User's Manual

R-IN32M4-CL3

User's Manual: CC-Link IE TSN edition

Document number: R18UZ0070EJ0106
Issue date: Jul 29, 2022

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 or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving 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, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. 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 shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
5. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended 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; industrial robots; etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or 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 damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
6. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, 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 and impractical, you are responsible for evaluating 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. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technologies shall 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 shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
11. This document shall not be reprinted, 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.

(Note1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.
(Note2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.4.0-1 November 2017)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan

www.renesas.com

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.

Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit:
www.renesas.com/contact/.

General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Precaution against Electrostatic Discharge (ESD)
   A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.
2. Processing at power-on
   The state of the product is undefined at the time 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 time 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 time 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 time when power is supplied until the power reaches the level at which resetting is specified.
3. Input of signal during power-off state
   Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.
4. Handling of unused pins
   Handle unused pins in accordance 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 the LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible.
5. Clock signals
   After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is 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. Additionally, 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.
6. Voltage application waveform at input pin
   Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (Max.) and V_{IH} (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (Max.) and V_{IH} (Min.).
7. Prohibition of access to reserved addresses
   Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.
8. Differences between products
   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

How to Use This Manual

1. Purpose and Target Readers

This manual is intended for users who wish to understand the remote device station communication functions of CC-Link IE TSN of the "R-IN32M4-CL3". Target users are expected to understand the fundamentals of electrical circuits, logic circuits, and microcomputers.

When designing an application system that includes this MCU, take all points to note into account.
Points to note are given in their contexts and at the final part of each section, and in the section giving usage notes.

The list of revisions is a summary of major points of revision or addition for earlier versions. It does not cover all revised items. For details on the revised points, see the actual locations in the manual.

Literature — 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 R-IN32M4-CL3

Document Name	Document Number
R-IN32M4-CL3 User's Manual: Hardware	R18UZ0073EJ****
R-IN32M4-CL3 User's Manual: Gigabit Ethernet PHY	R18UZ0075EJ****
R-IN32M4-CL3 User's Manual: Board Design	R18UZ0074EJ****
R-IN32M4-CL3 User's Manual: CC-Link IE TSN	This Manual
R-IN32M4-CL3 User's Manual: CC-Link IE Field	R18UZ0071EJ****
R-IN32M4-CL3 Programming Manual: Driver	R18UZ0076EJ****
R-IN32M4-CL3 Programming Manual: OS	R18UZ0072EJ****

2. Numbers and Symbols

Data significance: Higher digits on the left and lower digits on the right

Active low representation:

xxxZ (capital letter Z after pin or signal name)

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

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

Note:

Footnote for item marked with Note in the text

Caution:

Information requiring particular attention

Remark:

Supplementary information

Numeric representation:

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

Decimal … xxxx

Hexadecimal … xxxxH or n'hxxxx (n bits)

Prefix indicating power 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:

Word … 32 bits

Halfword … 16 bits

Byte … 8 bits

[Remark]

Communication speed

10 Mbps = 10×10^6 bps

100 Mbps = 100×10^6 bps

Contents

1. Introduction

1.1 Terms

Unless otherwise specified, this manual uses the following terms.

Term	Description
CANopen	A higher-layer protocol based on the CAN. CANopen in CC-Link IE TSN applies the application layer of the CANopen protocol to CC-Link IE TSN to send and receive communication objects (SDO and PDO).
GX Works 3	The product name of the sequencer software package for the MELSEC
IEEE1588	The standard protocol for synchronizing clocks between nodes in Ethernet
IEEE802.1AS	The standard for transport of precise timing and synchronization in Audio/Video Bridging (AVB) networks
Management I/F	An interface for accessing PHY registers from R-IN32M4-CL3. It consists of the MDIO and MDC.
PDO	Process data objects (PDOs) refers to the aggregate of application objects which are periodically transferred between several CANopen nodes. Data are sent as a TPDO (transmitted PDO) and received as an RPDO (received PDO) from the given TPDO.
R-IN32M4-CL2	An industrial Ethernet communications LSI from Renesas Electronics Corporation
R-IN32M4-CL3	A Gigabit Ethernet PHY (GbE-PHY) built-in communications LSI for remote stations of CC-Link IE TSN
R-IN32M4-CL3 application circuit	A communications circuit of CC-Link IE TSN that consists of R-IN32M4-CL3 and peripheral devices
R-IN32M4-CL3 application product	A CC-Link IE TSN compatible product manufactured with reference to this manual
RWr	A remote register of link devices. Information that is input from a slave station to the master station in 16-bit (one word) units.
RWw	A remote register of link devices. Information that is output from the master station to a slave station in 16-bit (one word) units.
RX	Remote input from link devices. Information that is input from a slave station to the master station in bit units.
RY	Remote output from link devices. Information that is output from the master station to a slave station in bit units.
SDO	A message for access to object entries in the CANopen node object dictionary. SDOs are assigned to transient SLMP frames for transmission and reception to and from other stations independently of the communications period.
End user	A purchaser and user of products which support CC-Link family connection developed by users
Grand master	The device to be the source of time synchronization in use of the PTP.
Cyclic transfer	The function by which data are periodically communicated among stations on the same network using link devices or CANopen's PDO.
Slave station	Stations other than a master station: local station and remote station

Term	Description
Device	Various memories (X, Y, M, D, or others) in a sequencer CPU, or memories in a user application where data communicated with R-IN32M4-CL3 are stored.
Transient transfer	The function to communicate with another station when requested by a user application
Buffer memory	Memory in a user application, where data (such as setting values and monitoring values) are stored
Master station	A station that controls CC-Link IE TSN. This station can perform cyclic and transient transfer with all stations.
Multicast filter	This filtering function is to select whether to send cyclic data for multicasting which have been received by a home station to the subsequent station. Since this function is automatically set up by the master station in accord with the system configuration, it does not require parameter settings.
User	A manufacturer who develops and sells CC-Link family connection supporting products based on this manual. The terms, vendor and partner manufacturer, are used with the same meaning.
Remote station	A station that performs cyclic and transient transfer with the master station
Link device	Devices (RX, RY, RWr, and RWw) in a network unit
Local station	A station that performs cyclic and transient transfer with the master station and other local stations
Safety PDU	Data used for safety communications
Safety protocol stack	Specific library software to perform safety communications over CC-Link IE network. Safety devices can perform CC-Link IE safety communications easily by implementing a safety protocol stack on them.
Safety output	A signal output from the safety station to execute safety function
Safety communications	A function to send/receive safety PDUs between safety stations on the same network
Safety input	A signal input to the safety station to execute the safety function
Standard communications	Communications (such as cyclic transmission and transient transmission over CC-Link IE TSN) other than safety communications
Disconnection	The process of stopping a data link in response to a data link error.
Station	An element that constitutes the network and sends, receives, and transfers data. The term, node, is used with the same meaning.
Station number	An identifier for uniquely identifying a station in the network
Home station	A remote station to be developed based on this manual
Another station	A station other than own station
CC-Link IE TSN Class	A rank of modules and switching hubs supporting CC-Link IE TSN based on their functions and performance. There are two Classes: CC-Link IE TSN Class A and CC-Link IE TSN Class B. R-IN32M4-CL3 can be used for the development of devices which are ranked as CC-Link IE TSN Class B/A.
Reconnection	The process of restoring a data link when a station has recovered from an error.
Reserved station	A station that is not connected to a network, but is included in the number of network units as a station to be connected in the future

1.2 General Terms and Abbreviations

Unless otherwise specified, this manual uses the following general terms and abbreviations

General Term/Abbreviation	Description
CAN	An abbreviation for Controller Area Network
CSP+	An abbreviation for Control & Communication System Profile. This specification is for describing the information required for CC-Link family compatible device startup, operation, and maintenance.
GbE-PHY	An abbreviation for Gigabit Ethernet PHY. In this manual, it refers to the one that has GMII and is compatible with 1000BASE-T.
GMII	An abbreviation for Gigabit Media Independent Interface. This interface is for communicating data between the MAC port (MAC layer) and PHY (physical layer) of R-IN32M4-CL3.
MDC	An abbreviation for Management Data Clock. It is an MDIO clock specified in GMII. It constitutes Management I/F together with MDIO.
MDI	An abbreviation for Medium Dependent Interface. It is an interface for communicating data between R-IN32M4-CL3 and the pulse transformer and between the pulse transformer and the RJ-45 connector.
MDIO	An abbreviation for Management Data Input/Output. It is a data input/output bus for accessing the PHY registers specified in GMII. It constitutes Management I/F together with MDC.
MIB	An abbreviation for Management Information Base. It is a management information base for saving the communications state of R-IN32M4-CL3.
PHY	An abbreviation for Physical layer. In this manual, it refers to a portion of R-IN32M4-CL3 functions that convert logic signals to actual electrical signals in an interface such as Ethernet.
PTP	An abbreviation for Precision Time Protocol. This protocol is used to synchronize the time among devices in a network.
RSPDU	An abbreviation for Received Safety Protocol Data Unit. This is the safety PDU that the own station receives.
SLMP	An abbreviation for SeamLess Message Protocol. This protocol is for accessing an SLMP compatible device and a sequencer connected to an SLMP compatible device from an external device.
SSPDU	An abbreviation for Send Safety Protocol Data Unit. This is the safety PDU sent by the own station.
UTC	An abbreviation for Coordinated Universal Time. The time to which leap seconds are added as required to adjust for differences between GMT (Greenwich Mean Time) and the precise time.
WDC	An abbreviation for Watchdog Counter
Data link	A general term for cyclic transfer and transient transfer
Safety station	A generic term for a station that performs safety communications and standard communications.

1.3 Related Manuals

This manual does not include CC-Link IE TSN details such as terminology and functions. If necessary, download and refer to the related manuals from the following.
Mitsubishi Electric Factory Automation website (http://www.MitsubishiElectric.co.jp/fa)

Manual Title (Manual Number)	Description
MELSEC iQ-R CC-Link IE TSN User's Manual (Startup) (SH-082127ENG)	Describes the CC-Link IE TSN specifications, procedures from preparation to operation, system configuration, wiring, and communication examples.
MELSEC iQ-R CC-Link IE TSN User's Manual (Application) (SH-082129ENG)	Describes the CC-Link IE TSN functions, parameter settings, programming, troubleshooting, input/output signals, buffer memory, and the like.
SLMP Reference Manual (SH-080956ENG)	Describes the protocol (SLMP) used for data reading and writing with SLMP compatible devices from an external device.

1.4 CC-Link Partner Association (CLPA)

(1) Specifications

The materials related to this manual include the specifications published by the CC-Link Partner Association below. For CC-Link IE TSN and SLMP details, download and refer to the following documents from the CC-Link Association website.

Document Title	Document No.
CC-Link IE TSN Specification (Overview)	BAP-C2011ENG-001
CC-Link IE TSN Specification (Physical Layer/Data Link Layer)	BAP-C2011ENG-002
CC-Link IE TSN Specification (Application Layer Service)	BAP-C2011ENG-003
CC-Link IE TSN Specification (Application Layer Protocol)	BAP-C2011ENG-004
CC-Link IE TSN Specification (Communication Profile)	BAP-C2011ENG-005
CC-Link IE TSN Specification (Implementation Rules)	BAP-C2011ENG-006
SLMP (Seamless Message Protocol) Specification (Overview)	BAP-C2006ENG-001
SLMP (Seamless Message Protocol) Specification (Services)	BAP-C2006ENG-002
SLMP (Seamless Message Protocol) Specification (Protocol)	BAP-C2006ENG-003
CC-Link IE Safety Communication Function Specification (Overview)	BAP-C2007ENG-001
CC-Link IE Safety Communication Function Specification (Application Layer Service and Protocol)	BAP-C2007ENG-002
CC-Link IE Safety Communication Function Specification (Communication profile)	BAP-C2007ENG-003
CC-Link IE Safety Communication Function Specification (Implementation Rules)	BAP-C2007ENG-004
CC-Link IE Safety Communication Function Specification (Communication Data Format)	BAP-C2007ENG-005

(2) Conformance Test

When a product is developed based on the information in this manual, the product must undergo a conformance test implemented by the CC-Link Partner Association. For conformance test details, download and refer to the following documents from the CC-Link Partner Association website.

Document Title	Document No.
CC-Link IE TSN Conformance Test Specifications Remote Station Version (Twist Pair Cable)	BAP-C0401ENG-049-A
CC-Link IE Safety Communication Function and CC-Link IE TSN Conformance Test Request	BAP-C0401ENG-065

(3) Creating a Control & Communication System Profile (CSP+)

The conformance test includes verification of CSP+. CSP+ must be created in advance. For CSP+ details, download and refer to the following specification from the CC-Link Partner Association website. Also, download and utilize the following related material and tools from the same website, which are available as an aid to CSP+ file creation.

Document Title/Related Material & Tool	Document No.
Control & Communication System Profile Specification	BAP-C2008ENG-001
Control & Communication System Profile Creation Guidelines	—
CSP+ Profile Creation support Tool	—
Sample CSP+ Files	—
CSP+ Templates	—

(4) Inquiries

To request materials published by the CC-Link Partner Association (CLPA) and for conformance test details, please contact the following:

CC-Link Partner Association	TEL: +81-52-919-1588
	FAX: +81-52-916-8655
	E-mail: info@cc-link.org
	Web: http://www.cc-link.org/

2. Overview

This manual describes how to develop a CC-Link IE TSN remote station using "CC-Link IE TSN remote station communications LSI R-IN32M4-CL3". The main information included in this manual is as follows:

- R-IN32M4-CL3 specifications
- R-IN32M4-CL3 application circuit design
- User program design
- R-IN32M4-CL3 driver specifications

2.1 Development Features

R-IN32M4-CL3 is an LSI that integrates the communications IP core for CC-Link IE TSN, CPU, and GbE-PHY. This integrated LSI allows you to reduce CPU and GbE-PHY related development costs and manhours. The following are the features of development using the R-IN32M4-CL3.

(1) Remote stations for CC-Link IE TSN can be developed without awareness of the protocol.
(2) The pattern design of the communication circuit is simplified by the integrated GbE-PHY. Also, with fewer components and circuits around the CPU and GbE-PHY, the board to be developed can be made compact.
(3) Sample codes are provided that can be easily customized in accordance with user hardware specifications and applications.
(4) R-IN32M4-CL3 includes HW-RTOS, reducing the CPU load and achieving low power consumption in the developed device.

2.2 R-IN32M4-CL3 Main Specifications

The following table lists the main specifications related to the R-IN32M4-CL3 hardware.

Table 2.1 R-IN32M4-CL3 Main Specifications

Item		Description
Outer appearance	Number of pins	BGA 484 pins
	Size	23 mm × 23 mm
Power supply voltage		3.3 ± 0.165 V, 2.5 ± 0.125 V, 1.15 ± 0.06 V
Operating ambient temperature		−40 to 85ºC
CPU		Built in Arm Cortex-M4 Processor (100 MHz)
Instruction RAM		768 Kbytes, built in (ECC compatible)
Data RAM		512 Kbytes (ECC compatible)
Buffer RAM		64 Kbytes (ECC compatible)
I/O ports		CMOS I/Os: Up to 106
Ethernet PHY		100BASE-TX, 1000BASE-T GbE-PHY (built-in) × 2 ports

2.3 R-IN32M4-CL3 Application Product Communications Specifications

The following table lists the main specifications for communications by R-IN32M4-CL3 application products.

Table 2.2 R-IN32M4-CL3 Application Product Communications Specifications (CC-Link IE TSN)

Item	Description
Station type	Remote station
Station number	1 to 254
Communications speed	1 Gbps, 100 Mbps
CC-Link IE TSN Protocol version	2.0*1
CC-Link IE TSN Class	CC-Link IE TSN Class B devices and CC-Link IE TSN Class A devices can be developed.
Communication method	CC-Link IE TSN Class A: Time managed polling method CC-Link IE TSN Class B: Time sharing method
Network topology	Line and star (coexistence of line topology and star topology is possible), and ring
Communications cable	1 Gbps: Ethernet cable that satisfies 1000BASE-T standards (category 5e or higher, shielded, STP) 100 Mbps: Ethernet cable that satisfies 100BASE-TX standards (category 5 or higher, shielded, STP)
Maximum station-to-station distance	100 m
Multicast filter	Supported
Cyclic transfer function	Maximum transmission size: The total size of RX, RWr, and SSPDU is within 2400 bytes. Maximum reception size: The total size of RY, RWw, RSPDU is within 2400 bytes.
Transient transfer function	Client function: Supported Server function: Supported
CANopen communication	PDO Send/Receive : Supported SDO Send/Receive : Supported Maximum number of object dictionaries (maximum number of control axes): 8
Safety PDU send/receive	RSPDU maximum size: 80 bytes SSPDU maximum size: 80 bytes

Note 1. Supported from R-IN32M4-CL3 sample code Ver.1.06.

Table 2.3 R-IN32M4-CL3 Application Product Communications Specifications (Ethernet)

Item	Description
Communication speed	1 Gbps, 100 Mbps
Communication mode	1000BASE-T (full-duplex), 100BASE-TX (full-duplex)
Interface	RJ-45 connector (AUTO MDI/MDI-X)
Maximum frame size	1518 bytes (Jumbo frames not supported)
Maximum segment length	Distance between a switching hub and a station: 100 m Distance between switching hubs: Consult with the manufacturer of the switching hub you are using.
Number of cascade connection stages	Consult with the manufacturer of the switching hub you are using.
IP version	Compliant with IPv4*1

Note 1. The IP address can be set in the range from 0.0.0.1 to 223.255.255.254. For the setting procedure, refer to section 3.5, Considering How to Write the IP Address.

2.3.1 Precautions when the product ranked as CC-Link IE TSN Class A operates

This section describes precautions of when the R-IN32M4-CL3 application product operates as CC-Link IE TSN Class A.

(1) Master station version
A master station supporting CC-Link IE TSN protocol version 2.0 is required.
To use the RJ71GN11-T2 as the master station, use a module with firmware version 15 or later.
To use the RD78G(H) as the master station, use a module with firmware version 24 or later.

(2) Cyclic transmission
To use the RJ71GN11-T2 or the RD78G(H) as the master station, the total cyclic data size of all slave stations (CC-Link IE TSN Class A) must not exceed 2K bytes. Set the total cyclic data size within 2K bytes.
When using a master station other than the above, follow the specifications of the master station used.

(3) Transient transmission (SLMP)
In a system where both CC-Link IE TSN Class B and CC-Link IE TSN Class A products exist, when frames are relayed from a station without the time sharing control (CC-Link IE TSN Class A) to a station with the time sharing control (CC-Link IE TSN Class B), frames may be lost in the process of relay due to the number of connected modules, which changes depending on the operating environment or frame size.

2.4 Folder Configuration of Sample Code

The following shows the folder structure and file overview of the sample code.

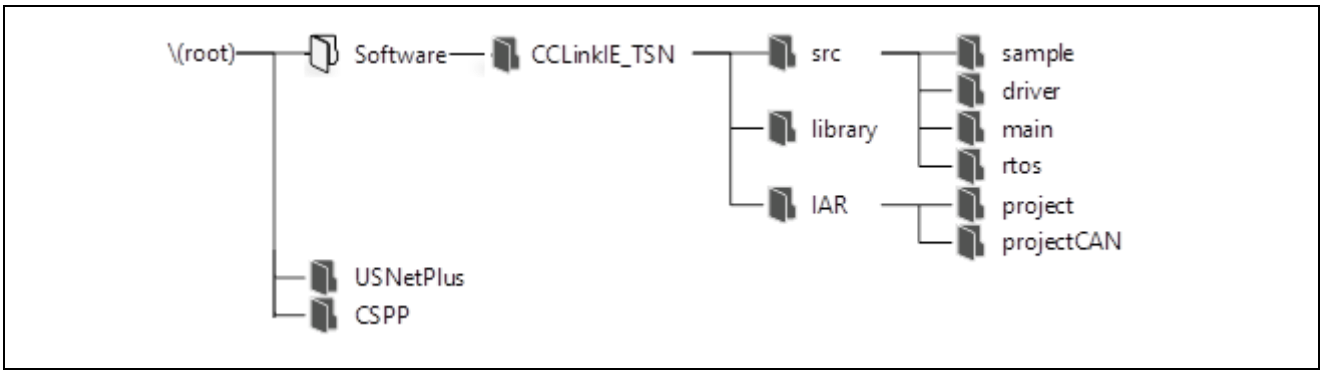


Figure 2.1 Folder Configuration Diagram

Table 2.4 File Overview

Folder Name			Description
Software	src	sample	Sample code for CC-Link IE TSN remote station. A folder for storing user program source files.
		driver	Sample code for CC-Link IE TSN remote station. A folder for storing source files of R-IN32M4-CL3 driver.
		main	A folder for tasks
		rtos	A folder for RTOS configuration files
	library		A folder for libraries (for GbE-PHY control and HW-RTOS control)
	IAR	project	Folders related to IAR development environment (projects, products, etc.) Stores two types of projects in cyclic transmission, one with linked devices and the other with CANopen's PDO.
		projectCAN	
USNetPlus	usnet_user_manual.pdf		User's Manual for USNetPlus (refer to section 1.8 of this manual)
CSPP			Sample CSP+ files and compressed files such as icons and images (See sections 2.6 and 3.8)

2.5 Sample Code Overview

The sample code is used to develop a CC-Link IE TSN remote station using R-IN32M4-CL3. It consists of the user program, R-IN32M4-CL3 driver interface functions, R-IN32M4-CL3 driver callback functions, and the R-IN32M4-CL3 driver main body. It only describes CC-Link IE TSN (the communications function).

(1) User program

The user program is an application program created by the user. The program in the sample code is provided for a reference for checking remote station logic. Customize the program in accordance with user requirement specifications. For details, refer to section 5, Creating User Programs.

(2) R-IN32M4-CL3 driver interface functions

R-IN32M4-CL3 driver interface functions are called when an R-IN32M4-CL3 driver function is used by the user program. Customization is not required. For details, refer to section 6, Specifications of the R-IN32M4-CL3 Driver Functions.

(3) R-IN32M4-CL3 driver callback functions

R-IN32M4-CL3 driver callback functions describe examples of processing on the user program side in response to events that occur on the R-IN32M4-CL3 driver side. Customize the functions in accordance with user requirement specifications. For details, refer to section 6, Specifications of the R-IN32M4-CL3 Driver Functions.

(4) R-IN32M4-CL3 driver main body

R-IN32M4-CL3 driver main body is called by R-IN32M4-CL3 driver interface functions and controls R-IN32M4-CL3.

2.6 Sample CSP+ file overview

Refer to the attached sample CSP+ file when developing a CC-Link IE TSN remote station using R-IN32M4-CL3. The following is a list of sample CSP+ files and their uses.

Table 2.5 File overview

File Name	Applications
0x1234_RemoteSample_1_en.CSPP.zip	Used when using a linked device for cyclic transmission.
0x1234_RemoteSample_CAN_1_en.CSPP.zip	Used when using CANopen's PDO for cyclic transmission.
0x1234_RemoteSample_CAN_Base_1_en.CSPP.zip	(When using CANopen's PDO for cyclic transmission) Of the equipment that consists of the basic unit and the expansion unit, it is used in the basic unit.
0x1234_RemoteSample_CAN_Ext_1_en.CSPP.zip	(When using CANopen's PDO for cyclic transmission) Of the equipment that consists of the basic unit and the expansion unit, it is used in the expansion unit.
0x1234_RemoteSample_Safe_1_en.CSPP.zip	When sending/receiving safety PDUs

Sample CSP+ file is for reference to display the R-IN32M4-CL3 applicable product in the "CC_IE_TSN configuration window" and to check "Slave station parameter processing / command execution"*1. Customize it according to the user's required specifications.
Refer to "3.8 Preparation for Creating CSP+ and Related Files" when creating CSP+.

Note 1. Describes the information required for "parameter batch read", "parameter batch write", and "parameter automatic setting". For details, refer to "3.7 Considering Support for Various Engineering Tool Functions".

2.7 System Configuration

(1) Software Configuration

The following describes an example of the software configuration of an R-IN32M4-CL3 application product. With the usage of the HW-RTOS library and various functions provided by the R-IN32M4-CL3 driver, the user program can utilize various R-IN32M4-CL3 functions, such as cyclic transfer and transient transfer.

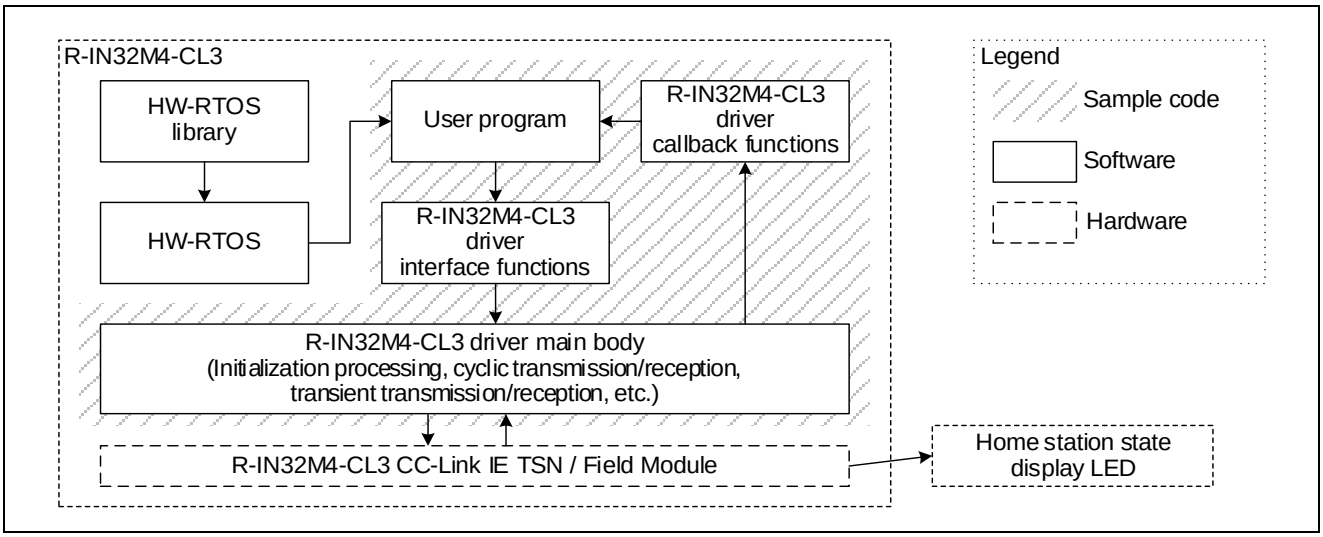


Figure 2.2 Software Configuration Overview

(2) Hardware Configuration

The following describes an example of the hardware configuration of an R-IN32M4-CL3 application product.
The hardware consists of R-IN32M4-CL3, peripheral components, and two Ethernet ports.
Note that the term "CPU" used in the following sections refers to the areas other than GbE-PHY areas in R-IN32M4-CL3.

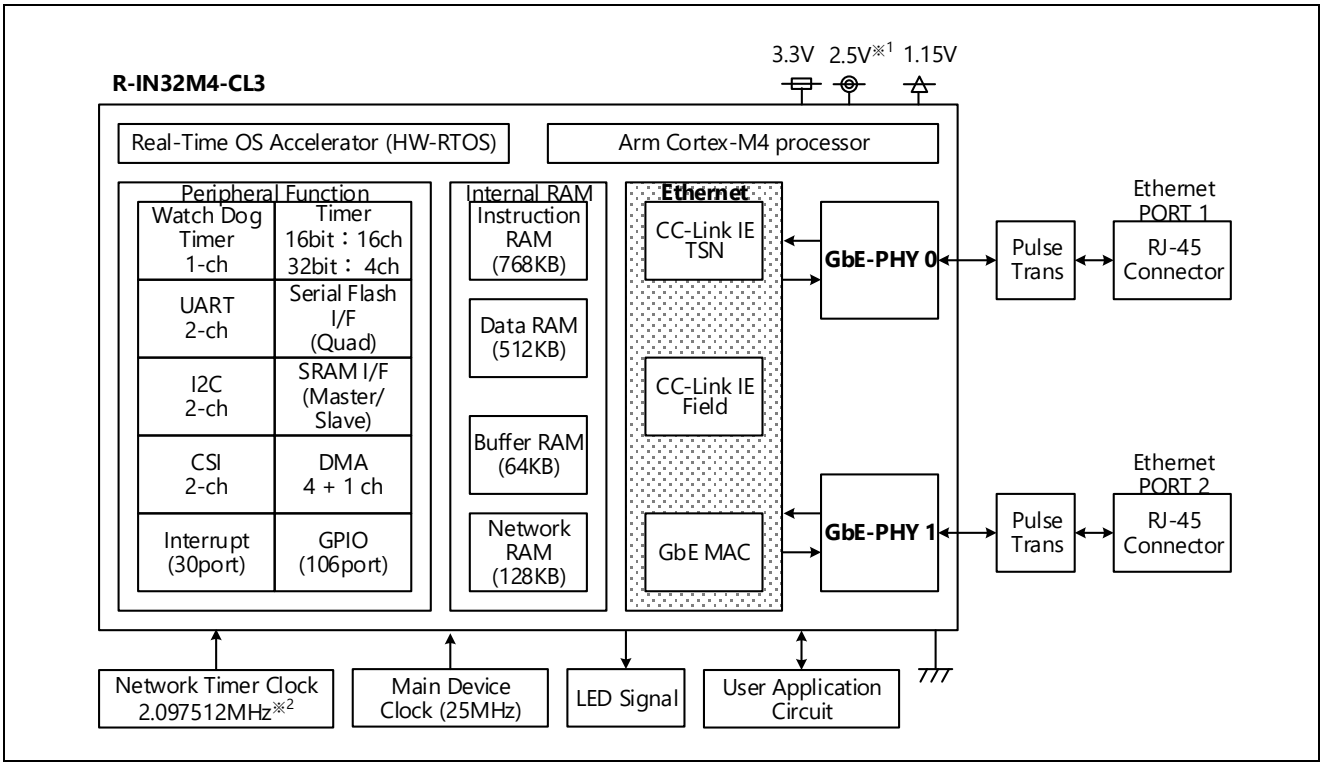


Figure 2.3 Hardware Configuration Overview

Note 1. +2.5 V power supply input to R-IN32M4-CL3 can be selected from +2.5 V power supply generated inside R-IN32M4-CL3 or +2.5 V power supply generated outside R-IN32M4-CL3 using the external pin (REG_EN).

Note 2. The Network Timer Clock for CC-Link IE Field operations can be selected from the internal PLL of R-IN32M4-CL3 or the external oscillator output of R-IN32M4-CL3 using the external pin (CLK2MSEL).

2.8 Protocols Supported by the R-IN32M4-CL3 Application Product

The protocols supported by the R-IN32M4-CL3 application product are as follows. Some protocols use the TCP/IP stack "USNetPlus®" from NISSIN SYSTEMS Co., Ltd.

Table 2.6 List of Supported Protocols

Protocol	USNetPlus Usage	Description
UDP	Used	For use in transient transfer • For use in SLMP communications for which UDP/IPv4 is used as the lower-layer protocol
IPv4	Used	
ICMP	Used	For use in responses to ping packets
ARP	Used	For use in MAC address acquisition
RARP	Used	For use in inquiry of the IP address
PING	Used	For use in checking interaction in communications
SNMPv2	Used	For use in collection of diagnostic information
GARP	—	For use in detecting duplication of the IP address setting
TCP	Used	For user extension (TCP and IP can also be used if this suits the implementation of the user application)

When an R-IN32M4-CL3 application product only performs cyclic transfer and transient transfer, user programs in the sample code do not control TCP/IP communications directly. Therefore, the user does not need to be aware of USNetPlus.
If you want to, for example, freely implement a TCP/IP communications function, refer to the included USNetPlus user's manual "usnet_user_manual.pdf" to check the API specifications, etc. of USNetPlus.

Note that processing related to general-purpose TCP/IP communications is not described in the sample code.

3. Considering the Specifications and Preparing for Development

This section describes the items which require consideration with regard to the specifications and preparations prior to the development of an R-IN32M4-CL3 application product.

The following illustrates the development process by the user.

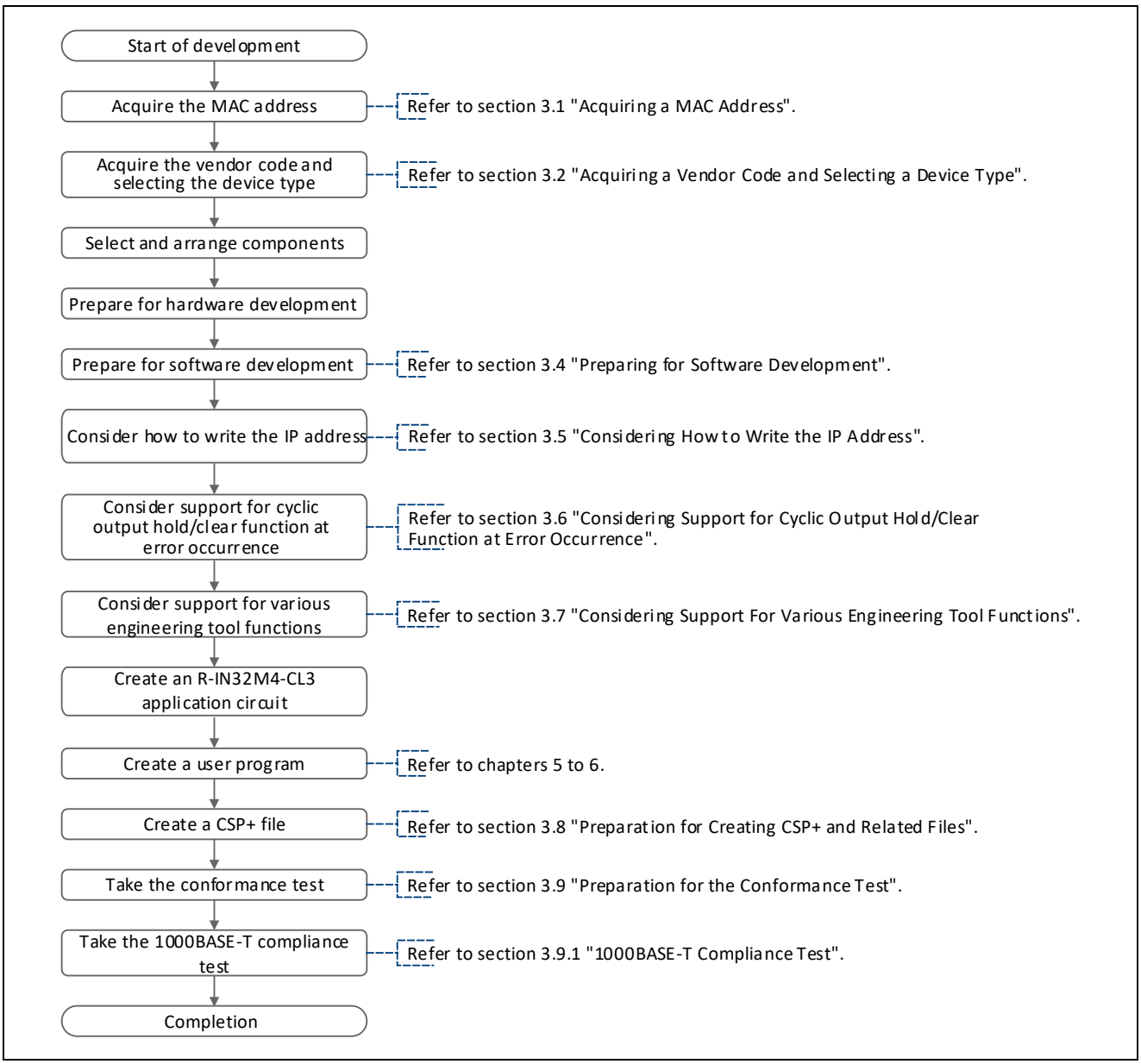


Figure 3.1 Development Process Example

3.1 Acquiring a MAC Address

R-IN32M4-CL3 application products are Ethernet (IEEE 802.3ab) compliant. Be sure to acquire a MAC address MA-L (MAC Address Block Large) unique to the device to be developed.
To acquire a MAC address, contact the following authority (department) in the USA.

The IEEE Registration Authority
Web: http://standards.ieee.org/develop/regauth/oui/ (as of March 2019)

3.2 Acquiring a Vendor Code and Selecting a Device Type

R-IN32M4-CL3 application products require registration of a vendor code and device type. The vendor code and device type are assigned and managed by the CC-Link Partner Association. If you have any questions, contact the CC-Link Partner Association.

Table 3.1 Vendor Code and Device Type

Item	Description
Vendor code (vendorCode)	ID number (fifth to eighth digits) issued when the vendor joined the CC-Link Partner Association. For example, if the ID number is 123-456-7890, the vendor code is 5678.
Device type (deviceType)	Select an applicable type from the device types listed on the CC-Link Association website. If an applicable device type does not exist, consult with the CC-Link Partner Association.

3.3 Pins Connected to Hardware Switches

In the R-IN32M4-CL3 application circuit, various settings are switched using hardware switches as indicated in the table below. The pins listed in the table below can be changed by the user. Investigate if you want to connect the pins to hardware switches.

If you are not using hardware switches, add processing for writing the various settings from the peripheral devices or the like of the R-IN32M4-CL3 application product.

Table 3.2 Pins Connected to Hardware Switches (CC-Link IE TSN)

Symbol	Switch	Pins
SW5	IP address 4th octet setting switch (×1)	RP10, RP11, RP12, RP13
SW3	IP address 4th octet setting switch (×10)	RP14, RP15, RP16, RP17

3.4 Preparing for Software Development

3.4.1 Software Development Procedure

The following shows an example of the software development procedure for R-IN32M4-CL3 application products.

(1) Creating User Program
Create a user program with reference to section 5, Creating User Programs.

(2) Compiling User Program and R-IN32M4-CL3 Driver Callback Function
Compile a customized user program and R-IN32M4-CL3 driver callback functions.

(3) Linking Object Module Files and Library Files
Create a load module file by linking the compiled files (object module files), OS driver library files, and library files of the R-IN32M4-CL3 driver.

(3) Writing an Executable File
Write the load module file to the R-IN32M4-CL3 application product (target) by using a debugger, ICE, or other devices.

