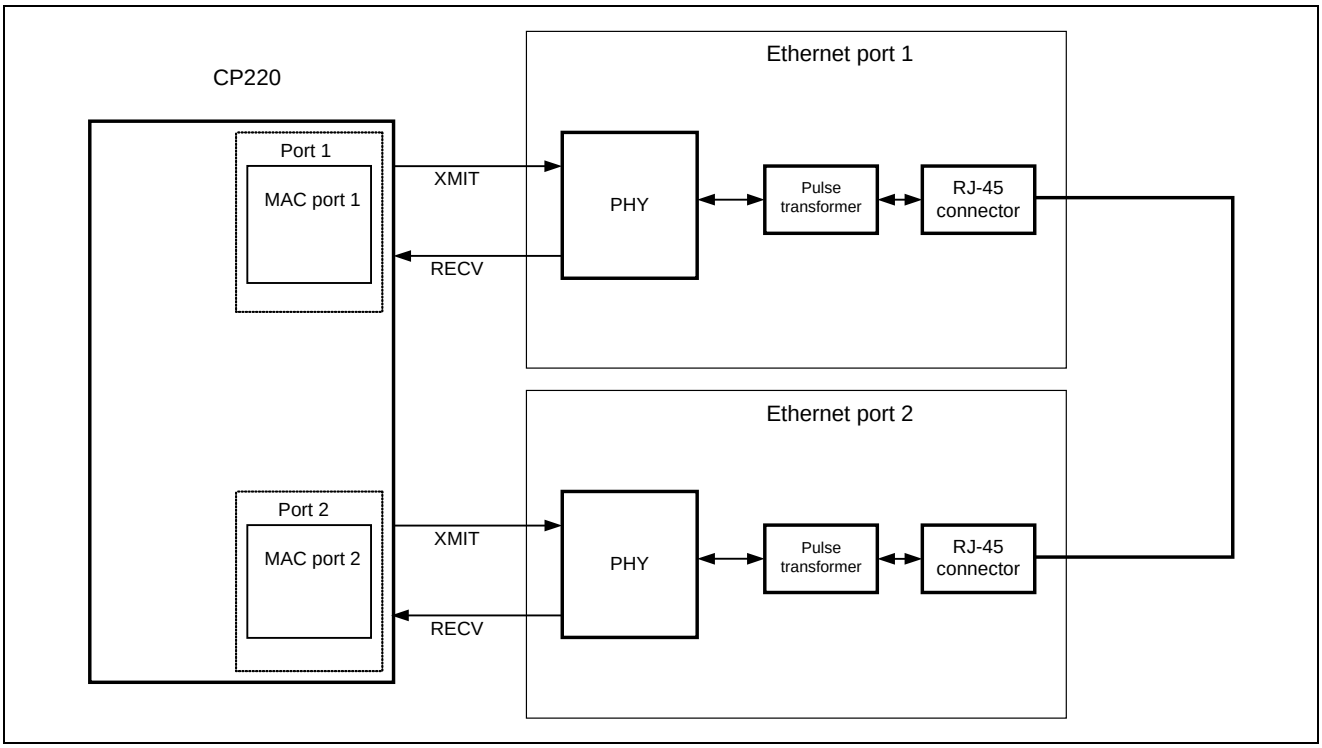


Figure 4.45 Port Schematic View

4.3 Sample Code File List

The sample code provided with this reference manual has been verified as free of compilation errors based on "GCC (GNU C Compiler) Version 4.3.4."

The sample code is neither operating system nor MPU dependent. Customize the code in accordance with the vendor environment.

4.3.1 Folder configuration

The following shows the folder configuration of the sample code.

```
                                                Contents
root -+- Japanese-+- sample   -+- include  ... user's code header files
       |          |             +- src      ... user's code files and callback code files
       |          |             +- obj      ... makefile (for building the user's application)
       |          |
       |          +- driver     -+- include  ... R-IN32M3 driver code header files
       |                         +- src      ... R-IN32M3 driver code files
       |                         +- obj      ... makefile
```

4.3.2 File list

The following presents the sample code file list.

```
root
 └─version.txt    ..... Version information


root ─ Japanese
    |    readme_e.txt ..... Help file (English version)
    |
    ├─driver
    |   ├─include
    |   |     R-IN32M3Driver.h          R-IN32M3 driver header
    |   |     R-IN32M3Function.h        Header file change to target-dependent function group
    |   |                               for R-IN32M3 driver (See Section 4.6.1.)
    |   |     R-IN32M3Types.h           Definitions described in Section (1), (2), and (3) of Section 4.5.
    |   |
    |   ├─obj
    |   |     makefile                  A file that requires customization by vendor.
    |   |
    |   └─src
    |         R_IN32.h
    |         R-IN32M3.h
    |         R-IN32M3_0.h
    |         R-IN32M3_1.h
    |         R-IN32M3_2.h
    |         R-IN32M3_3.h
    |         R_IN32C.h
    |         R_IN32C_R_IN32DInterface.c
    |         R_IN32C_Cyclic.c
    |         R_IN32C_Data.c
    |         R_IN32C_Indication.c
    |         R_IN32C_Init.c
    |         R_IN32C_I.h
    |         R_IN32C_Library.c
    |         R_IN32C_MainState.c
    |         R_IN32C_PortState.c
    |         R_IN32C_Time.c
    |         R_IN32D.h
```

R_IN32D_cyc.c	
R_IN32D_cyc_l.h	
R_IN32D_ihnd.c	
R_IN32D_ini.c	
R_IN32D_intr.c	
R_IN32D_intr_l.h	
R_IN32D_led.c	
R_IN32D_phy.c	
R_IN32D_phy_l.h	
R_IN32D_RcvCnt.c	
R_IN32D_RcvCnt_l.h	
R_IN32D_RcvPrm.c	
R_IN32D_RcvPrm_l.h	
R_IN32D_reg.c	
R_IN32D_reg_l.h	
R_IN32D_sub.c	
R_IN32D_sub_l.h	
R_IN32D_tran.c	
R_IN32D_tran_l.h	
R_IN32R.c	Function used by driver of target-dependent function group for R-IN32M3 driver (See Section 4.6.2.)
R_IN32R.h	Declaration of function used by driver of target-dependent function group for R-IN32M3 driver (See Section 4.6.2.)
R_IN32S.c	
R_IN32S.h	
R_IN32T.h	
R_IN32T_ASIC.c	
R_IN32T_ASIC.h	
R_IN32T_Cmu.h	
R_IN32T_CmuNCycRcv.c	
R_IN32T_CmuOutLpBak.c	
R_IN32T_CmuSub.h	
R_IN32T_CmuSub3.c	
R_IN32T_Com.c	
R_IN32T_Com.h	
R_IN32T_Data.c	
R_IN32T_Data.h	
R_IN32T_FrmForm.h	
R_IN32T_MACIP.c	
R_IN32T_MACIP.h	
R_IN32T_RegChk.c	
R_IN32T_RegChk.h	
R_IN32T_RING.c	
R_IN32T_RING.h	
R_IN32T_TxFrame.c	
R_IN32T_TxFrame.h	
R_IN32U.h	
R_IN32U_Init.c	
R_IN32_Frame.h	
R_IN32_Interface.c	Interface function group for R-IN32M3 driver (See Section 4.4.)
sample	
include	
R-IN32M3Callback.h	Declaration of callback function group for R-IN32M3 driver (See Section 4.7.)
obj	
makefile	A file that requires customization by vendor.
src	
R-IN32M3_Callback.c	Call-back function group for R-IN32M3 driver (See Section 4.7.)
R-IN32M3_HWTest.c	A file that needs to be created by vendor.
R-IN32M3_HWTest.h	A file that needs to be created by vendor.
R-IN32M3_sample.c	A file that needs to be created by vendor.
R-IN32M3_sample.h	A file that needs to be created by vendor.
R-IN32M3_Transient.c	A file that needs to be created by vendor.
R-IN32M3_Transient.h	A file that needs to be created by vendor.

4.4 Interface Function List for R-IN32M3-CL Driver

The following lists the interface functions of the R-IN32M3-CL driver.

Table 4.8 R-IN32M3-CL Driver Interface Function List

Function Category	Function Name	Function Type	Overview
Initial setup	gulR_IN32_GetResetStatus	ULONG	Acquires the reset status.
	gerR_IN32_Initialize	ERRCODE	Initializes R-IN32M3-CL.
	gerR_IN32_SetNodeAndNetworkNumber	ERRCODE	Sets the node number and network number.
	gerR_IN32_Start	ERRCODE	Starts R-IN32M3-CL communication.
Watchdog timer	gerR_IN32_ResetWDT	ERRCODE	Resets the R-IN32M3-CL internal WDT.
	gerR_IN32_DisableWDT	ERRCODE	Disables the R-IN32M3-CL internal WDT.
	gerR_IN32_EnableWDT	ERRCODE	Enables the R-IN32M3-CL internal WDT.
	gerR_IN32_SetWDT	ERRCODE	Sets the R-IN32M3-CL internal WDT time limit.
Event	gerR_IN32_GetEvent	ERRCODE	Detects R-IN32M3-CL events.
	gerR_IN32_Main	ERRCODE	Performs the main processing of R-IN32M3-CL event detection.
	gerR_IN32_RestartEvent	ERRCODE	Restarts the R-IN32M3-CL event.
	gerR_IN32_UpdatePortStatus	ERRCODE	Updates the PHY link status.
	gerR_IN32_UpdateMIB	ERRCODE	Updates MIB information.
Cyclic communication	gerR_IN32_SetCyclicStop	ERRCODE	Stops cyclic communication for device-side reasons.
	gerR_IN32_ClearCyclicStop	ERRCODE	Clears cyclic communication stop for device-side reasons.
	gerR_IN32_GetReceivedCyclicData	ERRCODE	Acquires received cyclic data.
	gerR_IN32_GetMasterNodeStatus	ERRCODE	Acquires the master station status.
	gerR_IN32_SetMyStatus	ERRCODE	Sets MyStatus transmission data.
	gerR_IN32_SetSendCyclicData	ERRCODE	Sets cyclic transmission data.
Host station status setup	gerR_IN32_SetNodeStatus	ERRCODE	Sets its host station status.
	gerR_IN32_ForceStop	ERRCODE	Sets a forced stop.
Host station status acquisition	gerR_IN32_GetNodeAndNetworkNumber	ERRCODE	Acquires the node number and network number.
	gerR_IN32_GetCurrentCyclicSize	ERRCODE	Acquires the cyclic communication size specified from the master station.
	gerR_IN32_GetCommumicationStatus	ERRCODE	Acquires the data link status.
	gerR_IN32_GetPortStatus	ERRCODE	Acquires the PHY link status.
	gerR_IN32_GetCyclicStatus	ERRCODE	Acquires the cyclic communication status.
	gerR_IN32_GetMIB	ERRCODE	Acquires MIB information.
	gerR_IN32_ClearMIB	ERRCODE	Clears MIB information.