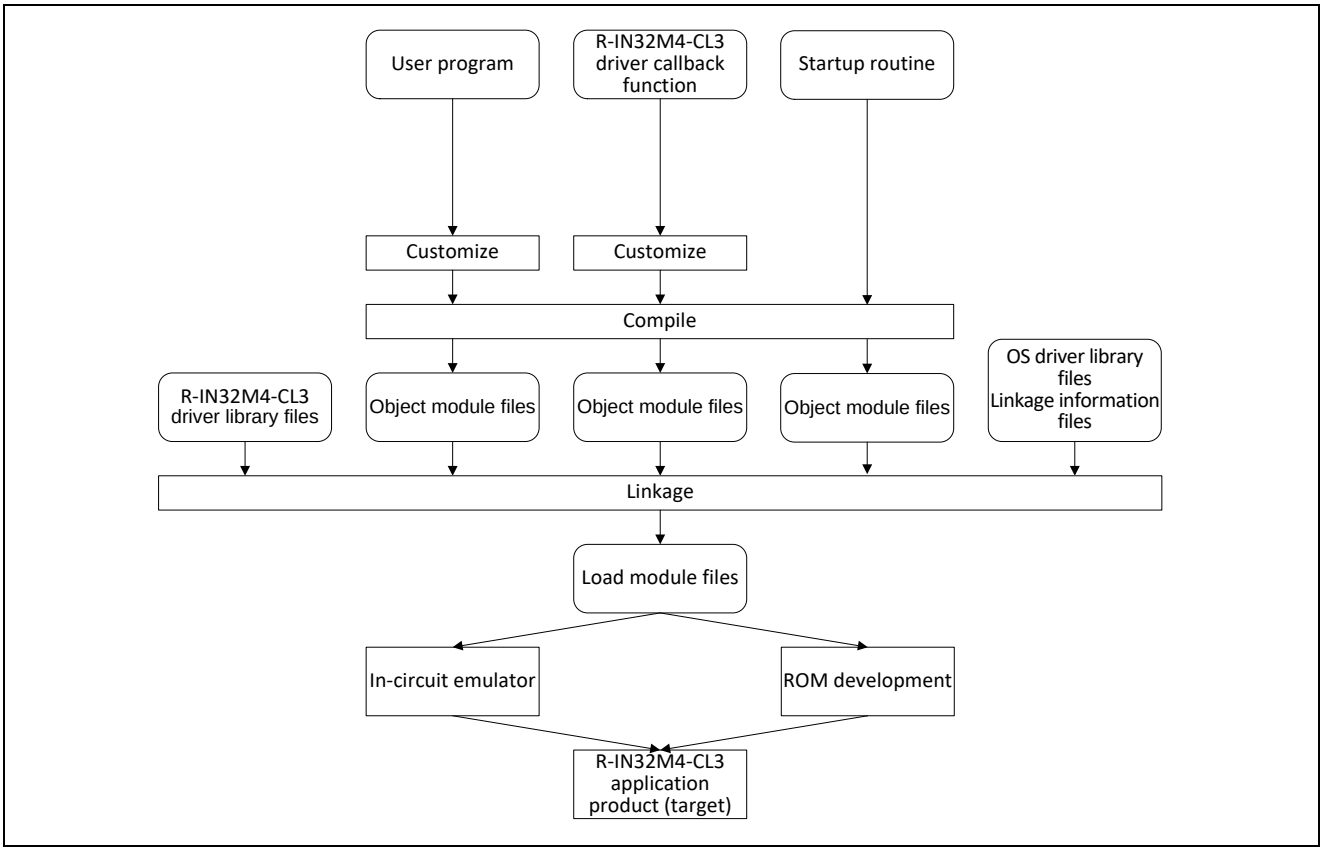


Figure 3.2 Software Development Procedure Example

3.4.2 Development Environment

The following environment is used as a software development environment for the Cortex-M4 microcontroller in R-IN32M4-CL3.

Table 3.3 Software Development Environment

Tool Chain	Compiler	Debugger	Emulator
Embedded Workbench for Arm V9.10.1 to the latest version (use the latest version) (IAR Systems)			i-Jet JTAGjet-Trace-CM (IAR Systems)

3.4.2.1 Execution Procedure

(a) Building procedure

Start the IAR project "ProjectIAR.eww" in the project folder of "CCLinkIE_TSN" and select "Project" → "Rebuilt All". This will cause rebuilding from the source code registered in your workspace.

(b) Firmware programming procedure

Start the IAR project "ProjectIAR.eww" in the project folder of "CCLinkIE_TSN" and select "Project" → "Download" → "Download File" to select the "main.out" file. The program will be downloaded to the flash memory.

3.4.3 Changing the Flash Loader Program

Using the "IAR Embedded Workbench for Arm (EWARM)" as a debugger may require changing the flash loader program which is used to write executable files to the flash memory.

This manual describes changing the flash loader program when the serial flash ROM (W25Q64JV) from Winbond® Electronics Corporation is used. For details of the flash loader, refer to "Flash Loader Development Guide for IAR Embedded Workbench® (UFLX-4) IAR SYSTEMS".

3.4.3.1 Development Environment for the Flash Loader

When EWARM is installed, the development environment for the flash loader is also installed at the same time. This development environment is used to change the flash loader program. The development environment for the flash loader which supports the serial flash ROM (W25Q64JV) from Winbond® Electronics Corporation is stored in the following location. However, the actual location for storage will differ with the destination for installation of the development environment.

C:¥Program Files (x86)¥IAR Systems¥Embedded Workbench 8.4¥arm¥src¥flashloader¥Renesas ¥FlashRIN32M4_SerialFlash

In the flash loader program (FlashRIN32M4_SerialFlash.c), information such as the ID of the loadable flash ROM is managed as a flash ROM table (flashType[]). If the information on the flash ROM in use is not defined in the table (flashType[]), it must be added to the table.

3.4.3.2 Flash ROM Table Settings

The table below lists the settings when adding the serial flash ROM (W25Q64JV) from Winbond® Electronics Corporation to the flash ROM table (flashType[]). The settings listed in the table below will be added to the flash ROM table (flashType[]) defined in the flash loader program (FlashRIN32M4_SerialFlash.c).

Table 3.4 Flash ROM Information

Table Information	Setting of W25Q64JV	Overview
ID	1740EFH	The JEDEC ID of W25Q64JV is set.
Block size	16	The size for block size per sector is specified as a power of 2. In the case of W25Q64JV, the size is 64 Kbytes, so the setting is 16.
Page size	8	The size for page programming is specified as a power of 2. In the case of W25Q64JV, the size is 256 bytes, so the setting is 8.
Device size	23	The number of bytes in the flash memory is specified as a power of 2. In the case of W25Q64JV, the size is 8 Mbytes, so the setting is 23. Two to the power of 23 is 8,388,608 bytes.

3.4.3.3 Changing the Flash Loader Program

The following shows the flash loader program before and after change.

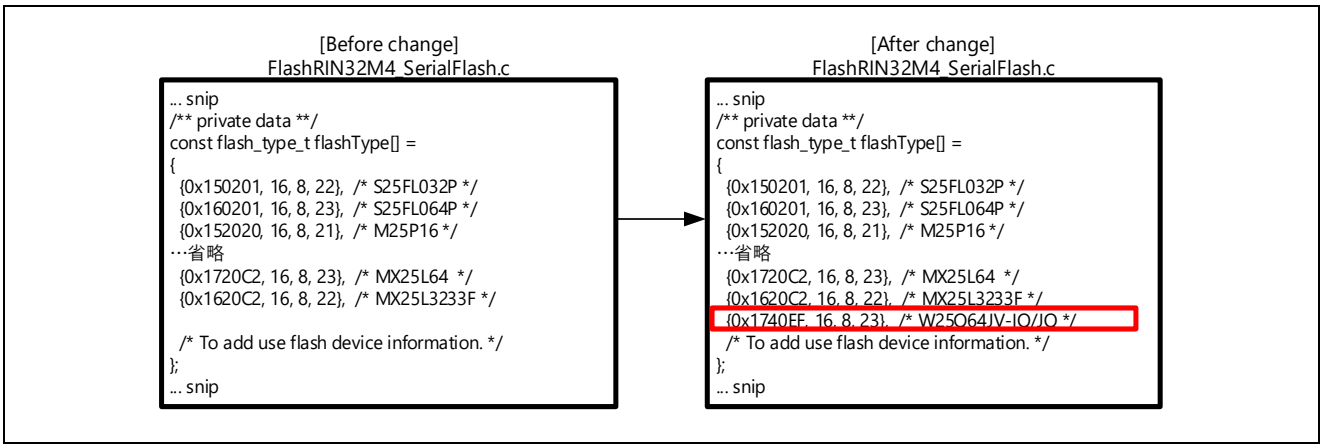


Figure 3.3 Changing the Flash Loader Program

3.4.4 Compilation Switches

The table below lists the macro definition names for use in switching control of compilation. Make or delete definitions to set up switching as required.

Table 3.5 Macro Definitions for the Compilation Switching

Macro Name	Operation when the Macro is Defined	Default Definition (Compilation Option)	Remarks
FOR_DEBUG	Disabling the WDT*1	Defined	
NOUSE_WDT	Disabling the WDT*2	Defined	
TSN_CAN_ENABLE	Enabling CANopen communications	Defined (projectCAN file only)	During CANopen communication
TSN_CAN_MULTIAXIS_ENABLE	Enable expansion unit for CANopen communication*3	Defined (projectCAN file only)	During CANopen communication
USE_LOOPBACKTEST	Enable loopback communication test	Not defined	
USE_COMPLIANCETEST	Enable IEEE802.3ab compliance test	Not defined	
SLMP_MULTI_DROP_DISABLE	Disable "Requested station processor sub number" of SLMP frame (0 setting)	Not defined	
SAFETY_PDU_ENABLE	Enables the safety PDU send/receive.	Not defined	When safety communications are performed
MCUIF_ENABLE	Enables the MCU-MCU interface function of the safety PDU send/receive. *4	Not defined	When safety communications are performed
CURERR_OPTIONINFO_ENABLE	MIB Current Enables processing related to error information option information	Not defined	When installing an expansion unit
ACCUMULATE_STATISTICS_INFORMATION	Enables holding of the total number of statistical information sets.	Not defined	-

Note 1. This prevents an WDT error from occurring during debugging.

Note 2. This disables the WDT not for use in CC-Link IE TSN (for use in CC-Link IE Field).

Note 3. The definition of "TSN_CAN_ENABLE" is required.

Note 4. The definition of "SAFETY_PDU_ENABLE" is required.

3.5 Considering How to Write the IP Address

To create a data link to the home station, writing an IP address (4th Octet) to R-IN32M4-CL3 is required. Therefore, you will need to consider in advance how to write the IP address in accordance with the specifications of the R-IN32M4-CL3 application product.

For reference, the following illustrates how to set the IP address by using a hardware switch block and peripheral devices with the R-IN32M4-CL3 application product. Either method uses "gerR_IN_SetIPAddress" (6.4.1(3)) of the R-IN32M4-CL3 driver interface functions in the user program "iUserInitialization" (5.3.1, Initialization Processing).

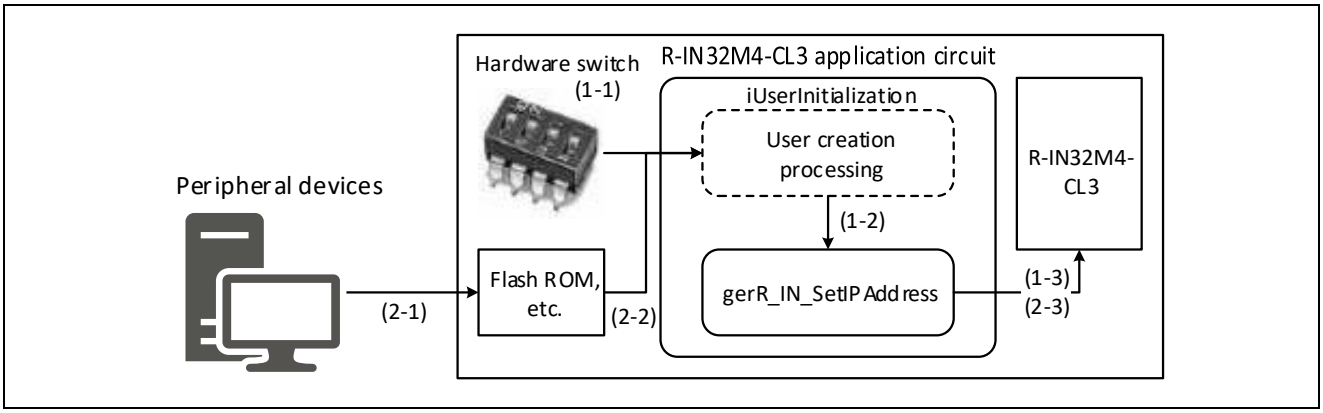


Figure 3.4 IP Address Setting Image

<Example 1: Using a hardware switch>

Step	Description
(1-1)	Set the IP address using a hardware switch.
(1-2)	The user creation process in iUserInitialization reads the current value of the hardware switch and sets it in the argument of "gerR_IN_SetIPAddress".
(1-3)	"gerR_IN_SetIPAddress" writes the argument value to R-IN32M4-CL3.

Select a hardware switch that corresponds to the value range of the IP address 4th octet.

- Range: 01H to FEH (1 to 254)

<Example 2: Using peripheral devices>

Step	Description
(2-1)	Write the IP address data to non-volatile memory using a peripheral device.
(2-2)	The user creation process in iUserInitialization reads the data written in the non-volatile memory and sets it in the argument of "gerR_IN_SetIPAddress".
(2-3)	"gerR_IN_SetIPAddress" writes the argument value to R-IN32M4-CL3.

3.6 Considering Support for the Cyclic Output Hold/Clear Function at Error Occurrence

The cyclic output retention / clear function at the time of abnormality is to send the received cyclic data (RY and RWw) to the outside of the own station when the master station application is stopped / abnormal or when the own station is disconnected from the data link. A function to hold or clear the output.
Consider in advance whether this function is to be supported as a fail-safe measure when an error described above occurs. To support the function, hold/clear processing needs to be added to the sample code.
For hold/clear processing when the master station application has stopped or is in error, refer to section 5.4.1, Cyclic Reception Processing. For hold/clear processing when the home station is disconnected from the data link, refer to section 5.4.4, Communications State Update Processing.

3.7 Considering Support for Various Engineering Tool Functions

The following functions can be performed using an engineering tool*1 connected to the master station's programmable controller CPU. Consider in advance whether these functions are to be supported as the specifications of the R-IN32M4-CL3 application product.
Note 1. This refers to GX Works 3 in the case of Mitsubishi Electric programmable controllers.

Table 3.6 Engineering Tool Functions

No.	Item		Items Required for R-IN32M4-CL3 Application Product
1	Parameter processing/command execution of slave stations		Write CSP+ files up to the range (3) in Figure 3.5.
	a	Parameter batch read/write	SLMP request reception and response transmission processing (SLMP command: Batch read (0613H), batch write (1613H))
	b	Parameter automatic setting	SLMP request reception and response transmission processing (SLMP command: Slave station parameter automatic setting related (0EB0Hand, etc.))

3.7.1 Parameter Processing/Command Execution of Slave Stations

By using this function, parameter setup and command execution of R-IN32M4-CL3 application products can be performed without programming. This makes it possible to reduce the programming required for parameter setup and command execution by the end user of the R-IN32M4-CL3 application product.
This function requires the creation of a CSP+ file. Then, the R-IN32M4-CL3 application product needs to respond to the SLMP command (SLMP request from the master station) described in the CSP+ file.

For the CSP+ file, refer to section 3.8, Preparation for Creating CSP+ and Related Files.
For SLMP request reception and response transmission processing, refer to section 5.6, User Program Details (SLMP Command Execution Related) and section 5.7, Details on Processing of User Programs (Slave Station Parameter Automatic Setting Related). (processing for transmission and reception of the individual commands of SLMP is described in the sample code).

(1) Batch reading/writing of parameters
This function reads and writes parameters required to start up the slave station.

(2) Automatic setting of parameters
This function compares slave station parameters stored in the master station with actual parameters of the slave station. If they do not match, the master station automatically writes parameters to the slave station.

3.8 Preparation for Creating CSP+ and Related Files

(1) Overview

CSP+ is specifications to describe required information for starting, operating, and maintaining CC-Link Family connection supporting products. Providing the CSP+ and related files (the image file, icon file, and object dictionary file) to the end users of the R-IN32M4-CL3 application product allows them to manage all stations of CC-Link IE TSN using one engineering tool.

For details of CSP+, refer to "Control & Communication System Profile Specification".
To create CSP+ and related files, use "Control & Communication System Profile Creation Guidelines" and "CSP+ Creation Support Tool". In addition, please use the attached sample CSP+. (See Section 2.6)

(2) Creation scope

The following shows the scope in which CSP+ and related files are to be created for the remote station.
The conformance test includes verification of CSP+, so be sure to create a CSP+ file of scope (1) and an object dictionary file*1 of scope (5). Then, consider in advance which functions (creation scopes (2) and (3)) are to be supported as the specifications of the R-IN32M4-CL3 application product.

Note 1. Handling CANopen communications requires creation of this file.

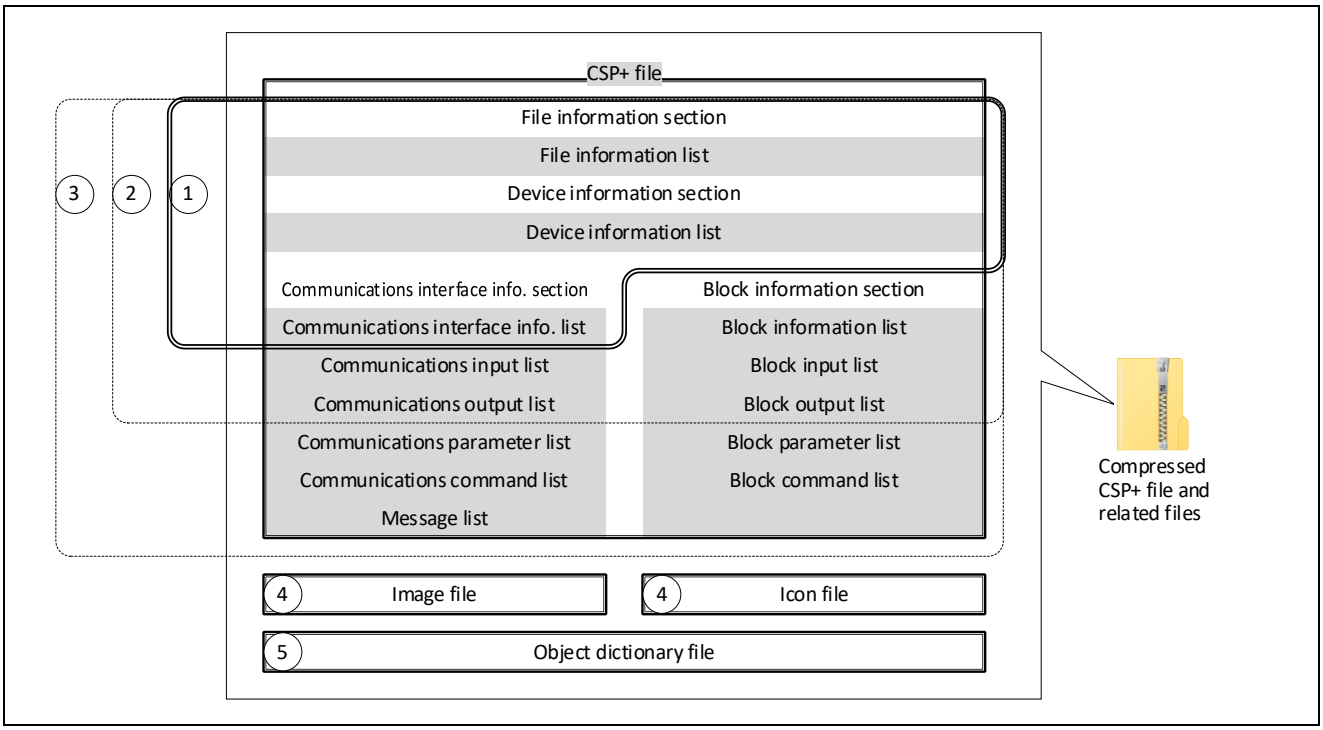


Figure 3.5 CSP+ File Section Configuration and Related Files

Scope	Description	Necessity
[1]	Information required to verify mandatory items in the CC-Link Partner Association's conformance test (for GX Works 3) R-IN32M4-CL3 application products are displayed in the CC_IE_TSN Configuration window, and the network configuration can be easily created.	Required
[2]	Information required to display slave station link device and master station device assignments	Optional
[3]	Information required for parameter processing/command execution of slave stations (for GX Works 3) The parameters of CC-Link IE TSN connection supporting products can be easily set from the CC_IE_TSN configuration window.	Optional
[4]	The file required to display the R-IN32M4-CL3 application product as an image and icon in the engineering tool (in the case of GX Works 3) In the CC_IE_TSN configuration window, the image will be displayed in the network configuration diagram and the icon will be displayed in the list of units.	Optional
[5]	The file required to set up the mapping of R-IN32M4-CL3 application products to PDOs (in the case of GX Works 3). Allows setting up the mapping of products that support CC-Link IE TSN connection to PDOs from the CC_IE_TSN configuration window.	*1

Note 1. Handling CANopen communications requires creation of this file.

Supplementary note on the PDO mapping setting
The object dictionary for CANopen communications in CC-Link IE TSN is defined in a CSV file. With the object dictionary file, RWr, RWw, and PDO objects can be linked to the stations which support CANopen communications.

(3) Expansion unit for CANopen communication

During CANopen communication, R-IN32M4-CL3 applicable products are "basic unit (axis 1)" that communicates and "expansion unit (axis 2 ~)" that does not communicate, such as multi-axis integrated servo amplifier. For the equipment to be configured, it is necessary to create two types of CSP+ files, the basic unit and the expansion unit.

Following information cannot be described in the CSP+ file of the expansion unit, so describe it in the CSP+ file of the basic unit.

- Communication input list
- List of communication parameters
- Block input list
- List of block parameters
- Communication output list
- Communication command list
- Block output list
- List of block commands
- Message list

3.9 Preparation for the Conformance Test

The conformance test is a test which needs to be conducted on each device to ensure high reliability in the communications by CC-Link IE TSN connection supporting products. The test verifies that the product developed by the user satisfies the CC-Link IE TSN communications specifications and is connectable to the network.
Obtain the conformance test specifications at the preliminary stage of development to design the R-IN32M4-CL3 application product so that it satisfies the test requirement specifications.

A CC-Link IE TSN connection supporting product that passes the conformance test can be included as a certified product in the "CC-Link Partner Product Catalog" and other media.

Point
The functions may not be supported depending on the development timing. When implementing the conformance test, contact the CC-Link Partner Association.

3.9.1 1000BASE-T Compliance Test

Since CC-Link IE TSN is compliant with 1000BASE-T, it is recommended to conduct the 1000BASE-T compliance test on R-IN32M4-CL3 application products based on the IEEE802.3ab specifications.

The 1000BASE-T compliance test measures four test waveforms from the Ethernet ports to verify waveforms of the transfer path. Process of outputting the waveform is described in the sample code "User IEEE Test", please use it.
For details on UserIEEE Test, refer to "5.11.1 Hardware Test Processing (IEEE802.3ab Compliance Test)".

Items required for the compliance test
(1) The UserIEEETest function is called by the idle task when the compiler switching definition "USE_COMPLIANCETEST" is valid. At this time, the R-IN32M4-CL3 application product does not start a data link but only runs the compliant test. Therefore, consider a method for calling the UserIEEETest function at a desired time by, for example, creating dedicated firmware which only implements the test without setting up a data link. (2) UserIEEETest executes "gerR_IN_IEEETest" with "test mode (1-4)" that outputs the test waveform as an argument, and writes the test waveform data to the PHY register. However, there is no processing for switching among test modes 1 to 4 as desired. Therefore, consider a method for switching four waveforms as desired by, for example, specifying test mode 1 to 4 by using the hardware switch.

4. Functions of the R-IN32M4-CL3 Application Product

This section gives an overview of the available functions of the R-IN32M4-CL3 application product (CC-Link IE TSN).

Table 4.1 Functions of the R-IN32M4-CL3 Application Product

No.	Function	Implementation necessity	Processing Category*1	Refer to
1	Cyclic transfer (CyclicMs, CyclicSs)	Required	User program	4.1
2	Transient transfer (NRSV-Transient)	—	—	4.2
	a. Transient transfer for users	Optional	User program	4.2.2.2(1)
	b. Transient transfer for management	Required	R-IN32M4-CL3 driver	4.2.2.2(2)
3	State display by LEDs	Optional	User program or R-IN32M4-CL3 driver	4.3
4	CC-Link IE TSN diagnostics	Optional	R-IN32M4-CL3 driver	4.4
5	Reserved station setting, temporary clearing of the reserved station setting	Optional	R-IN32M4-CL3 driver	—
6	Time synchronization specified in IEEE 1588 or IEEE 802.1AS	Required*2	R-IN32M4-CL3 driver	—
7	Network-synchronized communications	Optional*2, *3	User program	4.5
8	CANopen communications	Optional*3	User program	4.6
9	Expansion unit for CANopen communication	Optional	User program	4.6.1
10	Safety PDU send/receive in safety communications	Optional*3	User program	4.7
11	Communication speed and CC-Link IE TSN Class setting via SLMP	Optional	User program	4.8

Note 1. "user program" must implement the process shown in "5 Creating User Programs".
"R-IN32M4-CL3 driver" is already implemented in the R-IN32M4-CL3 driver, so the user does not need to be aware of it.

Note 2. Not required for CC-Link IE TSN Class A.

Note 3. Use the functions exclusively.
When the network synchronous communications are performed, the CANopen communications and the safety PDU send/receive cannot be performed. Furthermore, they cannot be used when the product ranked as CC-Link IE TSN Class A operates.
When the CANopen communications are performed, the network synchronous communications and the safety send/receive cannot be performed.
When the safety PDU send/receive is performed, the network synchronous communications and the CANopen communications cannot be performed.

4.1 Cyclic Transfer Function

The cyclic transfer function periodically communicates data with the master station using link devices. The R-IN32M4-CL3 application product receives CyclicMs and sends CyclicSs.
The following illustrates the flow of cyclic data.

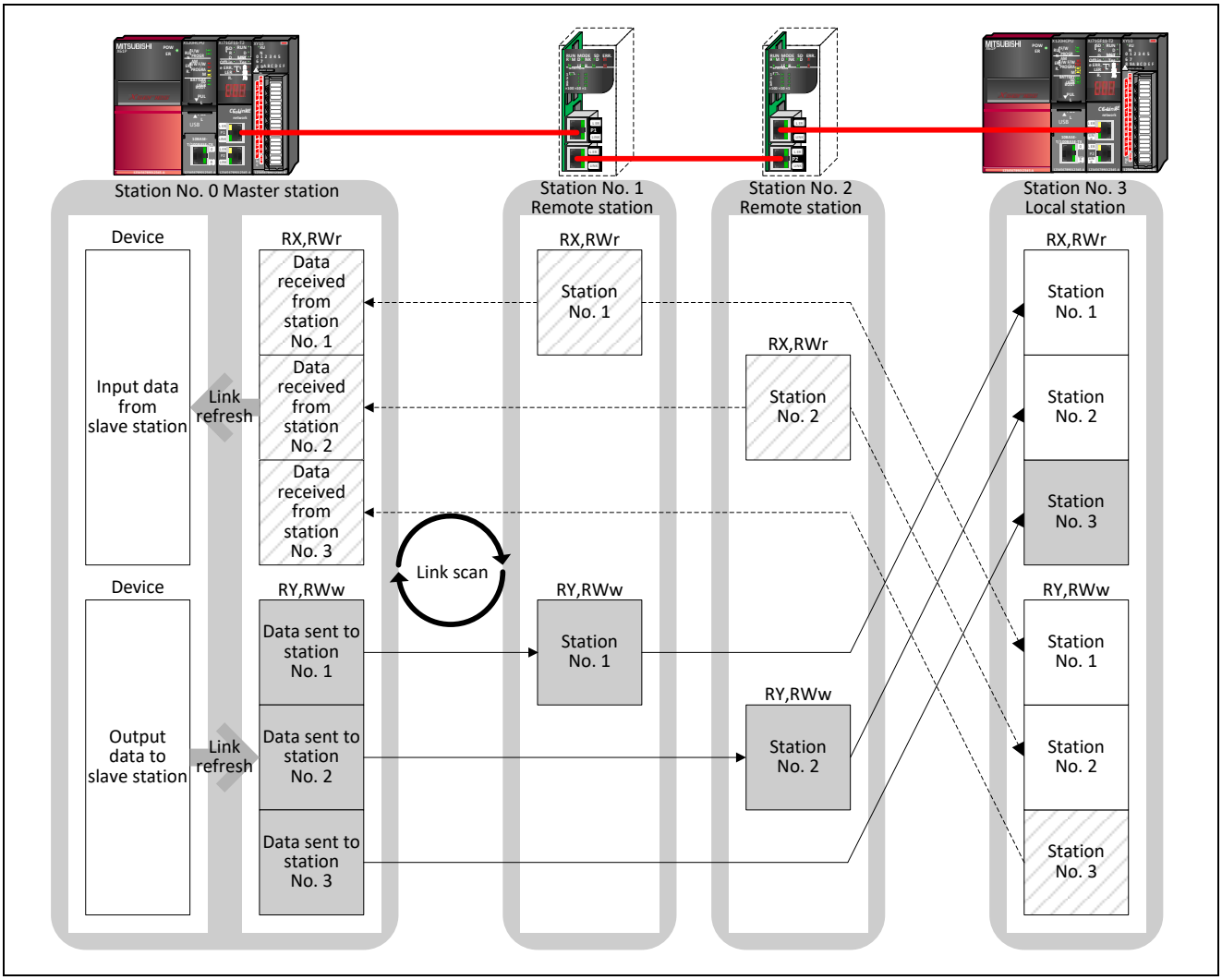


Figure 4.1 Flow of Cyclic Data

The state of each link device (RY and RWw) of the master station is stored in the link device (RY and RWw) of the home station by a link scan.
The state of each link device (RX and RWr) of the home station is stored in the link device (RX and RWr) of the master station by a link scan.
Cyclic transmission requires the processing described in "5.4.1 Cyclic Reception Processing" and "5.4.3 Cyclic Transmission Processing" of the user program.

4.2 Transient Transfer Function

Transient transfer communicates data when there is a request for communications from another station or the home station. Data are communicated through direct access to the device/buffer memory of the other station. The R-IN32M4-CL3 application product sends and receives data in NRSV-Transient.

There are the following two ways of transient transfer available for R-IN32M4-CL3 application products, which the user can send and receive desired data. In either way, the R-IN32M4-CL3 application product sends and receives data in NRSV-Transient.

- SLMP communications
- General-purpose TCP/IP communications

The following illustrates the flow of transient data in the case of an SLMP memory read instruction.

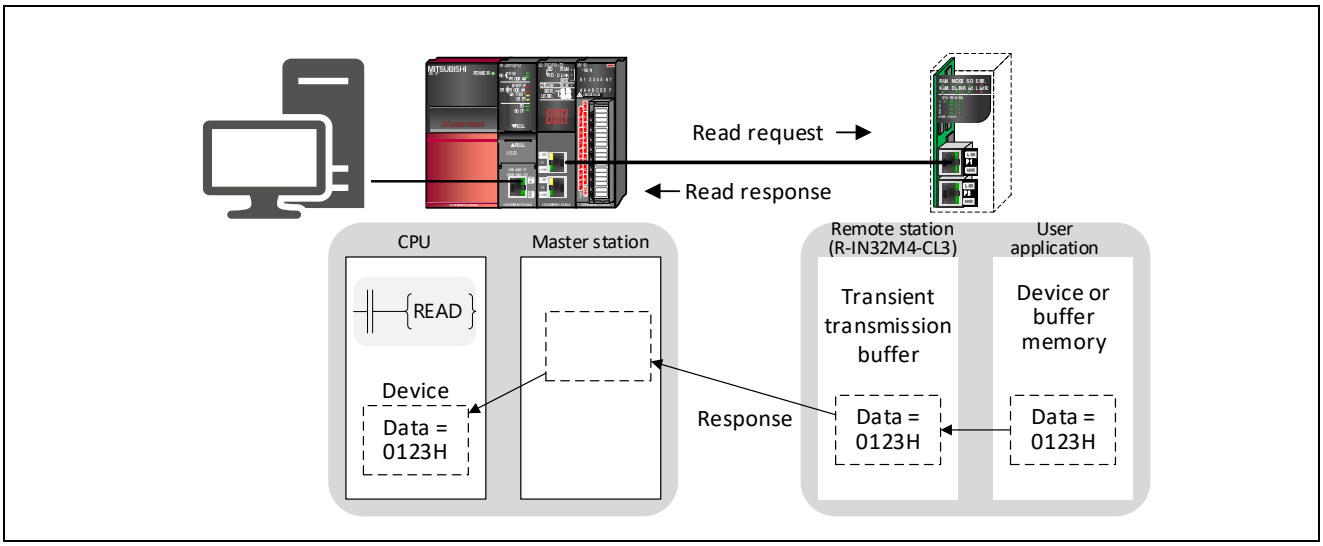


Figure 4.2 Flow of Transient Data

Transient transmission requires the processing described in "5.5 User Program Details (State Management and Transient Transfer Related)" of the user program. Regarding subsequent transient transfer, "communications using SLMP" is described.

4.2.1 Client and Server Functions of Transient Transfer

Transient transfer includes the client and server functions.
The client function sends transient requests to stations with the server function.
The server function sends transient responses to transient requests from stations with the client function.

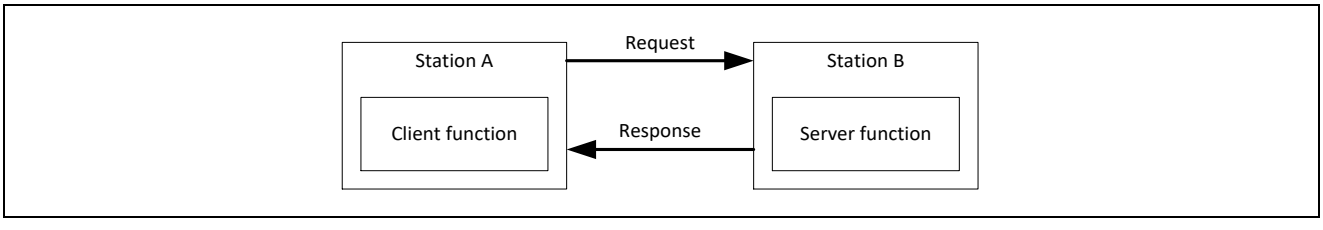


Figure 4.3 Transient Client/Server Function

4.2.2 About SLMP

4.2.2.1 Frames of SLMP

(1) IP Frames

The R-IN32M4-CL3 application product uses IP frames (Ethernet "Type" field: 0800H) for transmission and reception according to the SLMP.
Use IP frames when the R-IN32M4-CL3 application product, an engineering tool created by the user, etc. is to handle SLMP communications.

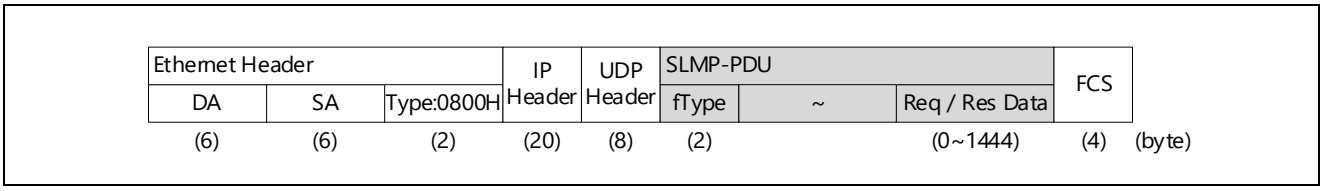


Figure 4.4 SLMP Structure of the IP Frame

(2) Frame Type

The SLMP has three frame types (fType).
The R-IN32M4-CL3 application product is capable of sending and receiving all SLMP frames. However, it only supports binary mode.
For details of each frame type, refer to "SLMP Specification" published by the CC-Link Partner Association or "SLMP Reference Manual".

Table 4.2 SLMP Frames which can be Transferred

No.	Type	Type Value (fType)	Symbol
1	LMT (Large-Node Number-Multi-Transmission type)	Request 0068H, Response 00E8H	6E*1
2	MT (Multi-Transmission type)	Request 0054H, Response 00D4H	4E*1
3	ST (Single-Transmission type)	Request 0050H, Response00D0H	3E*1

Note 1. The notations of 3E, 4E, and 6E are used as ST, MT, and LMT in above parts of the relevant manual "SLMP reference manual", sample code, etc.

4.2.2.2 Commands of SLMP

The SLMP has various commands to suit different forms of usage. Individual commands are sent and received by the user program for the R-IN32M4-CL3 application product and the R-IN32M4-CL3 driver.

- When you are sending and receiving desired data, use SLMP (port number: 45239). Processing for transmission and reception is handled by the user program (refer to section 5.6, User Program Details (SLMP Command Execution Related).

- When you are sending and receiving the data for network management or parameter management, use SLMP (port number 45238). Since processing for transmission and reception is handled by the R-IN32M4-CL3 driver, the user does not have to be particularly aware of this.

(1) SLMP commands to be sent and received by the user program (port number 45239)

The SLMP commands used in statements in the sample code are listed below.
To send and receive commands other than those listed below, add processing for each command with reference to "SLMP Specification" published by the CC-Link Partner Association or "SLMP Reference Manual".

Table 4.3 SLMP Commands to be Sent and Received by the User Program (Port Number 45239)

No	Type	Operation	Command	Sub-command	Implementation necessity	Refer to
When the home station is a client (when sending a request command and receiving a response command)						
1	Dual port memory	Batch reading	0613H	0000H	Optional	5.6.3 5.6.4
When the home station is a server (when receiving a request command and sending a response command)						
2	Dual port memory	Batch reading	0613H	0000H	Optional	5.6.1
3		Batch writing	1613H	0000H	Optional	5.6.2
4	Remote control	Remote reset	1006H	0000H	Optional	5.6.5
5		Indicator display	3070H	0000H	Optional	5.6.6
6	Device connection	Detecting connected devices	03E0H	0001H	Optional	5.8.1
7		Setting IP addresses of the connected devices	0E31H	0000H	Optional	5.6.7
8	Error history	Clearing error history	1619H	0000H	Optional	5.6.8
9	Slave station parameter automatic setting	Acquiring the communications settings	0EB0H	0001H	Optional	5.7.1
10		Checking the necessity of parameter distribution	0EBEH	0001H	Optional	5.7.2
11		Notifying the start of restoration	0EB8H	0001H	Optional	5.7.4
12		Notifying the end of restoration	0EB9H	0001H	Optional	5.7.5
13		Writing parameter data	0EBAH	0001H	Optional	5.7.6
14	Event history	Distributing network time offset	3062H	0000H	Optional	5.6.9
15		Distributing network time	3063H	0000H	Optional	5.6.10
16	Watchdog counter	Setting up the watchdog timer	3210H	0000H	*1	5.6.11
17	Access to the CAN application objects	Object reading	4020H	0001H	*2	5.9.4
18		Object writing	4020H	0002H	*2	5.9.5
19		Object sub-Index consecutive reading	4020H	0005H	*2	5.9.6
20		Object sub-index consecutive writing	4020H	0006H	*2	5.9.7
21		Acquiring the NMT state	4020H	0007H	*2	5.9.8
22		Setting the NMT state	4020H	0008H	*2	5.9.9
23	CC-Link IE TSN device parameter setting	Acquiring the function setting support information	3080H	0000H	Optional	5.8.2
24		Reading the function setting (communication speed)	3081H	0000H	Optional	5.8.3
25		Reading the function setting (CC-Link IE TSN Class)	3081H	0001H	Optional	5.8.5
26		Writing the function setting (communication speed)	3082H	0000H	Optional	5.8.4
27		Writing the function setting (CC-Link IE TSN Class)	3082H	0001H	Optional	5.8.6

*1: Required when handling network-synchronized communications
*2: Required when handling CANopen communications

[Usage of each SLMP command]

- Dual port memory: Reads or writes the general-purpose data from or to the memory of the target station (such as buffer memory).
- Remote control: Operates the target station via a network.
- Error history: Manages error information.
- Slave station parameter automatic setting: The master station automatically distributes parameters to a slave station.
- Event history: Distributes the correction data for synchronization with the time in the master station.
- Watchdog counter: The counter manages the monitoring of synchronization with the master station.
- Access to CAN application objects: Transmission and reception of SDOs and NMTs.
- CC-Link IE TSN device parameter setting: Changes the communication speed or CC-Link IE TSN Class via SLMP.

(2) SLMP commands to be sent and received by the driver (port number 45238)

For reference, the following table lists the SLMP commands to be sent and received by the R-IN32M4-CL3 driver in the sample code.

Table 4.4 SLMP Commands to be Sent and Received by the R-IN32M4-CL3 Driver (Port Number 45238)

No.	Type	Operation	Command	Sub-command	Processing Category
1	Network management	Network setting (main)	0E90H	0000H	Server
2		Network setting (time slot information)	0E90H	0001H	Server
3		Various notifications	0E94H	0000H	Server
4		Various notifications	0E94H	0000H	Client
5	Parameter management	Slave station setting	0E92H	0000H	Server
6		Cyclic transfer setting (main)	0E93H	0000H	Server
7		Cyclic transfer setting (send sub-payload information)	0E93H	0001H	Server
8		Cyclic transfer setting (received sub-payload information)	0E93H	0002H	Server
9		Cyclic transfer setting (target address for reception)	0E93H	0003H	Server

4.3 State Display Function

R-IN32M4-CL3 can display the state of the home station and the state of the Ethernet port by using LEDs.

4.3.1 State Display by LEDs

From the viewpoint of ease of use by the end user, mounting all LEDs other than the user LEDs listed in the table below is recommended.
Mount the LEDs so that the LED lights are visible from the housing of the R-IN32M4-CL3 application product.
Since the LED colors and shapes are not specified, select the LEDs to suit the specifications of the user.

Table 4.5 LED State Display List

LED Type	LED Name		Description	
Home station state display	RUN		Indicates the operating state.	
		On	Operating normally	
		Off	A hardware failure or a watchdog timer error has occurred.	
	SD/SDRD1*1		SD Mode	SDRD1 Mode
			Displays the state of data transmission.	Indicates the data transmission/reception state of Ethernet port 1.
		On	Data being sent	Port 1 data being sent/received
		Off	Data not sent	Port 1 data not sent/received
	RD/SDRD2*1		RD Mode	SDRD2 Mode
			Displays the state of data reception.	Indicates the data transmission/reception state of Ethernet port 2.
		On	Data being received	Port 2 data being sent/received
		Off	Data not received	Port 2 data not sent/received
	D LINK		Indicates the state of a data link.	
		On	Data link in progress (cyclic transfer in progress)	
		Off	Data link not performed (disconnected)	
		Blinking	Data link in progress (cyclic transfer stopped)	
	ERR.		Indicates the error state of R-IN32M4-CL3.	
		On	Error in the home station	
		Off	Operating normally	
	L ERR.		Indicates the error status of the received data and the line. When this LED is on, the port that detected the error can be checked using the L ER LED.	
		On	Abnormal data received or loopback being performed	
		Off	Normal data received or loopback not performed	
	User LED1		Indicates the state defined by the user.	
	User LED2		Indicates the state defined by the user.	
Ethernet Port 1 state display	LINK	On	Link up	
		Off	Link down	
	L ER	On	Abnormal data is received or loopback is being performed.	
		Off	Normal data is received or loopback is not performed	
Ethernet Port 2 state display	LINK	On	Link up	
		Off	Link down	
Ethernet Port 2 state display	L ER	On	Abnormal data is received or loopback is being performed.	
		Off	Normal data is received or loopback is not performed	

Note 1. The SD/SDRD1 LED and RD/SDRD2 LED can switch the mode between SD/RD and SDRD1/SDRD2.
For details, refer to "4.3.2 SD/SDRD1 and RD/SDRD2 LED Lighting Mode Setting".

4.3.2 SD/SDRD1 and RD/SDRD2 LED Lighting Mode Setting

The SD/SDRD1 LED and RD/SDRD2 LED have two lighting modes. Specify the mode in accordance with the user specifications. The lighting mode can be set by using the R-IN32M4-CL3 driver interface function "gerR_IN_SetSDRDLEDMode" (6.4.7(10)).

Table 4.6 Lighting Mode

Lighting Mode	Description
SD mode RD mode	The transmission state and the reception state are displayed by separate LEDs. However, no distinction is made between Ethernet port 1 and Ethernet port 2.
SDRD1 mode SDRD2 mode	The transmission/reception state of Ethernet port 1 and that of Ethernet port 2 are displayed by separate LEDs. However, no distinction is made between the transmission state and the reception state.

4.3.3 Controlling the LEDs

There are LEDs controlled by hardware and those controlled by software.

The LEDs controlled by hardware are turned on/off by the R-IN32M4-CL3 in accord with the state of the home station. These LEDs do not need to be controlled by software.
The LEDs controlled by the software are turned on/off by using R-IN32M4-CL3 driver interface functions in accord with the state of the home station. For details on the R-IN32M4-CL3 driver interface function, refer to "6.4.7 LED Control".
The following table lists the LED control at the time of a reset or error.

Table 4.7 LED Control List

Type	LED Name	R-IN32M4-CL3 Output Signal Name	Control Category	Power-on Reset	System Reset	Internal WDT error, External WDT error
Home station state display	RUN	CCI_RUNLEDZ	Hardware or Software	Off	Off	Off
	SD/SDRD1	CCI_SDLEDZ	Hardware	Off	—	—
	RD/RDSD2	CCI_RDLEDZ	Hardware	Off	—	—
	D LINK	CCI_DLINKLEDZ	Hardware or Software	Off	Off	Off
	ERR.	CCI_ERRLEDZ	Hardware or Software	Off	Off	On
	L ERR.	-	Hardware or software	-	-	-
	User LED1	RP20	Software	Off	Off	Off
	User LED2	RP21	Software	Off	Off	Off
Ethernet port 1 state display	LINK	PHY0_LED0	Hardware	Off	Off	—
	L ER	CCI_LERR1LEDZ	Hardware or Software	Off	Off	Off
Ethernet port 2 state display	LINK	PHY1_LED0	Hardware	Off	Off	—
	L ER	CCI_LERR2LEDZ	Hardware or Software	Off	Off	Off

4.3.4 Controlling User LEDs

User LEDs can be freely defined in accordance with the specifications of the R-IN32M4-CL3 application product.
For example, the on/off/blinking state of user LEDs can be controlled to indicate the following:

- State of online/offline mode (hardware test mode) of the home station
- Normal/error state of various tests such as the hardware test

Control the user LEDs using the R-IN32M4-CL3 driver interface functions "gerR_IN_SetUSER1LED" and "gerR_IN_SetUSER2LED". For details of the R-IN32M4-CL3 driver interface functions, refer to section 6.4.7 LED Control.

4.4 CC-Link IE TSN Diagnostics

This function diagnoses the entire CC-Link IE TSN and detects setting errors or communications errors.

With the function, the states of the individual stations are collected in the master station by responding to diagnostic requests from the master station.
Response processing is automatically performed by the R-IN32M4-CL3 driver. The user does not need to implement specific processing in the user program.
The diagnostic screen when using GX Work3 is shown below.

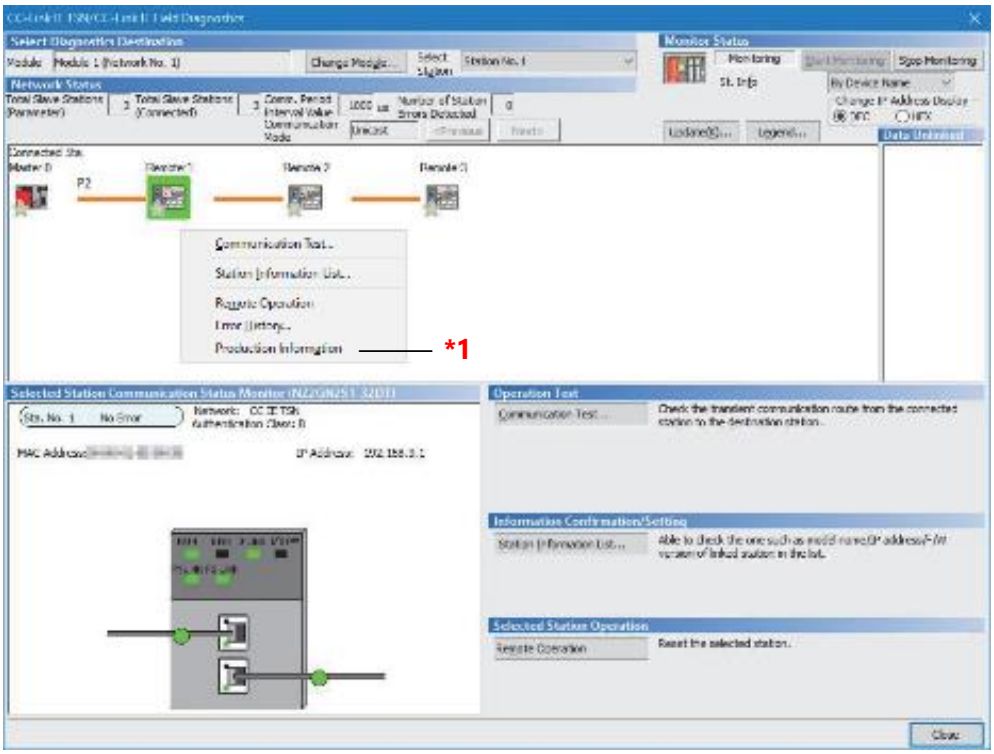


Figure 4.5 Diagnostic Window on GX Works3

* 1: R-IN32M4-CL3 application product does not support "Production Information". When "Product Information" is selected, an error message will be displayed on GX Works3.

4.5 Network-Synchronized Communications

Network synchronization communication is a function that performs internal processing of slave stations in the synchronization cycle of the master station. It allows alignment of the timing of the internal processing by slave stations with other slave stations that are connected to the same network.

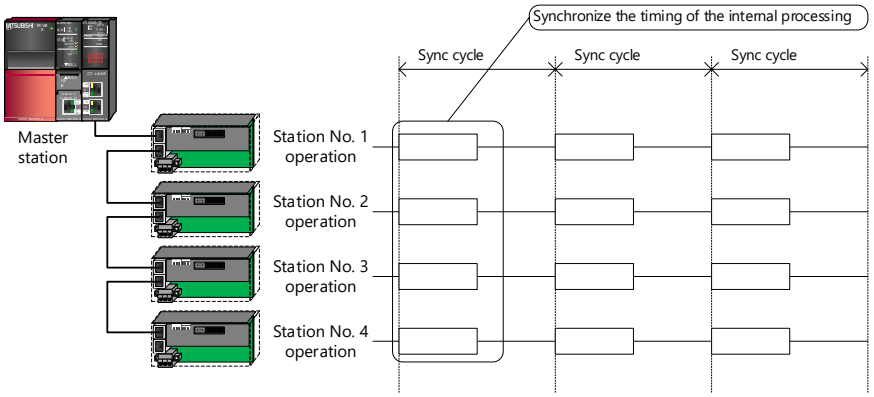


Figure 4.6 Schematic View of Network-Synchronized Communications

Using network-synchronized communications requires the following three tasks and implementation of processing to be called by the tasks.

Table 4.8 Tasks Related to Network-Synchronized Communications

No	Task	Task ID	Related Processing (for Reference)	reference
1	Low-priority interrupt task	TSKID_NX_LOW_INT	This task is awoken when CC-Link IE TSN generates a low-priority interrupt and calls back "synchronous cyclic communications processing" if the interrupt source is "cyclic transfer completed"	6.6(9)
2	Synchronous timing processing task	TSKID_SYNCPROC	This task is awoken with the timing of synchronous signal output*1 and executes "synchronous timing processing	5.2.5
3	Processing task at the time of synchronous disconnection	TSKID_PERIO_DLERR	This task is awoken with a fixed period if network-synchronized communications is enabled and executes "processing at the time of synchronous disconnection"	5.2.6

Note 1. When RJ71GN11-T2 is to be the master station, the start of TS1 is the time the synchronous timing processing task is awoken.

Additionally, a master station which supports network-synchronized communications is required.
When using the Mitsubishi Electric master station (RJ71GN11-T2), use a unit with firmware version "11" or later. For details on the setting method and functions, refer to the related manuals below.

- MELSEC iQ-R CC-Link IE TSN User's Manual (Application)
- MELSEC iQ-R Inter-Module Synchronization Function Reference Manual

<Points to note>
For CC-Link IE TSN Class A, this function cannot be used.
For CC-Link IE TSN Class B, this function cannot be used in CANopen communications or safety communications.

The following is the timing chart of processing to be called by three tasks.
(1) to (3) in the figure are related processing listed in Table 4.8.

(1) Network synchronous communication task/processing timing chart (Data Link is in Progress)
"1) Synchronous cyclic communications processing" is called back, and received data are read and data for transmission are written. "2) Synchronous timing processing" indicates execution of processing of the user application you want to synchronize.

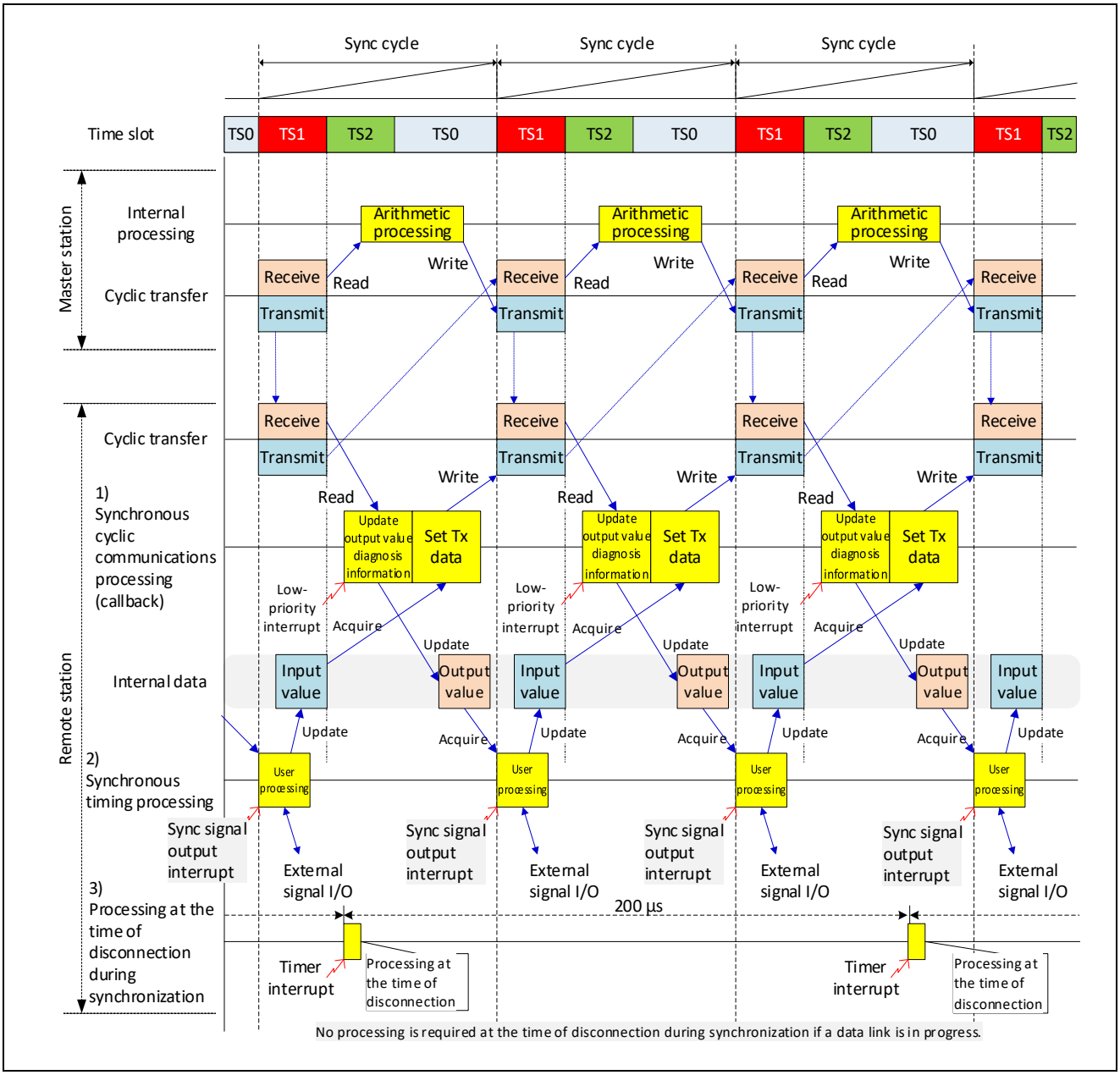


Figure 4.7 Synchronous Communications Task/Processing Timing Chart (Data Link in Progress)

(2) Network synchronous communication task/processing timing chart (Data Link not Performed (Disconnected))
In the disconnected state, "1) Synchronous cyclic communications processing" is not called back and "2) Synchronous timing processing" cannot also be executed. Processing required at the time of disconnection is executed by "3) Processing at the time of synchronous disconnection".

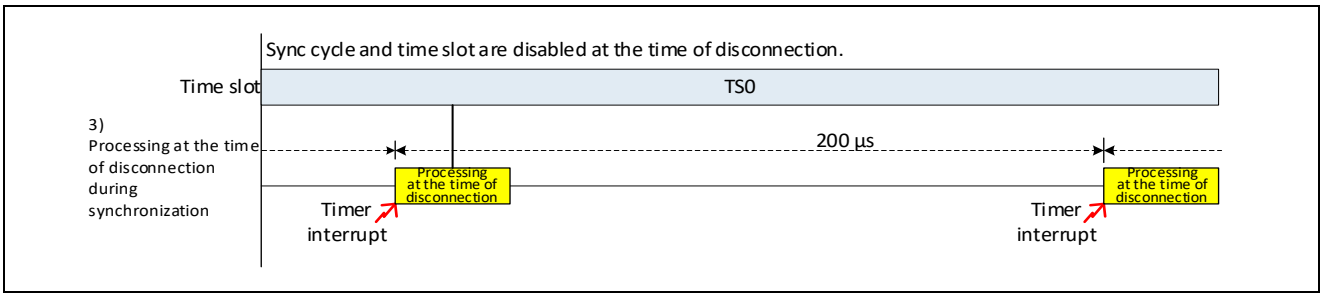


Figure 4.8 Synchronous Communications Task/Processing Timing Chart (Data Link not Performed (Disconnected))

4.6 CANopen Communications

CANopen communications is used to control devices which support the CANopen profile.
The target devices are controlled by operations equivalent to those for link devices by utilizing PDOs for transmission and reception in cyclic transfer or SDOs for transmission and reception in transient transfer (according to the SLMP).

Note that this manual mainly describes the method for sending and receiving SDO and PDO in CC-Link IE TSN. For details of CANopen, refer to the CANopen specification.

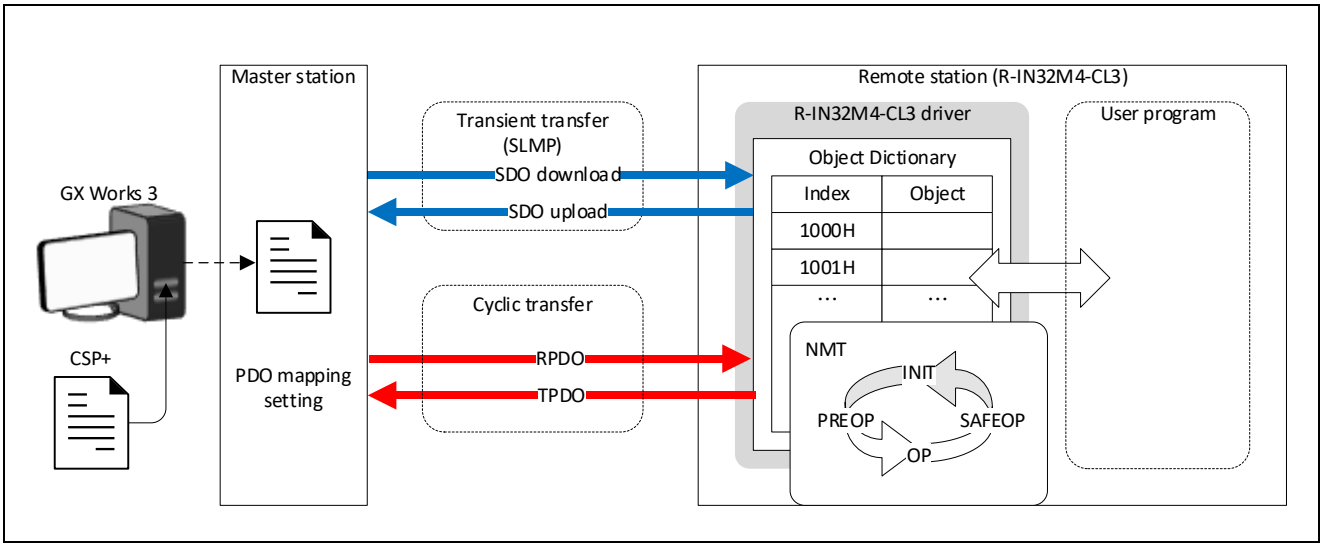


Figure 4.9 Schematic View of CANopen Communications in CC-Link IE TSN

Table 4.9 Overview of CANopen communication in CC-Link IE TSN

No	Item	Overview
1	Object Dictionary (OD)	The aggregate of various data such as control parameters and command values held by the R-IN32M4-CL3 application product is called the object dictionary. Each entry in the object dictionary is identified by anindex (16 bits) and sub-index (8 bits).
2	Network Management (NMT)	Network Management (NMT) manages the communication status of remote stations. Communication status changes by initial processing between the master station and the remote station or SLMP (NMTState Download).
3	Service Data Object (SDO)	Service data object (SDO) is a message for the master station to access the Object Dictionary of the remote station. SDO uses transient transmission (SLMP) and is transmitted and received aperiodically between the master station and the remote station.
4	Process Data Object (PDO)	A process data object (PDO) is the aggregate of application objects to be transferred synchronously between the stations for control and monitoring of devices. PDO uses cyclic transmission and is periodically transmitted and received between the master station and the remote station. The data stored in the PDO is determined by the PDO mapping settings, the PDO stored in the cyclic transmit data is called Transmit PDO (TPDO), and the PDO data stored in the cyclic receive data is called Receive PDO (RPDO). increase.

The PDO data contents are defined by the PDO mapping and set by "batch setting of PDO mapping" or "PDO mapping setting" (in the case of GX Works 3).

Using CANopen communications requires the following:

No	Requirements	reference
1	Use the development environment for CANopen communication.	2.4
2	Enable the compile switch "TSN_CAN_ENABLE".	3.4.4
3	Implement user programs related to CANopen communication	5.9
4	Create a CSP + file for CANopen communication.	2.6

Additionally, a master station which supports CANopen communications is required.
When using the Mitsubishi Electric master station (RJ71GN11-T2), use a unit with firmware version "12" or later. For details on the setting method and functions, refer to the related manuals below.

- MELSEC iQ-R CC-Link IE TSN User's Manual (Application)

4.6.1 Expansion unit for CANopen communication

During CANopen communication, by defining multiple Object Dictionary numbers for R-IN32M4-CL3 applicable products, the number of axes can be expanded, for example, like a servo amplifier.
Number of Object Dictionary can be changed by setting the macro definition "R_IN_CAN_MAX_ODTABLE_NUM" (R_IN32M4_CL3CanConst.h) of the R-IN32M4-CL3 sample code.
Following shows an image of an expansion unit, using as an example a multi-axis integrated servo amplifier consisting of a "basic unit (axis 1)" that communicates and an "expansion unit (axis 2-3)" that does not communicate.

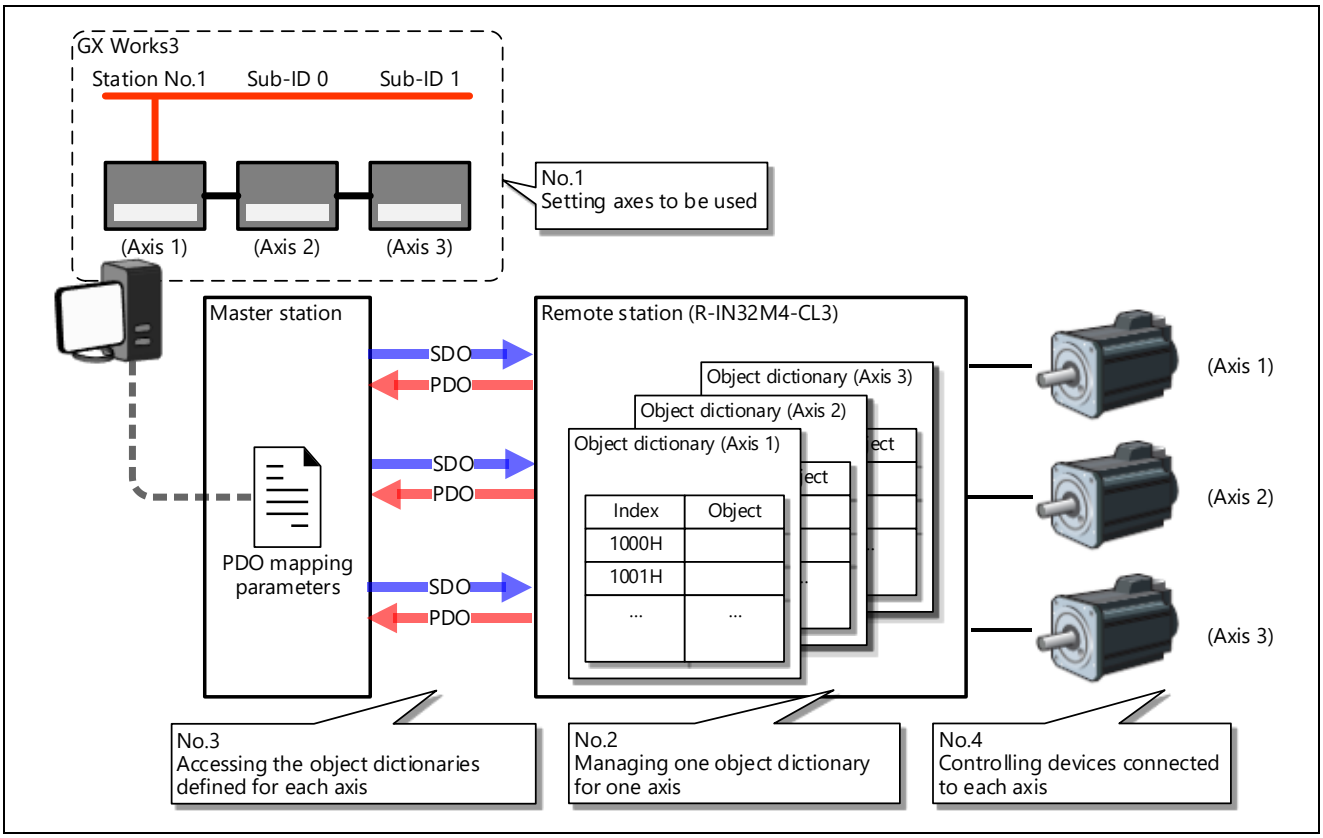


Figure 4.10 Image of Extension Modules used in CANopen Communications

No	Item	Overview
1	Setting the axis to use	In the network configuration diagram of the engineering tool, the axis to be used can be set arbitrarily by joining the expansion unit to the basic unit. (Example: Setting to use 1 to 2 axes in a unit that supports up to 3 axes)
2	Object of each axis Dictionary management	Object Dictionary of the remote station (R-IN32M4-CL3) is defined for each axis. When using in a maximum 3-axis configuration, define a total of 3 Object Dictionaries, a basic unit (axis 1) and an expansion unit (axis 2 to 3), and "gerR_IN_CanInit" (6.4.16(1) Initial of CANopen communication function Please set it in the argument of the conversion.
3	Object of each axis Access to Dictionary	Use SDO (SLMP) to access the Object Dictionary of each axis of the remote station (R-IN32M4-CL3) from the master station. The "Axis number (Object Dictionary number)" is stored in the "Requested station processor sub number" of the SLMP frame.
4	Control of equipment on each axis	The data in the Object Dictionary for each axis is used to control each axis from the user application.

Following items are required to use the CANopen communication expansion unit.

No	Necessary items	reference
1	Enable the compile switch "TSN_CAN_MULTIAXIS_ENABLE"	3.4.4
2	Change various settings according to the number of expansion units installed.	Table 5.30
3	Create CSP + files for the basic unit and expansion unit.	2.6

4.7 Safety PDU Send/Receive In Safety Communications

In safety communications, this function notifies the user program of a safety PDU received from the master station (safety) in cyclic transmission or sends a safety PDU created by the user program to the master station (safety) in cyclic transmission.

The function acquires or sets the safety PDU without the consideration of sub-payload configuration in cyclic transmission.

The safety PDU and RX/RY/RWr/RWw can be used together.

The following shows an image of safety communications.

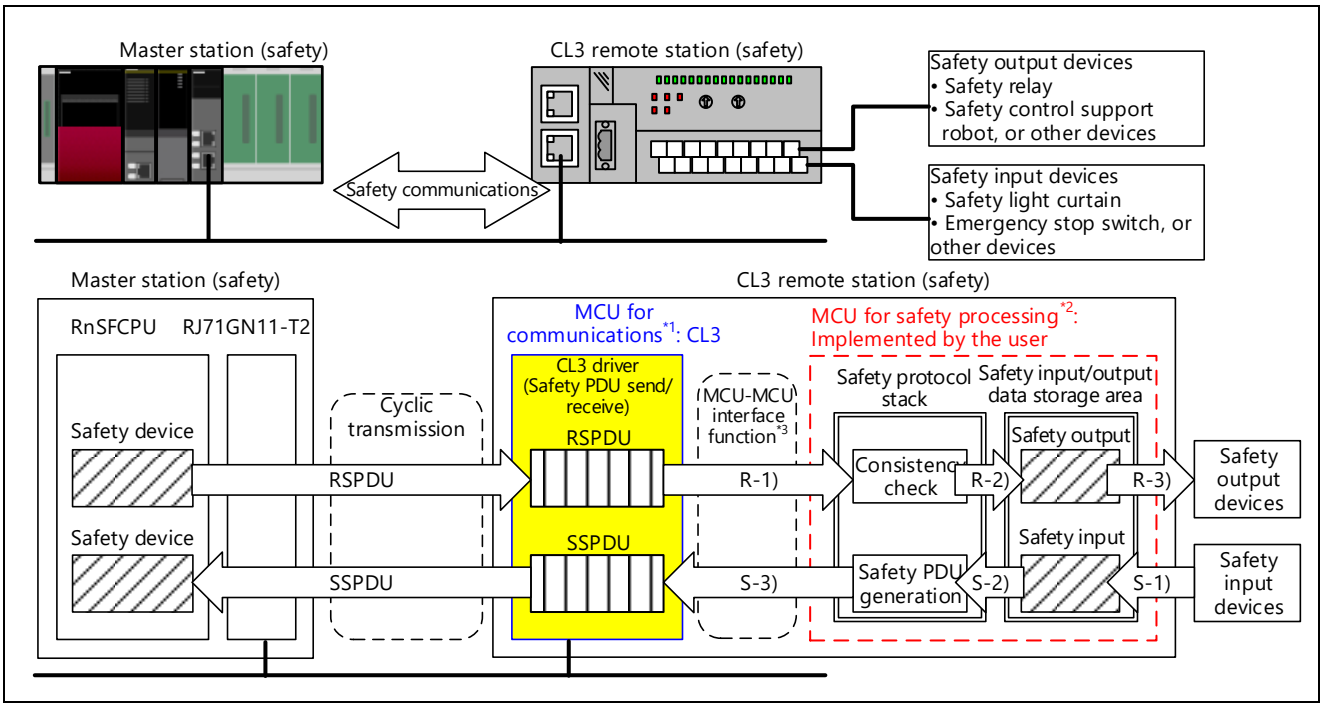


Figure 4.11 Image of Safety Communications

Category	Implementation location	Processing overview
For receive	R-1)	Reads the RSPDU data from the R-IN32M4-CL3 driver and delivers it to the safety protocol stack.
	R-2)	Acquires the safety output data extracted from the safety protocol stack and stores it into the safety output area.
	R-3)	Controls the safety output device based on the output value of the safety output area.
For send	S-1)	Sets the value input from the safety input device to the safety input area.
	S-2)	Delivers the safety input data to the safety protocol stack and generates the safety PDU.
	S-3)	Sets the generated safety PDU to the R-IN32M4-CL3 driver.

Note 1. Used for controlling the communications with the master station (safety) (including safety communications with the MCU for safety processing).

Note 2. Used for safety communications (safety PDU generation and consistency check) and the control related to the safety such as the safety input/output.

Note 3. The safety PDU is sent/received between the MCU for communications and MCU for safety processing. For details, refer to "Section 5.10 Details on Processing of User Programs (MCU-MCU Interface Related)".

To perform safety communications, satisfy the following.

No	Item	Reference
1	Select the MCU for safety processing (external MCU) and connect it to the R-IN32M4-CL3.	-
2	Enable the compiler switch "SAFETY_PDU_ENABLE" and "MCUIF_ENABLE".	3.4.4
3	Implement tasks related to the MCU-MCU interface.	5.2
4	Implement the user program related to the MCU-MCU interface.	5.10
5	Create a CSP+ file for safety communications.	2.6

Furthermore, the master station supporting safety communications is required.

For the master station manufactured by Mitsubishi Electric (RJ71GN11-T2), use the following modules.

- Safety CPU (RnSFCPU) with firmware version 20 or later and safety function module (R6SFM)
- RJ71GN11-T2 with firmware version 10 or later

For details on the setting methods and functions, refer to the following manual.

- MELSEC iQ-R CC-Link IE TSN User's Manual (Application)

4.8 Communication Speed and CC-Link IE TSN Class Setting via SLMP

This function stores the communication speed and CC-Link IE TSN Class of the own station to the non-volatile memory via SLMP and reflects them next time the power is turned on. The function switches communication speeds via SLMP (via a network) and based on the CC-Link IE TSN Class, as well as using a hardware switch.

Use CC-Link IE TSN Configurator (CC-Link IE TSN configuration tool) as an SLMP client. Download the tool from the CC-Link Partner Association website in advance.

To set the communication speed and CC-Link IE TSN Class via SLMP, satisfy the following.

No.	Item	Reference
1	Select non-volatile memory, and connect it to R-IN32M4-CL3.	-
2	Implement a user program related to the communication speed setting and CC-Link IE TSN Class.	5.8

This function is recommended to be executed before the system is started up (before the R-IN32M4-CL3 application product is implemented in the system).

<Points to note>
(When the R-IN32M4-CL3 application product operates as CC-Link IE TSN Class A)
When this function is executed, the SLMP command (Detecting connected devices (03E0H)) is sent/received in broadcast mode with a response so that CC-Link IE TSN Configurator detect the setting items (communication speed and CC-Link IE TSN Class). Response frames that are sent by the R-IN32M4-CL3 application product may be lost in the process of relay and CC-Link IE TSN Configurator may not be able to detect the R-IN32M4-CL3 application product due to the number of connected modules, which changes depending on the operating environment or frame size.
To avoid this failure, disconnect the modules in the entire system, and execute this function.

5. Creating User Programs

This section describes an overview of processing by user programs in the sample code.
A user program is sample processing for checking the communications processing logic of a remote station.

5.1 List of User Programs

The following tables list user programs and their implementation necessity.