Table 4.9 R-IN32M3-CL Driver Interface Function List (Continued)

Function Category	Function Name	Function Type	Overview
LED control	gerR_IN32_SetLERR1LED	ERRCODE	Sets the LED [L ER (port 1)]
	gerR_IN32_SetLERR2LED	ERRCODE	Sets the LED [L ER (port 2)]
	gerR_IN32_SetERRLED	ERRCODE	Sets the LED (ERR)
	gerR_IN32_SetDLINKLED	ERRCODE	Sets the LED (D LINK)
	gerR_IN32_SetUSER1LED	ERRCODE	Sets the LED (User LED 1)
	gerR_IN32_SetUSER2LED	ERRCODE	Sets the LED (User LED 2)
	gerR_IN32_SetRUNLED	ERRCODE	Sets the LED (RUN)
	gerR_IN32_DisableLED	ERRCODE	Disables the LED function.
	gerR_IN32_EnableLED	ERRCODE	Enables the LED function.
Network time	gerR_IN32_GetNetworkTime	ERRCODE	Acquires the network time (serial value).
	gerR_IN32_SetNetworkTime	ERRCODE	Sets the network time (serial value).
	gerR_IN32_NetworkTimeToDate	ERRCODE	Changes the network time (serial value) to clock information.
	gerR_IN32_DateToNetworkTime	ERRCODE	Changes the clock information to the network time (serial value).
MDIO access	gerR_IN32_EnableMACIPAccess	ERRCODE	Enables MAC IP access.
	gerR_IN32_DisableMACIPAccess	ERRCODE	Disables MAC IP access.
	gerR_IN32_WritePHY	ERRCODE	Writes to the PHY internal register.
	gerR_IN32_ReadPHY	ERRCODE	Reads the PHY internal register.
Transient reception	gerR_IN32_MainReceiveTransient1	ERRCODE	Main transient reception processing 1
	gerR_IN32_MainReceiveTransient2	ERRCODE	Main transient reception processing 2
	gerR_IN32_EnableReceiveTransient	ERRCODE	Enables transient reception for vendor reasons.
	gblR_IN32_GetReceiveTransientStatus	BOOL	Acquires the transient reception enable status for vendor reasons.
	gerR_IN32_SetMACAddressTableData	ERRCODE	Sets the node information distribution data (MAC address table).
Transient transmission	gerR_IN32_GetUnitInformation	ERRCODE	Acquires unit information.
	gusR_IN32_GetNodeID	USHORT	Acquires the node ID.
	gerR_IN32_GetMulticastMACAddress	ERRCODE	Acquires the multicast MAC address.
	gerR_IN32_GetUnicastMACAddress	ERRCODE	Acquires the unicast MAC address.
	gerR_IN32_GetSendTransientBuffer	ERRCODE	Acquires the transient transmission buffer.
	gerR_IN32_RequestSendingTransient	ERRCODE	Requests transient transmission.
	gerR_IN32_MainSendTransient	ERRCODE	Main transient transmission processing
Interrupt	gerR_IN32_DisableInterrupt	ERRCODE	Disables interrupts.
	gerR_IN32_EnableInterrupt	ERRCODE	Enables interrupts.
Hardware test	gerR_IN32_IEEETest	ERRCODE	IEEE 802.3ab compliance test
	gerR_IN32_InitializeLoopBackTest	ERRCODE	Initializes the internal loop-back / external loop-back communication test.
	gerR_IN32_InternalLoopBackTest	ERRCODE	Internal loop-back communication test
	gerR_IN32_ExternalLoopBackTest	ERRCODE	External loop-back communication test

4.5 R-IN32M3-CL Driver Interface Function Details

The R-IN32M3-CL driver interface functions are called from a user program written in C language. This section describes how to use the R-IN32M3-CL driver interface functions and the details of related functions.

This section uses the definitions below based on the sample code.

(1) Parameter data type and size

The R-IN32M3-CL driver interface function uses the parameter data and types below.

```
#define  VOID            void;
typedef  char            CHAR;
typedef  unsigned char   UCHAR;
typedef  short           SHORT;
typedef  unsigned short  USHORT;
typedef  int             INT;
typedef  unsigned int    UINT;
typedef  long            LONG;
typedef  unsigned long   ULONG;
typedef  int             ERRCODE;
typedef  int             BOOL;
```

(2) Error code definitions

The R-IN32M3-CL driver interface function uses the error codes returned as return values below.

```
#define  R_IN32_OK                0      /*!< Normal */
#define  R_IN32_ERR               (-1)   /*!< Abnormal end */
#define  R_IN32_ERR_OTHER         (-2)   /*!< Abnormal end
                                          (Error occurred in library internal driver.)*/
#define  R_IN32_ERR_OUTOFRANGE    (-3)   /*!< Out of range */
#define  R_IN32_ERR_EMPTY         (-4)   /*!< Empty */
#define  R_IN32_ERR_OVERFLOW      (-5)   /*!< Overflow */
#define  R_IN32_ERR_NOENTRY       (-6)   /*!< No entry */
#define  R_IN32_ERR_NOPERMIT      (-7)   /*!< Not permitted */
#define  R_IN32_ERR_NODATA        (-8)   /*!< No data */
#define  R_IN32_ERR_NOMYSTATUS    (-9)   /*!< Valid MyStatus non-existent */
```

(3) Other definitions

```
#define  R_IN32_TRUE    1
#define  R_IN32_FALSE   0
```

4.5.1 Initial setup

(1) gulR_IN32_GetResetStatus

Function	Gets reset status.			
Call format	ULONG gulR_IN32_GetResetStatus (VOID)			
Arguments	Name	Variable name	Description	I/O
	None			
Return value	R_IN32_RESET_PWRON(1): Power ON reset R_IN32_RESET_SYSTEM(2) : System reset			
Description	This function gets the reset status. Call this function before the function gerR_IN32_Initialize.			

(2) gerR_IN32_Initialize

Function	Initializes R-IN32M3-CL.			
Call format	ERRCODE gerR_IN32_Initialize (const UCHAR* puchMACAddr, const R_IN32_UNITINFO_T pstUnitInfo, const R_IN32_UNITINIT_T *pstUnitInit)			
Arguments	Name	Variable name	Description	I/O
	const UCHAR	*puchMACAddr	Host station MAC address Set as follows for 12-34-56-78-90-AB: puchMACAddr[0]:0x12 puchMACAddr[1]:0x34 puchMACAddr[2]:0x56 puchMACAddr[3]:0x78 puchMACAddr[4]:0x90 puchMACAddr[5]:0xAB	Input
	const R_IN32_UNITINFO_T	*pstUnitInfo	R-IN32M3-CL unit information initial setup For details, refer to Section A) "R_IN32_UNITINFO_T initial setup."	Input
	const R_IN32_UNITINIT_T	*pstUnitInit	R-IN32M3-CL initial setup For details, refer to Section B) “R_IN32_UNITINIT_T initial setup.”	Input
Return value	R_IN32_OK: Normal end			
Description	This function performs R-IN32M3-CL initialization and PHY reset (only at system reset and link down). Calling this function disables the R-IN32M3-CL internal WDT. When you want to use the R-IN32M3-CL internal WDT, be sure to start the function gerR_IN32_EnableWDT. For details, see Section 4.5.2 "Watchdog timer." *: When a fatal error occurs in R-IN32M3-CL, this function calls the function below created by the vendor. Be sure to execute error processing in accordance with the error code. gR_IN32_CallbackFatalError			

Arguments of gerR_IN32_Initialize

The following describes the structure of R_IN32_UNITINFO_T based on the sample code.

```
/* R-IN32M3-CL unit information */
typedef struct R_IN32_UNITINFO_TAG {
    /* Cyclic communication size maximum value */
    ULONG ulMaxRySize;                  /*!< RY size [bytes (octets)]*/
    ULONG ulMaxRWwSize;                 /*!< RWw size (words)*/
    ULONG ulMaxRxSize;                  /*!< RX size [bytes (octets)]*/
    ULONG ulMaxRWrSize;                 /*!< RWr size (words)*/

    /* Station information 1*/
    ULONG ulMyStationPortTotalNumber;   /*!< No. of host station ports */
    ULONG ulTokenHoldTime;              /*!< Token hold time */

    /* Station information 2 */
    ULONG ulIOType;                     /*!< Node information (I/O type)*/

    /* Network information */
    ULONG ulNetVersion;                 /*!< Network firmware version */
    ULONG ulNetModelType;               /*!< Network model type */
    ULONG ulNetUnitModelCode;           /*!< Network model code */
    ULONG ulNetVendorCode;              /*!< Network vendor code */
    UCHAR auchNetUnitModelName[20];     /*!< Network model name */
    UCHAR auchNetVendorName[32];        /*!< Network vendor name */

    /* Controller information */
    BOOL    blInfomationFlag;           /*!< Controller information status flag */
    ULONG ulCtrlVersion;                /*!< Controller firmware version */
    ULONG ulCtrlModelType;              /*!< Controller model type */
    ULONG ulCtrlUnitModelCode;          /*!< Controller model code */
    ULONG ulCtrlVendorCode;             /*!< Controller vendor code */
    UCHAR auchCtrlUnitModelName [20];   /*!< Controller model name */
    UCHAR auchCtrlVendorName [32];      /*!< Controller vendor name */
    ULONG ulVendorInformation;          /*!< Controller vendor device specific information */
} R_IN32_UNITINFO_T;
```

A) R_IN32_UNITINFO_T initial setup

The contents initially set up by R_IN32_UNITINFO_T are as follows:

(a) RY size [bytes (octets)]

Specifies the RY size (bytes) communicable by its own station in increments of 1 byte (multiple of 1). The maximum value for an intelligent device station is 256 bytes.

(b) RWw size (words)

Specifies the RWw size (words) communicable by its own station in 2-word boundary (multiple of 2). The maximum value for an intelligent device station is 1,024 words.

(c) RX size [bytes (octets)]

Specifies the RX size (bytes) communicable by its own station in increments of 1 byte (multiple of 1). The maximum value for an intelligent device station is 256 bytes.

(d) RWr size (words)

Specifies the RWr size (words) communicable by its own station in 2-word boundary (multiple of 2). The maximum value for an intelligent device station is 1,024 words.

(e) No. of own station ports

Specifies the number of physical communication ports of its own station. Set "2" if the station is an intelligent device station developed using R-IN32M3-CL.

(f) Token hold time

Specifies the maximum time its own station holds a token after token passing begins, in µs. (The token hold time is the time that results when the transmission time of the transient communication frame (Transient1, TransientAck, Transient2, etc.) is subtracted from the "time from receipt of a token frame addressed to its own station to the time of transmission of the token frame addressed to the next node.") When you want the RX/RWr size to be the maximum size, set 23 µs (23).
*The calculation method is as follows:

n = RWr transmission size (1,024 words, maximum)
m = RX transmission size (256 bytes, maximum)
i = No. of sent RWr frames

RWr size (words)	i
0	0
0-734	1
735-1024	2

j = No. of sent RX frames

RX size [bytes]	j
0	0
1-256	1

$$[\text{Token hold time}] = \frac{2344 + n \times 16 + m \times 8 + 672 \times (i + j)}{1000}$$

Example: When RWr = 1,024 words and RX = 256 bytes

$$[\text{Token hold time}] = \frac{2344 + 1024 \times 16 + 256 \times 8 + 672 \times (2 + 1)}{1000} = 22.792 \Rightarrow 23\ \mu s$$

(g) Node information (I/O type)

Specifies the I/O type.
00b (0x0) indicates front/back mixed, 01b (0x1) indicates input, 10b(0x2) indicates output), and 11b(0x3) indicates mixed.
Front/Back mixed is used in a case when the input and output are mixed and the input and output use the same address.
Mixed is used in a case where the input and output are mixed and the input and output do not use the same address.

(h) Network firmware version

Specifies the firmware version of the network.

(i) Network model type

Specifies the model type specified by the CC-Link Partner Association.

(j) Network model code

Specifies the model code of the network. The model code is any code defined by the vendor. Manage the code so that it is unique within the same vendor code.

(k) Network vendor code

Specifies the vendor code acquired when the vendor became a member of the CC-Link Partner Association, in BCD. (If the vendor code is 5678, 0x5678 is specified.)

(l) Network model name

Specifies the model name of the network. (in 20-byte character string (ASCII code)). The model name is any name defined by the vendor. Manage the name so that it is unique within the same vendor code.

(m) Network vendor name

Specifies the vendor name of the network. (in 32-byte character string (ASCII code)). The vendor name is any name defined by the vendor.

(n) Controller information status flag

Enables/Disables controller information ((o) Controller firmware version to (u) Controller vendor device specific information). R_IN32_FALSE indicates disabled, and R_IN32_TRUE indicates enabled.

(o) Controller firmware version

Specifies the firmware version of the controller.

(p) Controller model type

Specifies the model type specified by the CC-Link Partner Association.

(q) Controller model name

Specifies the model name of the controller. (in 20-byte character string (ASCII code)). The model name is any name defined by the vendor. Manage the name so that it is unique within the vendor code.

(r) Controller vendor code

Specifies the vendor code acquired when the vendor became a member of the CC-Link Partner Association, in BCD. (If the vendor code is 5678, 0x5678 is specified.)

(s) Controller model code

Specifies the model code of the controller. The model code is any code defined by the vendor. Manage the code so that it is unique within the same vendor code.

(t) Controller vendor name

Specifies the vendor name of the controller. (in 32-byte character string (ASCII code)). The vendor name is any name defined by the vendor.

(u) Controller vendor device specific information

Specifies the vendor device specific information of the controller. The vendor device specific information is any information defined by the vendor.