Table 5.1 List of User Programs Related to Initialization

No.	Function Name	Overview	Implementation Necessity	Refer to
1	iUserInitialization	Initialization processing	Required	5.3.1
2	iUserStart	Communications start processing	Required	5.3.2
3	UserSlmpReceiveInitial	SLMP reception initialization processing	Required	5.3.3
4	UserSlmpMakeRequestInitial	SLMP request frame creation initialization processing	Optional	5.3.4
5	UserInitialParameterOperation	Parameter operation initialization processing	Optional	5.3.5

Table 5.2 List of User Programs Related to Cyclic Transfer

No.	Function Name	Overview	Implementation Necessity	Refer to
1	UserReceiveCyclic	Cyclic reception processing	Required	5.4.1
2	UserSendNodeStatus	Home station state transmission processing	Required	5.4.2
3	UserSendCyclic	Cyclic transmission processing	Required	5.4.3
4	UserUpdateStatus	Communications state update processing	Required	5.4.4
5	UserGetCyclicStatus	Cyclic transfer state update processing	Optional	5.4.5

Table 5.3 List of User Programs Related to State Management and Transient Transfer

No.	Function Name	Overview	Implementation Necessity	Refer to
1	UserForceStop	Home station error processing	Required	5.5.1
2	iUserExecuteMain	Event processing	Required	5.5.2
3	UserUpdateLed	LED update processing	Required	5.5.3
4	UserGetMIB	MIB (statistical) information acquisition processing	Optional	5.5.4
5	UserSlmpReceive	SLMP reception processing	Required	5.5.5
6	UserSendSlmp	SLMP transmission processing	Required	5.5.6
7	UserSlmpMakeRequest	SLMP request frame creation processing	Optional	5.5.7
8	UserSlmpReceive3EFrame	SLMP ST (3E) response frame reception processing	Optional	5.5.8
9	UserConfirmFatalError	Processing for checking a fatal error detected in the R-IN32M4-CL3 driver	Optional	5.5.9

Table 5.4 List of User Programs Related to SLMP Command Execution

No.	Function Name	Overview	Type	Implementation Necessity	Refer to
1	erUserSlmpReceiveMemoryReadRequest	SLMP memory read request command reception processing	server	Optional	5.6.1
2	erUserSlmpReceiveMemoryWriteRequest	SLMP memory write request command reception processing	server	Optional	5.6.2
3	iUserSlmpMakeMemoryReadRequest	SLMP memory read request command creation processing	client	Optional	5.6.3
4	UserSlmpReceiveMemoryReadResponse	SLMP memory read response command reception processing	client	Optional	5.6.4
5	erUserSlmpRemoteReset	SLMP remote reset request command reception processing	server	Optional	5.6.5
6	erUserSlmpNodeIndication	SLMP indicator display request command reception processing	server	Optional	5.6.6
7	erUserSlmpSetIpAddress	SLMP IP address change request command reception processing	server	Optional	5.6.7
8	erUserSlmpClearErrorHistory	SLMP error history clearing request command reception processing	server	Optional	5.6.8
9	erUserSlmpClockOffsetDataSend	SLMP network time offset distribution command reception processing	server	Optional	5.6.9
10	erUserSlmpNetworkClockDataSend	SLMP network time distribution command receive processing	Server	Optional	5.6.10

Table 5.5 List of User Programs Related to Slave Station Parameter Automatic Setting

No.	Function Name	Overview	Type	Implementation Necessity	Refer to
1	erUserSlmpGetCommunicationSet	SLMP communications settings acquisition request command reception processing	server	Optional	5.7.1
2	erUserSlmpCheckParameterDelivery	SLMP parameter distribution necessity check request command reception processing	server	Optional	5.7.2
3	UserSlmpCheckParameterDeliveryMain	SLMP parameter distribution necessity check processing	server	Optional	5.7.3
4	erUserSlmpStartRestore	SLMP restoration start notification request command reception processing	server	Optional	5.7.4
5	erUserSlmpEndRestore	SLMP restoration end notification request command reception processing	server	Optional	5.7.5
6	erUserSlmpSetParameter	SLMP parameter data write request command reception processing	server	Optional	5.7.6
7	UserSlmpSetParameterMain	SLMP parameter data write processing	server	Optional	5.7.7

Table 5.6 List of User Programs Related to CC-Link IE TSN Device Parameter Setting

No.	Function Name	Overview	Type	Implementation Necessity	Refer to
1	erUserSlmpSearchNodeExtention	SLMP connected device detection (extended) request command reception processing	server	Optional	5.8.1
2	erUserSlmpGetFunctionSettingInfo	SLMP function setting (support information acquisition) Request command reception processing	server	Optional	5.8.2
3	erUserSlmpReadLinkSpeed	SLMP function setting read (communication speed) request command reception processing	server	Optional	5.8.3
4	erUserSlmpWriteLinkSpeed	SLMP function setting write (communication speed) request command reception processing	server	Optional	5.8.4
5	erUserSlmpReadCCIETSNClass	SLMP function setting read (CC-Link IE TSN Class) request command receive processing	Server	Optional	5.8.5
6	erUserSlmpWriteCCIETSNClass	SLMP function setting write (CC-Link IE TSN Class) request command receive processing	Server	Optional	5.8.6

Table 5.7 List of User Programs Related to Network-Synchronized Communications

No.	Function Name	Overview	Implementation Necessity	Refer to
1	UserSyncTimingRoutine	Synchronous timing processing	*1	5.2.5
2	UserSyncComLinkErrorRoutine	Processing at the time of disconnection during synchronization	*1	5.2.6
3	erUserSlmpSetWatchdogCounterInfo	SLMP watchdog counter information setting request command reception processing	*1	5.6.11

Note 1. Handling network-synchronized communications requires implementation of this function.

Table 5.8 List of User Programs Related to CANopen Communications

No.	Function Name	Overview	Implementation Necessity	Refer to
1	iUserCanInitial	CANopen communications function initialization processing	*1	5.9.1
2	UserReceiveCyclic	Cyclic reception processing (updating RPDOs)	*1	5.9.2
3	UserSendCyclic	Cyclic transmission processing (updating TPDOs)	*1	5.9.3
4	erUserSlmpCanReadObject	SLMP ReadObject request command reception processing	*1	5.9.4
5	erUserSlmpCanWriteObject	SLMP WriteObject request command reception processing	*1	5.9.5
6	erUserSlmpCanObjectSubIDReadBlock	SLMP ObjectSubIDReadBlock request command reception processing	*1	5.9.6
7	erUserSlmpCanObjectSubIDWriteBlock	SLMP ObjectSubIDWriteBlock request command reception processing	*1	5.9.7
8	erUserSlmpCanNmtStateUpload	SLMP NMTStateUpload request command reception processing	*1	5.9.8
9	erUserSlmpCanNmtStateDownload	SLMP NMTStateDownload request command reception processing	*1	5.9.9
10	usUserSlmpCanGetFinishCode	SLMP end code acquisition processing	*1	5.9.10
11	iUserCanSetParameter	CANopen parameter setting processing	*1	5.9.11
12	gulWriteFunc1010	Index1010 write processing	*1	5.9.12
13	gulWriteFunc1011	Index1011 write processing	*1	5.9.13

Note 1. Handling CANopen communications requires implementation of this function.

Table 5.9 List of User Programs Related to the MCU-MCU Interface

No.	Function Name	Overview	Implementation Necessity	Refer to
1	UserMcuIfInitial	MCU-MCU interface initialization processing	*1	5.10.1
2	UserMcuIfAckGpioSet	GPIO communication acceptance processing	*1	5.10.2
3	UserMcuIfAckGpioClear	GPIO communication acceptance clear processing	*1	5.10.3
4	erUserMcuIfRequestGpioSet	GPIO communication event type/code setting processing	*1	5.10.4
5	UserMcuIfSafetyPduTransfer	Safety PDU transfer processing (internal → external MCU)	*1	5.10.5
6	UserMcuIfSafetyPduReceived	Safety PDU transfer processing (internal ← external MCU)	*1	5.10.6
7	UserMcuIfSafetyPduTransferResponse	Safety PDU transfer ready (internal → external MCU) acceptance receive processing	*1	5.10.7
8	UserMcuIfSafetyPduReceivedResponse	Safety PDU transfer ready (internal ← external MCU) acceptance receive processing	*1	5.10.8
9	UserMcuIfSafetyPduReceivedComplete	Safety PDU transfer completion processing (internal ← external MCU)	*1	5.10.9

Note 1. Implementation is required for sending/receiving safety PDUs.

Table 5.10 List of User Programs Related to Hardware Test

No.	Function Name	Overview	Implementation Necessity	Refer to
1	UserIEEETest	Hardware test processing (IEEE802.3ab compliance test)	Required	5.11.1
2	UserLoopBackTest	Hardware test processing (loopback communications test)	Optional	5.11.2
3	UserLoopBackTestInitial	Loopback communications test preparation processing	Optional	5.11.3
4	UserLoopBackTestLinkCheck	Loopback communications test link state check processing	Optional	5.11.4
5	UserLoopBackTestSend	Loopback communications test data transmission processing	Optional	5.11.5
6	UserLoopBackTestReceived	Loopback communications test reception result determination processing	Optional	5.11.6
7	UserLoopBackTestExit	Loopback communications test completion processing	Optional	5.11.7

5.2 User Program Tasks

Describe the generated information to be described in the RTOS configuration file in the user program. For details of the settings, refer to "R-IN32M4-CL3 Programming Manual: OS edition".
Settings should not be changed unless there is a specific reason.
When creating a new task by the user, also check the generation information described in "6.3 R-IN32M4-CL3 Driver Tasks" before creating it.

Table 5.11 List of User Program Tasks

No.	Task Name	Task ID	T_CTSK Structure Member					
			Task Attribute	Extension Information	Start Address (Described by function name)	Priority at Startup	Stack Size	Start Address of Stack Area
1	Initial task	TSKID_INITIAL(1)	TA_HLNG | TA_ACT	0000H	vTask_Initial	1	400H	NULL
2	Periodic processing task	TSKID_PERIODIC(3)	TA_HLNG	0000H	vTask_Periodic	3	400H	NULL
3	Idle task	TSKID_IDLE(2)	TA_HLNG	0000H	vTask_Idle	15	1000H	NULL
4	Error management task	TSKID_ERR(5)	TA_HLNG	0000H	vTask_Err	10 (13)※1	1800H	NULL
5	Synchronous timing processing task	SKID_SYNCPROC(9)	TA_HLNG	0000H	vTask_SyncProc	2	400H	NULL
6	Task for processing at the time of disconnection during synchronization	TSKID_PERIO_DLERR (10)	TA_HLNG	0000H	vTask_Periodic_ DLinkErr	3	400H	NULL
7	Scheduler task	TSKID_SCHEDULER(11)	TA_HLNG	0000H	vTask_Scheduler	12	500H	NULL
8	GPIO communication send task	TSKID_MSGP_SEND(12)	TA_HLNG	0000H	vTask_MsgGpio_S end	3	600H	NULL
9	GPIO communication response receive task	TSKID_MSGP_ANS(13)	TA_HLNG	0000H	vTask_MsgGpio_A ns	3	400H	NULL
10	GPIO communication receive task	TSKID_MSGP_RECV(14)	TA_HLNG	0000H	vTask_MsgGpio_R ecv	3	500H	NULL
11	RT DMA transfer completion task	TSKID_RTDMA_DONE (15)	TA_HLNG	0000H	vTask_RtDma_Do ne	3	500H	NULL
12	Cyclic frame send processing task (CC-Link IE TSN Class A)	TSKID_SENDCYCFRM _CLASS_A(16)	TA_HLNG	0000H	vTask_SendCyclic FrameClassA	4	400H	NULL

Note 1. The priority is lowered from 10 to 13 not to interfere with the operation of the scheduler task for time measurement when the MCU-MCU interface function is enabled (the compiler switch "MCUIF_ENABLE" is enabled).

The tasks to start differ depending on the CC-Link IE TSN Class and whether network synchronous communications are set. The following table lists the tasks to start in each case.

Network synchronous communications are performed when the product that is ranked as CC-Link IE TSN Class B operates and the master station has specified "Synchronous" to the "Network Synchronous Communication" parameter. Network synchronous communications cannot be performed when the product that is ranked as CC-Link IE TSN Class A operates.

Table 5.12 List of Tasks to Start for Each Communications Setting

No.	Task Name	Without network synchronous communications		With network synchronous communications
		CC-Link IE TSN Class A	CC-Link IE TSN Class B	CC-Link IE TSN Class B
1	Initial task	○	○	○
2	Fixed scan processing task	○	○	○
3	Idle task	○	○	○
4	Error management task	○	○	○
5	Synchronization timing processing task	×	×	○
6	Synchronization disconnection processing task	×	×	○
7	Scheduler task	○*1	○*1	×
8	GPIO communication send task	○*1, *2	○*1, *2	×
9	GPIO communication response receive task	○*1, *2	○*1, *2	×
10	GPIO communication receive task	○*1, *2	○*1, *2	×
11	RT DMA transfer completion task	○*1, *2	○*1, *2	×
12	Cyclic frame send processing task (CC-Link IE TSN Class A)	○	×	×

○: Start, ×: Do not start

Note 1. These tasks start when the MCU-MCU interface function is enabled (the compiler switch "MCUIF_ENABLE" is enabled).

Note 2. These tasks start when the GPIO communications with the external MCU start.

Table 5.13 Semaphore Generation Information

No.	Name	ID	Semaphore Attribute*1	Initial Value of Number of Resources	Max. Number of Resources
1	Fatal error management semaphore	SEMID_FATALERROR (128)	TA_TFIFO	1	1
2	Error history management semaphore	SEMID_ERRHIS (127)	TA_TFIFO	1	1
3	PHY management semaphore	SEMID_PHY (126)	TA_TFIFO	1	1
4	SLMP reception information management semaphore	SEMID_SLMP_RES_INFO (125)	TA_TFIFO	1	1

Note 1. µITRON defines semaphore attributes as follows (the µITRON specification, ver 4.03).

TA_TFIFO(00H): Task queues are managed in FIFO order.

TA_TPRI (01H): Task queues are managed in the priority order of the tasks.

Table 5.14 Mailbox Creation Information

No.	Name	ID	T_CMBX structure member		
			Mailbox attribute	Highest message priority	Start address of the area for message queue headers for each message priority
1	GPIO communication send*1	MBXID_MSGP(50)	TA_TPRI | TA_MFIFO	1	NULL

Note 1. When the MCU-MCU interface function is enabled (the compiler switch "MCUIF_ENABLE" is enabled)

Table 5.15 Hardware ISR Settings

No.	Hardware ISR Name	Target Interrupt Number		Exception No.	Service Call to be Automatically Executed in Response to an Interrupt	Hardware ISR object ID
1	Periodic processing	0 (TAUJ2 channel 0 interrupt)		16	HWISR_WUP_TSK	TSKID_PERIODIC
2	Error management	Standard communications	62 (INTPZ15 input/TAUJ2 channel 9 interrupt)	78	HWISR_WUP_TSK	TSKID_ERR
		Safety communications*2	2 (TAUJ2 channel 2 interrupt)			
3	Synchronous timing processing*1	114 (synchronization signal output (for application))		130	HWISR_WUP_TSK	TSKID_SYNCPROC
4	Processing at the time of synchronous disconnection*1	1 (TAUJ2 channel 1 interrupt)		17	HWISR_WUP_TSK	TSKID_PERIO_DLERR
5	Scheduler task*2	1 (TAUJ2 channel 1 interrupt)		17	HWISR_WUP_TSK	TSKID_SCHEDULER
6	RT DMA transfer completion task*2	26 (Real-time port DMAC transfer completion interrupt)		42	HWISR_WUP_TSK	TSKID_RTDMA_DONE
7	GPIO communication response receive task*2	61 (INTPZ14 input/TAUD channel 8 interrupt)		77	HWISR_WUP_TSK	TSKID_MSGP_ANS
8	GPIO communication receive task*2	62 (INTPZ15 input/TAUD channel 9 interrupt)		78	HWISR_WUP_TSK	TSKID_MSGP_RECV

Note 1. When network synchronous communications are performed

Note 2. When the MCU-MCU interface function is enabled (the compiler switch "MCUIF_ENABLE" is enabled)

Table 5.16 Timer Usage List

No.	Category	Channel Number	Application		Base Time Number	Operating Clock	Time
1	32-bit timer (TAUJ2)	0	To perform the fixed scan processing in a fixed cycle.		Base time 1	CK0	0.01 μs (PCLK/20)
2		1	Standard communications	To perform the synchronization disconnection processing in a fixed cycle.			
			Safety communications*1	To perform the scheduler task in a fixed cycle.			
3		2	Standard communications	Not used			
			Safety communications*1	To perform the error management processing (SNMP) in a fixed cycle.			
4	16-bit timer (TAUD)	9	Standard communications	To perform the error management processing (SNMP) in a fixed cycle.	Base time 4	CK3_PRE	0.16 μs (PCLK/24)
			Safety communications*1	Not used			

Note 1. When the MCU-MCU interface function is enabled (the compiler switch "MCUIF_ENABLE" is enabled)

For details of the timers, refer to the R-IN32M4-CL3 User's Manual: Hardware edition.

5.2.1 Initial Task

(1) Task Processing Overview

This task initializes, sets up, and starts up the timers to be used (TAUJ2, TAUD) and starts up the idle task.
This task executes processing only once at start-up.

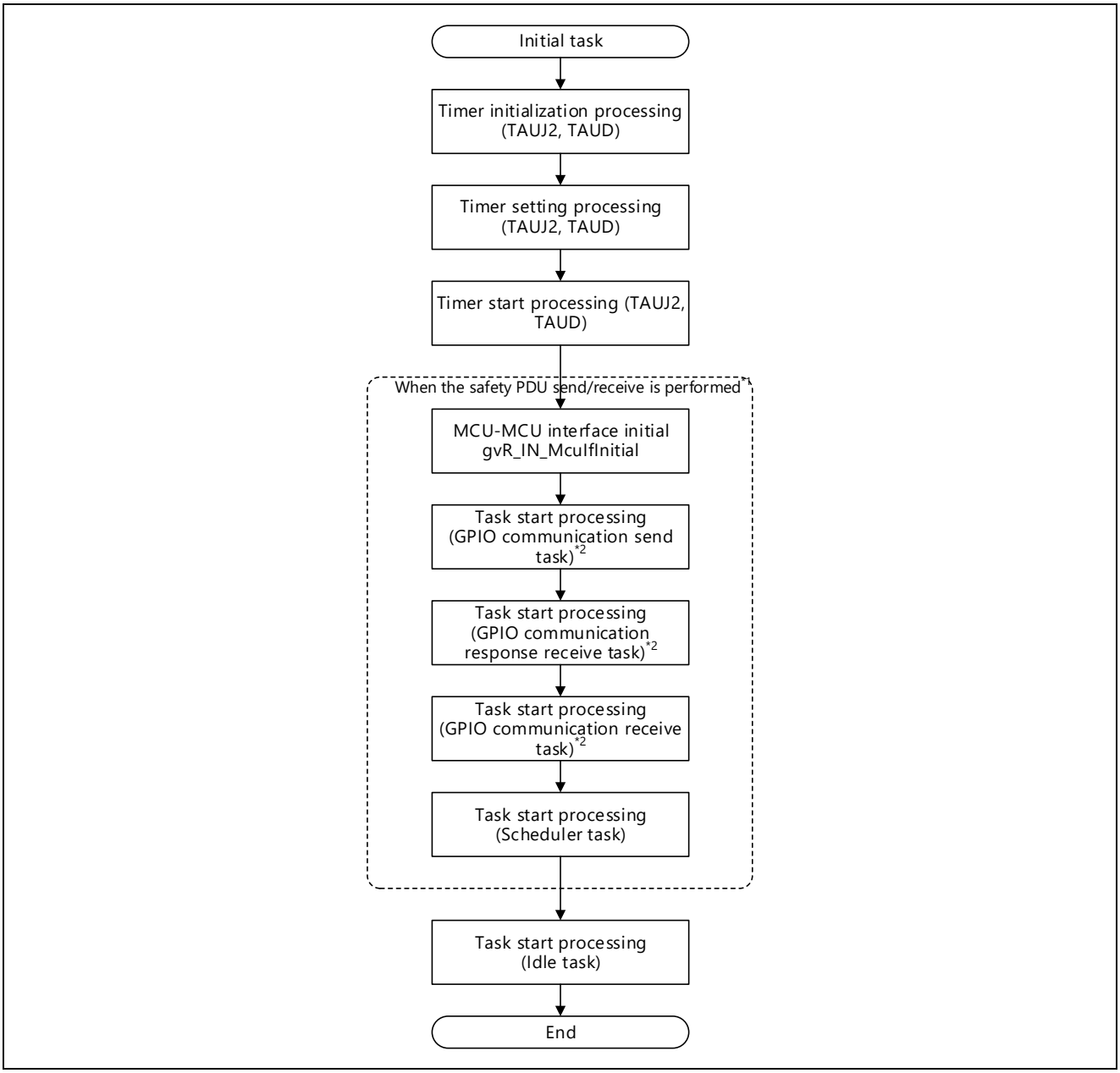


Figure 5.1 Initial Task Processing Overview

Note 1. These processing are performed when the MCU-MCU interface function is enabled (the compiler switch "MCUIF_ENABLE" is enabled).

Note 2. These tasks start when the master station specifies "1b: Supported" in "Safety communications".

The values to be set for each timer are as follows.

Table 5.17 Timer Usage List

Timer to be Used		Clock Setting	Timeout Setting
32-bit timer (TAUJ2)	TM00 TM01	CK0 (0.01us = PCLK / (2^0))	20000 – 1 (1 clock cycle is 10 ns so the period is 200 µs)
16-bit timer (TAUD)	TM09	CK3 (0.16us = PCLK / (2^4))	62500 – 1 (1 clock cycle is 160 ns so the period is 10 ms)

5.2.2 Idle Task

(1) Task Processing Overview

This task initializes the firmware and hardware only once at start-up. The task repeats state management and transient main processing by an endless loop after the initialization processing.

This task performs processing with less-restricted time by an endless loop. The priory of this task is the lowest.
The task is executed when no other tasks are being executed. The following is the task processing overview.

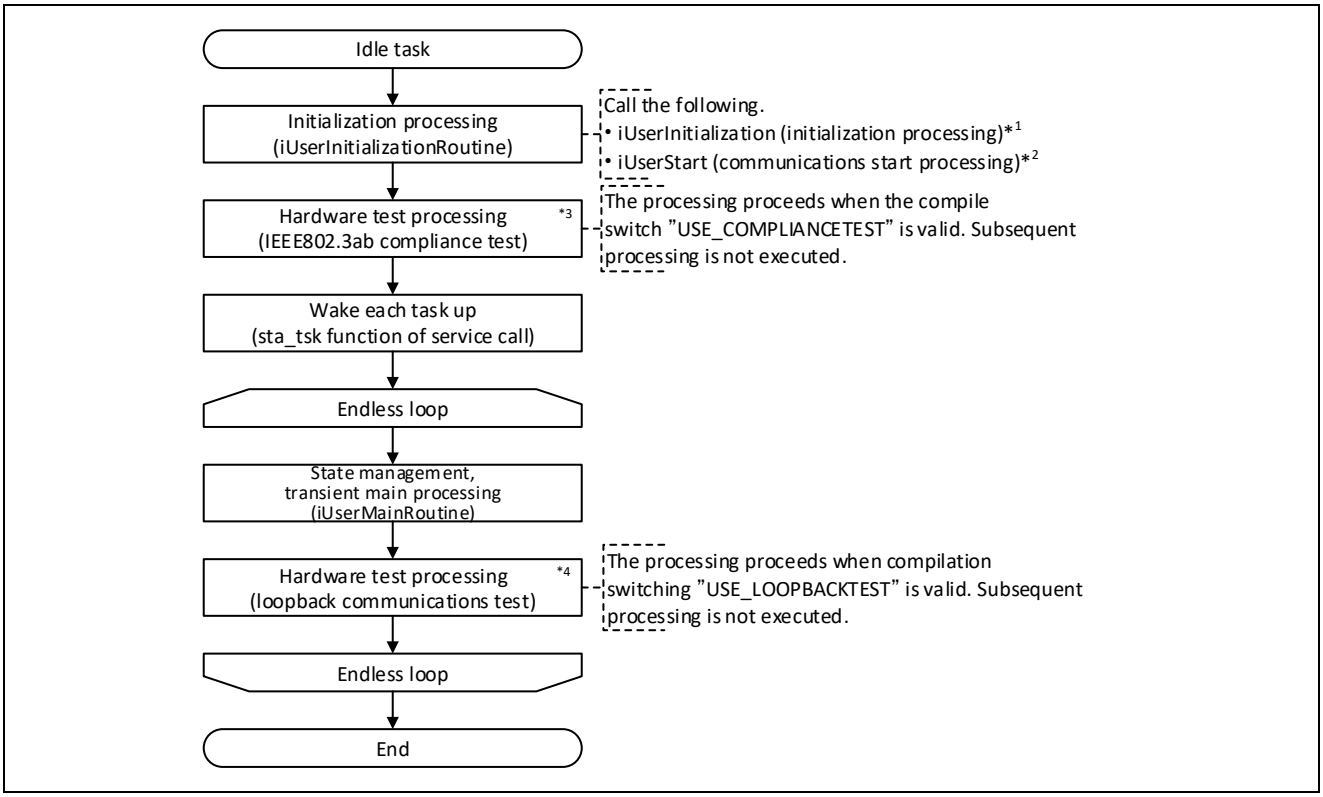


Figure 5.2 Idle Task Processing Overview

Note 1. Refer to "iUserInitialization" (5.3.1, Initialization Processing).

Note 2. Refer to "iUserStart" (5.3.2, Communications Start Processing).

Note 3. Refer to "UserIEEETest "(5.11.1, Hardware Test Processing (IEEE802.3ab Compliance Test)).

Note 4. Refer to "UserLoopBackTest "(5.11.2, Hardware Test Processing (Loopback Communications Test)).

(2) Main Processing of the Task

The following is the general flow of state management and transient main processing of the idle task.

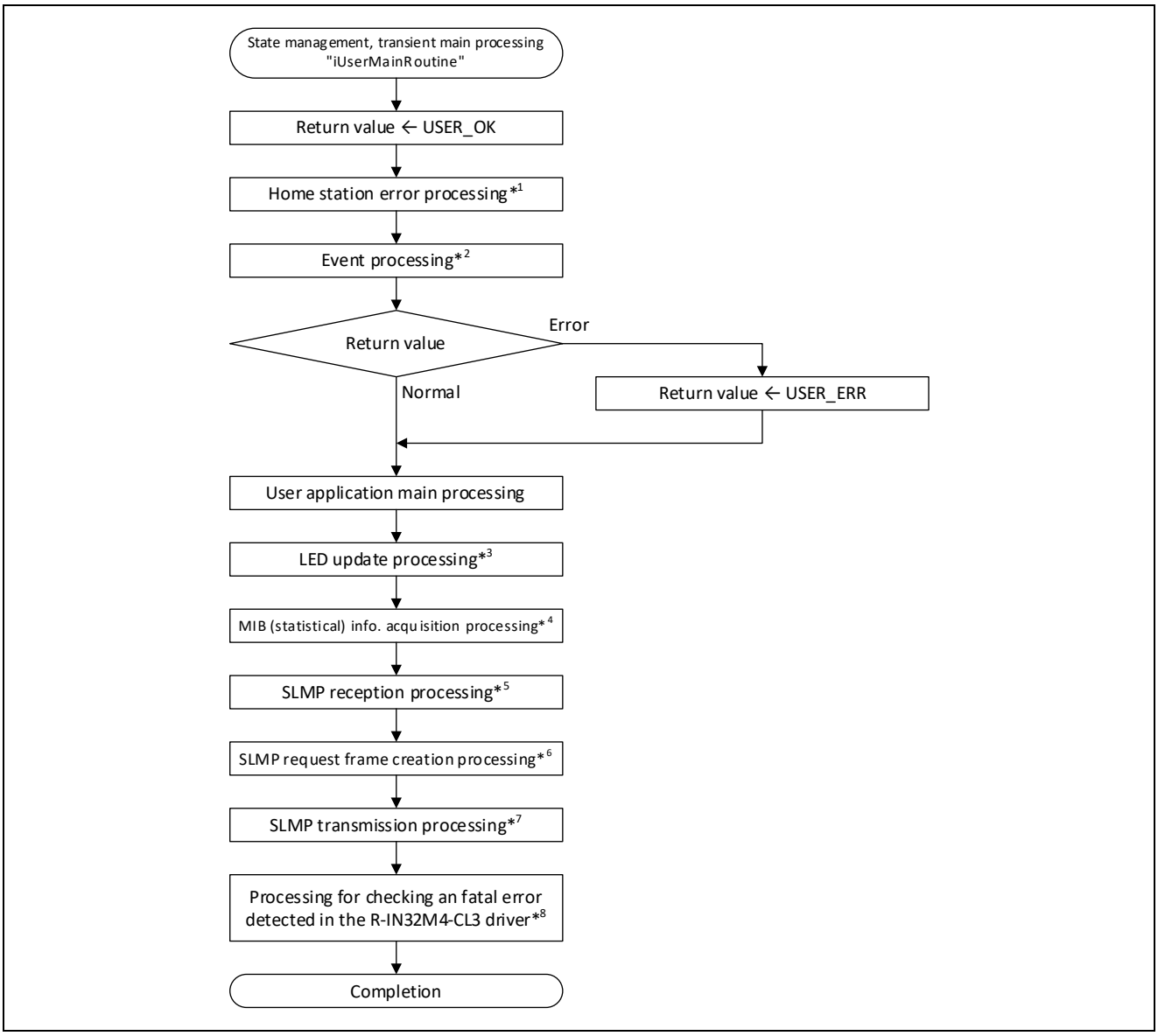


Figure 5.3 Flowchart for State Management and Transient Main Processing

Note 1. Refer to "UserForceStop" (5.5.1, Home Station Error Processing).

Note 2. Refer to "iUserExecuteMain" (5.5.2, Event Processing).

Note 3. Refer to "UserUpdateLed" (5.5.3, LED Update Processing).

Note 4. Refer to "UserGetMIB" (5.5.4, MIB (Statistical) Information Acquisition Processing).

Note 5. Refer to "UserSlmpReceive" (5.5.5, SLMP Reception Processing).

Note 6. Refer to "UserSlmpMakeRequest" (5.5.7, SLMP Request Frame Creation Processing).

Note 7. Refer to "UserSendSlmp" (5.5.6, SLMP Transmission Processing).

Note 8. Refer to "UserConfirmFatalError" (5.5.9, Processing for Checking a Fatal Error Detected in the R-IN32M4-CL3 Driver).

5.2.3 Periodic Processing Task

This task updates cyclic data on a regular basis. An infinite loop repeatedly executes "waiting for task to wake up" and "cyclic main processing".

(1) Task Processing Overview

This task repeats the service calls “Put task to sleep” and “Cyclic main processing” by an endless loop. When the RTOS detects a timer interrupt, the hardware ISR wakes the task up. The following is the task processing overview.

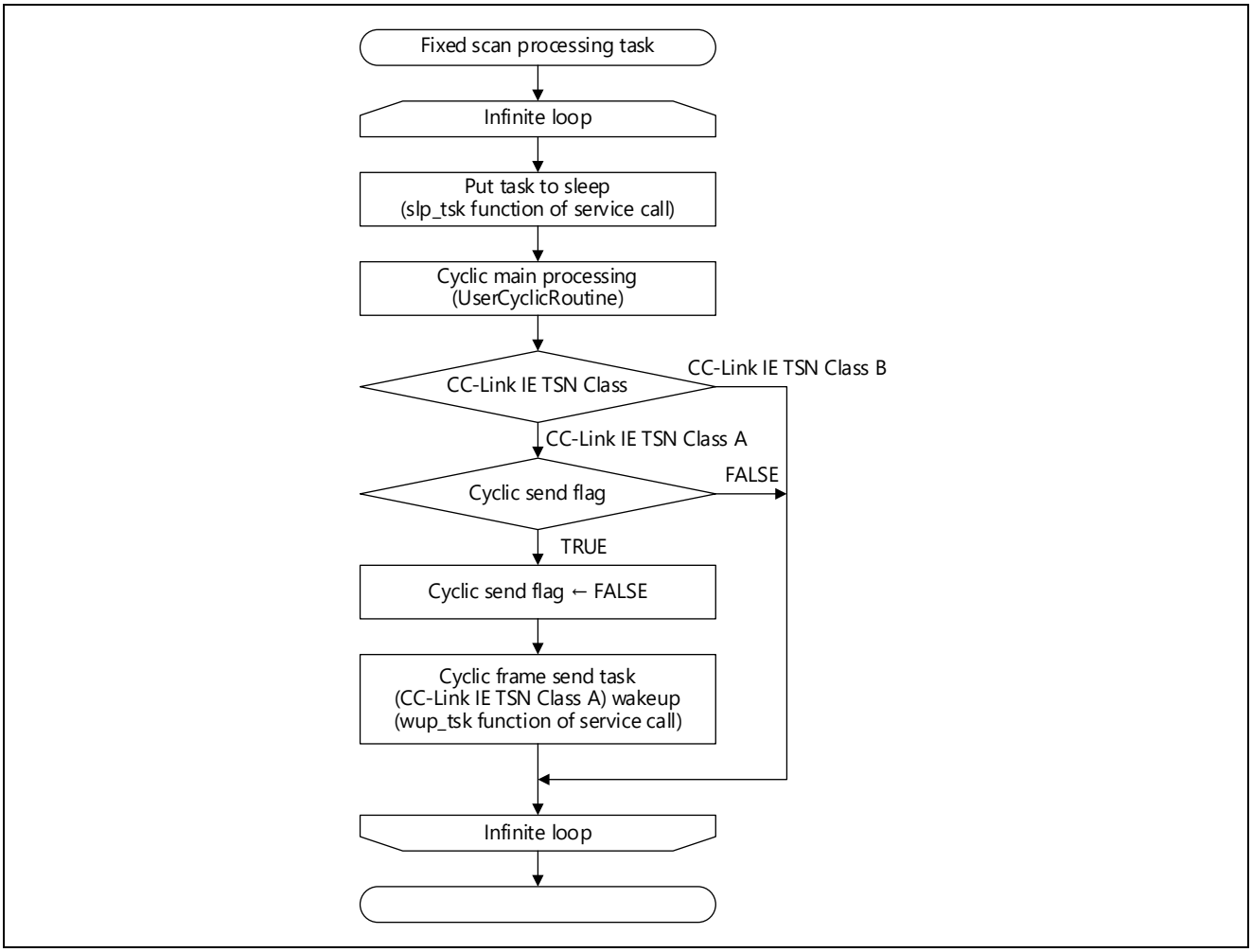


Figure 5.4 Periodic Processing Task Processing Overview

<Point to note>
This task is started up when the master station specifies “Not synchronizing” in the network-synchronized communications setting.

Changing the state of a remote station by enabling or disabling network synchronous communications is not possible after enabling or disabling (synchronizing or not synchronizing) of network synchronous communications has been fixed. Changing the network synchronous communications setting requires switching the power of the target device off and then on or applying a system reset to stop any started tasks.

(2) Main Processing of the Task

The following is the general flow of cyclic main processing.

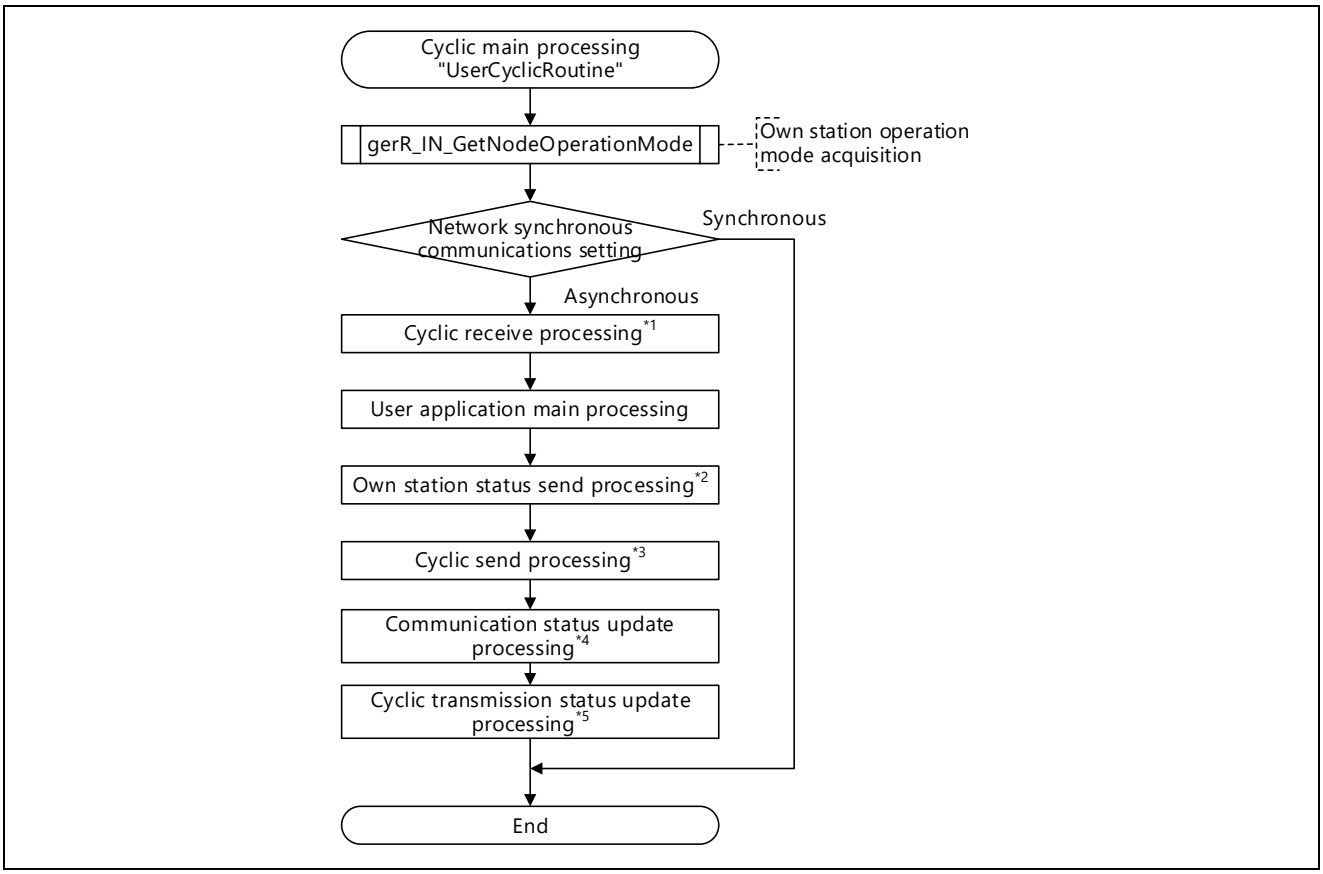


Figure 5.5 Flowchart for Cyclic Main Processing

Note 1. Refer to "UserReceiveCyclic" (5.4.1, Cyclic Reception Processing).

Note 2. Refer to "UserSendNodeStatus" (5.4.2, Home Station State Transmission Processing).

Note 3. Refer to "UserSendCyclic" (5.4.3, Cyclic Transmission Processing).

Note 4. Refer to "UserUpdateStatus" (5.4.4, Communications State Update Processing).

Note 5. Refer to "UserGetCyclicStatus" (5.4.5, Cyclic Transfer State Update Processing).

5.2.4 Error Management Task

This task uses SNMP to send the diagnostic information of your own station to the master station in CC-Link IE TSN diagnosis. It repeats the service calls "Put task to sleep" and "SNMP periodic processing" by an endless loop.

This task handles processing with a fixed period. When the RTOS detects a timer interrupt, the hardware ISR wakes the task up. The following is the task processing overview.
"SNMP periodic processing" is executed by the RIN32M4-CL3 driver.

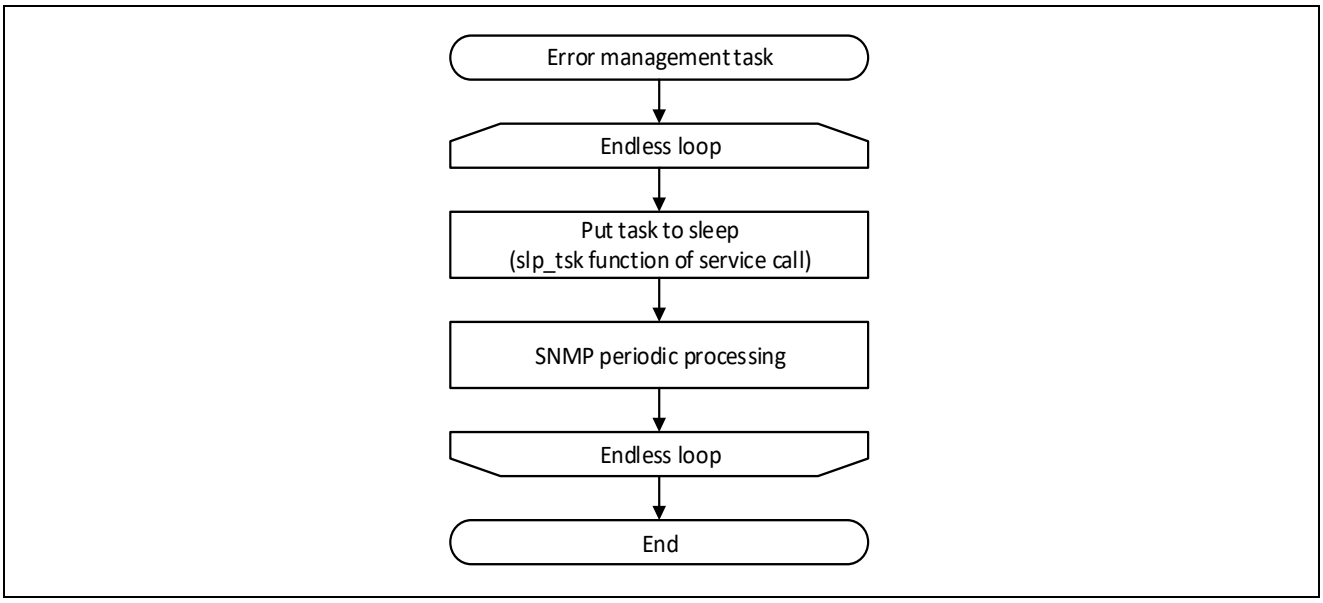


Figure 5.6 Error Management Task Processing Overview

5.2.5 Synchronous Timing Processing Task

(1) Task Processing Overview

This task executes user processing for which the user wants the timing to be aligned with that of another slave station, i.e., communications are to be synchronous.
It repeats the service calls "Put task to sleep" and "Synchronous timing processing" by an endless loop.