Arguments of gerR_IN32_Initialize

The following describes the structure of R_IN32_UNITINIT_T based on the sample code

```
/* R-IN32M3-CL initial setup */
typedef struct R_IN32_UNITINIT_TAG {
    BOOL   blNMIUse;                                  /*!< NMI interrupt use */
    BOOL   blInterruptUse;                            /*!< MPU interrupt function use */
    BOOL   blFailedProcess1;                          /*!< Failed process setting 1*/
    BOOL   blFailedProcess2;                          /*!< Failed process setting 2*/
    ULONG ulNodeType;                                 /*!< Node type */
    BOOL   blTransientReceiveEnable;                  /*!< Transient reception function */
    BOOL   blNodeAndNetworkNumberFromMasterPermission;
                /*!< Enable node number and network number setting from master station */
    BOOL   blMACAddressTableRequest;                  /*!< Request node information distribution */
    ULONG ulRunStatus;                                /*!< Initial value of detailed application run status */
    ULONG ulErrorStatus;                              /*!< Initial value of error detection status */
    ULONG ulErrorCode;                                /*!< Initial value of error code */
    ULONG ulUserInformation;                          /*!< Initial value of vendor specific node information */

} R_IN32_UNITINIT_T;
```

B) R_IN32_UNITINIT_T initial setup

The contents initially set by R_IN32_UNITINIT_T are as follows:

(a) NMI interrupt use (Only when you want to use the R-IN32M3-CL internal WDT function)
(R_IN32_TRUE: Use, R_IN32_FALSE: Do not use)

Specify “R_IN32_TRUE” when you want to use the R-IN32M3-CL internal WDT function, and “R_IN32_FALSE” when you do not. Specifying “R_IN32_TRUE” changes the terminal NMIL to "Low" when the R-IN32M3-CL internal WDT overflows.

(b) MPU interrupt function use

Specify “R_IN32_TRUE” when you want to use the R-IN32M3-CL MPU interrupt function, and “R_IN32_FALSE” when you do not. Specifying “R_IN32_TRUE” changes the terminal INTL to "Low" when a R-IN32M3-CL interrupt occurs.

(c) Failed process setting 1

Specify R_IN32_TRUE. When any of the signals below are true, R-IN32M3-CL changes to bypass mode. (Communication frames are neither sent nor received. A received frame is forwarded as is to the other port.)

[1] When the WDTIL terminal setting is True (Low)
[2] When the R-IN32M3-CL internal WDT times out

When you want to clear bypass mode, power ON reset or system reset is required.

(d) Failed process setting 2

Specify R_IN32_TRUE. When a forced stop is executed (gerR_IN32_ForceStop function is called), R-IN32M3-CL changes to bypass mode. (Communication frames are neither sent nor received. A received frame is forwarded as is to the other port.)
When you want to clear a forced stop, power ON reset or system reset is required. For gerR_IN32_ForceStop function details, see Section 0 of Section 4.5.5 “Own station status setup.”

(e) Node type

Specifies the node type of its own station. Specify intelligent device station (0x33).

(f) Transient reception function

Specifies whether or not the transient reception function is present. R_IN32_FALSE indicates the function is not present, and R_IN32_TRUE indicates the function is present.

(g) Enable node number and network number setting from master station

Specifies whether or not the node number and network number setting from the parameter frame is enabled. R_IN32_TRUE enables the node number and network number setting from the master station. R_IN32_FALSE causes the master station not to set the node number and network number.

(h) Request node information distribution

Node information is information indicating the correspondence between the node number and MAC address, and is distributed to the master station. Setting this to R_IN32_TRUE distributes the node information from the master station.*1

*1: If "(f) Transient reception function" is set to R_IN32_FALSE, be sure to set this setting to R_IN32_FALSE.

The master station distributes node information to all slave stations using a Transient1 frame.

When "(f) Transient reception function" is set to R_IN32_FALSE, discard the node information distributions (Transient1 frames) received from the master station using the user program.

*: When Transient2 frames are transmitted:
A response returned to the source can be returned using the source MAC address. When transmissions are actively sent, the MAC address table is used.
The MAC address table is created using the Distribute Node Information request frame (Transient1 frame) distributed from the master station.

(i) Initial value of detailed application run status

Specifies the initial value of the detailed application run status within nodeStatus of the MyStatus frame.

Table 4.10 List of Initial Values of Detailed Application Run Status

Value	Communication Operation
R_IN32_RUNSTS_UNSUPPORTED	Detailed application run status notification not supported
R_IN32_RUNSTS_STOP	Application stopped
R_IN32_RUNSTS_RUN	Application running
R_IN32_RUNSTS_NOTEXIST	Application user does not exist

(j) Initial value of error detection status

Specifies the initial value of the error detection status within nodeStatus of the MyStatus frame.

Value	Communication Operation
R_IN32_ERRSTS_NONE	No error
R_IN32_ERRSTS_WARNING	Minor error
R_IN32_ERRSTS_ERROR	Intermediate error
R_IN32_ERRSTS_FATALERROR	Fatal error

(k) Initial value of error code

Specifies the initial value of the errorCode of the MyStatus frame.

(l) Initial value of vendor specific node information

Specifies the initial value of vendorSpfNodeInfo of the MyStatus frame.

(3) gerR_IN32_SetNodeAndNetworkNumber

Function	Sets node number and network number.			
Call format	ERRCODE gerR_IN32_SetNodeAndNetworkNumber (UCHAR uchNetworkNumber,USHORT usNodeNumber)			
Arguments	Name	Variable name	Description	I/O
	UCHAR	uchNetworkNumber	Network number (value range: 1-239)	Input
	USHORT	usNodeNumber	Node number (value range: 1-120)	Input
Return value	R_IN32_OK: Normal end R_IN32_ERR: Abnormal error (error status in library) R_IN32_ERR_OUTOFRANGE: Node number out of range or network number out of range			
Description	This function sets the node number and network number in R-IN32M3-CL. When the return value is R_IN32_ERR_OUTOFRANGE, the node number and network number are not set. *:This function needs to be called before the function gerR_IN32_Start in Section 4.2.3 “Start Communication processing,” after Section 4.2.2 “Initialization processing.” When this function is executed before the above processing, a R_IN32_ERR (abnormal end; status error in library) occurs.			

(4) ger R_IN32_Start

Function	Starts R-IN32M3-CL communication.			
Call format	ERRCODE gerR_IN32_Start(VOID)			
Arguments	Name	Variable name	Description	I/O
	None			
Return value	R_IN32_OK: Normal end R_IN32_ERR: Abnormal end			
Description	This function provides instructions to start communication to R-IN32M3-CL. *:When a fatal error occurs in R-IN32M3-CL, this function calls the function below created by the vendor. Be sure to execute error processing in accordance with the error code. gR_IN32_CallbackFatalError			

4.5.2 Watchdog timer

(1) gerR_IN32_ResetWDT

Function	Resets R-IN32M3-CL internal WDT.			
Call format	ULONG gerR_IN32_ResetWDT (VOID)			
Arguments	Name	Variable name	Description	I/O
	None			
Return value	R_IN32_OK: Normal end			
Description	This function resets the R-IN32M3-CL internal WDT. *:If you want to call a function within Section 4.5.2 "Watchdog timer" after this function is called, wait 1.032 µs or longer.			

(2) gerR_IN32_DisableWDT

Function	Disables R-IN32M3-CL internal WDT.			
Call format	ULONG gerR_IN32_DisableWDT (VOID)			
Arguments	Name	Variable name	Description	I/O
	None			
Return value	R_IN32_OK: Normal end			
Description	This function disables the R_IN32M3-CL internal WDT. *:If you want to call a function within Section 4.5.2 "Watchdog timer" after this function is called, wait 1.032 µs or longer. R-IN32M3-CL enables the R-IN32M3-CL internal WDT immediately after reset. (Initial value of R-IN32M3-CL internal WDT time limit setting: 3.2 s.) The R-IN32M3-CL internal WDT is disabled when the function gerR_IN32_Initialize is called. Implement one of the following when startup of the function gerR_IN32_Initialize takes time: Call this function to disable the R-IN32M3-CL internal WDT. Call the function gerR_IN32_ResetWDT to reset the R-IN32M3-CL internal WDT. (Make sure that the R-IN32M3-CL internal WDT does not time out.)			

(3) gerR_IN32_EnableWDT

Function	Enables R-IN32M3-CL internal WDT.			
Call format	ULONG gerR_IN32_EnableWDT (VOID)			
Arguments	Name	Variable name	Description	I/O
	None			
Return value	R_IN32_OK: Normal end			
Description	This function enables the R_IN32M3-CL internal WDT. *:If you want to call a function within Section 4.5.2 "Watchdog timer" after this function is called, wait 1.032 µs or longer. R-IN32M3-CL disables the R-IN32M3-CL internal WDT when the function gerR_IN32_Initialize is called. Be sure to implement this function when you want to use the R-IN32M3-CL internal WDT.			

(4) gerR_IN32_SetWDT

Function	Sets R-IN32M3-CL internal WDT time limit.			
Call format	ULONG gerR_IN32_SetWDT (USHORT usWDTCOUNT)			
Arguments	Name	Variable name	Description	I/O
	USHORT	usWDTCOUNT	Sets the R-IN32M3-CL internal WDT time limit. 0x0000:100ms 0x0000:100ms 0x0002:300ms : 0x001F: 3.2 s	Input
Return value	R_IN32_OK: Normal end			
Description	This function sets the R-IN32M3-CL internal WDT time limit. *:If you want to call a function within Section 4.5.2 "Watchdog timer" after this function is called, wait 1.032 µs or longer. If the R-IN32M3-CL internal WDT time limit setting is changed by this function while the R-IN32M3-CL internal WDT is running (after the function gerR_IN32_EnableWDT is called), the R-IN32M3-CL internal WDT runs using the new time limit setting after the function gerR_IN32_ResetWDT is called. (Until the function gerR_IN32_ResetWDT is called, the R-IN32M3-CL internal WDT runs using the R-IN32M3-CL internal WDT time limit setting prior to the change.)			

4.5.3 Event

(1) gerR_IN32_GetEvent

Function	Detects R-IN32M3-CL events.			
Call format	ERRCODE gerR_IN32_GetEvent (R_IN32_EVTPRM_INTERRUPT_T *pstEvent)			
Arguments	Name	Variable name	Description	I/O
	R_IN32_EVTPRM_INTERRUPT_T	*pstEvent	Interrupt cause	Output
Return value	R_IN32_OK: Normal end			
Description	This function detects R-IN32M3-CL events.			

Arguments of gerR_IN32_GetEvent

The following describes the configuration of R_IN32_EVTPRM_INTERRUPT_T based on the sample code.

```
/* Interrupt cause */
typedef struct R_IN32_EVTPRM_INTERRUPT_TAG    {
    union {
        ULONG           ulAll;
        struct {
            ULONG   b1ZCommConnect:                 1;      /* b0: Connect communication */
            ULONG   b1ZCommDisconnect:              1;      /* b1: Disconnect communication */
            ULONG   b1ZCommConnectToDisconnect:     1;      /* b2: Connect communication
                                                               → Disconnect communication */
            ULONG   b1ZCommDisconnectToConnect:     1;      /* b3: Disconnect communication
                                                               → Connect communication */
            ULONG   b1ZChangeStNoNetNo:             1;      /* b4: Change node number and network  number */
            ULONG   b1ZChangeActCommand:            1;      /* b5: Change run command */
            ULONG   b1ZPrmFrmRcv_OK:                1;      /* b6: Parameter frame reception */
            ULONG   b1ZReserve1:                    1;      /* b7: Reserved */
            ULONG   b1ZPrmChkFrmRcv_OK:             1;      /* b8: ParamCheck frame reception
                                                               (when parameters match)*/
            ULONG   b3ZReserve2:                    3;      /* b9-11: Reserved */
            ULONG   b1ZRecvNonCyclic:               1;      /* b12: Transient reception */
            ULONG   b1ZSendFinNonCyclic:            1;      /* b13: Transient transmission complete */
            ULONG   b7ZReserve3:                    7;      /* b14-20: Reserved */
            ULONG   b1ZMasterWatchTimeout:          1;      /* b21: Master watchdog timer  timeout occurred */
            ULONG   bAZReserve4:                    10;     /* b22-31: Reserved */
        } stBit;
    } uniFlag;
} R_IN32_EVTPRM_INTERRUPT_T;
```

(2) gerR_IN32_Main

Function	Main R-IN32M3-CL event detection processing			
Call format	ERRCODE gerR_IN32_Main (const R_IN32_EVTPRM_INTERRUPT_T *pstEvent)			
Arguments	Name	Variable Name	Description	I/O
	const R_IN32_EVTPRM_ INTERRUPT_T	*pstEvent	Interrupt cause	Input
Return value	R_IN32_OK: Normal end R_IN32_ERR: Abnormal end (status error in library)			
Description	This function performs processing in response to a R-IN32M3-CL event. *:This function needs to be called after the processing described in Section 4.2.2 “Initialization processing" and Section 4.2.3 "Start Communication processing." Calling this function before executing this processing results in a R_IN32_ERR (abnormal end; status error in library).			

(3) gerR_IN32_RestartEvent

Function	Restarts R-IN32M3-CL events.			
Call format	ERRCODE gerR_IN32_RestartEvent (VOID)			
Arguments	Name	Variable Name	Description	I/O
	None			
Return value	R_IN32_OK: Normal end			
Description	This function restarts events stopped by R-IN32M3-CL event detection (the function gerR_IN32_GetEvent).			

(4) gerR_IN32_UpdatePortStatus

Function	Updates PHY link status.			
Call format	ERRCODE gerR_IN32_UpdatePortStatus (ULONG ulPort)			
Arguments	Name	Variable Name	Description	I/O
	ULONG	ulPort	Port specification R_IN32_PORT1(0) : Port 1 R_IN32_PORT2(1) : Port 2	Input
Return value	R_IN32_OK: Normal end			
Description	This function updates the PHY link status.			

(5) gerR_IN32_UpdateMIB

Function	Updates MIB information.			
Call format	ERRCODE gerR_IN32_UpdateMIB (VOID)			
Arguments	Name	Variable Name	Description	I/O
	None			
Return value	R_IN32_OK: Normal end R_IN32_ERR: Abnormal end [MIB information collection error (status error in library / mismatch)] R_IN32_ERR_OTHER: Abnormal end [MIB information collection error (error occurred in driver inside library)]			
Description	This function updates the MIB information. *:When the return value of this function is a value other than R_IN32_OK, the function calls the function below created by the vendor. Be sure to execute error processing in accordance with the error code. gR_IN32_CallbackFatalError			

4.5.4 Cyclic communication

(1) gerR_IN32_SetCyclicStop

Function	Stops cyclic communication for device-side reasons.			
Call format	ERRCODE gerR_IN32_SetCyclicStop (VOID)			
Arguments	Name	Variable Name	Description	I/O
	None			
Return value	R_IN32_OK:Normal end			
Description	This function stops cyclic communication for device-side reasons. If you want to clear the stop status, call the function gerR_IN32_ClearCyclicStop.			

(2) ger R_IN32_ClearCyclicStop

Function	Clears cyclic communication stop for device-side reasons.			
Call format	ERRCODE gerR_IN32_ClearCyclicStop (VOID)			
Arguments	Name	Variable Name	Description	I/O
	None			
Return value	R_IN32_OK:Normal end			
Description	This function clears cyclic communication stop that was called by the function gerR_IN32_SetCyclicStop.			

(3) ger R_IN32_GetReceivedCyclicData

Function	Gets received cyclic data.			
Call format	ERRCODE gerR_IN32_GetReceivedCyclicData (VOID *pRyDst, VOID *pRWwDst, BOOL blEnable)			
Arguments	Name	Variable Name	Description	I/O
	VOID	*pRyDst	RY area	Output
	VOID	*pRWwDst	RWw area*1	Output
	BOOL	blEnable	Enables/Disables copying. R_IN32_TRUE: Enable R_IN32_FALSE: Disable	Input
Return value	R_IN32_OK: Normal end (received data present) R_IN32_ERR: Abnormal end (no received data)			
Description	This function stores cyclic data received from the master station in the addresses indicated by pRyDst and pRWwDst. Note, however, that when blEnable is set to R_IN32_FALSE, the received cyclic data is discarded. (The return value changes to R_IN32_ERR.) *: R_IN32_ERR: Abnormal end (no received data) While a R_IN32_ERR occurs when no cyclic communication is received from the previous call of the function gerR_IN32_GetReceivedCyclicData to the current call of the function gerR_IN32_GetReceivedCyclicData, this does not indicate an error. *1: Set the head address of the RWw area in increments of 4 bytes (0 or multiple of 4).			

(4) ger R_IN32_GetMasterNodeStatus

Function	Gets master station status.			
Call format	ERRCODE gerR_IN32_GetMasterNodeStatus (BOOL *pblRunSts, BOOL *pblErrSts, ULONG *pulErrCode)			
Arguments	Name	Variable Name	Description	I/O
	BOOL	*pblRunSts	Application run status R_IN32_TRUE: Running R_IN32_FALSE: Stopped	Output
	BOOL	*pblErrSts	Application error status R_IN32_TRUE: Error R_IN32_FALSE: No error	Output
	ULONG	*pulErrCode	Master station error code	Output
Return value	R_IN32_OK: Normal end (MyStatus frame received from master station) R_IN32_ERR: Abnormal end [MyStatus frame not received from master station due to no data link (data link disconnected)]			
Description	This function acquires the status of the master station from the MyStatus frame received from the master station. When the MyStatus frame is not received from the master station due to no data link (data link disconnected), the arguments are as follows: pblRunStsR_IN32_FALSE pblErrStsR_IN32_FALSE pulErrCode0			

(5) ger R_IN32_SetMyStatus

Function	Sets MyStatus transmission data.			
Call format	ERRCODE gerR_IN32_SetMyStatus (VOID)			
Arguments	Name	Variable Name	Description	I/O
	None			
Return value	R_IN32_OK: Normal end			
Description	This function sets its own station status specified by the function gerR_IN32_SetNodeStatus in R-IN32M3-CL.			

(6) ger R_IN32_SetSendCyclicData

Function	Sets cyclic transmission data.			
Call format	ERRCODE gerR_IN32_SetSendCyclicData (const VOID *pRxSrc, const VOID *pRWwSrc, BOOL blEnable)			
Arguments	Name	Variable Name	Description	I/O
	const VOID	*pRxSrc	RX area	Input
	const VOID	*pRWwSrc	RWw area*1	Input
	BOOL	blEnable	Enables/Disables update. R_IN32_TRUE: Enable R_IN32_FALSE: Disable	Input
Return value	R_IN32_OK: Normal end			
Description	This function sets the cyclic transmission data stored in the addresses indicated in pRxSrc and pRWwSrc in R-IN32M3-CL. Now, however, that when blEnable is set to R_IN32_FALSE, cyclic transmission data is not set. (The return value changes to R_IN32_ERR.) *1: Set the head address of the RWw area in increments of 4 bytes (0 or multiple of 4).			

4.5.5 Own station status setup

(1) gerR_IN32_SetNodeStatus

Function	Sets its own station status.			
Call format	ERRCODE gerR_IN32_SetNodeStatus (ULONG ulRunSts, ULONG ulErrSts, ULONG ulErrCode,ULONG ulUserInformation)			
Arguments	Name	Variable Name	Description	I/O
	ULONG	ulRunSts	Detailed application run status R_IN32_RUNSTS_UNSUPPORTED(0): Detailed application run status notification not supported R_IN32_RUNSTS_STOP(1): Application stopped R_IN32_RUNSTS_RUN(2): Application running R_IN32_RUNSTS_NOTEXIST(3): Application user does not exist	Input
	ULONG	ulErrSts	Application error status R_IN32_ERRSTS_NONE(0):No error R_IN32_ERRSTS_WARNING(1):Minor error R_IN32_ERRSTS_ERROR(2):Intermediate error R_IN32_ERRSTS_FATALERROR(3):Fatal error	Input
	ULONG	ulErrCode	Error code	Input
	ULONG	ulUserInformation	Vendor specific node information	Input
Return value	R_IN32_OK: Normal end			
Description	This function sets its own station status as information to be sent in a MyStatus frame.			

(2) gerR_IN32_ForceStop

Function	Sets a forced stop.			
Call format	ERRCODE gerR_IN32_ForceStop (VOID)			
Arguments	Name	Variable Name	Description	I/O
	None			
Return value	R_IN32_OK: Normal end			
Description	This function forcibly stops R-IN32M3-CL. To clear the forced stop, power ON reset or system reset is required.			

4.5.6 Host station status acquisition

(1) gerR_IN32_GetNodeAndNetworkNumber

Function	Gets node number and network number.			
Call format	ERRCODE gerR_IN32_GetNodeAndNetworkNumber (USHORT *pusNodeNumber,UCHAR *puchNetworkNumber)			
Arguments	Name	Variable Name	Description	I/O
	USHORT	*pusNodeNumber	Node number	Output
	UCHAR	*puchNetworkNumber	Network number	Output
Return value	R_IN32_OK: Normal end			
Description	This function acquires the node number and network number.			

(2) gerR_IN32_GetCurrentCyclicSize

Function	Gets specified cyclic communication size from master station.			
Call format	ERRCODE gerR_IN32_GetCurrentCyclicSize (R_IN32_CYCLIC_SIZE_T *pstCyclicSize)			
Arguments	Name	Variable Name	Description	I/O
	R_IN32_CYCLIC_SIZE_T	*pstCyclicSize	Cyclic communication size ulRySize: RY size [bytes (octets)] ulRWwSize: RWw size [bytes (octets)] ulRxSize: RX size [bytes (octets)] ulRWrSize: RWr size [bytes (octets)]	Output
Return value	R_IN32_OK: Normal end			
Description	This function acquires the specified cyclic communication from the master station in the parameter frame. The functions gerR_IN32_GetReceivedCyclicData and gerR_IN32_SetSendCyclicData input and output cyclic transmission and reception data in the size acquired by this function.			

Arguments of gerR_IN32_GetCurrentCyclicSize

The following describes the structure of R_IN32_CYCLIC_SIZE_T based on the sample code.

```
/* Cyclic communication size */
typedef struct R_IN32_CYCLIC_SIZE_TAG {
    ULONG    ulRySize;      /*!< RY size [bytes (octets)]*/
    ULONG    ulRWwSize;     /*!< RWw size [bytes (octets)]*/
    ULONG    ulRxSize;      /*!< RX size [bytes (octets)]*/
    ULONG    ulRWrSize;     /*!< RWr size [bytes (octets)]*/
} R_IN32_CYCLIC_SIZE_T;
```

(3) gerR_IN32_GetCommumicationStatus

Function	Gets data link status.			
Call format	ERRCODE gerR_IN32_GetCommumicationStatus (ULONG *pulCommSts)			
Arguments	Name	Variable Name	Description	I/O
	ULONG	*pulCommSts	Data link status R_IN32_COMMSTS_CYC_DLINK(2): Data link in progress (cyclic communication in progress) R_IN32_COMMSTS_TOKEN_PASS(1): Data link in progress (cyclic communication stopped) R_IN32_COMMSTS_DISCONNECT(0): No data link (disconnected)	Output
Return value	R_IN32_OK: Normal end			
Description	This function acquires the data link status. Turn the D LINK LED on/off according to the data link status. R_IN32_COMMSTS_CYC_DLINK: LED on Other: LED off *: For D LINK LED on/off control, see Section 4.2.12 “Update Communication Status processing.”			

(4) gerR_IN32_GetPortStatus

Function	Gets PHY link status.			
Call format	ERRCODE gerR_IN32_GetPortStatus (ULONG ulPort, ULONG *pulLinkStatus, ULONG *pulSpeed, ULONG *pulDuplex)			
Arguments	Name	Variable Name	Description	I/O
	ULONG	ulPort	Port specification R_IN32_PORT1(0) : Port 1 R_IN32_PORT2(1) : Port 2	Input
	ULONG	*pulLinkStatus	Link status R_IN32_LINKUP(1): Link up R_IN32_LINKDOWN(0) : Link down	Output
	ULONG	*pulSpeed	Speed R_IN32_SPEED_1G(0): 1 Gbps R_IN32_SPEED_100M(1): 100 Mbps R_IN32_SPEED_10M(2): 10 Mbps (This is enabled when the second argument *pulLinkStatus is set to R_IN32_LINKUP(1). Do not use when the argument is set to R_IN32_LINKDOWN(0).)	Output
	ULONG	*pulDuplex	Full duplex / Half duplex R_IN32_DUPLEX_FULL(0): Full duplex R_IN32_DUPLEX_HALF(1): Half duplex (This is enabled when the second argument *pulLinkStatus is set to R_IN32_LINKUP(1). Do not use when the argument is set to R_IN32_LINKDOWN(0).)	Output
Return value	R_IN32_OK: Normal end			
Description	This function acquires the PHY link status.			

(5) gerR_IN32_GetCyclicStatus

Function	Gets cyclic communication status.			
Call format	ULONG gerR_IN32_GetCyclicStatus (R_IN32_CYCLIC_STA_T *pstCyclicStatus)			
Arguments	Name	Variable Name	Description	I/O
	R_IN32_CYCLIC_STA_T	*pstCyclicStatus	Cyclic communication status Bits 2-0 Cyclic communication parameter hold status 001b: Parameter normally received 010b: Not received or ID mismatch 011b: Checking 100b: Parameter abnormally received Bit 3 Cyclic communication parameter check status 0: Checked 1: Checking Bit 4 Node number invalid setting status 0: In range 1: Out of range Bit 5 Reserved node setting status 0: Non-reserved node 1: Reserved node Bit 6 Cyclic communication implementation instruction (batch) setting status 0: Run 1: Stop Bit 7 Cyclic communication implementation instruction (individual) setting status 0: Run 1: Stop Bit 8 Reserved Bit 9 Cyclic communication continuation not possible error status 0: No error 1: Cyclic communication not possible error Bit 10 Node number duplication status 0: No duplication 1: Duplication Bit 11 Reserved Bit 12 Node type invalid / Specified size invalid status 0: Normal 1: Invalid Bit 13 Reserved Bit 14 Disconnection status 0:Cyclic communication in progress or token passing in progress 1: Disconnected Bit 15 Stop status due to own reasons 0: Not stopped 1: Cyclic communication stopped due to reason other than the above	Output
Return value	R_IN32_OK: Normal end			
Description	This function acquires the cyclic communication status.			