This task executes processing when the synchronous output signal is switched from off to on by using the interrupt function of R-IN32M4-CL3.

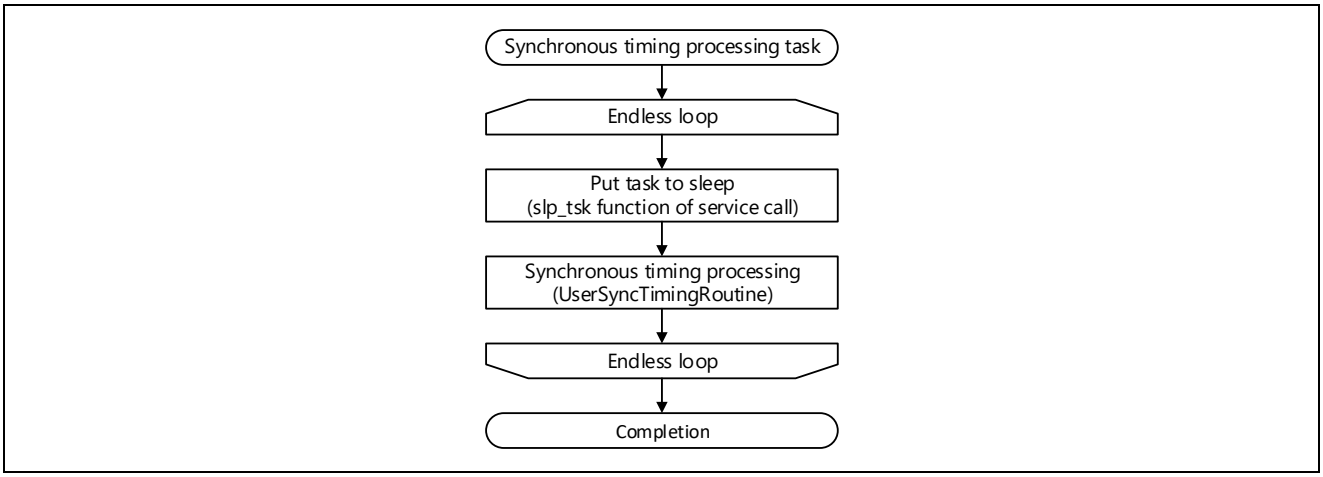


Figure 5.7 Synchronous Timing Processing Task Processing Overview

<Point to note>
This task is started up when the master station specifies "Synchronizing" in the network-synchronized communications setting.

Changing the state of a remote station by enabling or disabling network synchronous communications is not possible after enabling or disabling (synchronizing or not synchronizing) of network synchronous communications has been fixed. Changing the network synchronous communications setting requires switching the power of the target device off and then on or applying a system reset to stop any started tasks.

(2) Main Processing of the Task

The following is a schematic flowchart of synchronous timing processing.

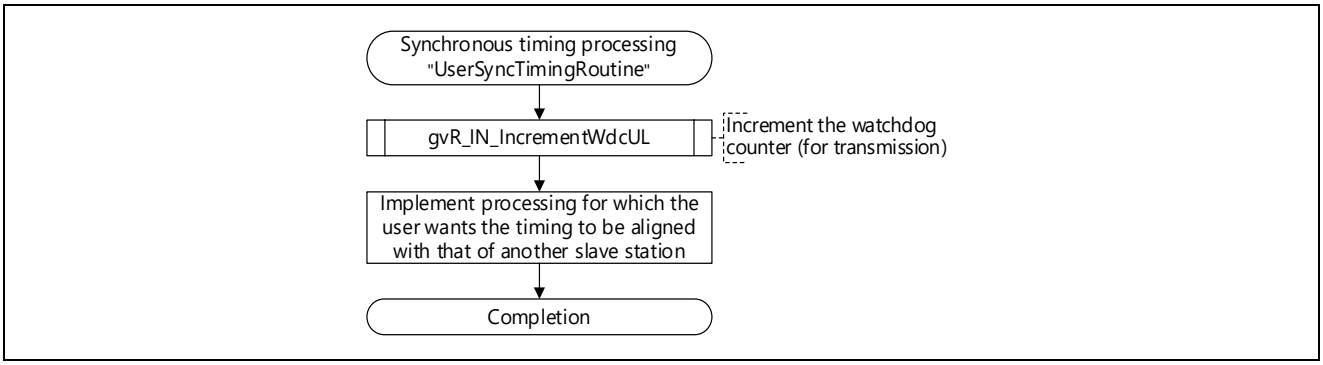


Figure 5.8 Flowchart for Synchronous Timing Processing

5.2.6 Processing Task at the Time of Disconnection during Synchronization

(1) Task Processing Overview

This task executes processing which the user will need to run on the home station when it is disconnected during synchronous communications.
It repeats the service calls "Put task to sleep" and "Processing at the time of disconnection during synchronization" by an endless loop.

This task handles processing with a fixed period. When the RTOS detects a timer interrupt, the hardware ISR wakes the task up. The following is the task processing overview.

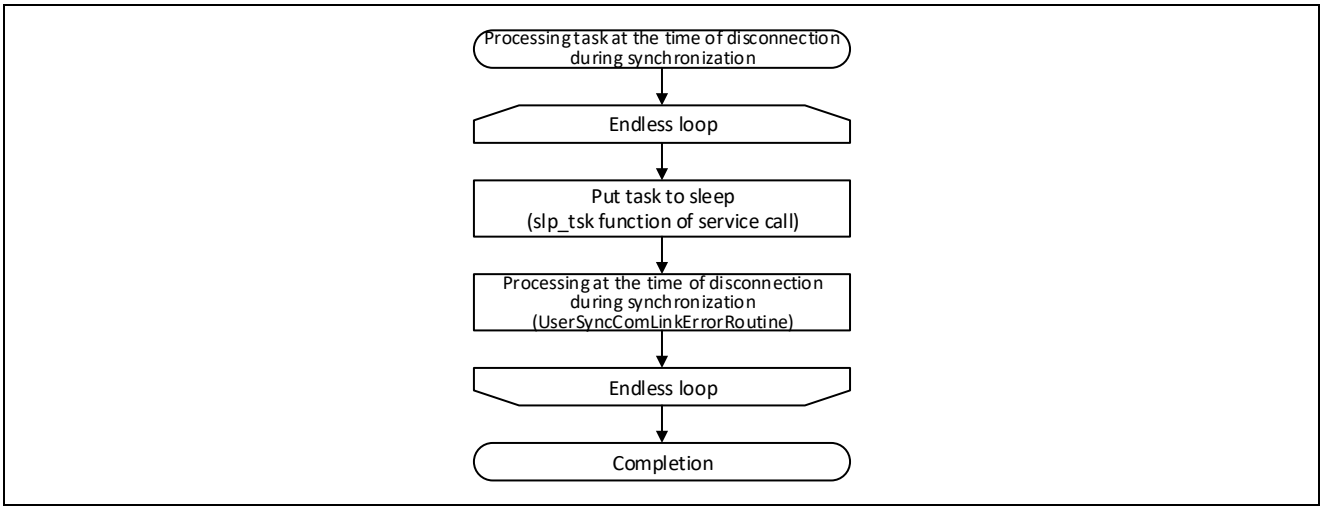


Figure 5.9 Overview of the Processing Task at the Time of Disconnection during Synchronization

<Point to note>
This task is started up when the master station specifies "Not synchronizing" in the network-synchronized communications setting.

Changing the state of a remote station by enabling or disabling network synchronous communications is not possible after enabling or disabling (synchronizing or not synchronizing) of network synchronous communications has been fixed. Changing the network synchronous communications setting requires switching the power of the target device off and then on or applying a system reset to stop any started tasks.

(2) Main Processing of the Task

The following is a schematic flowchart of processing at the time of disconnection during synchronization.

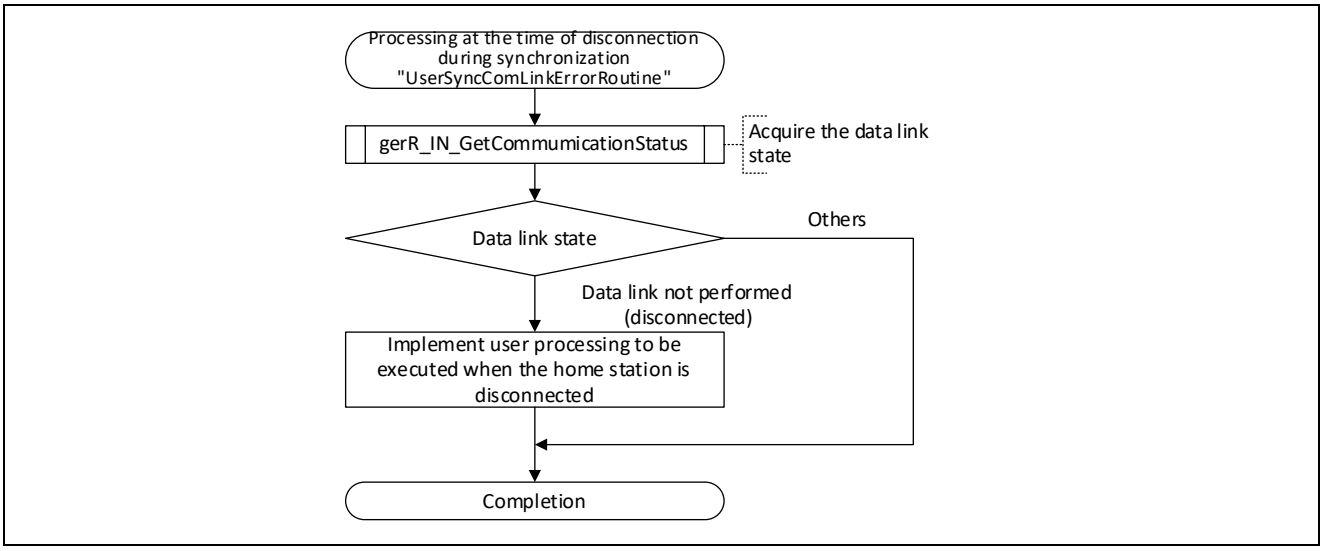


Figure 5.10 Flowchart for Processing at the Time of Disconnection during Synchronization

5.2.7 Scheduler task

(1) Processing overview

This task repeats the service call "Put task to sleep" and the "Scheduler task main processing" by an infinite loop.
The task periodically performs processing. When the RTOS detects a timer interrupt, the hardware ISR wakes up the task. The following is the task processing overview.

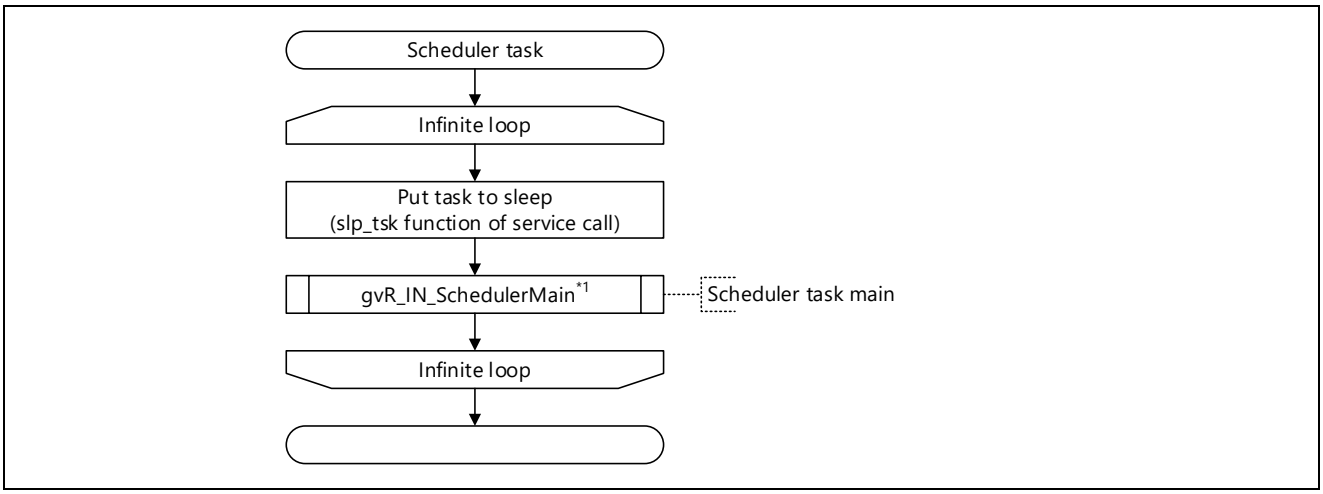


Figure 5.11 Scheduler Task Processing Overview

Note 1. Refer to "6.4.17(1) gvR_IN_SchedulerMain".

<Point to note>
This task starts when the MCU-MCU interface function is enabled (the compiler switch "MCUIF_ENABLE" is enabled).

5.2.8 GPIO communication send task

(1) Processing overview

This task repeats the GPIO communication send task main processing by an infinite loop.

The task performs the processing when it receives a message. If there are no messages, the task is moved to the message waiting state. The following is the task processing overview.

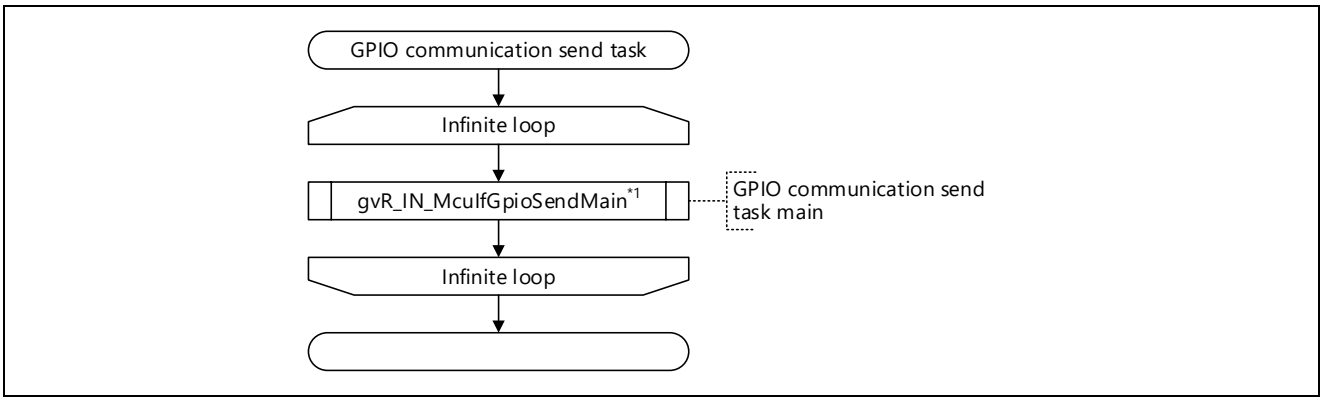


Figure 5.12 GPIO Communication Send Task Processing Overview

Note 1. Refer to "6.4.17(2) gvR_IN_McuIfGpioSendMain".

The following table lists the messages that the GPIO communication send task receives.

Table 5.18 Messages

No.	Name	ID	Send source
1	Event notification send request (internal → external MCU)	MSGP_MID_SEND_EVENTNOTICE_MCUA (1)	GPIO communication receive task
2	Event response receive timeout (internal → external MCU)	MSGP_MID_TIMOUT_RESPNOTICE_MCUA (2)	Scheduler task
3	Next event notification check (internal → external MCU)	MSGP_MID_CHECK_NEXTEVENT_MCUA (3)	GPIO communication receive task

<Point to note>
This task starts when the master station specifies "1b: Supported" in "Safety communications" after the MCU-MCU interface function is enabled (the compiler switch "MCUIF_ENABLE" is enabled).

5.2.9 GPIO communication response receive task

(1) Processing overview

This task repeats the service call "Interrupt waiting state" and the "GPIO communication response receive task main processing" by an infinite loop.

The task performs processing when an interrupt occurs. When the RTOS detects an INTPZ14 interrupt, the hardware ISR wakes up the task. The following is the task processing overview.

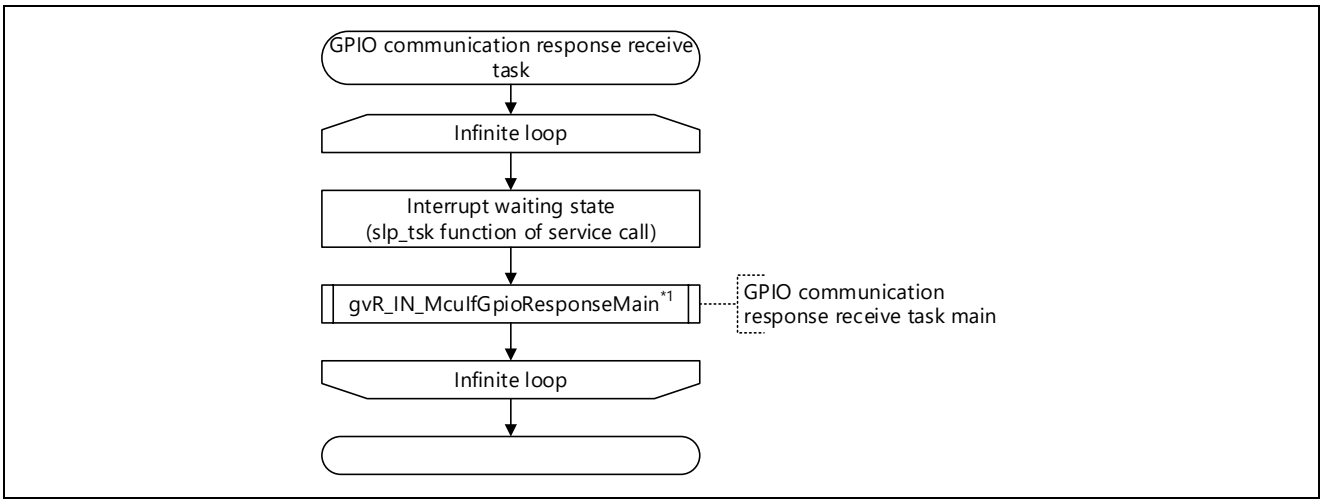


Figure 5.13 GPIO Communication Response Receive Task Processing Overview

Note 1. Refer to "6.4.17(3) gvR_IN_McuIfGpioResponseMain".

<Point to note>
This task starts when the master station specifies "1b: Supported" in "Safety communications" after the MCU-MCU interface function is enabled (the compiler switch "MCUIF_ENABLE" is enabled).

5.2.10 GPIO communication receive task

(1) Processing overview

This task repeats the service call "Interrupt waiting state" and the "GPIO communication receive task main processing" by an infinite loop.

The task performs processing when an interrupt occurs. When the RTOS detects an INTPZ15 interrupt, the hardware ISR wakes up the task. The following is the task processing overview.

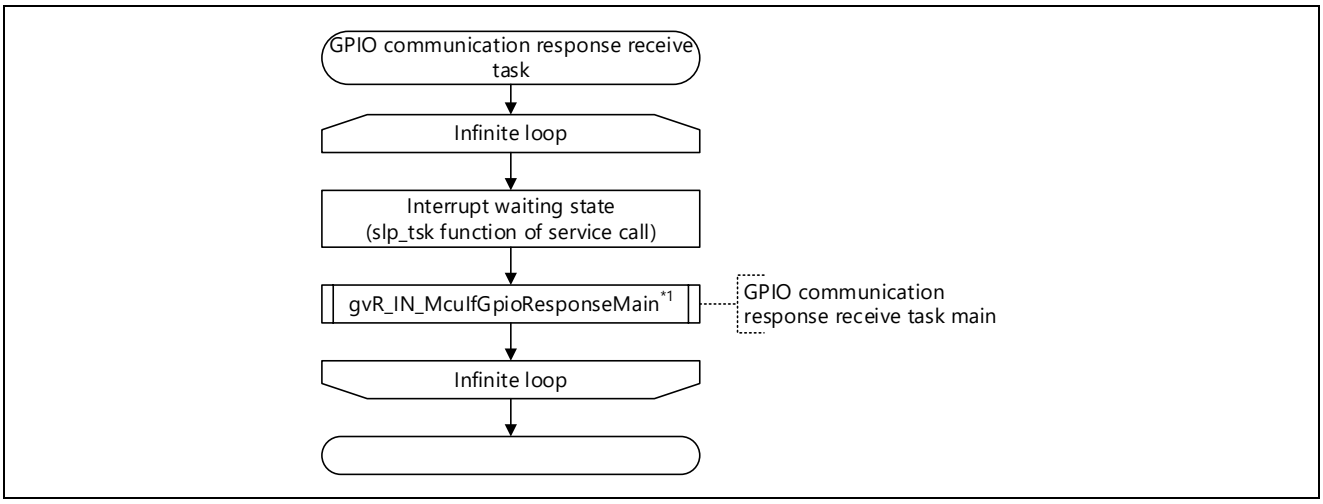


Figure 5.14 GPIO Communication Response Receive Task Processing Overview

Note 1. Refer to "6.4.17(4) gvR_IN_McuIfGpioReceivedMain".

<Point to note>
This task starts when the master station specifies "1b: Supported" in "Safety communications" after the MCU-MCU interface function is enabled (the compiler switch "MCUIF_ENABLE" is enabled).

5.2.11 RT DMA transfer completion task

(1) Processing overview

This task repeats the service call "Interrupt waiting state" and the "RT DMA transfer completion task main processing" by an infinite loop.

The task performs processing when an interrupt occurs. When the RTOS detects a real-time port DMAC transfer completion interrupt, the hardware ISR wakes up the task. The following is the task processing overview.

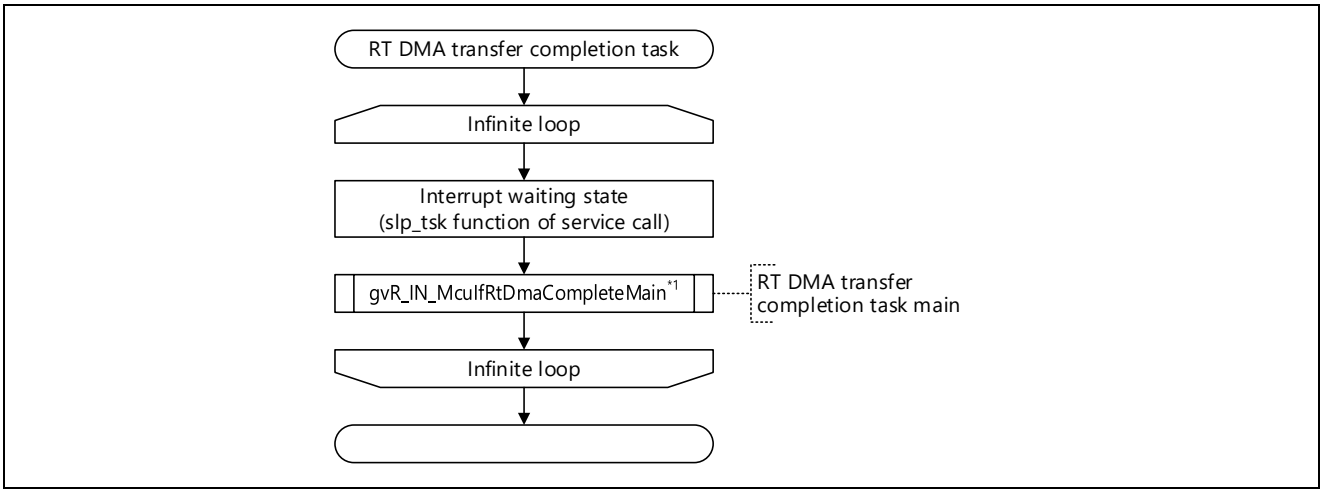


Figure 5.15 RT DMA Transfer Completion Task Processing Overview

Note 1. Refer to "6.4.17(5) gvR_IN_McuIfRtDmaCompleteMain".

<Point to note>
This task starts when the master station specifies "1b: Supported" in "Safety communications" after the MCU-MCU interface function is enabled (the compiler switch "MCUIF_ENABLE" is enabled).

5.2.12 Cyclic frame send task (CC-Link IE TSN Class A)

(1) Processing overview

This task sends cyclic frames when the own station operates as CC-Link IE TSN Class A station.
The task repeats the service call "Put task to sleep" and the "Cyclic frame send processing (CC-Link IE TSN Class A)" by an infinite loop.
The task is started by the idle task and moved from the standby state to the wake up state by the fixed scan processing task to perform processing.
The following is the task processing overview.

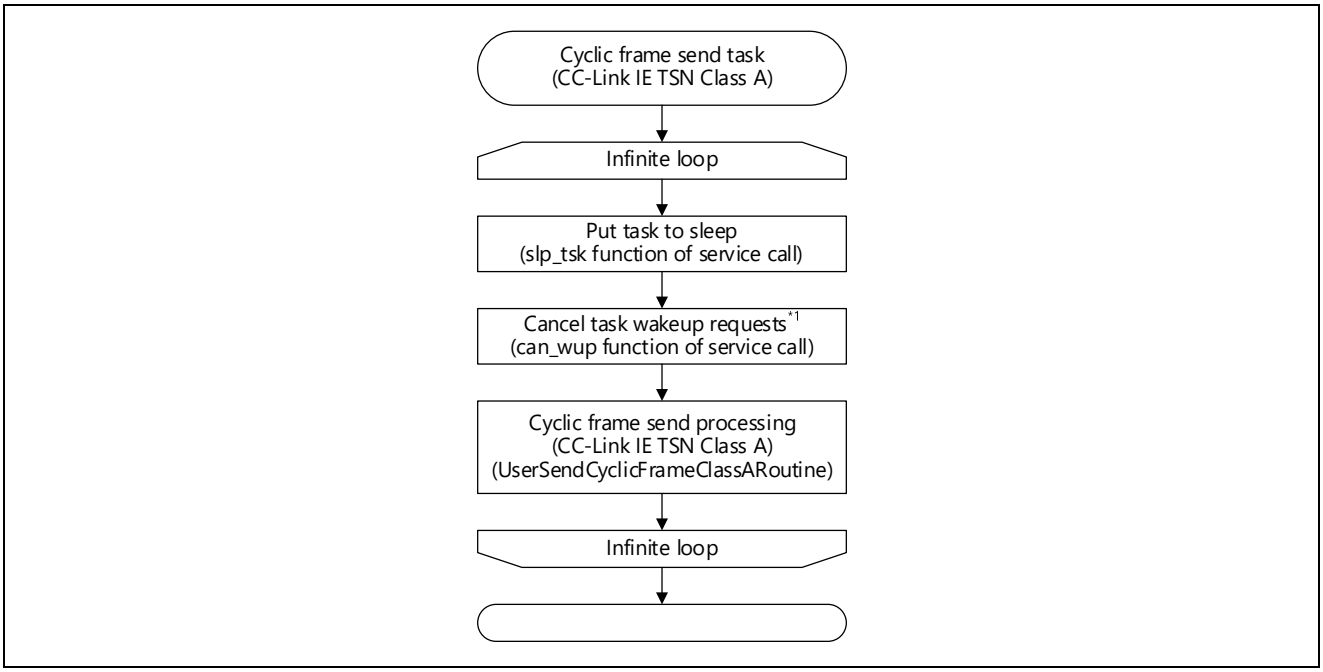


Figure 5.16 Cyclic Frame Send Task (CC-Link IE TSN Class A) Processing Overview

Note 1. The fixed scan processing task and this task operate one-to-one. However, in circumstances such as when the load of the fixed scan processing task is high, cases where this task operates only once while the fixed scan processing task operates several times can be expected.

In such cases, to prevent unnecessary wake-up of this task, unneeded wake-up requests are canceled with the can_wup function.

(2) Main processing

The following is the general flow of the cyclic frame send processing (CC-Link IE TSN Class A).

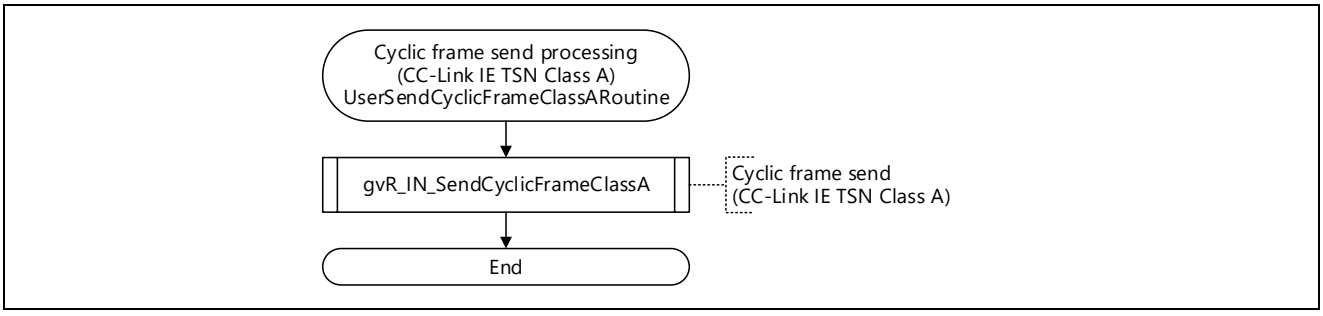


Figure 5.17 Flowchart for Cyclic Frame Send Processing (CC-Link IE TSN Class A)

5.3 User Program Details (Initialization Related)

5.3.1 Initialization Processing

This function initializes R-IN32M4-CL3 and sets an IP address.

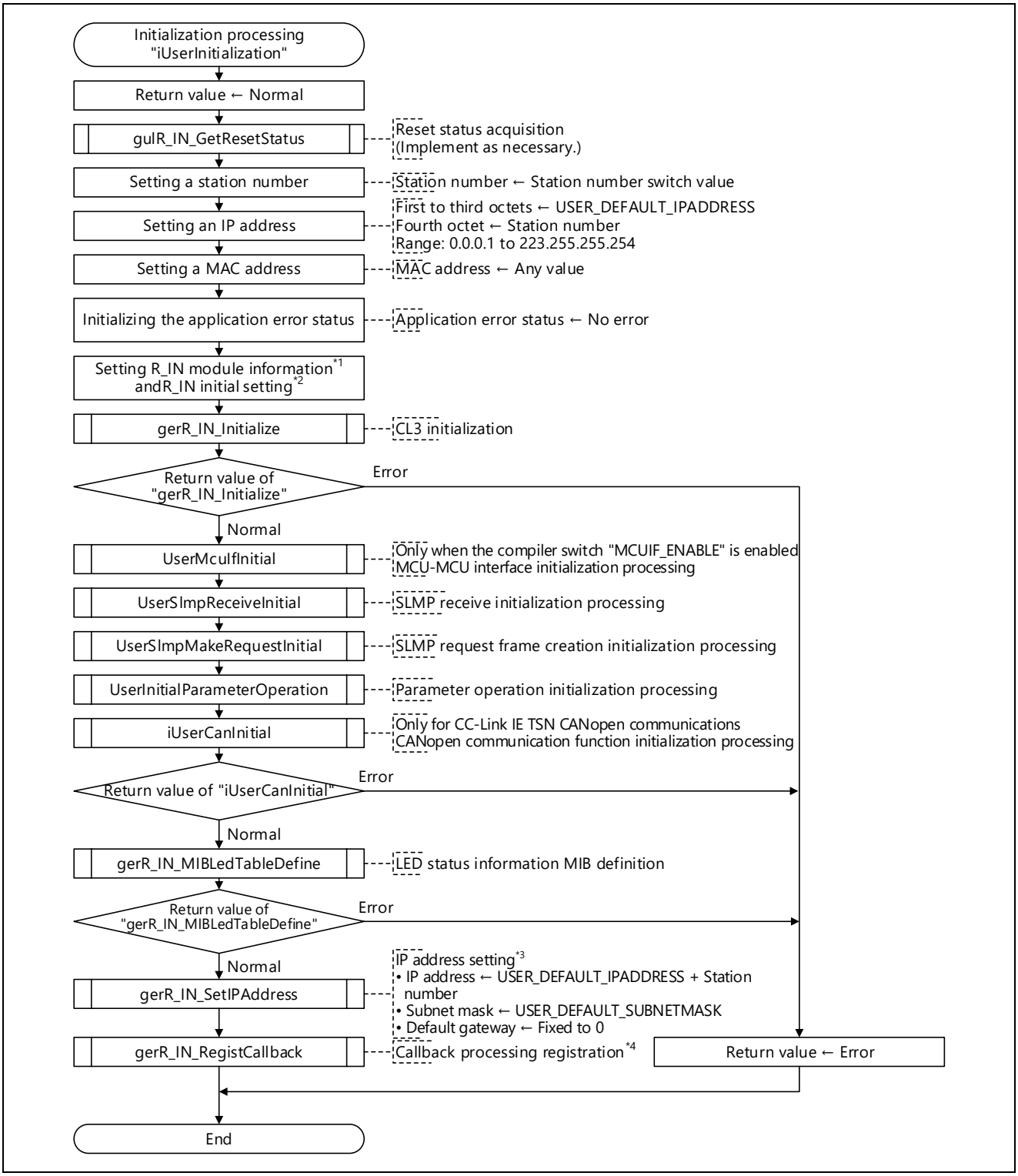


Figure 5.18 Flowchart for Initialization Processing

Note 1. Refer to Table 6.9, R_IN_UNITINFO_T List.

Note 2. Refer to Table 6.17, R_IN_UNITINIT_T List.

Note 3. This sets the initial value related to the IP address in the R-IN32M4-CL3 driver. The first to third octets of the IP address and the subnet mask are changed to the values specified by the master station by the R-IN32M4-CL3 driver at the start of CC-Link IE TSN communications. For the timing, refer to Figure 5.19 Writing image of IP address 1st to 3rd octets.

Note 4. For processing to be registered, refer to Table 6.19, Callback Processing Specifications.

<Supplementary information for the first to third octets of the IP address>

When iUserInitialization is executed, the initial value "USER_DEFAULT_IPADDRESS" is set to the first to third octets of the IP address by the gerR_IN_SetIPAddress function (Section 6.4.1(3)).

When CC-Link IE TSN communications are started with iUserStart (Section 5.3.2 "Communications Start Processing") after iUserInitialization is completed, the value specified by the master station (the value calculated based on the Detection frame) is set to the first to third octets of the IP address by the R-IN32M4-CL3 driver.

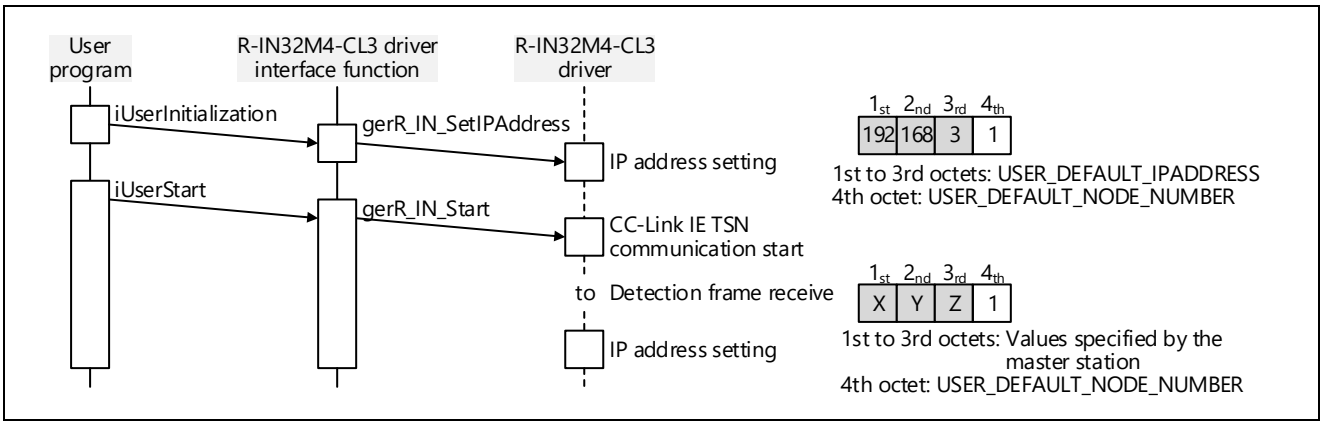


Figure 5.19 Writing image of IP address 1st to 3rd octets

5.3.2 Communications Start Processing

This function instructs R-IN32M4-CL3 to start communications. Execute this processing only once after the initialization processing.