Arguments of gerR_IN32_GetCyclicStatus

The following describes the structure of R_IN32_CYCLIC_STA_T based on the sample code.

```
/* Cyclic communication status */
typedef struct R_IN32_CYCLIC_STA_TAG{
union {
    USHORT      usAll;
    struct    {
      USHORT    b3ZComonParamkeepCond:      3;/* b2-0   : Cyclic communication parameter
                                                          hold status */
      USHORT    b1ZParamCheckCond:          1;/* b3     : Cyclic communication parameter
                                                          check status */
      USHORT    b1ZMyNodeNoRangeOut:        1;/* b4     : Node number invalid setting status */
      USHORT    b1ZMyNodeReserveSetup:      1;/* b5     : Reserved node setting status */
      USHORT    b1ZCyclicOpeInstructPackage: 1;/* b6     : Cyclic communication implementation
                                                          instruction (batch) setting status */
      USHORT    b1ZCyclicOpeInstructVarious: 1;/* b7     : Cyclic communication implementation
                                                          instruction (individual) setting status */
      USHORT    b1ZReserved1:               1;/* b8:    Reserved */
      USHORT    b1ZMyMpuAbnomal:            1;/* b9     : Cyclic communication continuation not
                                                          possible error status */
      USHORT    b1ZMyNodeNumberDuplicate:   1;/* b10    : Node number duplication status */
      USHORT    b1ZReserved2:               1;/* b11:   Reserved */
      USHORT    b1ZNodeTypeWrong:           1;/* b12 :  Node type invalid / Specified size invalid
                                                          status */
      USHORT    b1ZReserved3:               1;/* b13    : Reserved */
      USHORT    b1ZDLinkState:              1;/* b14    : Disconnection status */
      USHORT    b1ZCyclicState:             1;/* b15    : Stop status due to own reasons */
      } stBit;
    } uniCycSta;
} R_IN32_CYCLIC_STA_T;
```

(6) gerR_IN32_GetMIB

Function	Gets MIB information.			
Call format	ERRCODE gerR_IN32_GetMIB (R_IN32_MIB_T *pstMIB)			
Arguments	Name	Variable Name	Description	I/O
	R_IN32_MIB_T	*pstMIB	R-IN32M3-CL MIB information	Output
Return value	R_IN32_OK: Normal end			
Description	This function acquires MIB information.			

Precaution:
MIB information is non-disclosed information. Regard this information as information deployed by the vendor only.

Arguments of gerR_IN32_GetMIB

The following describes the structure of R_IN32_MIB_T based on the sample code.

```
/* MIB information */
typedef struct R_IN32_MIB_TAG {
    R_IN32_MIBSDRD_T    stSDRD;          /*!< Transmission/Reception area counter value */
    R_IN32_MIBMACIP_T   stMACIP1;        /*!< MAC IP area counter value (port 1)*/
    R_IN32_MIBMACIP_T   stMACIP2;        /*!< MAC IP area counter value (port 2)*/
    R_IN32_MIBRGCNT_T   stRING1;         /*!< Ring control area counter value (port 1)*/
    R_IN32_MIBRGCNT_T   stRING2;         /*!< Ring control area counter value (port 2)*/
    ULONG               ulP1DownCounter; /*!< Link down counter (port 1)*/
    ULONG               ulP2DownCounter; /*!< Link down counter (port 2)*/
    ULONG               ulMasterWatchCount; /*!< Master watchdog timer error counter */
} R_IN32_MIB_T;
```

The following describes the configuration of the tags included in R_IN32_MIB_T.

```
/* MIB information (counter)*/
typedef struct R_IN32_MIBSDRD_TAG {
    ULONG               ulCyclicRecNomalFrameCnt;  /*!< Received cyclic frame counter */
    ULONG               ulNonCyclicRecValidCnt;    *!< Received transient frame counter */
    ULONG               ulNonCyclicRecRejectCnt;   /*!< Received transient frame discarded counter */
} R_IN32_MIBSDRD_T;

/* MIB information (ring control area)*/
typedef struct R_IN32_MIBRGCNT_TAG {
    ULONG               ulHecErr;          /*!< MIB1: No. of HEC error frames */
    ULONG               ulDcsFcsErr;       /*!< MIB2: No. of DCS/FCS error frames */
    ULONG               ulUnderErr;        /*!< MIB3: No. of undersize error frames */
    ULONG               ulRpt;             /*!< MIB4: No. of forwarded frames */
    ULONG               ulUp;              /*!< MIB5: No. of upper layer transmission frames */
    ULONG               ulRptFullDrop;     /*!< MIB6: No. of discarded frames due to full forward buffer */
    ULONG               ulUpFullDrop;      /*!< MIB7: No. of discarded frames due to full upper layer
                                               transmission buffer */
} R_IN32_MIBRGCNT_T;

/* MIB information (MAC IP) */
typedef struct R_IN32_MIBMACIP_TAG {
    ULONG               ulRFrm;            /*!< Received frame counter */
    ULONG               ulTFrm;             /*!< Sent frame counter */
    ULONG               ulRUnd;            /*!< Received undersized frame counter */
    ULONG               ulROvr;            /*!< Received oversized frame counter */
    ULONG               ulRFcs;            /*!< Received frame FCS error counter */
    ULONG               ulRFgm;            /*!< Received frame fragment error counter */
    ULONG               ulRIFGErr;         /*!< Minimum IFG frame detection counter */
    ULONG               ulREps;            /*!< Received frame SFD or less detection counter */
    ULONG               ulRCde;            /*!< Received code error counter */
    ULONG               ulRFce;            /*!< Received invalid carrier error counter */
    ULONG               ulRCEE;            /*!< Received carrier extension error counter */
} R_IN32_MIBMACIP_T;
```

(7) gerR_IN32_ClearMIB

Function	Clears MIB information.			
Call format	ERRCODE gerR_IN32_ClearMIB (VOID)			
Arguments	Name	Variable Name	Description	I/O
	None			
Return value	R_IN32_OK: Normal end			
Description	This function clears the MIB information.			

4.5.7 LED control

(1) gerR_IN32_SetLERR1LED

Function	Sets LED (L ER (port 1)).			
Call format	ERRCODE gerR_IN32_SetLERR1LED (ULONG ulCtrl)			
Arguments	Name	Variable Name	Description	I/O
	ULONG	ulCtrl	LED control R_IN32_LED_OFFLED off R_IN32_LED_ONLED on	Input
Return value	R_IN32_OK: Normal end			
Description	This function turns on and off the L ER LED of port 1. *: The LED cannot be turned on when a R-IN32M3-CL internal WDT, external WDT, or forced stop occurs.			

(2) gerR_IN32_SetLERR2LED

Function	Sets LED (L ER (port 2)).			
Call format	ERRCODE gerR_IN32_SetLERR2LED (ULONG ulCtrl)			
Arguments	Name	Variable Name	Description	I/O
	ULONG	ulCtrl	LED control R_IN32_LED_OFFLED off R_IN32_LED_ONLED on	Input
Return value	R_IN32_OK: Normal end			
Description	This function turns on and off the L ER LED of port 2. *: The LED cannot be turned on when a R-IN32M3-CL internal WDT, external WDT, or forced stop occurs.			

(3) gerR_IN32_SetERRLED

Function	Sets LED (ERR).			
Call format	ERRCODE gerR_IN32_SetERRLED (ULONG ulCtrl)			
Arguments	Name	Variable Name	Description	I/O
	ULONG	ulCtrl	LED control R_IN32_LED_OFFLED off R_IN32_LED_ONLED on R_IN32_LED_BLINKLED blinking	Input
Return value	R_IN32_OK: Normal end			
Description	This function turns on and off the ERR LED. *: The LED cannot be turned off or set to blinking when a R-IN32M3-CL internal WDT, external WDT, or forced stop occurs.			

(4) gerR_IN32_SetDLINKLED

Function	Sets LED (D LINK).			
Call format	ERRCODE gerR_IN32_SetDLINKLED (ULONG ulCtrl)			
Arguments	Name	Variable Name	Description	I/O
	ULONG	ulCtrl	LED control R_IN32_LED_OFFLED off R_IN32_LED_ONLED on R_IN32_LED_BLINKLED blinking	Input
Return value	R_IN32_OK: Normal end			
Description	This function turns on and off the D LINK LED. *: The LED cannot be turned on or set to blinking when a R-IN32M3-CL internal WDT, external WDT, or forced stop occurs.			

(5) gerR_IN32_SetUSER1LED

Function	Sets LED (User LED 1).			
Call format	ERRCODE gerR_IN32_SetUSER1LED (ULONG ulCtrl)			
Arguments	Name	Variable Name	Description	I/O
	ULONG	ulCtrl	LED control R_IN32_LED_OFFLED off R_IN32_LED_ONLED on R_IN32_LED_BLINKLED blinking	Input
Return value	R_IN32_OK: Normal end			
Description	This function turns on and off User LED 1. *: The LED cannot be turned on or set to blinking when a R-IN32M3-CL internal WDT, external WDT, or forced stop occurs.			

(6) gerR_IN32_SetUSER2LED

Function	Sets LED (User LED 2).			
Call format	ERRCODE gerR_IN32_SetUSER2LED (ULONG ulCtrl)			
Arguments	Name	Variable Name	Description	I/O
	ULONG	ulCtrl	LED control R_IN32_LED_OFFLED off R_IN32_LED_ONLED on R_IN32_LED_BLINKLED blinking	Input
Return value	R_IN32_OK: Normal end			
Description	This function turns on and off User LED 2. *: The LED cannot be turned on or set to blinking when a R-IN32M3-CL internal WDT, external WDT, or forced stop occurs.			

(7) gerR_IN32_SetRUNLED

Function	Sets LED (RUN).			
Call format	ERRCODE gerR_IN32_SetRUNLED (ULONG ulCtrl)			
Arguments	Name	Variable Name	Description	I/O
	ULONG	ulCtrl	LED control R_IN32_LED_OFFLED off R_IN32_LED_ONLED on	Input
Return value	R_IN32_OK: Normal end			
Description	This function turns on and off the RUN LED. *: The LED cannot be turned on when a R-IN32M3-CL internal WDT, external WDT, or forced stop occurs.			

(8) gerR_IN32_DisableLED

Function	Disables LED function.			
Call format	ERRCODE gerR_IN32_DisableLED (USHORT usBitPattern)			
Arguments	Name	Variable Name	Description	I/O
	ULONG	usBitPattern	Disables the LED function. (ON: Disable, OFF: Hold previous vale) Bit 0: Disable RUN LED Bit 2: Disable User LED 2 Bit 4: Disable User LED 1 Bit 6: Disable D LINK LED Bit 8: Disable ERR LED Bit10: Disable port 1 L ER LED Bit11: Disable port 2 L ER LED (Bits 1, 3, 5, 7, 9, and 12-15: Not used)	Input
Return value	R_IN32_OK: Normal end			
Description	This function disables the LED function. *: The function cannot be disabled when a R-IN32M3-CL internal WDT, external WDT, or forced stop occurs.			

(9) gerR_IN32_EnableLED

Function	Enables LED function.			
Call format	ERRCODE gerR_IN32_EnableLED (USHORT usBitPattern)			
Arguments	Name	Variable Name	Description	I/O
	ULONG	usBitPattern	Enables the LED function. (ON: Enable, OFF: Hold previous vale) Bit 0: Enable RUN LED Bit 2: Enable User LED 2 Bit 4: Enable User LED 1 Bit 6: Enable D LINK LED Bit 8: Enable ERR LED Bit10: Enable port 1 L ER LED Bit11: Enable port 2 L ER LED (Bits 1, 3, 5, 7, 9, and 12-15: Not used)	Input
Return value	R_IN32_OK: Normal end			
Description	This function enables the LED function.			

4.5.8 Network time

(1) gerR_IN32_GetNetworkTime

Function	Acquires network time (serial value).			
Call format	ERRCODE gerR_IN32_GetNetworkTime (USHORT *pusSerial)			
Arguments	Name	Variable Name	Description	I/O
	USHORT	*pusSerial	Network time pusSerial[0]: Network time (bits 15-0) pusSerial[1]: Network time (bits 31-16) pusSerial[2]: Network time (bits 47-32)	Output
Return value	R_IN32_OK: Normal end			
Description	This function acquires the network time (serial value in increments of 15.2587890625 µs given a starting point of January 1, 2000, 00:00:00).			

(2) gerR_IN32_SetNetworkTime

Function	Sets network time (serial value).			
Call format	ERRCODE gerR_IN32_SetNetworkTime (const USHORT *pusSerial)			
Arguments	Name	Variable Name	Description	I/O
	const USHORT	*pusSerial	Network time pusSerial[0]: Network time (bits 15-0) pusSerial[1]: Network time (bits 31-16) pusSerial[2]: Network time (bits 47-32)	Input
Return value	R_IN32_OK: Normal end			
Description	This function sets the network time (serial value in increments of 15.2587890625 µs given a starting point of January 1, 2000, 00:00:00).			

(3) gerR_IN32_NetworkTimeToDate

Function	Converts network time (serial value) → Clock information.			
Call format	ERRCODE gerR_IN32_NetworkTimeToDate (R_IN32_TIMEINFO_T *pstTimeInfo, const USHORT *pusSerial)			
Arguments	Name	Variable Name	Description	I/O
	R_IN32_TIMEINFO_T	*pstTimeInfo	Clock information	Output
	const USHORT	*pusSerial	Network time pusSerial[0]: Network time (bits 31-16) pusSerial[1]: Network time (bits 47-32)	Input
Return value	R_IN32_OK: Normal end			
Description	This function converts the network time (serial value in increments of seconds given a starting point of January 1, 2000, 00:00:00) to clock information [year/month/day/hour/minute/second/microsecond (fixed to 0)/day of the week].			

Arguments of gerR_IN32_NetworkTimeToDate

The following describes the structure of R_IN32_TIMEINFO_T based on the sample code.

```
/* Clock information */
typedef struct R_IN32_TIMEINFO_TAG {
    USHORT    usYear;     /*!< Year        (2000 -  2136)*/
    USHORT    usMonth;    /*!< Month       (   1 -    12)*/
    USHORT    usDay;      /*!< Day         (   1 -    31)*/
    USHORT    usHour;     /*!< Hour        (   0 -    23)*/
    USHORT    usMin;      /*!< Minute      (   0 -    59)*/
    USHORT    usSec;      /*!< Second      (   0 -    59)*/
    USHORT    usMsec;     /*!< Microsecond (   0 -   999)*/
    USHORT    usWday;     /*!< Weekday     (0(Sunday)- 6(Saturday))*/
} R_IN32_TIMEINFO_T;
```

(4) gerR_IN32_DateToNetworkTime

Function	Convertsclock information → Network time (serial value).			
Call format	ERRCODE gerR_IN32_DateToNetworkTime (const R_IN32_TIMEINFO_T *pstTimeInfo, USHORT *pusSerial)			
Arguments	Name	Variable Name	Description	I/O
	const R_IN32_TIMEINFO_T	*pstTimeInfo	Clock information	Input
	USHORT	*pusSerial	Network time pusSerial[0]: Network time (bits 15-0) pusSerial[1]: Network time (bits 31-16) pusSerial[2]: Network time (bits 47-32)	Output
Return value	R_IN32_OK: Normal end R_IN32_ERR: Abnormal end			
Description	This function converts clock information (year/month/day/hour/minute/second) to network time (serial value in increments of seconds given a starting point of January 1, 2000, 00:00:00). (ausSerial[0]: Network time (bits 15-0) is fixed to 0.) *: A year other than 2000-2136 results in a R_IN32_ERR. The R-IN32M3-CL driver does not check for any errors other than the above. Implement error processing in the user program to ensure that there are no leap year or date errors.			

4.5.9 MDIO access

(1) gerR_IN32_EnableMACIPAccess

Function	Enables MAC IP access.			
Call format	ERRCODE gerR_IN32_EnableMACIPAccess (VOID)			
Arguments	Name	Variable Name	Description	I/O
	None			
Return value	R_IN32_OK: Normal end R_IN32_ERR: Abnormal end (MDIO command end wait error)			
Description	This function enables MAC IP access. *:Shorten the period from MAC IP access enable (function gerR_IN32_EnableMACIPAccess) to MAC IP access disable (function gerR_IN32_DisableMACIPAccess) to the extent possible. (If the vendor uses interrupts, disable the interrupts from MAC IP access enable to MAC IP access disable.) When the return value of this function is a value other than R_IN32_OK, the function calls the function below created by the vendor. Be sure to execute error processing in accordance with the error code. gR_IN32_CallbackFatalError			

(2) gerR_IN32_DisableMACIPAccess

Function	Disables MAC IP access.			
Call format	ERRCODE gerR_IN32_DisableMACIPAccess (VOID)			
Arguments	Name	Variable Name	Description	I/O
	None			
Return value	R_IN32_OK: Normal end			
Description	This function disables MAC IP access.			

(3) gerR_IN32_WritePHY

Function	Writes to PHY internal register.			
Call format	ERRCODE gerR_IN32_WritePHY (ULONG ulPort, ULONG ulAddr, ULONG ulData)			
Arguments	Name	Variable Name	Description	I/O
	ULONG	ulPort	Port subject to register writing R_IN32_PORT1(0): Port 1 R_IN32_PORT2(1): Port 2	Input
	ULONG	ulAddr	PHY register address	Input
	ULONG	ulData	Data to be written to PHY	Input
Return value	R_IN32_OK: Normal end R_IN32_ERR: Abnormal end (MDIO command end wait error)			
Description	This function writes to the PHY internal register in MDIO. *:Use this function during the period from MAC IP access enable (function gerR_IN32_EnableMACIPAccess) to MAC IP access disable (function gerR_IN32_DisableMACIPAccess). When the return value of this function is a value other than R_IN32_OK, the function calls the function below created by the vendor. Be sure to execute error processing in accordance with the error code. gR_IN32_CallbackFatalError			

(4) gerR_IN32_ReadPHY

Function	Reads PHY internal register.			
Call format	ERRCODE gerR_IN32_ReadPHY (ULONG ulPort, ULONG ulAddr, ULONG *ulData)			
Arguments	Name	Variable Name	Description	I/O
	ULONG	ulPort	Port subject to register reading R_IN32_PORT1(0): Port 1 R_IN32_PORT2(1): Port 2	Input
	ULONG	ulAddr	PHY register address	Input
	ULONG	*ulData	Data read from PHY	Output
Return value	R_IN32_OK: Normal end R_IN32_ERR: Abnormal end (MDIO command end wait error)			
Description	This function reads the PHY internal register in MDIO. *:Use this function during the period from MAC IP access enable (function gerR_IN32_EnableMACIPAccess) to MAC IP access disable (function gerR_IN32_DisableMACIPAccess). When the return value of this function is a value other than R_IN32_OK, the function calls the function below created by the vendor. Be sure to execute error processing in accordance with the error code. gR_IN32_CallbackFatalError			

4.5.10 Transient reception processing

(1) gerR_IN32_MainReceiveTransient1

Function	Main transient reception processing 1			
Call format	ERRCODE gerR_IN32_MainReceiveTransient1 (VOID)			
Arguments	Name	Variable Name	Description	I/O
	None			
Return value	R_IN32_OK: Normal end			
Description	This function acquires the transient frames received by R-IN32M3-CL.			

(2) gerR_IN32_MainReceiveTransient2

Function	Main transient reception processing 2			
Call format	ERRCODE gerR_IN32_MainReceiveTransient2 (VOID)			
Arguments	Name	Variable Name	Description	I/O
	None			
Return value	R_IN32_OK: Normal end			
Description	This function delivers the received transient frames acquired by the function gerR_IN32_MainReceiveTransient1 to the user program using the call-back function gerR_IN32_CallbackReceivedTransient.			

(3) gerR_IN32_EnableReceiveTransient

Function	Enables transient reception for vendor reasons.			
Call format	ULONG gerR_IN32_EnableReceiveTransient (BOOL blEnable)			
Arguments	Name	Variable Name	Description	I/O
	BOOL	blEnable	Enables reception. R_IN32_TRUE: Enable reception R_IN32_FALSE: Disable reception	Input
Return value	R_IN32_OK: Normal end			
Description	This function enables or disables transient reception for vendor reasons. When the return value of the function below created by the vendor is R_IN32_ERR, the transient reception for vendor reasons status is set to disabled. Be sure to set this function to "Reception enabled" if reception becomes enabled. gerR_IN32_CallbackReceivedTransient			

(4) gblR_IN32_GetReceiveTransientStatus

Function	Gets enabled/disabled status of transient reception for vendor reasons.			
Call format	BOOL gblR_IN32_GetReceiveTransientStatus (VOID)			
Arguments	Name	Variable Name	Description	I/O
	None			
Return value	Reception enabled/disabled status R_IN32_TRUE: Reception enabled R_IN32_FALSE: Reception disabled			
Description	This function acquires the enabled/disabled status of transient reception for vendor reasons.			

(5) gerR_IN32_SetMACAddressTableData

Function	Sets node information distribution data (MAC address table).			
Call format	ERRCODE gerR_IN32_SetMACAddressTableData (UCHAR uchSeqNumber, R_IN32_MACADDRESSDATA_T *pstMacAddrDat)			
Arguments	Name	Variable Name	Description	I/O
	UCHAR	uchSeqNumber	Sequential distribution number (value range: 1-7)	Input
	R_IN32_MACADDRESSDATA_T	*pstMacAddrDat	Information such as MAC address (MAC address table)	Input
Return value	R_IN32_OK: Normal end R_IN32_ERR_OUTOFRANGE: Node number out of range or sequential distribution number out of range			
Description	This function sets the information (MAC address table), such as the MAC address, acquired by node information distribution from the master station, and the sequential distribution number. *:Register the node number of the master station as 0x7D. If R_IN32_FALSE is set as a result of a (h) Distribute Node Information request in B) R_IN32_UNITINIT_T initial setup of the function gerR_IN32_Initialize, this function does not need to be called.			

Arguments of gerR_IN32_SetMACAddressTableData

The following describes the structure of R_IN32_MACADDRESSDATA_T based on the sample code.

```
/* Information (MAC address table), such as MAC address */
typedef struct _R_IN32_MACADDRESSDATA_TAG {
    USHORT    usNodeNumber;              /*!< Node number (1-120, master station: 0x7d)*/
    UCHAR     uchTransientReceiveEnable; /*!< Transient reception function
                                              (R_IN32_ENABLE/R_IN32_DISABLE) */
    UCHAR     auchMacAddress[6];         /*!< MAC address */
} R_IN32_MACADDRESSDATA_T;
```

4.5.11 Transient transmission processing

(1) gerR_IN32_GetUnitInformation

Function	Gets unit information.			
Call format	ERRCODE gerR_IN32_GetUnitInformation(R_IN32_UNITINFO_T *pstUnitInfo, R_IN32_UNITNETWORKSETTING_T*pstUnitNetworkSetting)			
Arguments	Name	Variable Name	Description	I/O
	R_IN32_UNITINFO_T	*pstUnitInfo	Unit information	Output
	R_IN32_UNITNETWORK SETTING_T	*pstUnitNetworkSetting	Network operation setting	Output
Return value	R_IN32_OK: Normal end			
Description	This function acquires the setting information of its own station. The acquired setting information is used when creating a Get Detailed Node Information response.			

Arguments of getR_IN32_GetUnitInformation

The following describes the structure of R_IN32_UNITNETWORKSETTING_T based on the sample code.

```
/* Network operation setting */
typedef struct R_IN32_UNITNETWORKSETTING_TAG {
    ULONG    ulFrameSendCount;      /*!< No. of transmissions during token hold */
    ULONG    ulFrameSendInterval;   /*!< Frame transmission interval */
    ULONG    ulTokenSendCount;      /*!< No. of token transmissions */
} R_IN32_UNITNETWORKSETTING_T;
```

(2) gusR_IN32_GetNodeID

Function	Gets node ID.			
Call format	USHORT gusR_IN32_GetNodeID (VOID)			
Arguments	Name	Variable Name	Description	I/O
	None			
Return value	Node ID			
Description	This function acquires the node ID. The acquired node ID is used when performing transient transmission.			

(3) gerR_IN32_GetMulticastMACAddress

Function	Gets multicast MAC address.			
Call format	ERRCODE gerR_IN32_GetMulticastMACAddress (UCHAR *puchMACAddr)			
Arguments	Name	Variable Name	Description	I/O
	UCHAR	*puchMACAddr	Multicast address When 13-34-56-78-90-AB is set, the following addresses are returned: puchMACAddr[0]:0x13 puchMACAddr[1]:0x34 puchMACAddr[2]:0x56 puchMACAddr[3]:0x78 puchMACAddr[4]:0x90 puchMACAddr[5]:0xAB	Output
Return value	R_IN32_OK: Normal end R_IN32_ERR: Abnormal end [The multicast MAC address cannot be acquired due to no data link (data link disconnected).]			
Description	This function acquires the multicast MAC address. The acquired multicast MAC address is used as the destination address when transient transmission is performed to all nodes connected to the network.			