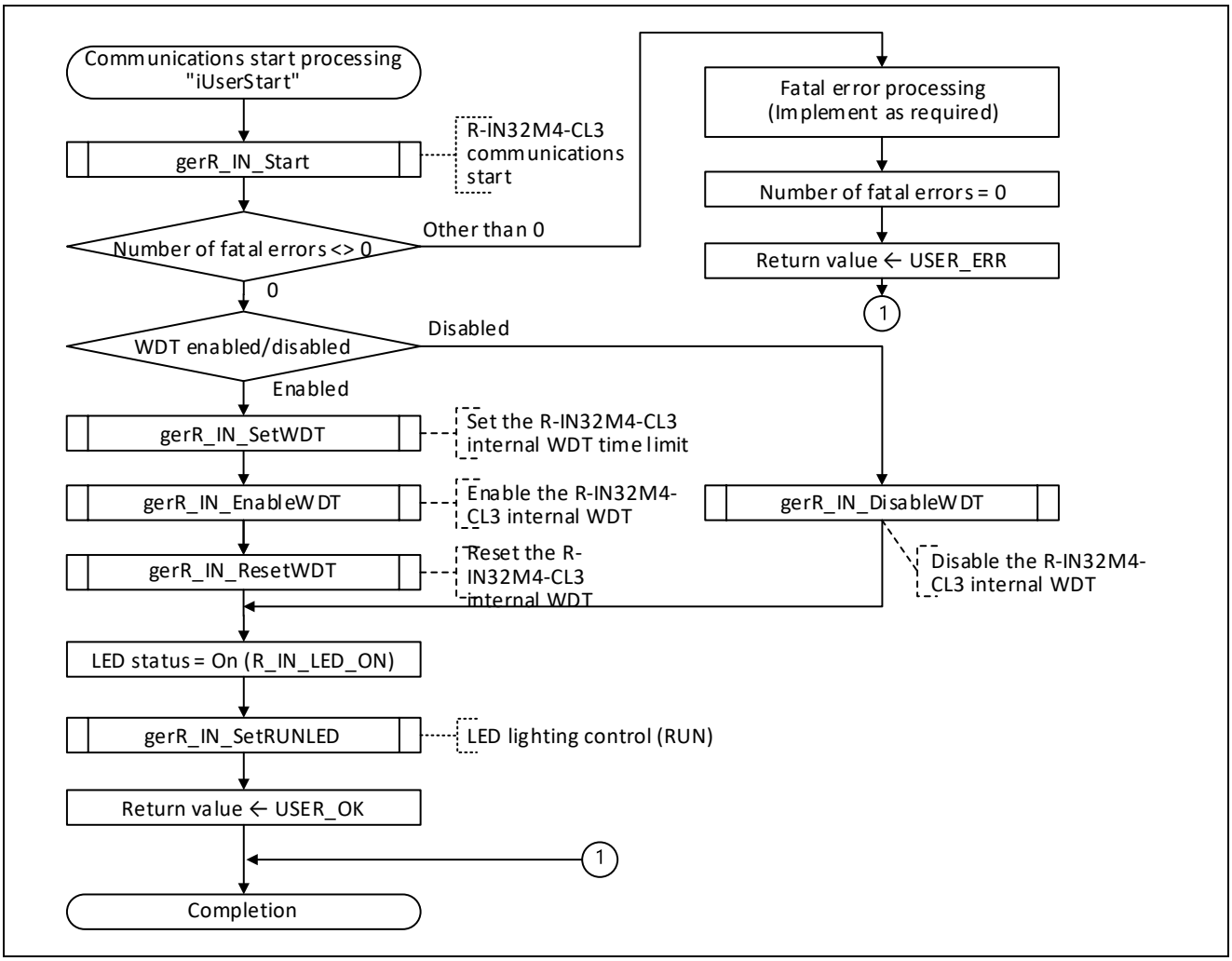


Figure 5.20 Flowchart for Communications Start Processing

<Points to note when creating the socket program by using the API functions of USNetPlus>

■ When performing general-purpose Ethernet communications without connecting the master station

Since the initialization of USNetPlus is executed by "gerR_IN_Start" (6.4.1(4)), implement it so that the Socket program is executed after executing gerR_IN_Start.

■ When performing general-purpose Ethernet communications by connecting the master station

Since the initialization of USNetPlus is executed in gerR_IN_Start, after executing gerR_IN_Start, make sure that it is in the data link and implement it so that the Socket program is executed.

To check the data link, execute "gerR_IN_GetCommunicationStatus" (6.4.6(4)) to get the data link status.

Implement to execute the Socket program when the data link status of the argument is other than the data link not executed (during untied).

5.3.3 SLMP Reception Initialization Processing

The pointers to the SLMP request command execution function table "R_IN_SLMP_FUNCTION_REQUEST_TBL_T" and the SLMP response command execution function table "R_IN_SLMP_FUNCTION_RESPONSE_TBL_T" are conveyed to the R-IN32M4-CL3 driver.

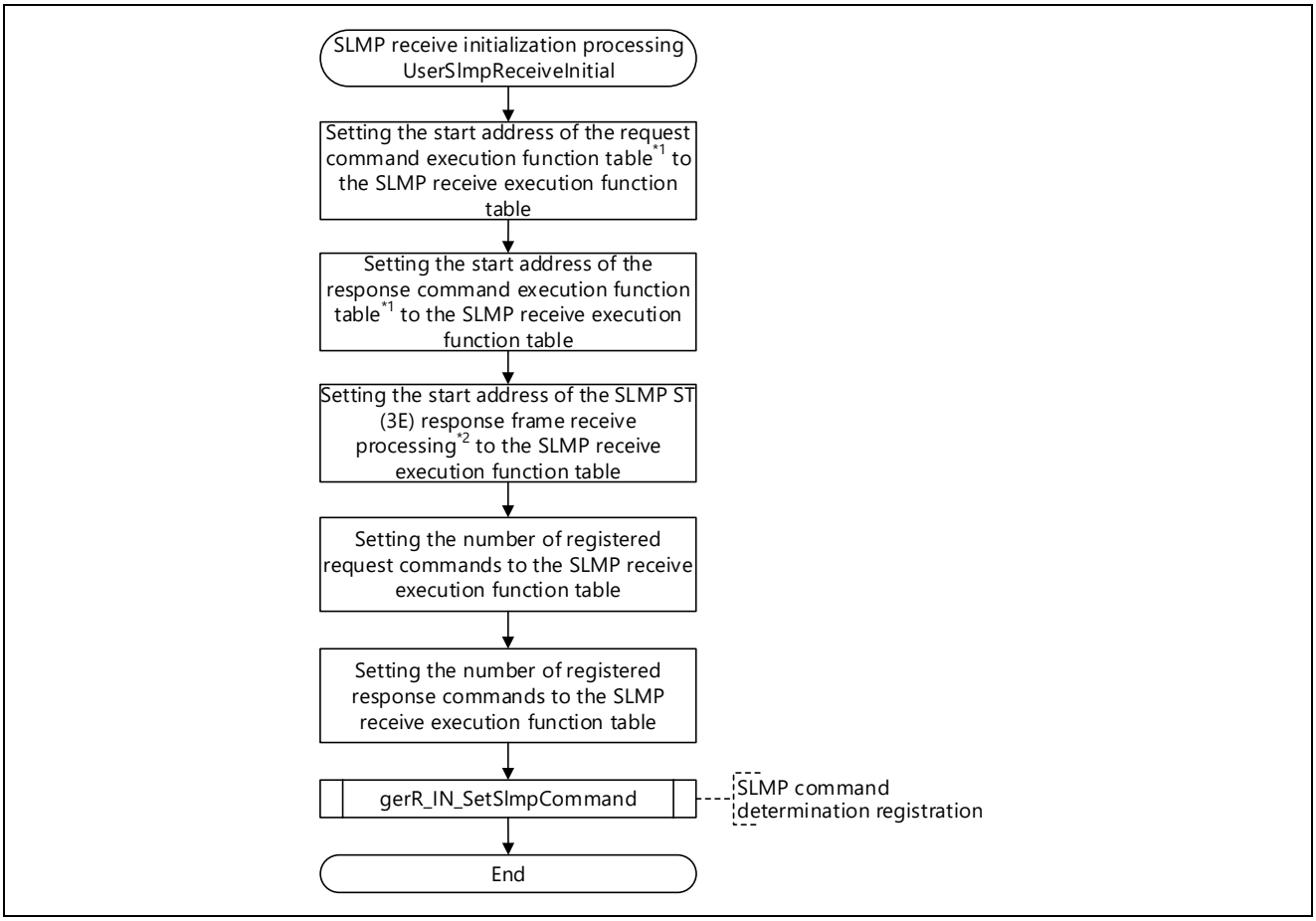


Figure 5.21 Flowchart for SLMP Reception Initialization Processing

Note 1. For details on the execution function table, refer to "Table 6.30" and "Table 6.33".

Note 2. For details, refer to section 5.5.8, SLMP ST (3E) Response Frame Reception Processing.

5.3.4 SLMP Request Frame Creation Initialization Processing

Initialize the global variables used in "UserSlmpMakeRequest" (Section 5.5.7 SLMP Request Frame Creation Processing).

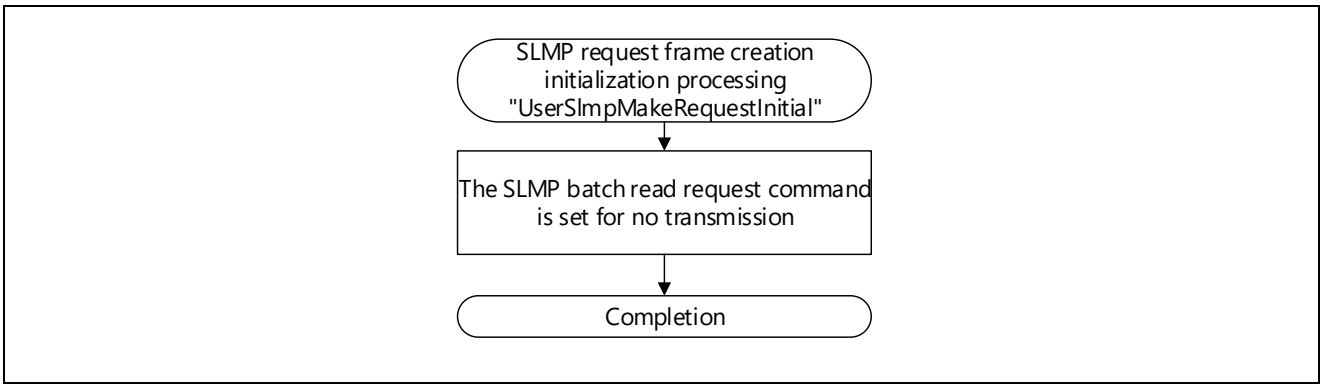


Figure 5.22 Flowchart for SLMP Request Frame Creation Initialization Processing

5.3.5 Parameter Operation Initialization Processing

This function is for initializing global variables for use in automatic setting (parameter operations) of slave station parameters.

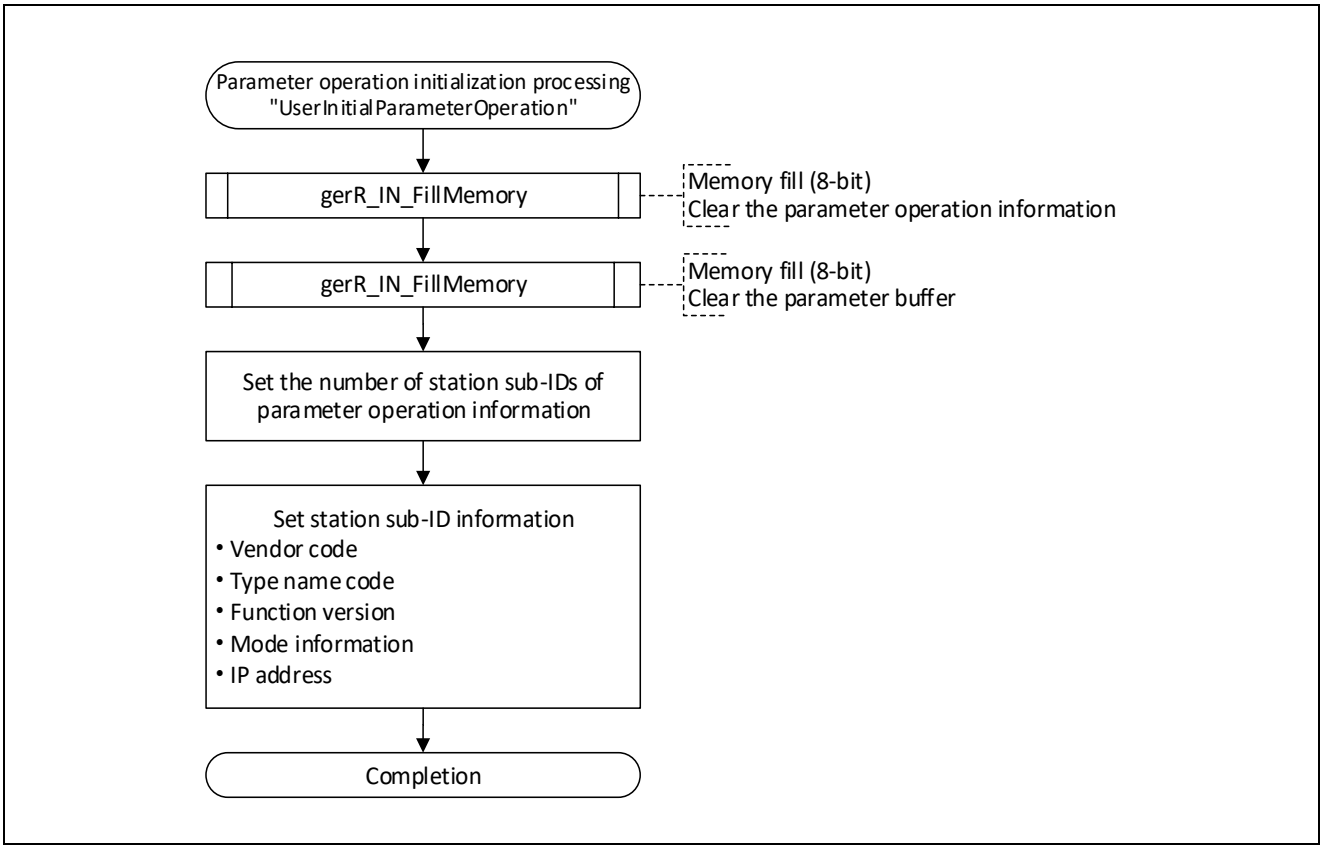


Figure 5.23 Flowchart for Parameter Operation Initialization Processing

5.4 User Program Details (Cyclic Transfer Related)

5.4.1 Cyclic Reception Processing

This function acquires cyclic data (RY and RWw) from the received cyclic frame.
The function also acquires RSPDU when the safety PDU send/receive is performed (the compiler switch "SAFETY_PDU_ENABLE" is enabled).
Three types of processing are described here. Use one type of the processing at a time, and comment out other two types.
For CANopen communications, use the function described in section 5.9.2, Cyclic Reception Processing (Updating RPDOs). The following three processing cannot be used.

(1) Cyclic Reception Processing (High-Speed)
[Outline] Data are read by directly specifying the address ranges of RY, RWw, and RSPDU without calling the R-IN32M4-CL3 driver interface function.
[Feature] Data can be processed at high speeds even if the amount of received data is from 1420 to 2400 bytes. Use this processing if you want to give priority to performance.

(2) Cyclic Reception Processing (Batch)
[Outline] The whole RY, RWw, and RSPDU areas are read in every period by calling the R-IN32M4-CL3 driver interface function.
[Feature] The processing is included for compatibility with the "UserReceiveCyclic" function of the R-IN32M4-CL2 sample code. Use this processing if you have developed products which support CC-Link IE Field connection in the past.

(3) Cyclic Reception Processing (for Individual Sections of Data)
[Outline] Only parts of the RY, RWw, and RSPDU areas are read in every period by calling the R-IN32M4-CL3 driver interface function.
[Feature] Use this processing when the RY, RWw, and RSPDU areas are divided into an area from which data are frequently read and one from which data are less frequently read.

Point

The maximum size of data that can be sent and received using the cyclic transfer function varies depending on the load variations caused by scale of applications.
The following are the approximate maximum data sizes when the periodic processing task is performed at every 200 µs.

User program	CC-Link IE TSN Class B	CC-Link IE TSN Class A	
	1 Gbps/100 Mbps	1 Gbps	100 Mbps
Cyclic send processing (high-speed) and cyclic receive processing (high-speed)	1426	1116	606
Cyclic send processing (all data) and cyclic receive processing (all data)	678	548	368
Cyclic send processing (part of data) and cyclic receive processing (part of data)	490	368	232

To send and receive up to 2400 bytes of data, use cyclic transmission processing (high-speed) and cyclic reception processing (high-speed). At this time, set the processing interval of the periodic processing task to around a 400-µs period.

(1) Cyclic Reception Processing (High-Speed)

This function reads data by directly specifying the destination addresses for storage of RY, RWw, and RSPDU.
Use this processing when up to 2400 bytes of data for transmission/reception are to be used.
It cannot be used together with any other type of cyclic reception processing.

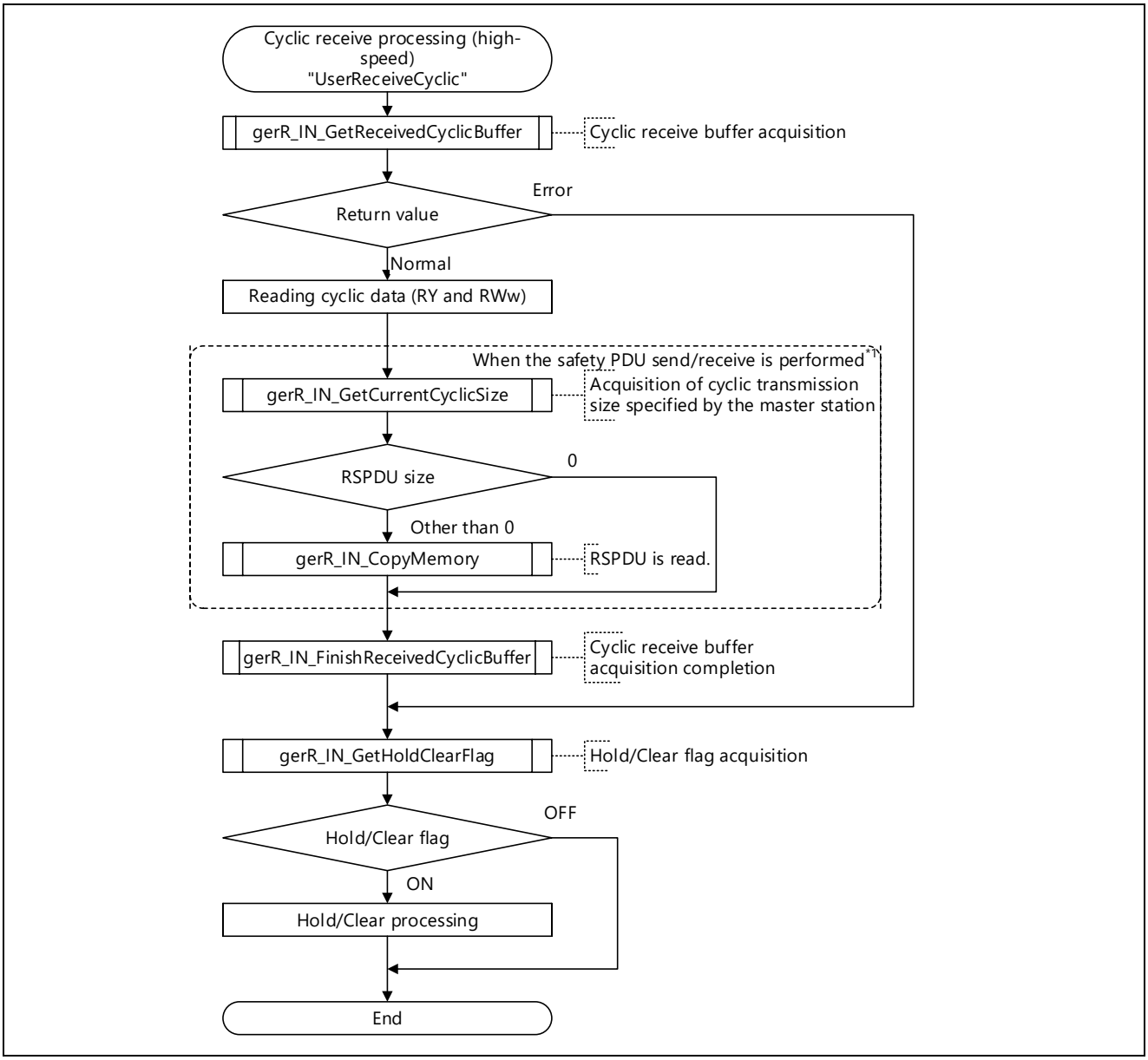


Figure 5.24 Flowchart for Cyclic Reception Processing (High-Speed)

Note 1. The function is executed when the compiler switch "SAFETY_PDU_ENABLE" is enabled.

(2) Cyclic Reception Processing (all data)

This function batch-reads the data in RY, RWw, and RSPDU (from the start address to the end address).
(Use this function to read all data in RY and RWw at every scan.)
The function cannot be used together with any other type of cyclic receive processing.

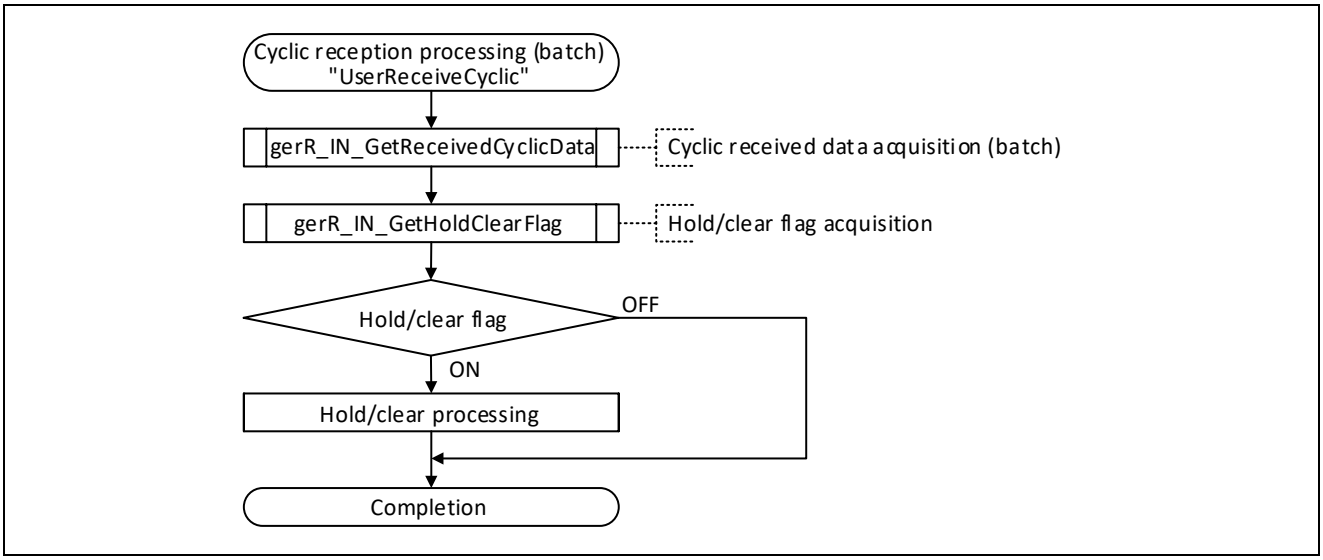


Figure 5.25 Flowchart for Cyclic Reception Processing (All Data)

(3) Cyclic Reception Processing (for Individual Sections of Data)

This function reads the specified data in RY, RWw, and RSPDU (from the start address to the specified offset address).
(Use this function to read only part of data in RY, RWw, and RSPDU at every scan.)
The function cannot be used together with any other type of cyclic receive processing.

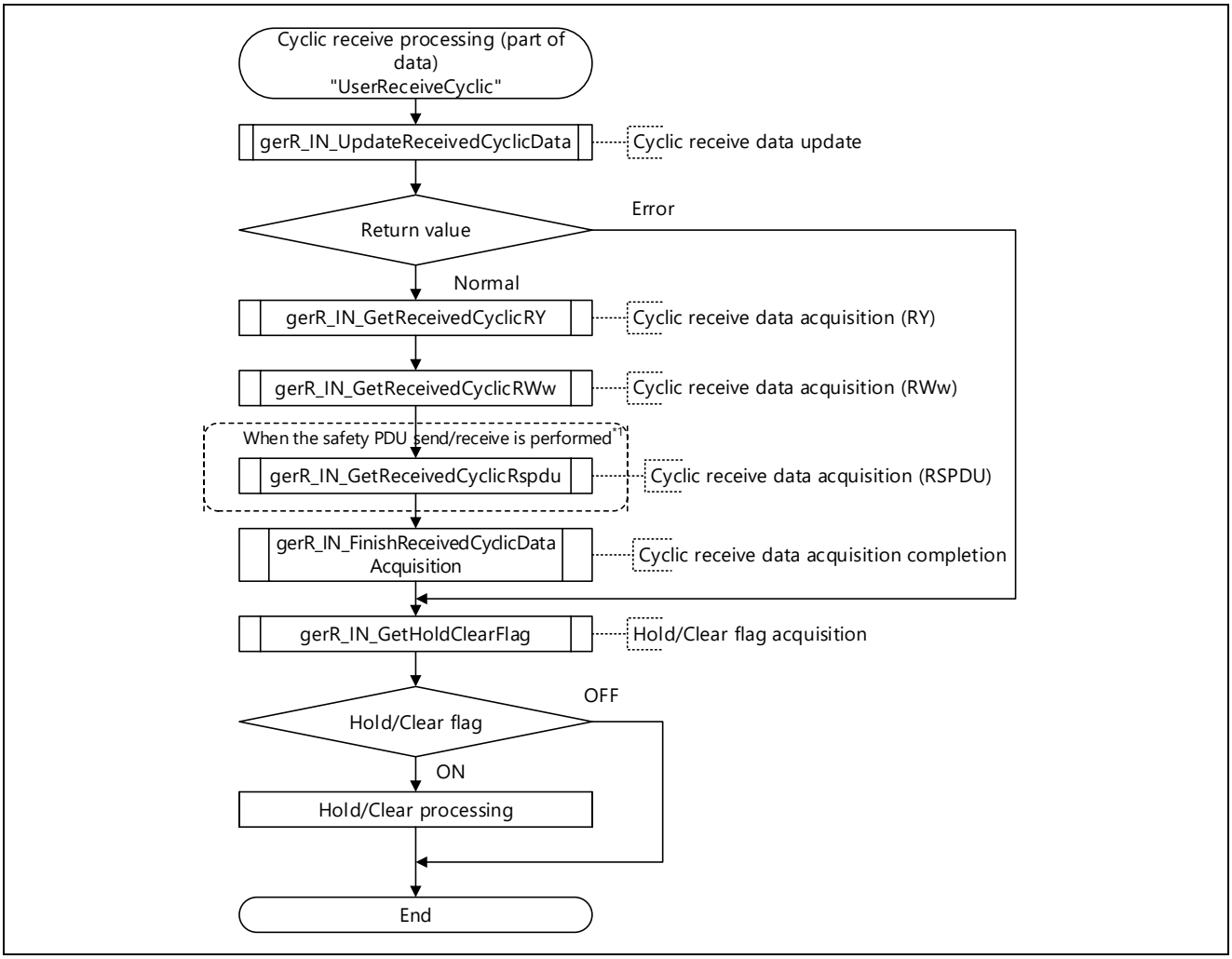


Figure 5.26 Flowchart for Cyclic Reception Processing (for Individual Sections of Data)

Note 1. The function is executed when the compiler switch "SAFETY_PDU_ENABLE" is enabled.

(4) Hold/Clear Processing

Hold/clear processing is a process to continue (hold) or stop (clear) the output of the received cyclic data to the other station, when the home station is disconnected from a data link or when the application of the master station is stopped or abnormal.

As a failsafe, implement hold/clear processing (user optional processing) in consideration of the following:

[About the data being cyclically sent by the master station (RY and RWw)]
When the master station application is stopped or in error, the data being cyclically sent by the master station have been held or cleared depending on the setting of the master station. There is no way for the slave station (home station) to detect in advance the state of the data being cyclically sent by the master station (in terms of whether the data have been held or cleared).
When a master station manufactured by Mitsubishi Electric is in use, hold/clear processing is set in the "Output Setting during CPU STOP" and the "Error Time Output Mode" parameter by using the engineering tool.

5.4.2 Home Station State Transmission Processing

This function sets the application error status*1, which is stored to StsW.
The application error status is stored to StsW, and then notified to the master station during cyclic transmission.

* 1: Only the application error status is set in this processing. Other items are set by the R-IN32M4-CL3 driver.

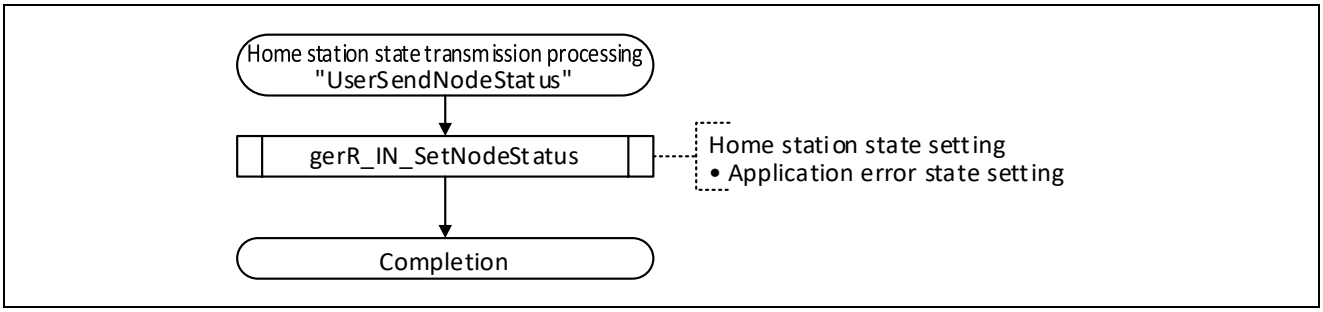


Figure 5.27 Flowchart for Home Station State Transmission Processing

There are the following four types of application error states to be set in the argument. There is no regulation on the degree of error status, so define it arbitrarily. For reference, the error status of the Mitsubishi Electric remote I / O unit is illustrated.

Table 5.19 Example of application error status (for Mitsubishi Electric remote I/O unit)

Application error status	Details
No error (0)	-
Mild abnormality (1)	Error that the unit continues to operate
Moderate abnormality (2)	unit has stopped operating and can be dealt with by the end user's operation.
Severe abnormalities (3)	unit has stopped operating and the end user cannot recover the error.

When RJ71GN11-T2 is used as the master station, the application error status of each station can be monitored with the following link special register (SW). If there is something wrong with the application of each station, the previous cyclic data is retained.

- SW0100 --SW0107 (Severe abnormal occurrence of CPU in each station)
- SW0110 --SW0117 (each station CPU minor abnormality occurrence state)

5.4.3 Cyclic Transmission Processing

This function sets the cyclic data (RX and RWr) in frames for cyclic transmission.
The function also acquires SSPDU when the safety PDU send/receive is performed (the compiler switch "SAFETY_PDU_ENABLE" is enabled).

As an example of processing for setting cyclic data, three types of processing are described here. Use one type of the processing at a time, and comment out other two types.
For CANopen communications, use processing described in section 5.9.3, Cyclic Transmission Processing (Updating TPDOs). The following three types of processing cannot be used.

(1) Cyclic Transmission Processing (High-Speed)
[Outline] Data are written by directly specifying the address ranges of RX, RWr, and SSPDU without calling the R-IN32M4-CL3 driver interface function.
[Feature] Data can be processed at high speeds even if the amount of data for transmission is from 1420 to 2400 bytes. Use this processing if you want to give priority to performance.

(2) Cyclic Transmission Processing (Batch)
[Outline] The whole RX, RWr, and SSPDU areas are written in each period by calling the R-IN32M4-CL3 driver interface function.
[Feature] The processing is included for compatibility with the "UserSendCyclic" function of the R-IN32M4-CL2 sample code. Use this processing if you have developed products which support CC-Link IE Field connection in the past.

(3) Cyclic Transmission Processing (for Individual Sections of Data)
[Outline] Only parts of the RX, RWr, and SSPDU areas are written in every period by calling the R-IN32M4-CL3 driver interface function.
[Feature] Use this processing when the RX, RWr, and SSPDU areas are divided into an area to which data are frequently written and one to which data are less frequently written.

Point

The maximum size of data that can be sent and received using the cyclic transfer function varies depending on the load variations caused by scale of applications.
The following are the approximate maximum data sizes when the periodic processing task is performed at every 200 µs.

User program	CC-Link IE TSN Class B	CC-Link IE TSN Class A	
	1 Gbps/100 Mbps	1 Gbps	100 Mbps
Cyclic send processing (high-speed) and cyclic receive processing (high-speed)	1426	1116	606
Cyclic send processing (all data) and cyclic receive processing (all data)	678	548	368
Cyclic send processing (part of data) and cyclic receive processing (part of data)	490	368	232

To send and receive up to 2400 bytes of data, use cyclic transmission processing (high-speed) and cyclic reception processing (high-speed). At this time, set the processing interval of the periodic processing task to around a 400-µs period.

(1) Cyclic Transmission Processing (High-Speed)

This function writes data by directly specifying the destination addresses for storage of RX, RWr, and SSPDU.
Use this processing when up to 2400 bytes of data for transmission/reception are to be used.
The processing cannot be used together with any other type of cyclic transmit processing.

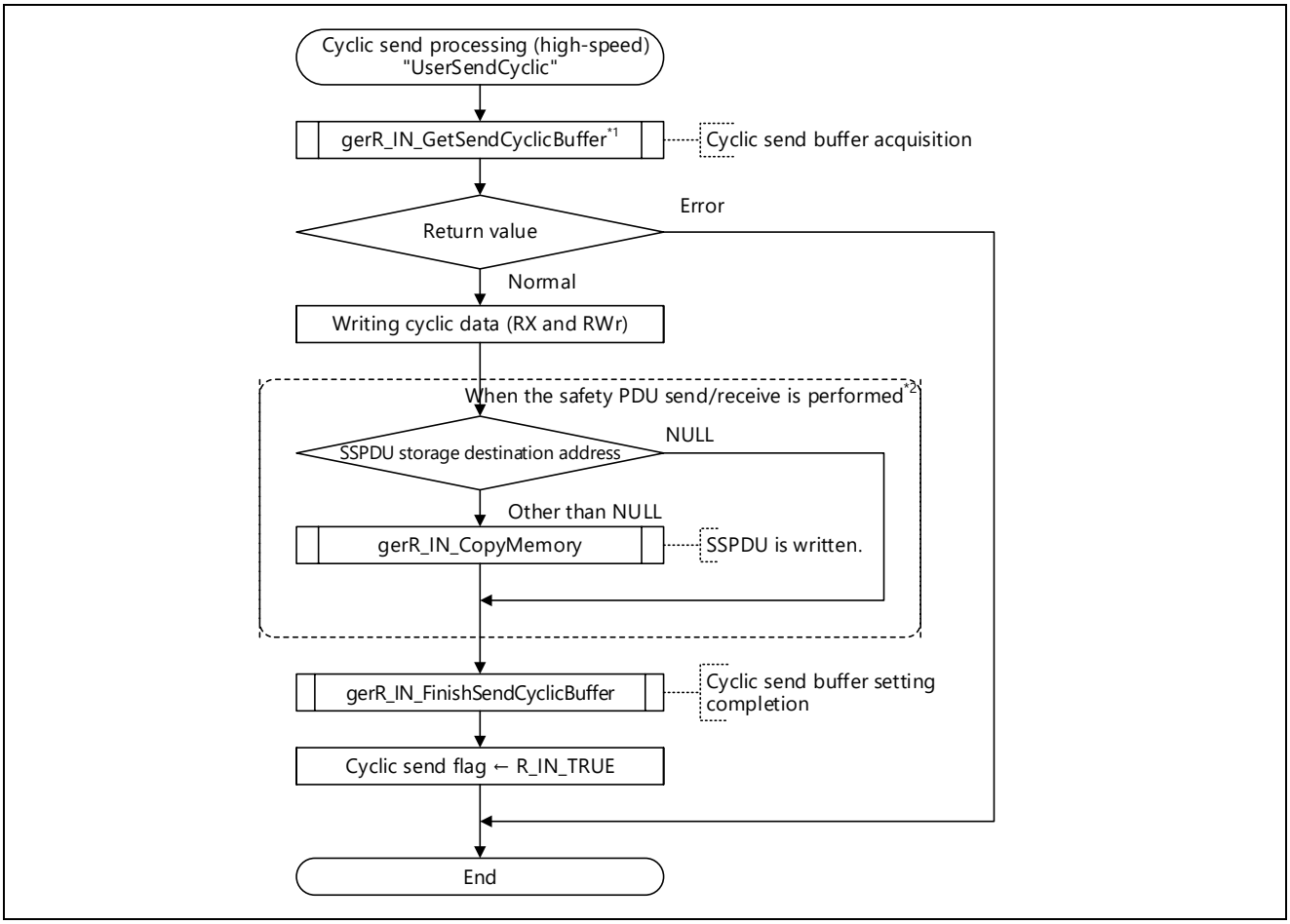


Figure 5.28 Flowchart for Cyclic Transmission Processing (High-Speed)

Note 1. If a frame for transmission would exceed 1518 bytes and thus has to be divided into two frames, use the cyclic split transmission buffer acquisition function described in section 6.4.4(17), gerR_IN_GetSplitSendCyclicBuffer.

Note 2. The function is executed when the compiler switch "SAFETY_PDU_ENABLE" is enabled.

(2) Cyclic Transmission Processing (All Data)

This function batch-writes data to the whole (from start to end) of both the RX, RWr, and SSPDU areas.
Use this processing when data are to be written to the whole RX, RWr, and SSPDU areas in every period.
It cannot be used together with any other type of cyclic transmission processing.

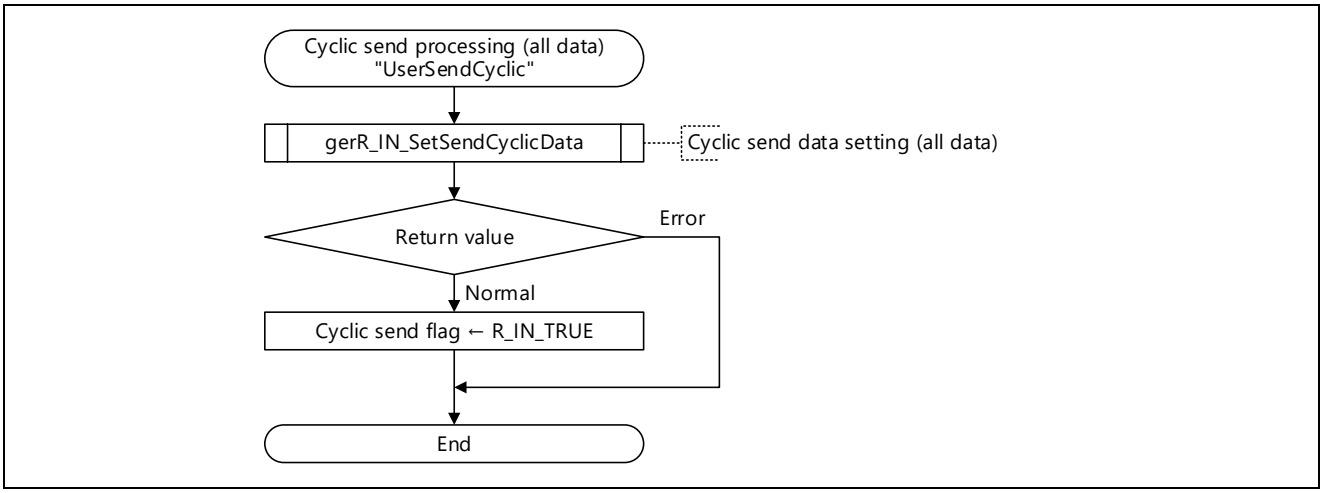


Figure 5.29 Flowchart for Cyclic Transmission Processing (All Data)

(3) Cyclic Transmission Processing (for Individual Sections of Data)

This function individually writes to the specified locations (from the start to the location at the given offset) of the RX, RWr, and SSPDU areas. Use this processing when only parts of the RX and RWr areas are to be written in every period.
The function cannot be used together with any other type of cyclic transmission processing.