(4) gerR_IN32_GetUnicastMACAddress

Function	Gets unicast MAC address.			
Call format	ERRCODE gerR_IN32_GetUnicastMACAddress (USHORT usNodeNumber,UCHAR *puchMACAddr)			
Arguments	Name	Variable Name	Description	I/O
	USHORT	usNodeNumber	Node number (value range: 1-120, master station: 0x7D)	
	UCHAR	*puchMACAddr	Unicast address When 13-34-56-78-90-AB is set, the following addresses are returned: puchMACAddr[0]:0x12 puchMACAddr[1]:0x34 puchMACAddr[2]:0x56 puchMACAddr[3]:0x78 puchMACAddr[4]:0x90 puchMACAddr[5]:0xAB	Output
Return value	R_IN32_OK: Normal end R_IN32_ERR_NOENTRY: No entry R_IN32_ERR_OUTOFRANGE: Node number out of range			
Description	This function acquires the unicast MAC address corresponding to the node number from the node information distribution received from the master station. *:When there is no data link (data link disconnected), the unicast MAC address cannot be acquired (the return value becomes R_IN32_ERR_NOENTRY). Be sure to set the node number of the master station to 0x7D.			

(5) gerR_IN32_GetSendTransientBuffer

<table>
<tr><td>Function</td><td colspan="4">Gets transient transmission buffer.</td></tr>
<tr><td>Call format</td><td colspan="4">ERRCODE gerR_IN32_GetSendTransientBuffer(USHORT usSize, VOID** ppvSendBuffAddr,
UCHAR *puchSendBuffNo, UCHAR *puchConnectionInfo)</td></tr>
<tr><td rowspan="5">Arguments</td><td>Name</td><td>Variable Name</td><td>Description</td><td>I/O</td></tr>
<tr><td>USHORT</td><td>usSize</td><td>Transmission data size excluding DCS/FCS</td><td>Input</td></tr>
<tr><td>VOID</td><td>**ppvSendBuffAddr</td><td>Transient transmission buffer address</td><td>Output</td></tr>
<tr><td>UCHAR</td><td>*puchSendBuffNo</td><td>Transient transmission buffer number</td><td>Output</td></tr>
<tr><td>UCHAR</td><td>*puchConnectionInfo</td><td>Transient connection information</td><td>Output</td></tr>
<tr><td>Return value</td><td colspan="4">R_IN32_OK: Normal end (transient transmission buffer acquired)
R_IN32_ERR: Abnormal end (transient transmission buffer acquisition error)</td></tr>
<tr><td>Description</td><td colspan="4">This function inquires whether or not there is space in the transient transmission area for transmission of the "transmission data size," and returns the following information if there is space:
Transient transmission buffer address
Transient transmission buffer number
Transient connection information

*:In the following cases, transient transmission cannot be performed and the process ends in error (R_IN32_ERR: abnormal end).
When there is no data link (data link disconnected)
When the transmission data size is greater than 1,510 bytes

When you want to perform transient transmission, execute the following:
Acquire the transient transmission buffer number using this function.
Store the transmission data in the acquired transient transmission buffer.
Request transient transmission using the function gerR_IN32_RequestSendingTransient.</td></tr>
</table>

(6) gerR_IN32_RequestSendingTransient

Function	Requests transient transmission.			
Call format	ERRCODE gerR_IN32_RequestSendingTransient (UCHAR uchSendBuffNo, USHORT usSize)			
Arguments	Name	Variable Name	Description	I/O
	UCHAR	uchSendBuffNo	Transient transmission buffer number	Input
	USHORT	usSize	Transmission data size excluding DCS/FCS	Input
Return value	R_IN32_OK: Normal end R_IN32_ERR: Abnormal end (transient transmission request error)			
Description	This function specifies transmission to the transient transmission buffer number acquired by the function gerR_IN32_GetSendTransientBuffer. If you want to perform transient transmission, execute the following: Acquire the transient transmission buffer using the function gerR_IN32_GetSendTransientBuffer. Store the transmission data in the acquired transient transmission buffer. Request transient transmission using this function. *:In the following case, transient transmission cannot be performed and the process ends in error (R_IN32_ERR: Abnormal end): When there is no data link (data link disconnected) Any error that occurs after transmission is requested by this function is notified by the return value of the function gerR_IN32_MainSendTransient. Set the transmission size to the same size as the value specified in gerR_IN32_GetSendTransientBuffer.			

(7) gerR_IN32_MainSendTransient

Function	Main transient transmission processing			
Call format	ULONG gerR_IN32_MainSendTransient (VOID)			
Arguments	Name	Variable Name	Description	I/O
	None			
Return value	R_IN32_OK: Normal end			
Description	This function acquires the transient transmission end result. This function calls the function gerR_IN32_CallbackTransientSendingComplete to issue a notification regarding the status (transmission result) of the target transmission descriptor.			

4.5.12 Interrupts

(1) gerR_IN32_DisableInterrupt

Function	Disables interrupts.			
Call format	ERRCODE gerR_IN32_DisableInterrupt (VOID)			
Arguments	Name	Variable Name	Description	I/O
	None			
Return value	R_IN32_OK: Normal end			
Description	This function disables interrupts.			

(2) gerR_IN32_EnableInterrupt

Function	Enables interrupts.			
Call format	ERRCODE gerR_IN32_EnableInterrupt (VOID)			
Arguments	Name	Variable Name	Description	I/O
	None			
Return value	R_IN32_OK: Normal end			
Description	This function enables interrupts.			

4.5.13 Hardware tests

(1) gerR_IN32_IEEETest

Function	IEEE 802.3ab compliance test			
Call format	ERRCODE gerR_IN32_IEEETest (USHORT usMode)			
Arguments	Name	Variable Name	Description	I/O
	USHORT	usMode	IEEE 802.3ab compliance test mode R_IN32_IEEE_MODE1(1):MODE1 R_IN32_IEEE_MODE2(2):MODE2 R_IN32_IEEE_MODE3(3):MODE3 R_IN32_IEEE_MODE4(4):MODE4 R_IN32_IEEE_END(5): Test end	Input
Return value	R_IN32_OK: Normal end R_IN32_ERR: Abnormal end			
Description	This function sets the waveform output for test mode in PHY in accordance with the IEEE 802.3ab compliance test mode of the argument. *:When the return value of this function is a value other than R_IN32_OK, the function calls the function below created by the vendor. Be sure to execute error processing in accordance with the error code. gR_IN32_CallbackFatalError			

(2) gerR_IN32_InitializeLoopBackTest

Function	Initializes internal loop-back / external loop-back communication test.			
Call format	ERRCODE gerR_IN32_InitializeLoopBackTest (VOID)			
Arguments	Name	Variable Name	Description	I/O
	None			
Return value	R_IN32_OK: Normal end R_IN32_ERR: Abnormal end			
Description	This function performs initialization for executing the internal loop-back / external loop-back communication test. *:When the return value of this function is a value other than R_IN32_OK, the function calls the function below created by the vendor. Be sure to execute error processing in accordance with the error code. gR_IN32_CallbackFatalError			

(3) gerR_IN32_InternalLoopBackTest

Function	Internal loop-back communication test			
Call format	ERRCODE gerR_IN32_InternalLoopBackTest (ULONG ulPort)			
Arguments	Name	Variable Name	Description	I/O
	ULONG	ulPort	Test target port R_IN32_PORT1(0) : Port 1 R_IN32_PORT2(1) : Port 2	Input
Return value	R_IN32_OK: Normal end R_IN32_ERR: Abnormal end			
Description	This function sends a frame from the test target port specified in the argument, and verifies the received result by internal loop-back. *:When the return value of this function is a value other than R_IN32_OK, the function calls the function below created by the vendor. Be sure to execute error processing in accordance with the error code. gR_IN32_CallbackFatalError			

(4) gerR_IN32_ExternalLoopBackTest

Function	External loop-back communication test			
Call format	ERRCODE gerR_IN32_ExternalLoopBackTest (ULONG ulPort)			
Arguments	Name	Variable Name	Description	I/O
	ULONG	ulPort	Source port R_IN32_PORT1(0) : Port 1 R_IN32_PORT2(1) : Port 2	Input
Return value	R_IN32_OK: Normal end R_IN32_ERR: Abnormal end			
Description	This function sends a frame from the test target port specified in the argument, and verifies the received result using the other port. When implementing this test, connect port 1 and port 2 using an Ethernet cable. *:When the return value of this function is a value other than R_IN32_OK, the function calls the function below created by the vendor. Be sure to execute error processing in accordance with the error code. gR_IN32_CallbackFatalError			

4.6 Customizing the Target-Dependent Function Group for the R-IN32M3-CL Driver

4.6.1 Changing the header file

Change each item defined in the header file "R_IN32M3Function.h"<R> in accordance with the system environment of the vendor.

(1) R-IN32M3-CL address setting

[1] R-IN32M3-CL head address

Specifies the address for R-IN32M3-CL access by the R-IN32M3-CL driver.

```
#define     R_IN32_BASE_ADR                 0x40100000<R>   /* R-IN32M3-CL head address */
```

(2) PHY reset setting

Defines the setup for resetting PHY during initialization.

[1] PHY reset assert time setting

Sets the time at which the R-IN32M3-CL driver is to assert the PHY reset signal in units of µs. The assertion time varies according to the PHY used. See the manual of your PHY.

```
#define R_IN32_WAITUS_PHYRESET_ASSERT       10000UL     /* PHY reset assertion time */
```

[2] Time after PHY reset clear to normal operation

Specifies the time after PHY reset is cleared by the R-IN32M3-CL driver to normal PHY operation in units of µs.

The time after reset clear to normal PHY operation varies according to the PHY used. See the manual of your PHY.

```
#define     R_IN32_WAITUS_PHYRESET_END          5000UL
                                                /* Time after PHY reset clear to normal operation */
```

(3) No. of transient reception buffers

Defines the number of transient reception buffers.
The R-IN32M3-CL driver uses an area (memory) equivalent to
R_IN32_TRANSIENT_BUFFER_NUM x 1,520 bytes.
Set a value greater than or equal to 2.

```
#define     R_IN32_TRANSIENT_BUFFER_NUM     (64)        /* No. of transient reception buffers */
```

4.6.2 Creating a target-dependent function group for the R-IN32M3-CL driver

Point

Be sure to implement the target-dependent function group in "Table 4.11Target-Dependent Function Group for R-IN32M3-CL Driver."

The target-dependent function group for the R-IN32M3-CL driver must be customized in accordance with the target hardware environment. The following lists the functions to be customized by the vendor.

Table 4.11 Target-Dependent Function Group for R-IN32M3-CL Driver

Function Category	Function Name	Function Type	Overview
Wait processing	gR_IN32R_WaitUS	VOID	Waits a period of time.
Time measurement	gR_IN32R_StartStopwatchTimer	VOID	Starts time measurement.
	gR_IN32R_GetElapsedTime	VOID	Acquires the elapsed time.
Interrupt	gR_IN32R_DisableInt	VOID	Disables interrupts.
	gR_IN32R_EnableInt	VOID	Enables interrupts.
Hardware test	gerR_IN32R_IEEETest	ERRCODE	IEEE 802.3ab compliance test

(1) gR_IN32R_WaitUS

Function	Waits a certain period of time.			
Call format	VOID gR_IN32R_WaitUS (ULONG ulWaitTime)			
Arguments	Name	Variable Name	Description	I/O
	ULONG	ulWaitTime	Wait time (µs)	Input
Return value	None			
Description	This function waits for the wait time specified in the argument to elapse. *:The maximum wait time used by the R-IN32M3-CL driver is 10 ms (10000 UL). If the assertion time of the used PHY is longer than 10 ms (10000 UL), change the value below so that that value can be counted: #define R_IN32_WAITUS_PHYRESET_ASSERT 10000UL /* PHY reset assertion time */ [For details, see Section (2) of Section 4.6.1 "PHY reset setting."]			

(2) gR_IN32R_StartStopwatchTimer

Function	Starts time measurement.			
Call format	VOID gR_IN32R_StartStopwatchTimer (R_IN32R_STOPWATCH_T *pstStopWatch,ULONG ulUnit)			
Arguments	Name	Variable Name	Description	I/O
	R_IN32R_STOPWATCH_T*	pstStopWatch	Stopwatch work area	Input/Output
	ULONG	ulUnit	Measurement unit (1: µs)	Input
Return value	None			
Description	This function starts time measurement.			