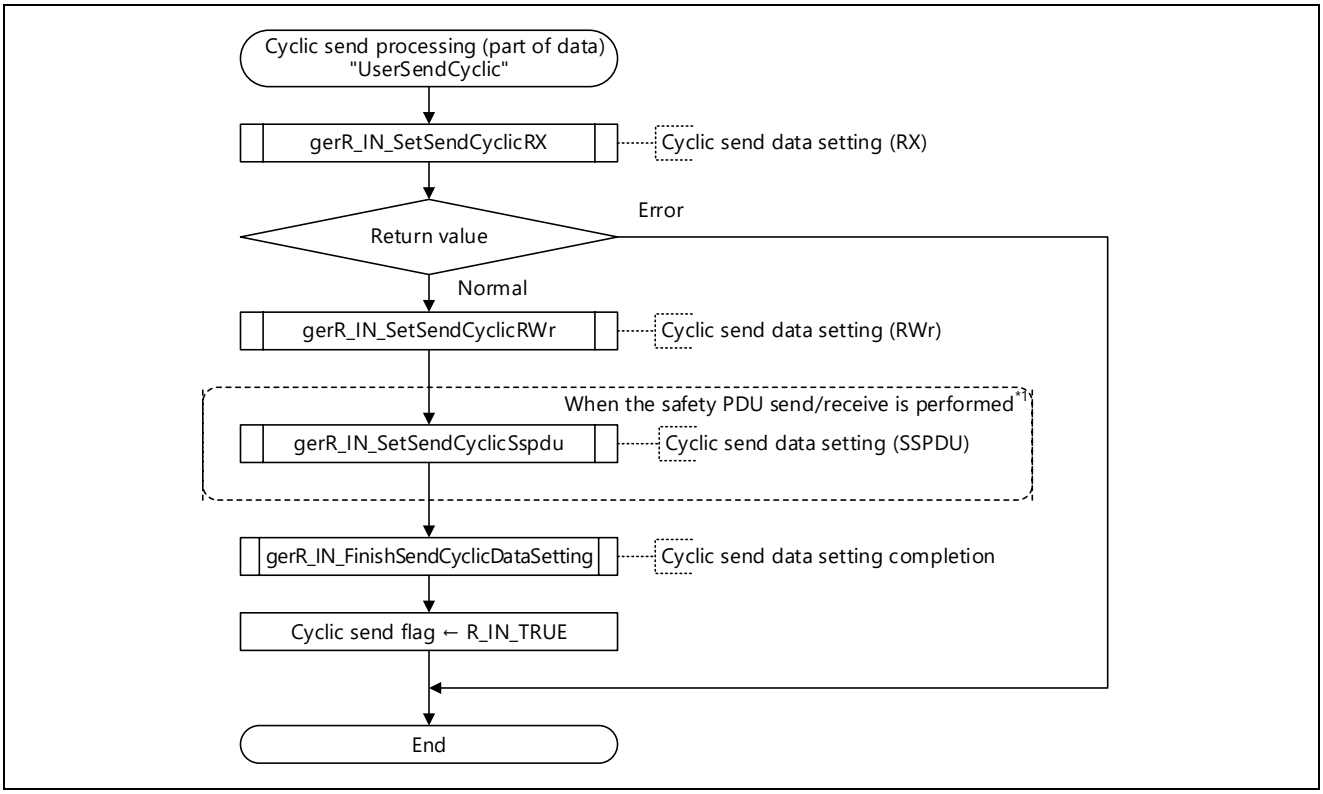


Figure 5.30 Flowchart for Cyclic Transmission Processing (for Individual Sections of Data)

Note 1. The function is executed when the compiler switch "SAFETY_PDU_ENABLE" is enabled.

5.4.4 Communications State Update Processing

This function acquires the data link state of the home station, and sets the ERR LED control flag (gulErrCtrl) in accord with the data link state.
The ERR. LED control flag (gulErrCtrl) is used to control the lighting / extinguishing / blinking of the ERR. LED in " UserUpdateLed" (5.5.3 LED Update Processing).

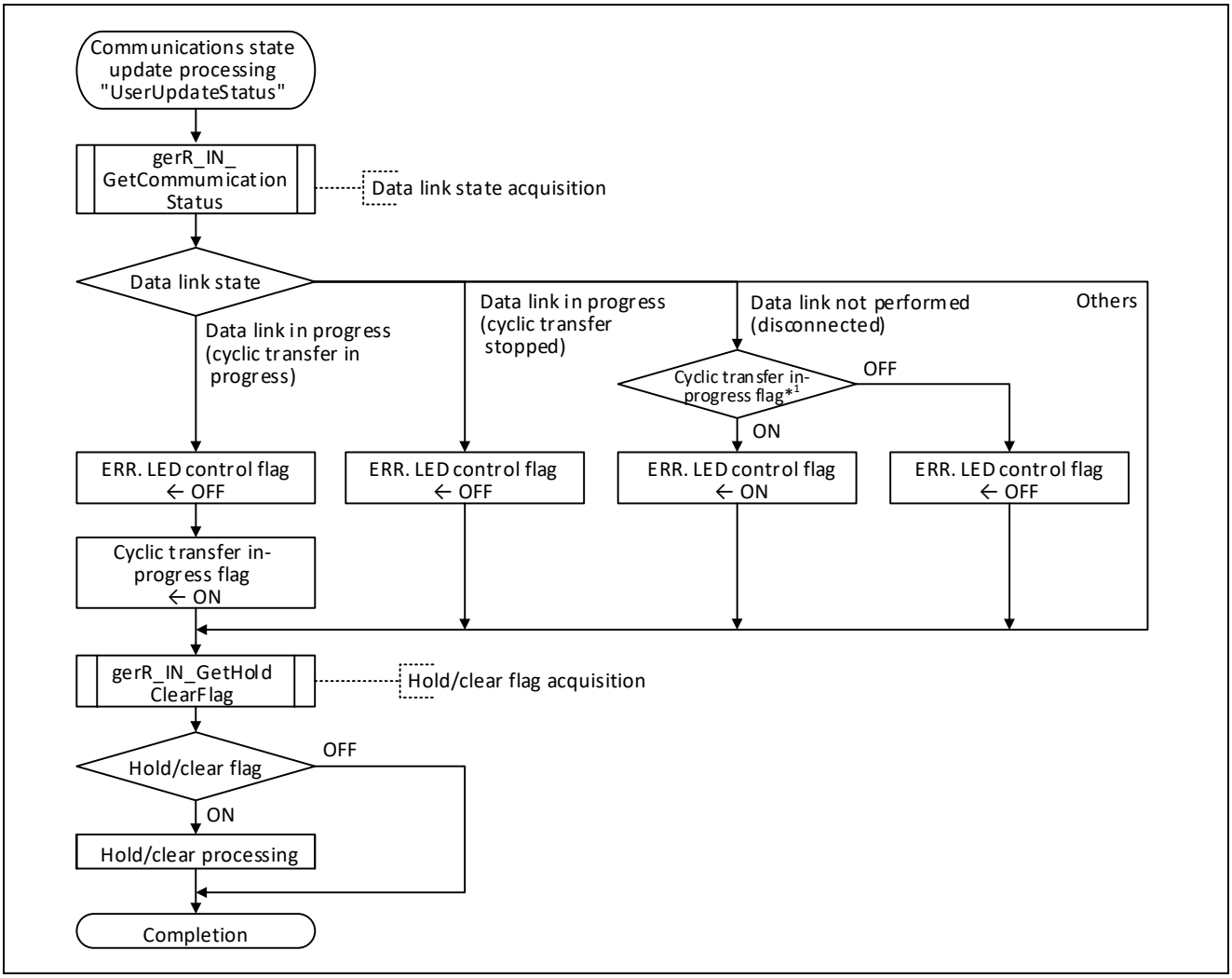


Figure 5.31 Flowchart for Communications State Update Processing

Note 1. This flag determines whether the disconnection occurred during cyclic transfer or before cyclic transfer (after linkup).

About hold/clear processing

Hold/clear processing is a process to continue (hold) or stop (clear) the output of the received cyclic data to the other station, when the home station is disconnected from a data link or when the application of the master station is stopped or abnormal.
For details, see section 5.4, User Program Details (Cyclic Transfer Related).

5.4.5 Cyclic Transfer State Update Processing

This function acquires the size of cyclic transfer specified by the master station and the state of cyclic transfer.

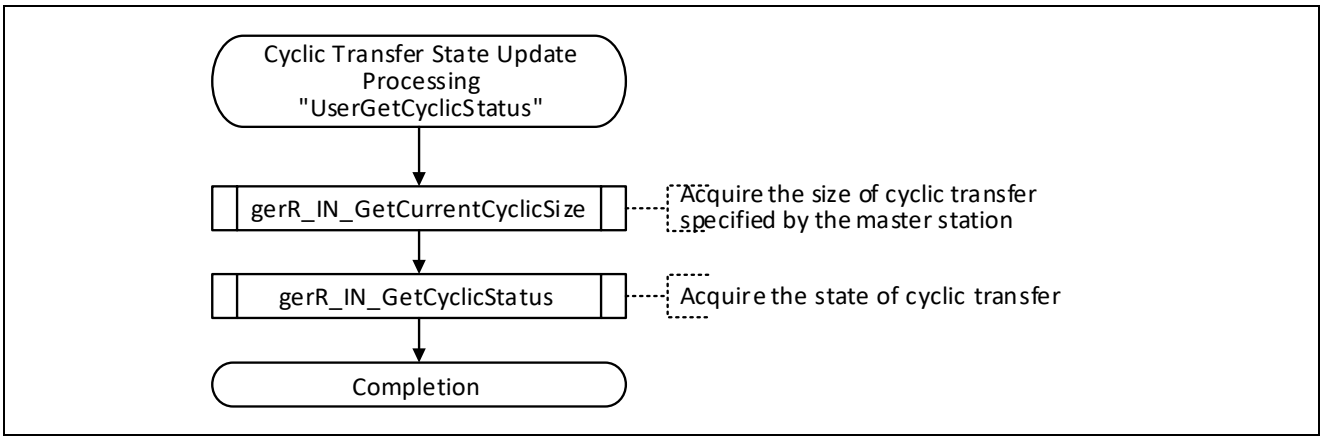


Figure 5.32 Flowchart for Cyclic Transfer State Update Processing

5.5 User Program Details (State Management and Transient Transfer Related)

5.5.1 Home Station Error Processing

Add error processing to this function when describing error processing of a user application in an idle task of a sample code instead of a task created or added by the user.
This processing is optional.

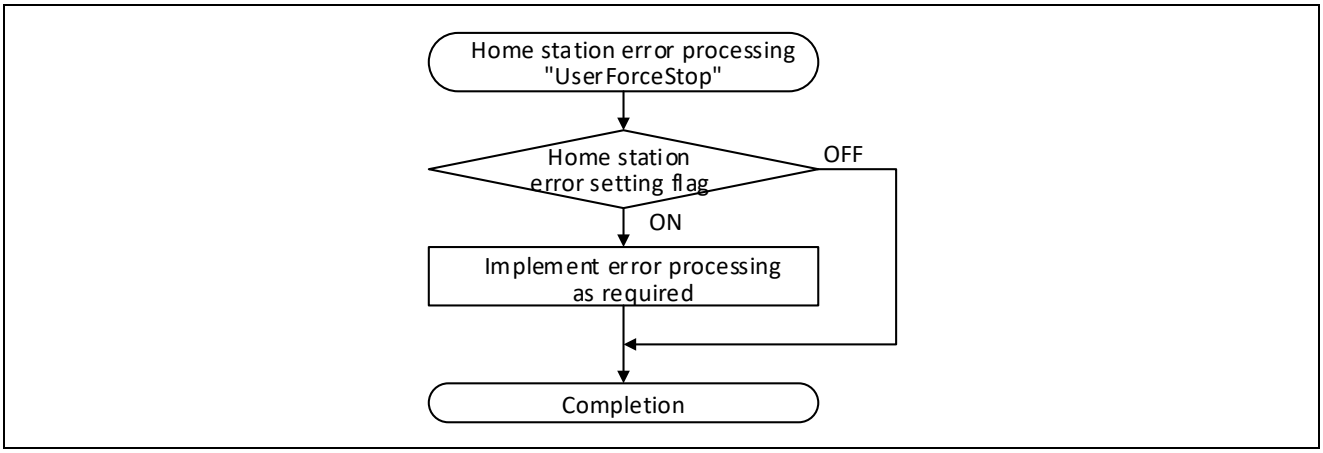


Figure 5.33 Flowchart for Home Station Error Processing

5.5.2 Event Processing

This function performs processing in response to an event occurred in R-IN32M4-CL3.

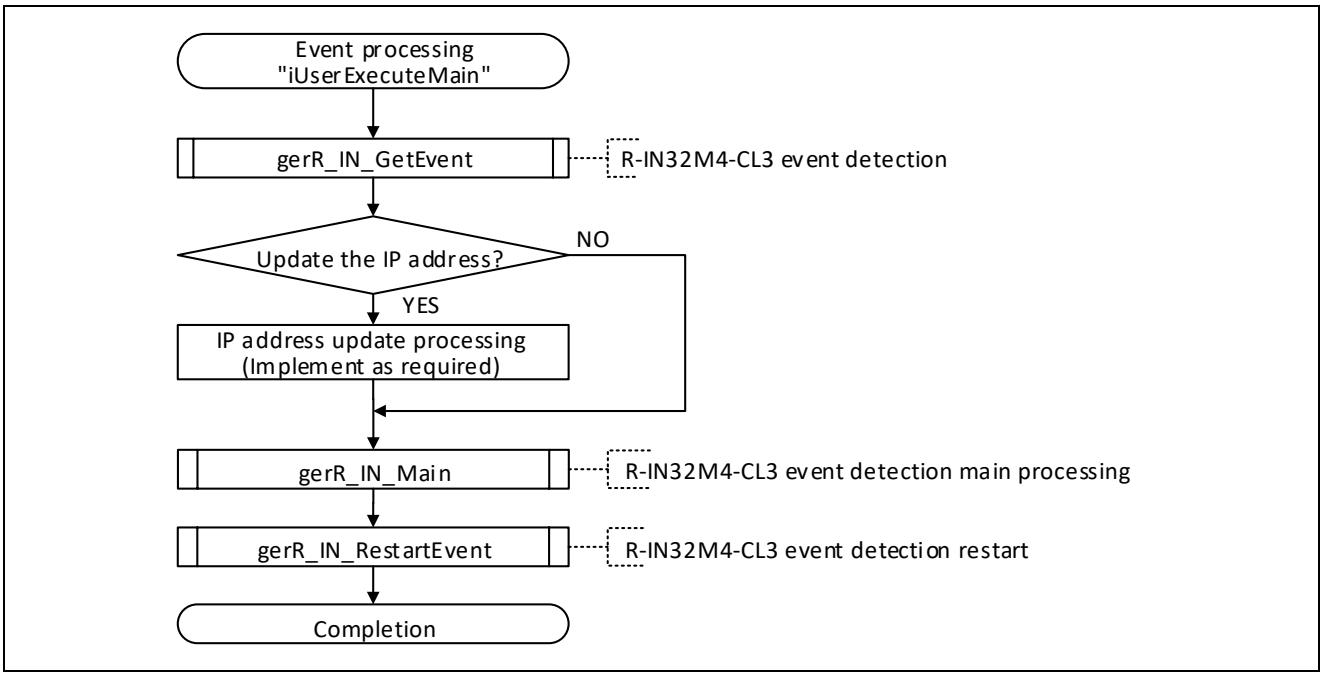


Figure 5.34 Flowchart for Event Processing

5.5.3 LED Update Processing

This function controls on, off, and blinking states of RUN LED, D LINK LED, and ERR LED in accord with the data link state of the home station.

The lighting / extinguishing / blinking of the ERR. LED is controlled according to the value of the ERR. LED control flag (gulErrCtrl) set in "UserUpdateStatus" (5.4.4 Communications State Update Processing).
When controlling on, off, or blinking of ERR LED for the convenience of the user application, change the value of the ERR LED control flag (gulErrCtrl) within this function.

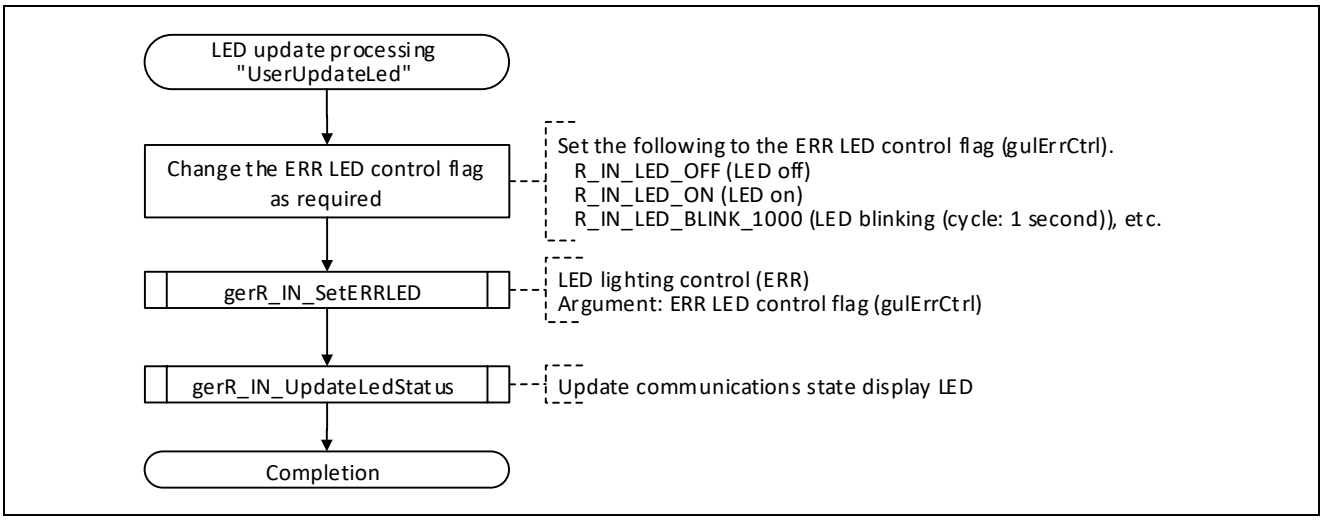


Figure 5.35 Flowchart for LED Update Processing

5.5.4 MIB (Statistical) Information Acquisition Processing

This function acquires MIB (statistical) information.

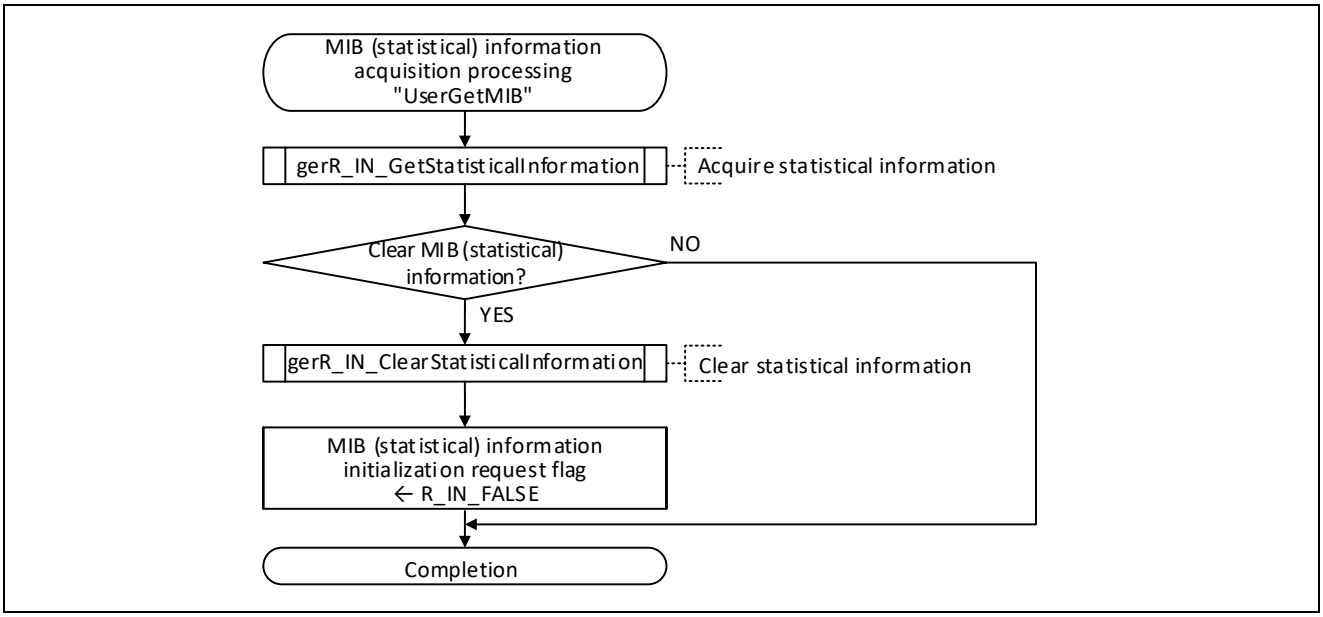


Figure 5.36 Flowchart for MIB (Statistical) Information Acquisition Processing

5.5.5 SLMP Reception Processing

This function acquires SLMP frames that the home station has received, and processes the data.

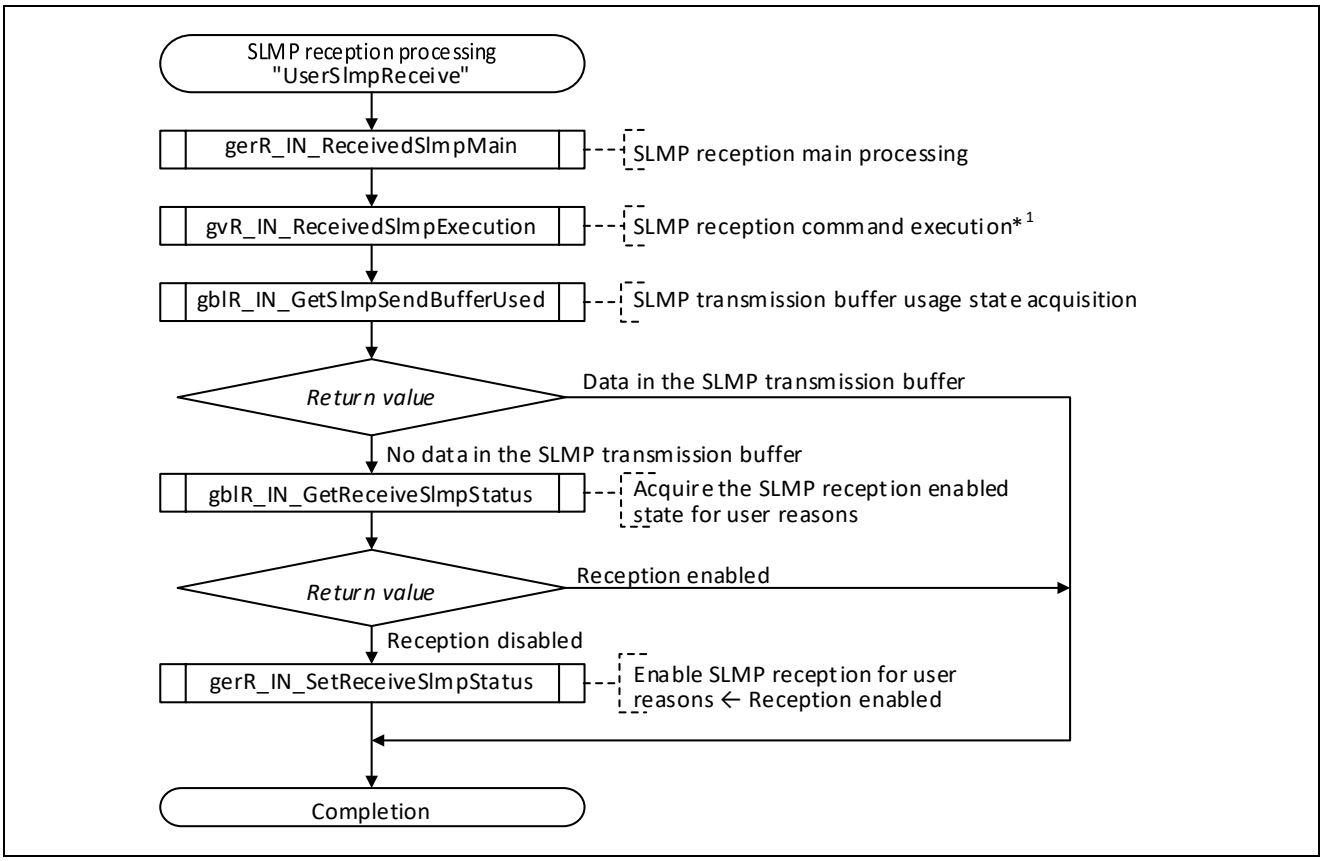


Figure 5.37 Flowchart for SLMP Reception Processing

Note 1. If a command of the received SLMP frame is supported, the corresponding command reception processing function is called. For the commands supported in the initial sample code, refer to Table 4.3.

(1) When the home station is a server (when receiving a request command)

When a request command (a command requested from another station to the home station) listed in Table 4.3 is received, the corresponding command function performs processing for the request command (request reception processing and response transmission processing).

To receive SLMP request commands not listed in Table 4.3, add desired SLMP commands, SLMP sub-commands, and command reception processing functions created by users in the execution function table "R_IN_SLMP_FUNCTION_REQUEST_TBL_T" (stored in the R_IN32M4_CL3_Slmp_Receive.c file).

(2) When the home station is a client (when receiving a response command)

When a response command (a response to the command requested from the home station to another station) listed in Table 4.3 is received, the corresponding command function performs processing for the response command (response reception processing).

To receive SLMP response commands not listed in Table 4.3, add desired SLMP commands, SLMP sub-commands, and command reception processing functions created by users in the execution function table "R_IN_SLMP_FUNCTION_RESPONSE_TBL_T" (stored in the R_IN32M4_CL3_Slmp_Receive.c file).
Then, add the request frame creation process for the command you want to send to "UserSlmpMakeRequest" (5.5.7 SLMP Request Frame Creation Processing).

When an ST type response frame is received, there is no information (serial number, command, subcommand) that identifies the command in the response frame, so "UserSlmpReceive3EFrame" (5.5.8 SLMP ST (3E) Response Frame Reception Processing) that performs reception processing regardless of the command is called.

5.5.6 SLMP Transmission Processing

This function sends SLMP frames.

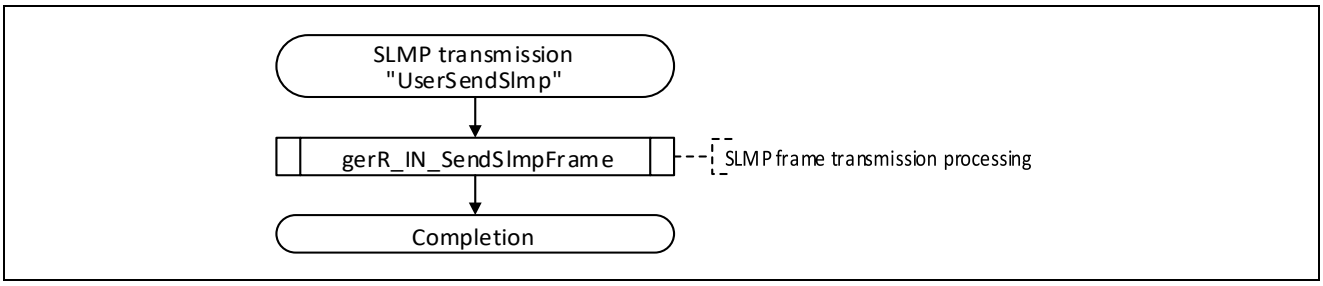


Figure 5.38 Flowchart for SLMP Transmission Processing

5.5.7 SLMP Request Frame Creation Processing

This processing is performed when the home station is a client.

If there is an SLMP command to be sent, this function calls the SLMP request frame creation processing function for this SLMP command. This sample code calls frame creation processing for the SLMP memory read request as an example. Add the frame creation processing for the request command to be sent.

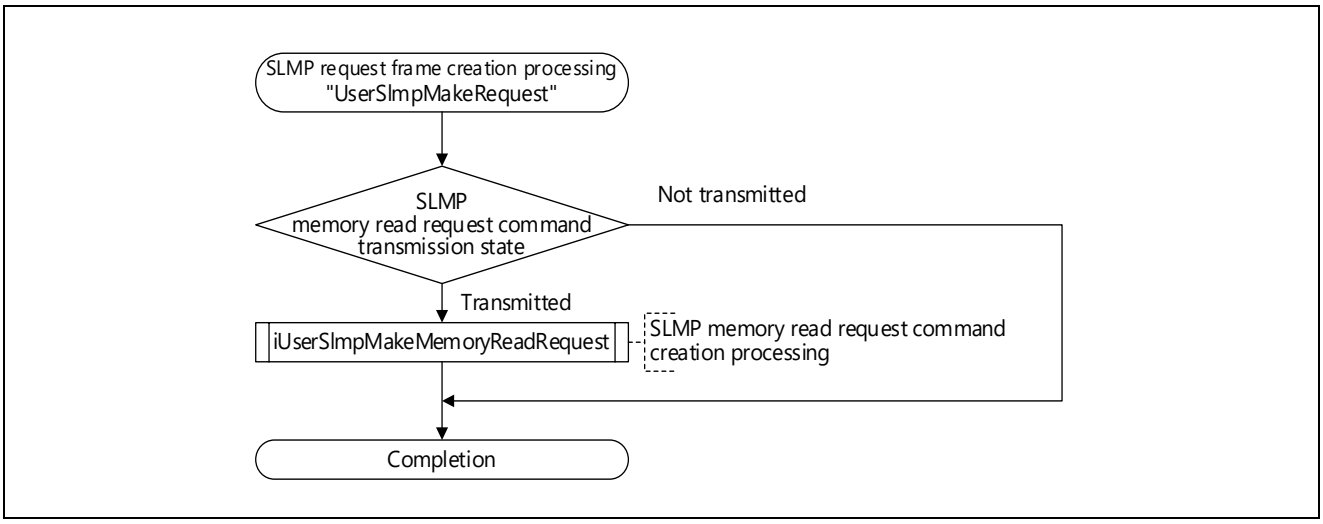


Figure 5.39 Flowchart for SLMP Request Frame Creation Processing

5.5.8 SLMP ST (3E) Response Frame Reception Processing

This processing is performed when the home station is a client.

Requests to other stations in SLMP ST (3E) frames and executes common reception processing when the response frame is received.
Since there is no information (serial number, command, subcommand) that identifies the command in the ST (3E) response frame, this process is called regardless of which command is used.

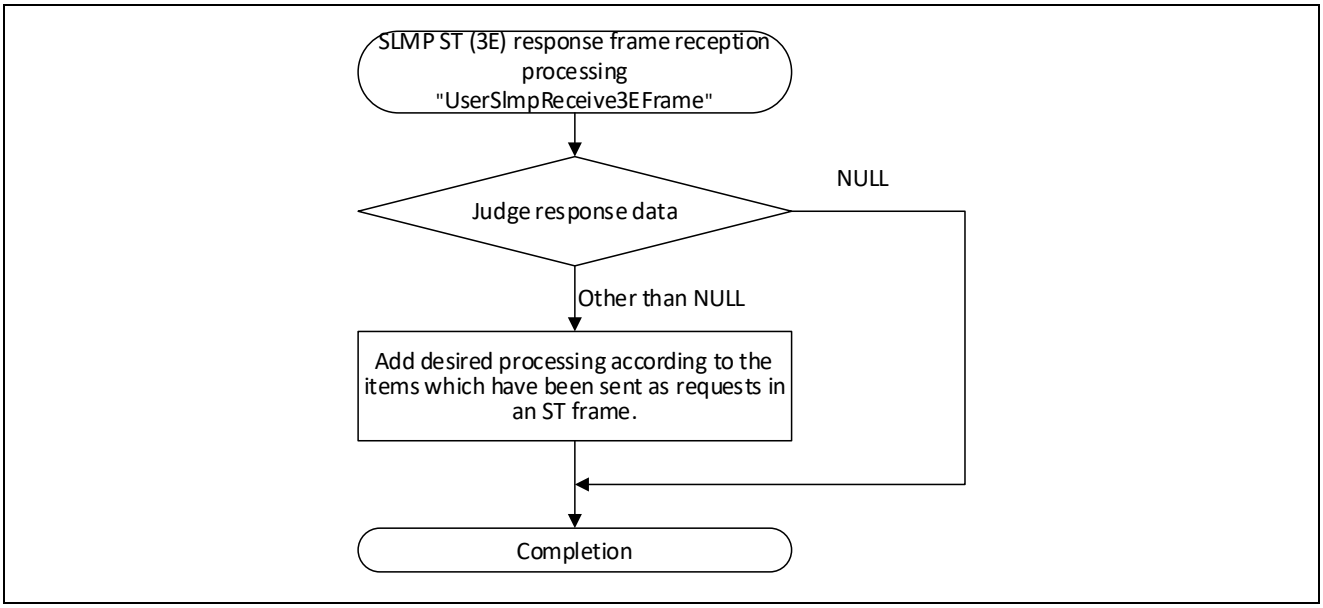


Figure 5.40 Flowchart for SLMP ST (3E) Response Frame Reception Processing

5.5.9 Processing for Checking a Fatal Error Detected in the R-IN32M4-CL3 Driver

This function checks for the detection of a fatal error by the R-IN32M4-CL3 driver regardless of any requests from the user program. If a fatal error has occurred, processing for the fatal error proceeds.

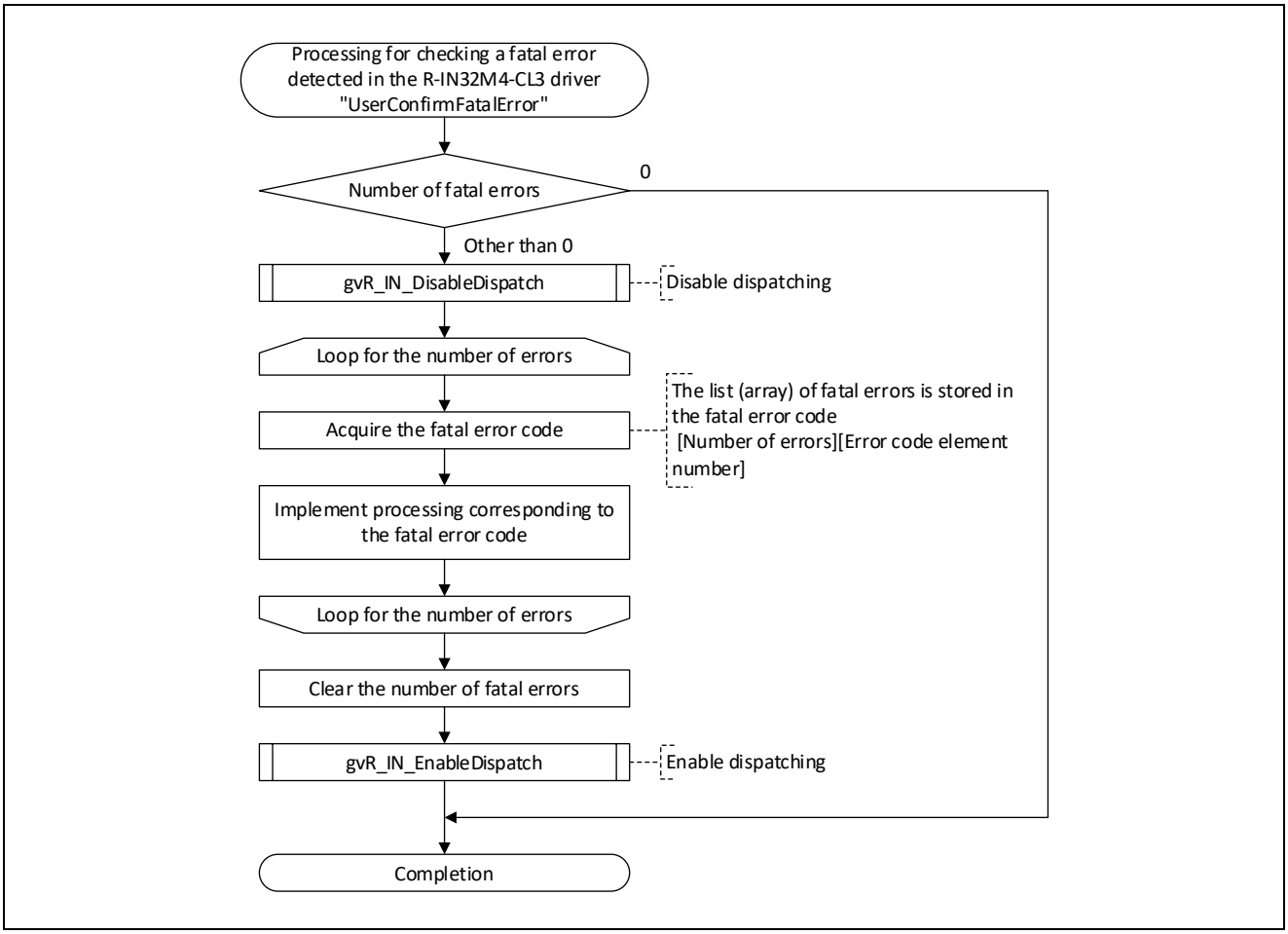


Figure 5.41 Flowchart for Processing for Checking a Fatal Error Detected in the R-IN32M4-CL3 Driver

Table 5.20 Fatal Error Codes

Fatal Error Code	Value	Description
R_IN_FATALERROR_MDIOCOMMAND_TIMEOUT_ERROR	0000 D52AH	MDIO command wait error
R_IN_FATALERROR_LOOPBACKTEST_SEND_ERROR	0000 D530H	A transmission error occurred in the loopback test.
R_IN_FATALERROR_LOOPBACKTEST_RECEIVE_FRAME_ERROR	0000 D531H	An FCS error frame was received in the loopback test.
R_IN_FATALERROR_LOOPBACKTEST_RECEIVE_COUNT_ERROR	0000 D532H	The number of received frames is abnormal in the loopback test.
R_IN_FATALERROR_LOOPBACKTEST_RECEIVE_DISCARD_COUNT_ERROR	0000 D533H	The received frame has been discarded in the loopback test.
R_IN_FATALERROR_SYNCMODE_CHANGE	1000 0001H	Synchronous mode was changed.

There are two general cases of the generation of a fatal error: One case consists of those fatal errors which occur in response to requests for processing from the user program; the other consists of those fatal errors which occur within the R-IN32M4-CL3 driver independently of requests from the user program.

(1) Case where fatal errors occur in response to requests from the user program
Communications start processing, hardware testing, etc. in initialization processing within the idle task.
In these cases, a check for fatal errors proceeds after each function is called.