Arguments of gR_IN32R_StartStopwatchTimer

The following describes the configuration of R_IN32R_STOPWATCH_T based on the sample code.

```
typedef struct _R_IN32R_STOPWATCH_TAG {
    ULONG    ulUnit;              /* Unit of measured time */
    ULONG    ulFirstTmr1Cnt;      /* General-purpose timer 1 counter value (at startup) */
    ULONG    ulLastTmr1Cnt;       /* General-purpose timer 1 counter value (previous value) */
} R_IN32R_STOPWATCH_T;
```

(3) gR_IN32R_GetElapsedTime

Function	Gets elapsed time.			
Call format	VOID gR_IN32R_GetElapsedTime (R_IN32R_STOPWATCH_T *pstStopWatch,ULONG *pulElapsedTime)			
Arguments	Name	Variable Name	Description	I/O
	R_IN32R_STOPWATCH_T	*pstStopWatch	Stopwatch work area	Input/Output
	ULONG	*pulElapsedTime	Elapsed time (Unit: The unit specified by the function gR_IN32R_StartStopwatchTimer.)	Output
Return value	None			
Description	This function acquires the elapsed time after the time measurement startup function gR_IN32R_StartStopwatchTimer is called. The R-IN32M3-CL driver monitors timeouts using the functions gR_IN32R_StartStopwatchTimer and gR_IN32R_GetElapsedTime. *:Implement the function so that Unsigned Long (0-4294967295) can be counted. If timeout monitoring is not required, set *ulElapsedTime to "0" (elapsed time: 0 µs).			

(4) gR_IN32R_DisableInt

Function	Disables interrupts.			
Call format	VOID gR_IN32R_DisableInt (VOID)			
Arguments	Name	Variable Name	Description	I/O
	None			
Return value	None			
Description	This function disables interrupts. *:This function is a dummy function. Regard the processing as no processing.			

(5) gR_IN32R_EnableInt

Function	Enables interrupts.			
Call format	VOID gR_IN32R_EnableInt (VOID)			
Arguments	Name	Variable Name	Description	I/O
	None			
Return value	None			
Description	This function enables interrupts. *:This function is a dummy function. Regard the processing as no processing.			

(6) gerR_IN32R_IEEETest

Function	IEEE 802.3ab compliance test			
Call format	ERRCODE gerR_IN32R_IEEETest (USHORT usIEEETestMode)			
Arguments	Name	Variable Name	Description	I/O
	USHORT	usIEEETestMode	IEEE 802.3ab compliance test mode R_IN32R_IEEE_MODE1(1):MODE1 R_IN32R_IEEE_MODE2(2):MODE2 R_IN32R_IEEE_MODE3(3):MODE3 R_IN32R_IEEE_MODE4(4):MODE4 R_IN32R_IEEE_END(5): Test end	Input
Return value	R_IN32_OK: Normal end R_IN32_ERR: Abnormal end			
Description	This function specifies the waveform output for test mode in PHY in accordance with the IEEE 802.3ab compliance test mode of the argument. The processing differs according to PHY. Implement the function after verifying the specification of the PHY used.			

4.7 Customizing the Call-Back Function Group for the R-IN32M3-CL Driver

The internal processing of the call-back function group for the R-IN32M3-CL driver needs to be customized by the vendor.

The following lists the call-back functions to be called by the R-IN32M3-CL driver.

Table 4.12 List of Call-Back Functions Used by R-IN32M3-CL Driver

Function Category	Function Name	Function Type	Overview
Error processing	gR_IN32_CallbackFatalError	VOID	Acquires R-IN32M3-CL fatal errors.
Own station status acquisition	gerR_IN32_CallbackCommandFromMaster	ERRCODE	Acquires commands from the master station.
	gerR_IN32_CallbackNodeAndNetwork Number	ERRCODE	Changes the node number and network number from the master station.
Transient transmission/ reception	gerR_IN32_CallbackReceivedTransient	ERRCODE	Acquires received transient frames.
	gerR_IN32_CallbackTransientSending Complete	ERRCODE	Acquires the transient transmission completion status.

(1) gR_IN32_CallbackFatalError

Function	Gets R-IN32M3-CL fatal errors.			
Call format	VOID gR_IN32_CallbackFatalError (ULONG ulErrorCode, ULONG ulErrorInfo)			
Arguments	Name	Variable Name	Description	I/O
	ULONG	ulErrorCode	Fatal error code	Input
	ULONG	ulErrorInfo	Fatal error information (Address of function when error occurred.)	Input
Return value	None			
Description	This function acquires R-IN32M3-CL fatal errors. The R-IN32M3-CL driver calls this function when a R-IN32M3-CL fatal error is detected. Function internal processing is freely implemented by the vendor.			

Table 4.13 List of Fatal Error Codes of gR_IN32_CallbackFatalError Function

Fatal Error Code (ulErrorCode)	Fatal Error Information (ulErrorInfo)	Fatal Error Description	Processing
D529	Driver internal startup source function Address of the function gerR_IN32D_ClearTxRxRAM	Communication LSI error	・The error is most likely a malfunction caused by noise, etc. Check the distance between lines and cables as well as device grounding, and implement noise countermeasures accordingly. ・Implement a module unit test. If the error occurs again, most likely the hardware of the module is faulty.
D52A	Driver internal startup source function Address of the function erR_IN32D_MDIO_WaitCommandComplete	Communication LSI error	
D52B	Driver internal startup source function Address of the function erR_IN32D_ResetMAC	Communication LSI error	
D52C	Driver internal startup source function Address of the function gerR_IN32D_StartRing	Communication LSI error	

(2) gerR_IN32_CallbackCommandFromMaster

<table>
<tr><td>Function</td><td colspan="4">Gets command from master station.</td></tr>
<tr><td>Call format</td><td colspan="4">ERRCODE gerR_IN32_CallbackCommandFromMaster (ULONG pulCommand)</td></tr>
<tr><td rowspan="2">Arguments</td><td>Name</td><td>Variable Name</td><td>Description</td><td>I/O</td></tr>
<tr><td>ULONG</td><td>pulCommand</td><td>Command status from master module
ulCommand
Bit 0: Stop cyclic communication instruction (node number out of range)
1: Stop instruction
Bit 1:Stop cyclic communication instruction (reserved node setting)
1: Stop instruction
Bit 2 :Stop cyclic communication instruction (master station instruction)
1: Stop instruction
Bit 3: Stop cyclic communication instruction (node number duplication)
1: Stop instruction
Bits 15-4: Reserved
Bit 16: Node type invalid (own station node type does not match node type specified by master station)
1: Node type invalid
Bit 17: Specified size invalid (The cyclic communication size specified by the master station is greater than the allowable maximum size (size specified by the function gerR_IN32_initialize) for own station cyclic communication.)
1: Specified size invalid
Bits 31-18: Reserved</td><td>Input</td></tr>
<tr><td>Return value</td><td colspan="4">R_IN32_OK: Normal end</td></tr>
<tr><td>Description</td><td colspan="4">This function acquires the command by parameter frame reception from the master station.

The R-IN32M3-CL driver calls this function when a parameter frame is received from the master station. Function internal processing is freely implemented by the vendor.</td></tr>
</table>

(3) gerR_IN32_CallbackNodeAndNetworkNumber

Function	Changes node number and network number from master station.			
Call format	ERRCODE gerR_IN32_CallbackNodeAndNetworkNumber (UCHAR uchNetworkNumber,USHORT usNodeNumber)			
Arguments	Name	Variable Name	Description	I/O
	UCHAR	uchNetworkNumber	Network number (value range: 1-239)	Input
	USHORT	usNodeNumber	Node number (value range: 1-120)	Input
Return value	R_IN32_OK: Normal end			
Description	This function issues a notification of a change in the node number and network number by reception of a parameter frame from the master station. The R-IN32M3-CL driver calls this function when it receives a parameter frame from the master station and the node number and network number are changed. Function internal processing is freely implemented by the vendor. *:When (g) Enable node number and network number setting from master station of B) R_IN32_UNITINIT_T initial setup of the function gerR_IN32_Initialize is set to R_IN32_FALSE (no function), the master station does not change the node number or network number. In such a case, this function does not need to be processed.			

(4) gerR_IN32_CallbackReceivedTransient

Function	Gets received transient frames.			
Call format	ERRCODE gerR_IN32_CallbackReceivedTransient (VOID *pvRcv, USHORT usFrameSize)			
Arguments	Name	Variable Name	Description	I/O
	VOID	*pvRcv	Reception buffer	Input
	USHORT	usFrameSize	Frame size excluding FCS	Input
Return value	R_IN32_OK: Normal end R_IN32_ERR: Abnormal end			
Description	This function acquires received transient frames. The R-IN32M3-CL driver calls this function when a transient frame is received. Function internal processing is freely implemented by the vendor. *1:Set the head address of the reception buffer in increments of 4 bytes (0 or multiple of 4). If the return value is a value other than R_IN32_OK, the transient reception enable status for vendor reasons is set to reception disabled. Be sure to set it to “Reception enabled” by calling the function gerR_IN32_EnableReceiveTransient after it becomes to be able to receive.			

(5) gerR_IN32_CallbackTransientSendingComplete

<table>
<tr><td>Function</td><td colspan="4">Gets transient transmission completion status.</td></tr>
<tr><td>Call format</td><td colspan="4">ERRCODE gerR_IN32_CallbackTransientSendingComplete
(UCHAR uchSendBuffNo, ERRCODE erSendStatus)</td></tr>
<tr><td rowspan="3">Arguments</td><td>Name</td><td>Variable Name</td><td>Description</td><td>I/O</td></tr>
<tr><td>UCHAR</td><td>uchSendBuffNo</td><td>Transient transmission buffer number</td><td>Input</td></tr>
<tr><td>ERRCODE</td><td>erSendStatus</td><td>Status of target transient transmission buffer (transmission result)

R_IN32_OK:
Transient transmission normal completion
R_IN32_ERR:
Transient transmission abnormal completion</td><td>Input</td></tr>
<tr><td>Return value</td><td colspan="4">R_IN32_OK: Normal end</td></tr>
<tr><td>Description</td><td colspan="4">This function acquires the transmission status (transmission result) of the transient transmission buffer.

The R-IN32M3-CL driver calls this function when transmission of a transient frame ends.
Function internal processing is freely implemented by the vendor.</td></tr>
</table>

REVISION HISTORY	R-IN32M3 Series CC-Link IE Field Intelligent device station

Rev.	Date	Description	
		Page	Summary
1.00	Jul 26, 2013	-	First edition issued
2.00	Dec 25, 2014	2	Modification of PHY address 2 of ***Table 1.2 Circuit Design Check Sheet***
		6	Modification of product name of ***Figure 2.1 External AND Logic for Turning L.ERR On***
		9	Modification of product name ***Figure 3.1 Transient1 Response Procedure (Request Source: Master Station)***
		10	Modification of product name ***Figure 3.2 Transient2 Response Procedure***
		140	Modification of header file name and R-IN32M3-CL head address of ***4.6.1 Changing the header file***

[Memo]

SALES OFFICES

Renesas Electronics Corporation

http://www.renesas.com

Renesas Electronics America Inc.
2880 Scott Boulevard Santa Clara, CA 95050-2554, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130
Renesas Electronics Canada Limited
1101 Nicholson Road, Newmarket, Ontario L3Y 9C3, Canada
Tel: +1-905-898-5441, Fax: +1-905-898-3220
Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-651-700, Fax: +44-1628-651-804
Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-65030, Fax: +49-211-6503-1327
Renesas Electronics (China) Co., Ltd.
7th Floor, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100083, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679
Renesas Electronics (Shanghai) Co., Ltd.
Unit 204, 205, AZIA Center, No.1233 Lujiazui Ring Rd., Pudong District, Shanghai 200120, China
Tel: +86-21-5877-1818, Fax: +86-21-6887-7858 / -7898
Renesas Electronics Hong Kong Limited
Unit 1601-1613, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2886-9318, Fax: +852 2886-9022/9044
Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670
Renesas Electronics Singapore Pte. Ltd.
80 Bendemeer Road, Unit #06-02 Hyflux Innovation Centre Singapore 339949
Tel: +65-6213-0200, Fax: +65-6213-0300
Renesas Electronics Malaysia Sdn.Bhd.
Unit 906, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510
Renesas Electronics Korea Co., Ltd.
11F., Samik Lavied' or Bldg., 720-2 Yeoksam-Dong, Kangnam-Ku, Seoul 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

R-IN32M3 Series
User's Manual
CC-Link IE Network Field

Renesas Electronics Corporation