(2) Case where fatal errors occur within the R-IN32M4-CL3 driver independently of requests from the user program
An example of such cases is where state management and transient main processing within the idle task leads to an invalid SLMP request message from the master station.
In these cases, a check for fatal errors from the R-IN32M4-CL3 driver proceeds when this processing (UserConfirmFatalError) is run at the end of main processing.

5.6 User Program Details (SLMP Command Execution Related)

5.6.1 SLMP Memory Read Request Command Reception Processing

This function reads data in the specified buffer memory areas of the home station and sets the data to a response frame when the SLMP dual port memory batch read request frame (SLMP command: 0613H) is received.
A frame of the LMT, MT, or ST type is received through processing in response to reception of this command. If the received frame is of a different type from the specified one, an error response is sent and received data are discarded.If an error in the data length after the subcommand section of the received LMT frame is detected, the R-IN32M4-CL3 driver sets C05CH to the end code and sends a response. Therefore, this command receive processing is not performed.

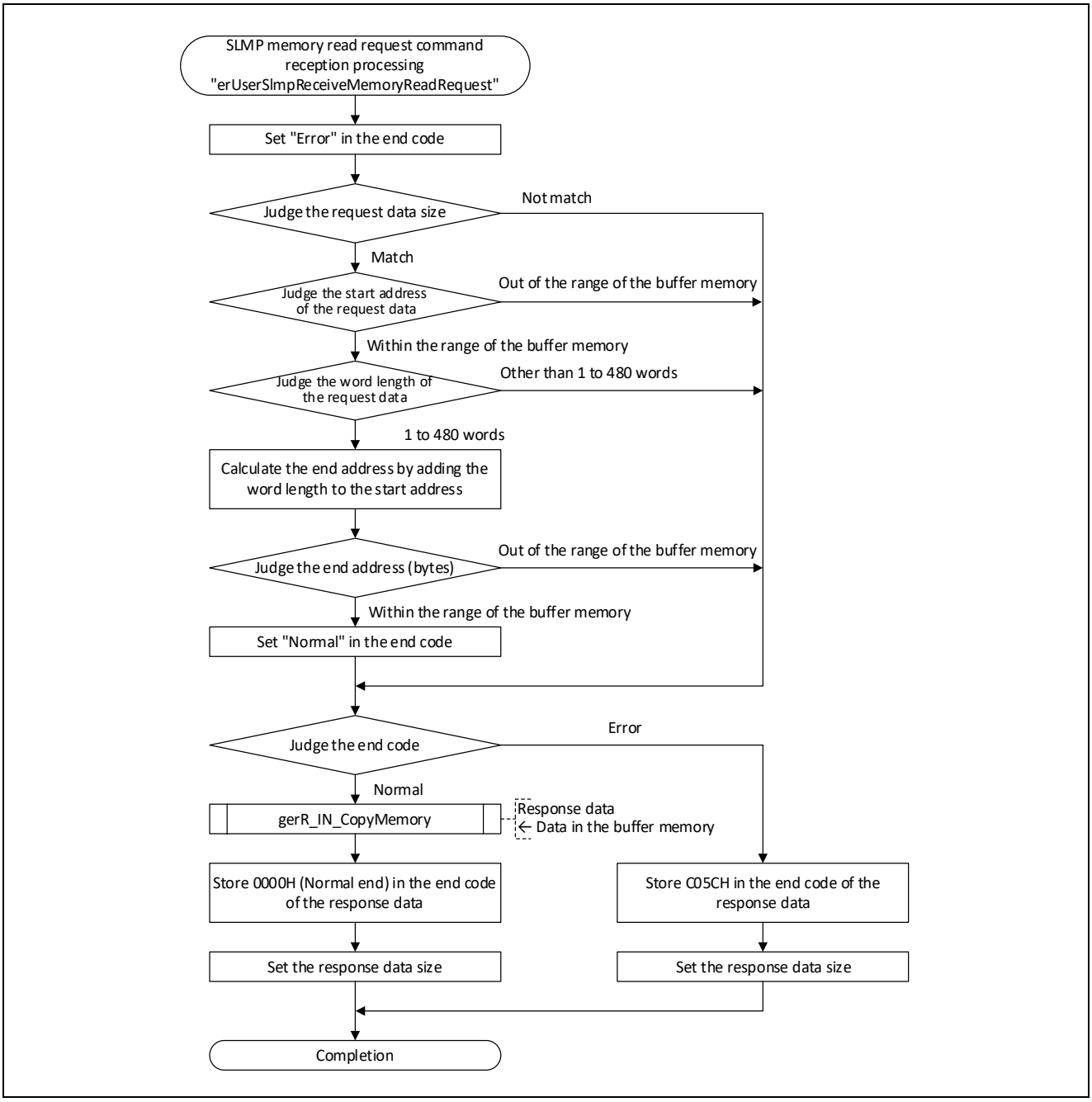


Figure 5.42 Flowchart for SLMP Memory Read Request Command Reception Processing

5.6.2 SLMP Memory Write Request Command Reception Processing

This function writes data to the specified buffer memory areas of the home station and sends a response frame when the SLMP dual port memory batch write request frame (SLMP command: 1613H) is received.
A frame of the LMT, MT, or ST type is received through processing in response to reception of this command.
If the received frame is of a different type from the specified one, an error response is sent and received data are discarded. If an error in the data length after the subcommand section of the received LMT frame is detected, the R-IN32M4-CL3 driver sets C05CH to the end code and sends a response. Therefore, this command receive processing is not performed.

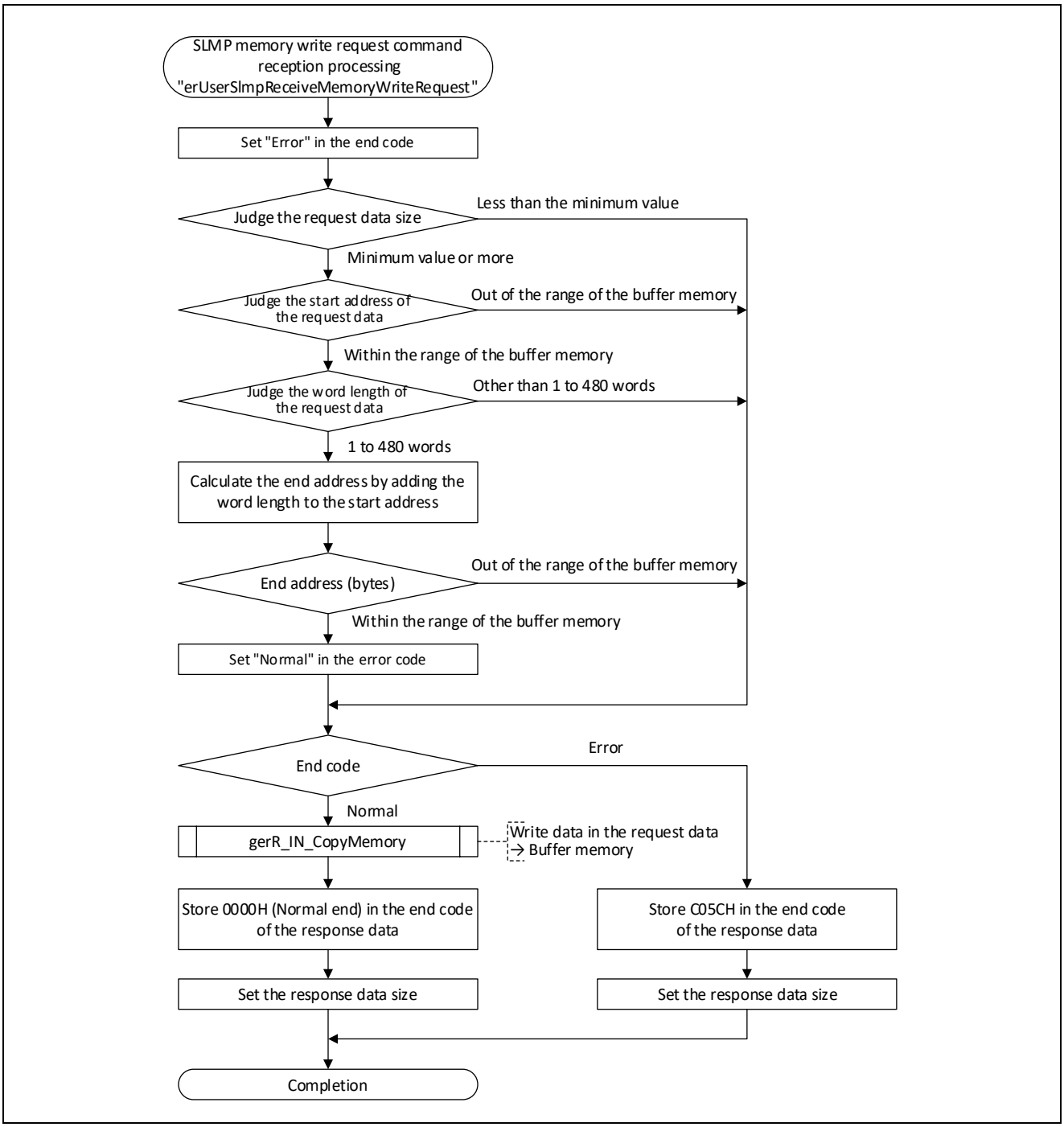


Figure 5.43 Flowchart for SLMP Memory Write Request Command Reception Processing

5.6.3 SLMP Memory Read Request Command Creation Processing

This function creates a request frame for the SLMP dual port memory batch read command (SLMP command: 0613H).

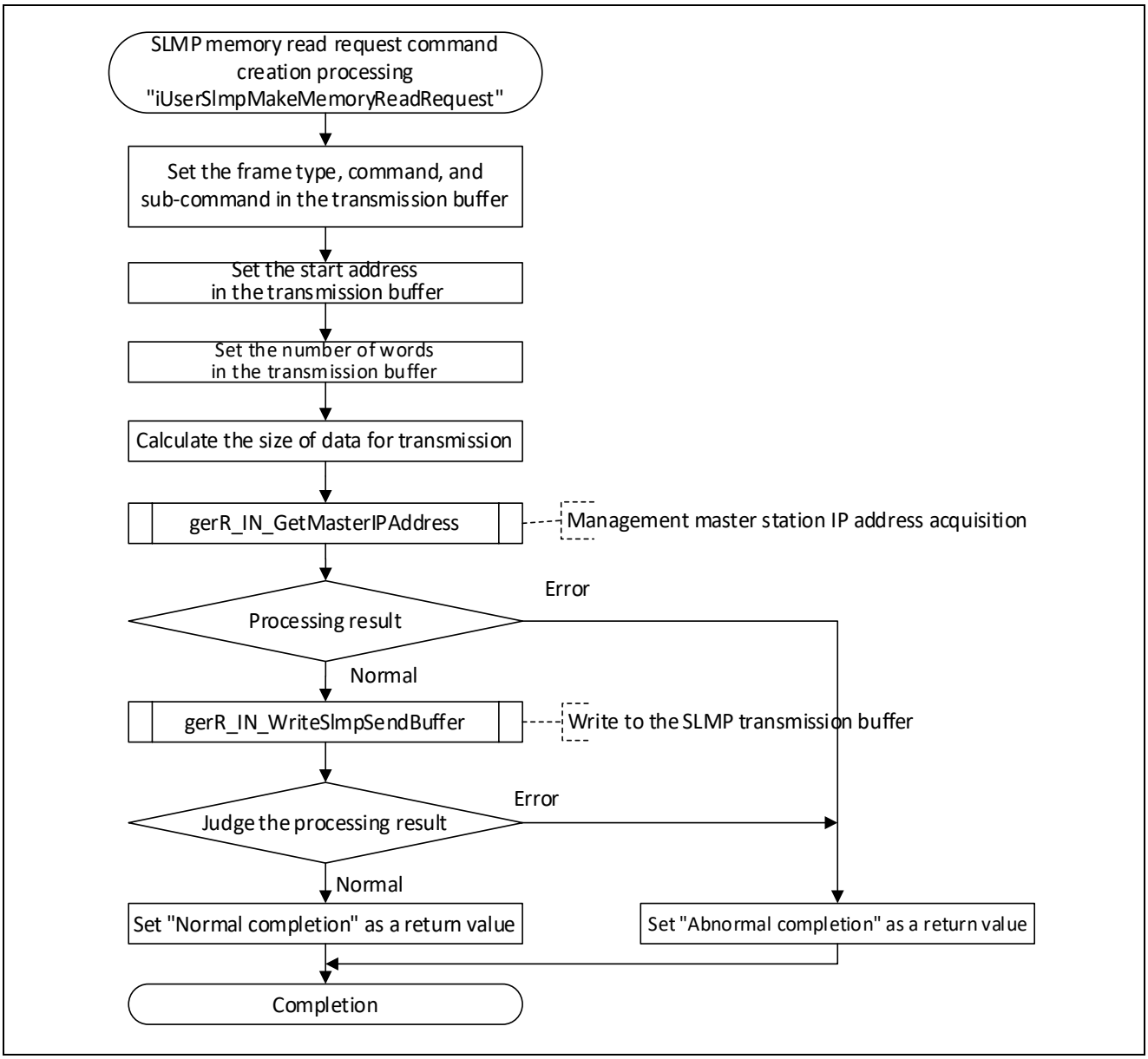


Figure 5.44 Flowchart for SLMP Memory Read Request Command Creation Processing

The following tables shows the configuration of the request frame. For details on each member, refer to "SLMP Specification" published by the CC-Link Partner Association or the SLMP Reference Manual.

Table 5.21 R_IN_SLMP_6E_FRAME_REQUEST_T

No	Member		Description
1	R_IN_SLMP_6E_FRAME_HEAD_T	stSlmpHead	SLMP header information (Table 5.22)
2	USHORT	usExtendedStationNumber	Expanded station number of request destination station
3	USHORT	usByteSize	Request data length
4	USHORT	usTimer	Timer
5	USHORT	usCommand	Command
6	USHORT	usSubCommand	Sub-command
7	UCHAR	uchReserve	Reserved
8	UCHAR	uchDataId	Message identification value
9	USHORT	usDataDevideNum	Total number of divided data sets
10	USHORT	usDataNumber	Division number
11	ULONG	aulDataArea [361]	Request data [Dword size]

Table 5.22 R_IN_SLMP_6E_FRAME_ HEAD_T

No	Member		Description
1	USHORT	usFrameType	Frame Type
2	USHORT	usSerialNumber	Serial Number
3	USHORT	usReserve1	Reserved
4	UCHAR	uchNetworkNumber	Network Number of request destination station
5	UCHAR	uchStationNumber	Station number of request destination station
6	USHORT	usProcessorNumber	Processor number of request destination station
7	UCHAR	uchMultiDropNo	Requested station processor sub-number
8	UCHAR	uchReserve3	Reserved

5.6.4 SLMP Memory Read Response Command Reception Processing

This function handles processing for reception of the SLMP dual port memory batch read response frame (SLMP command: 0613H).
A response to the request which has been sent to another station by the home station is received.
A frame of the LMT type is received through processing in response to reception of this command. If the received frame is of a different type from the specified one, an error response is sent and received data are discarded. If an error in the data length after the subcommand section of the received LMT frame is detected, the received data is discarded. Therefore, this command receive processing is not performed.

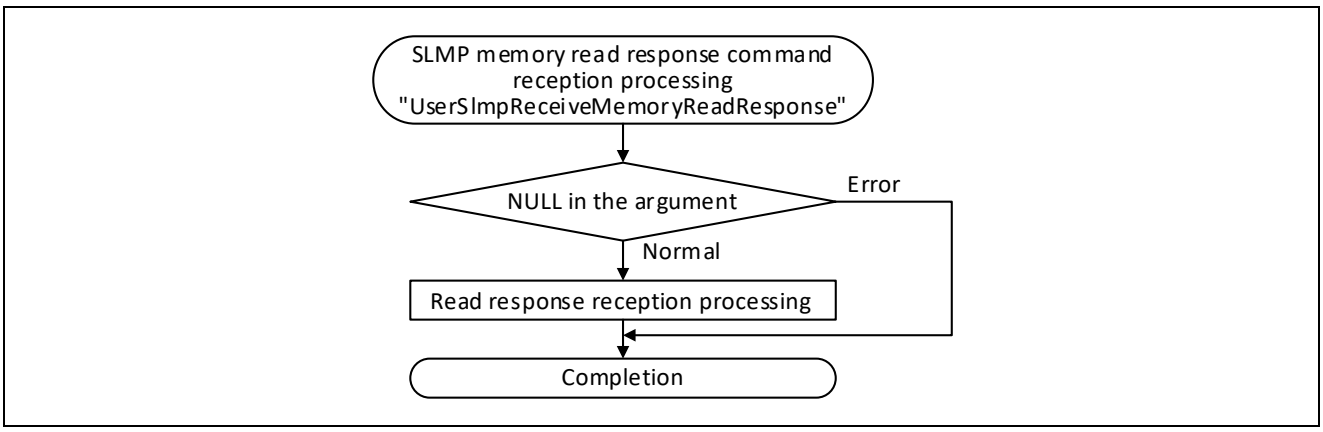


Figure 5.45 Flowchart for SLMP Memory Read Response Command Reception Processing

5.6.5 SLMP Remote Reset Request Command Reception Processing

This function resets the home station (R-IN32M4-CL3) when the SLMP remote reset request frame (SLMP command: 1006H) is received.
A frame of the LMT, MT, or ST type is received through processing in response to reception of this command. If the received frame is of a different type from the specified one, an error response is sent and received data are discarded. If an error in the data length after the subcommand section of the received LMT frame is detected, the R-IN32M4-CL3 driver sets C05CH to the end code and sends a response. Therefore, this command receive processing is not performed.

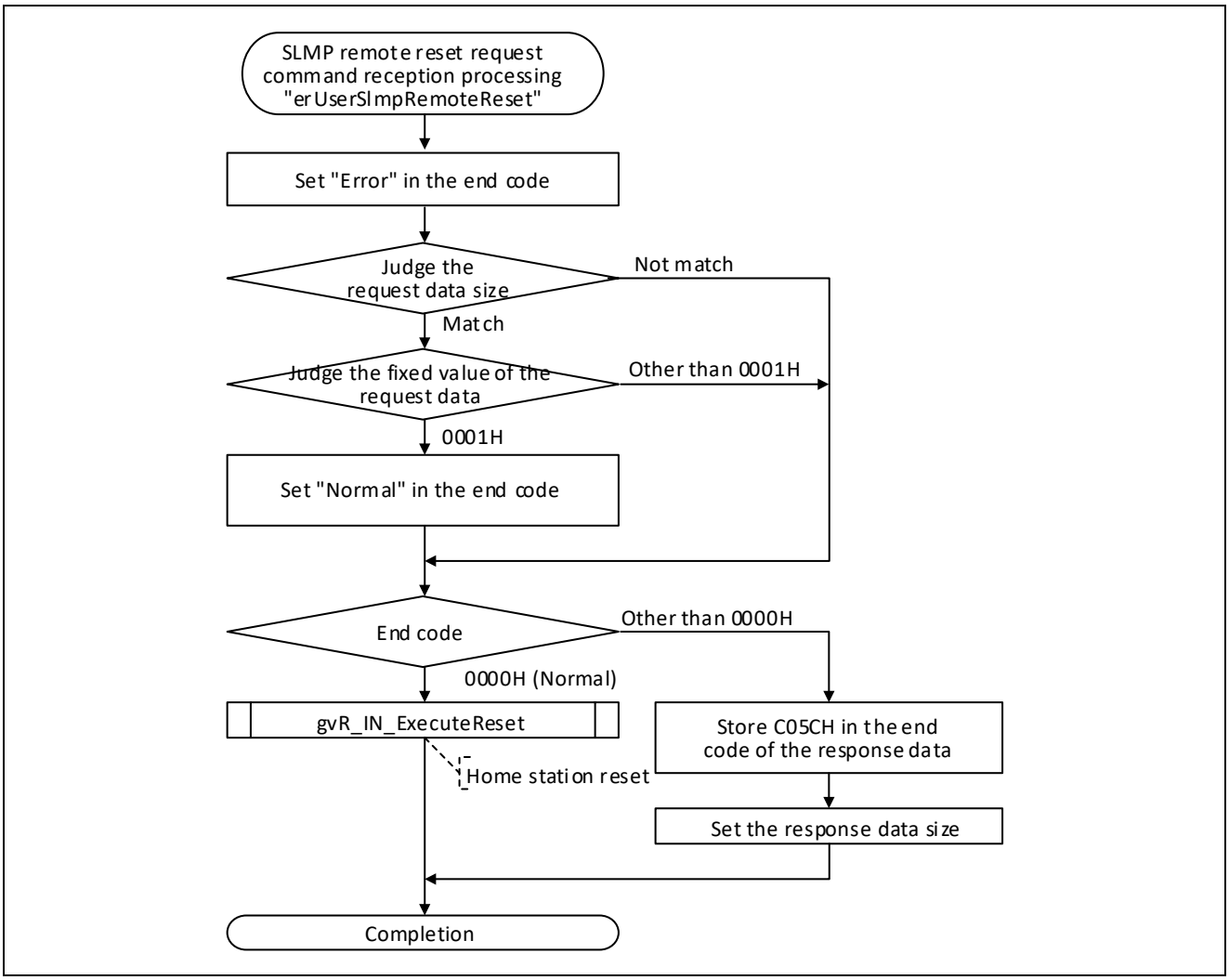


Figure 5.46 Flowchart for SLMP Remote Reset Request Command Reception Processing

5.6.6 SLMP Indicator Display Request Command Reception Processing

This function turns the user LED on or off in accord with the display instruction specified by SLMP.
A frame of the LMT, MT, or ST type is received through processing in response to reception of this command. If the received frame is of a different type from the specified one, an error response is sent and received data are discarded. If an error in the data length after the subcommand section of the received LMT frame is detected, the R-IN32M4-CL3 driver sets an error to the SLMP receive result in the request data (receive) included information.

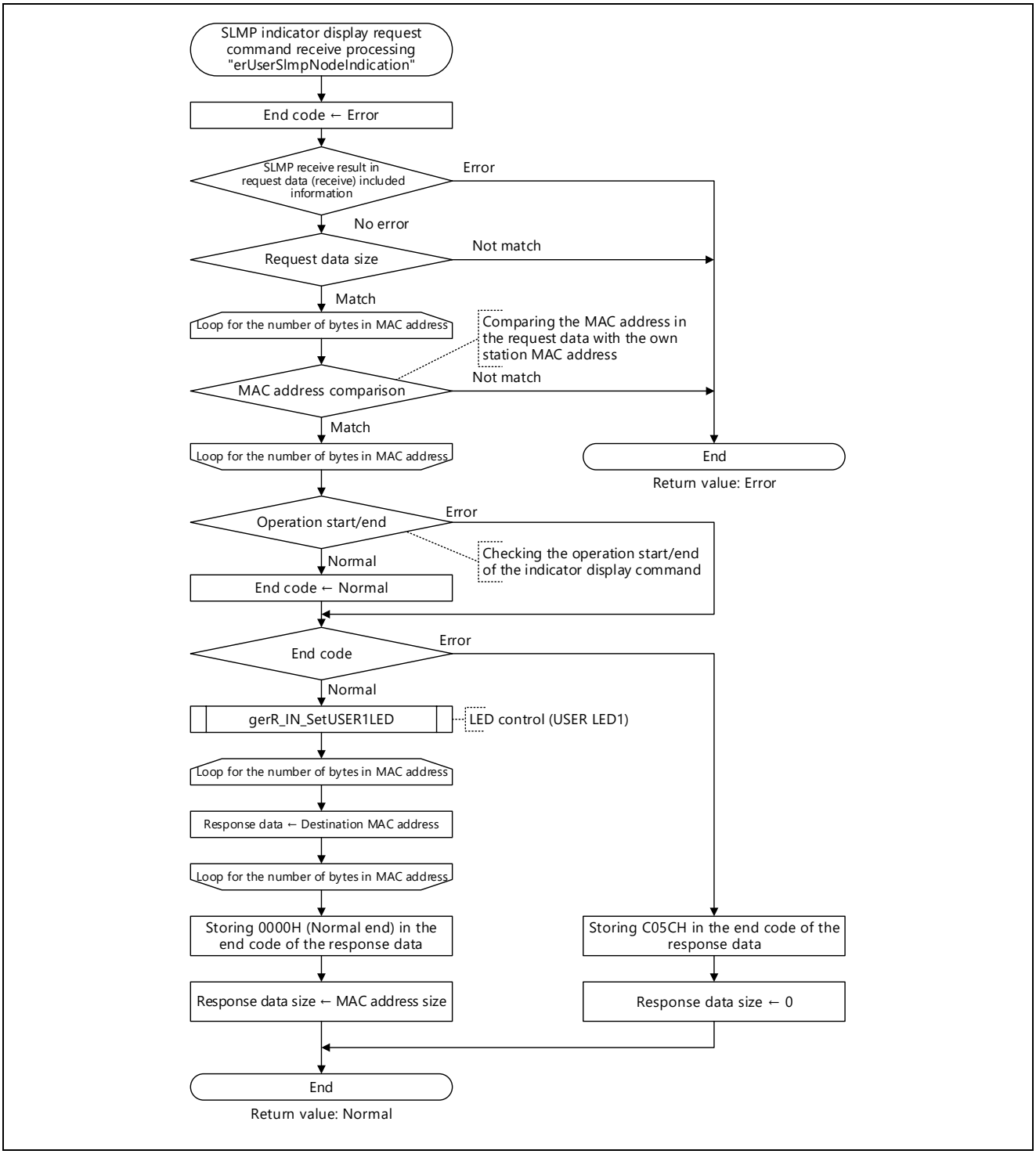


Figure 5.47 Flowchart for SLMP Indicator Display Request Command Reception Processing

5.6.7 SLMP IP Address Change Request Command Reception Processing

This function acquires the IP address of the server specified by SLMP.
A frame of the LMT, MT, or ST type is received through processing in response to reception of this command. If the received frame is of a different type from the specified one, an error response is sent and received data are discarded. If an error in the data length after the subcommand section of the received LMT frame is detected, the R-IN32M4-CL3 driver sets an error to the SLMP receive result in the request data (receive) included information.

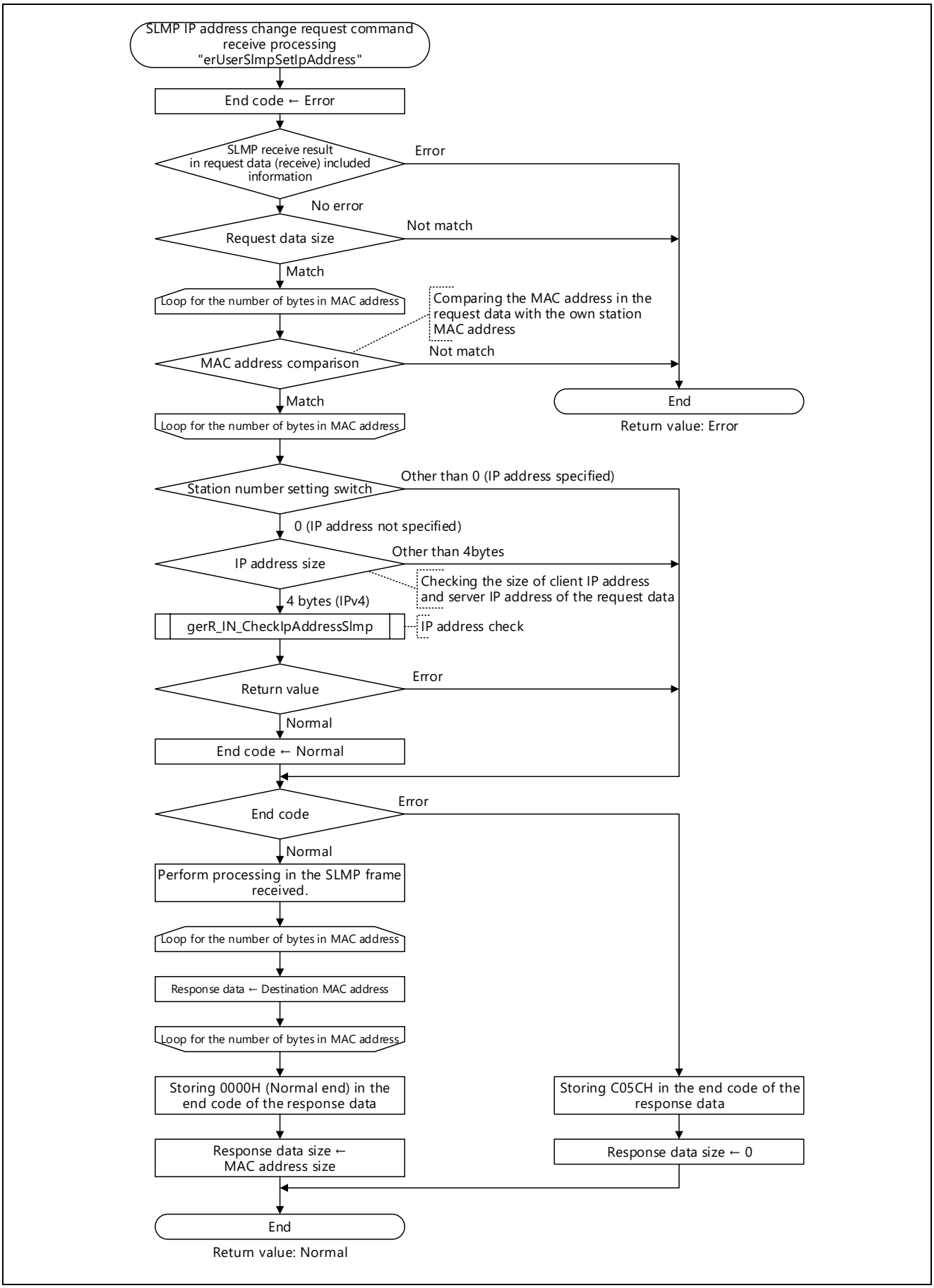


Figure 5.48 Flowchart for SLMP IP Address Change Request Command Reception Processing

5.6.8 SLMP Error History Clearing Request Command Reception Processing

This function clears all of the error histories.
When extension modules/slice remote I/O modules are used and the option information of the MIB current error information is used (the compiler switch "CURERR_OPTIONINFO_ENABLE" is enabled), the function checks the request destination station processor subnumber of the request data (receive) included information structure. And then, the function clears the error history logs of the controller information or the option information.
A frame of the LMT, MT, or ST type is received through processing in response to reception of this command. If the received frame is of a different type from the specified one, an error response is sent and received data are discarded. If an error in the data length after the subcommand section of the received LMT frame is detected, the R-IN32M4-CL3 driver sets C05CH to the end code and sends a response. Therefore, this command receive processing is not performed.

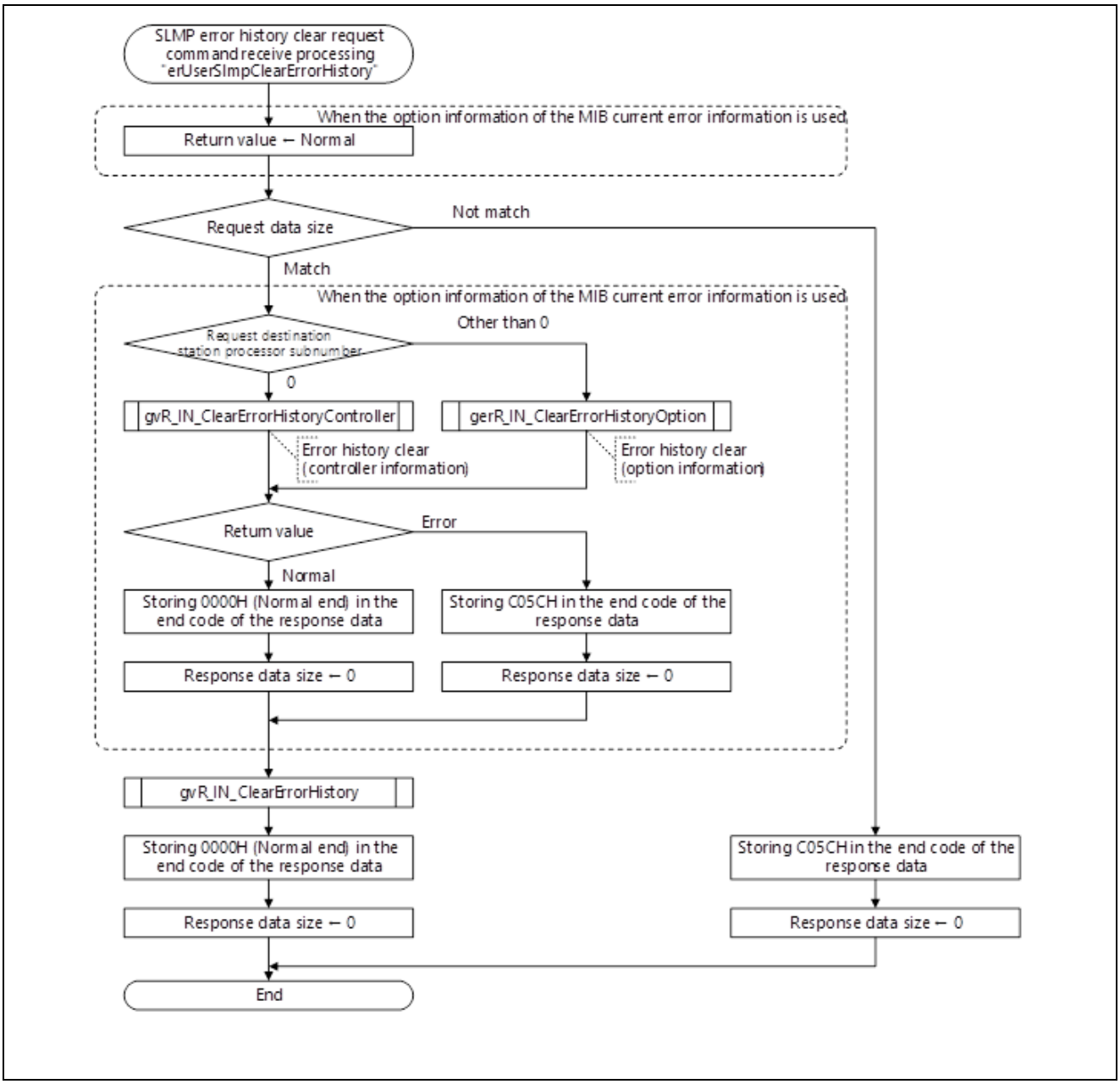


Figure 5.49 Flowchart for SLMP Error History Clearing Request Command Reception Processing

5.6.9 SLMP Network Time Offset Distribution Command Reception Processing

This function sets the network time offset value to the clock function.
A frame of the LMT, MT, or ST type is received through processing in response to reception of this command. If the received frame is of a different type from the specified one, an error response is sent and received data are discarded. If an error in the data length after the subcommand section of the received LMT frame is detected, the R-IN32M4-CL3 driver sets an error to the SLMP receive result in the request data (receive) included information.

For this command, sending a response is not required. To prevent response processing, even for normal end, "Error (no data)" is set as the return value.

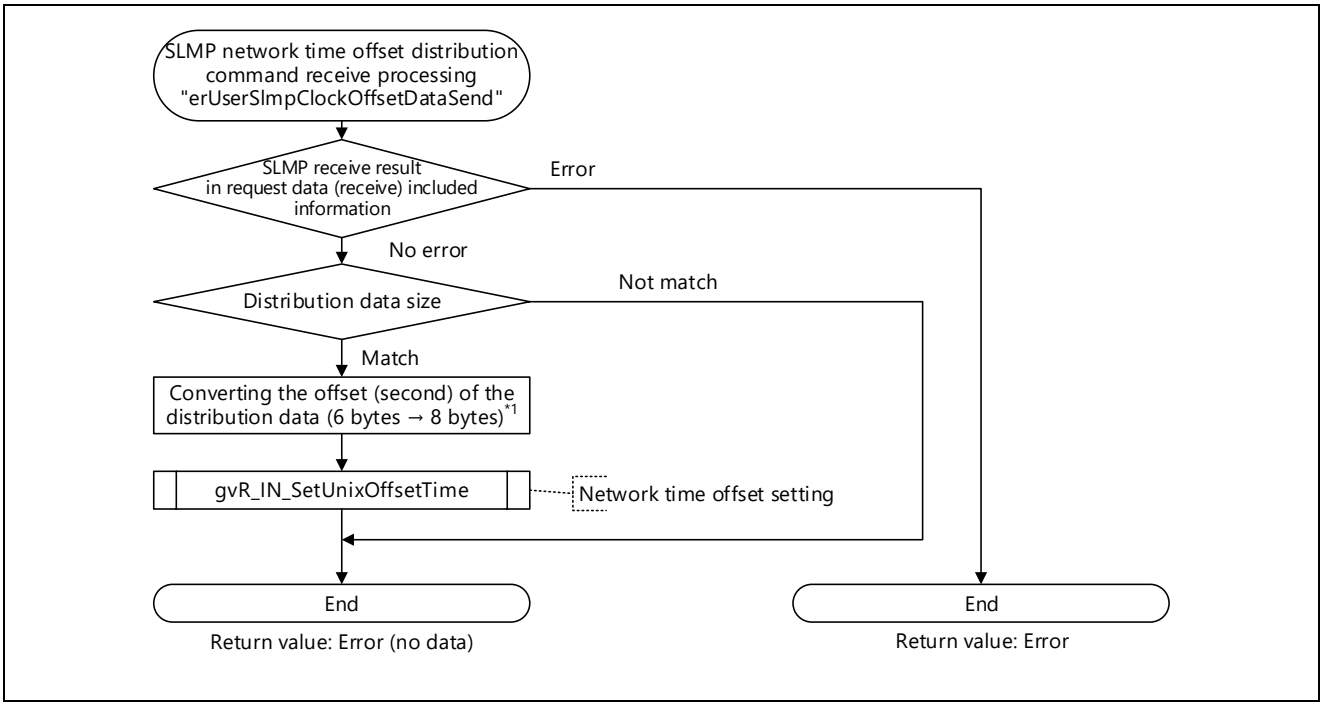


Figure 5.50 Flowchart for SLMP Network Time Offset Distribution Command Reception Processing

Note 1. An offset value (second) of the distributed data is 6-byte signed data. When the offset value is converted from 6-byte data to 8-byte data, the most significant bit of 6 bytes is checked. If the bit is negative, the higher-order 2 bytes of the converted 8 bytes are set to all Fs.

5.6.10 SLMP network time distribution command receive processing

This processing can be performed only for CC-Link IE TSN Class A.
The function sets the network time to the clock function.

The function receives LMT frames. When a frame other than above is received, the receive data is discarded. If an error in the data length after the subcommand section of the received LMT frame is detected, the R-IN32M4-CL3 driver sets an error to the SLMP receive result in the request data (receive) included information.

For this command, sending a response is not required. To prevent response processing, even for normal end, "Error (no data)" is set as the return value..

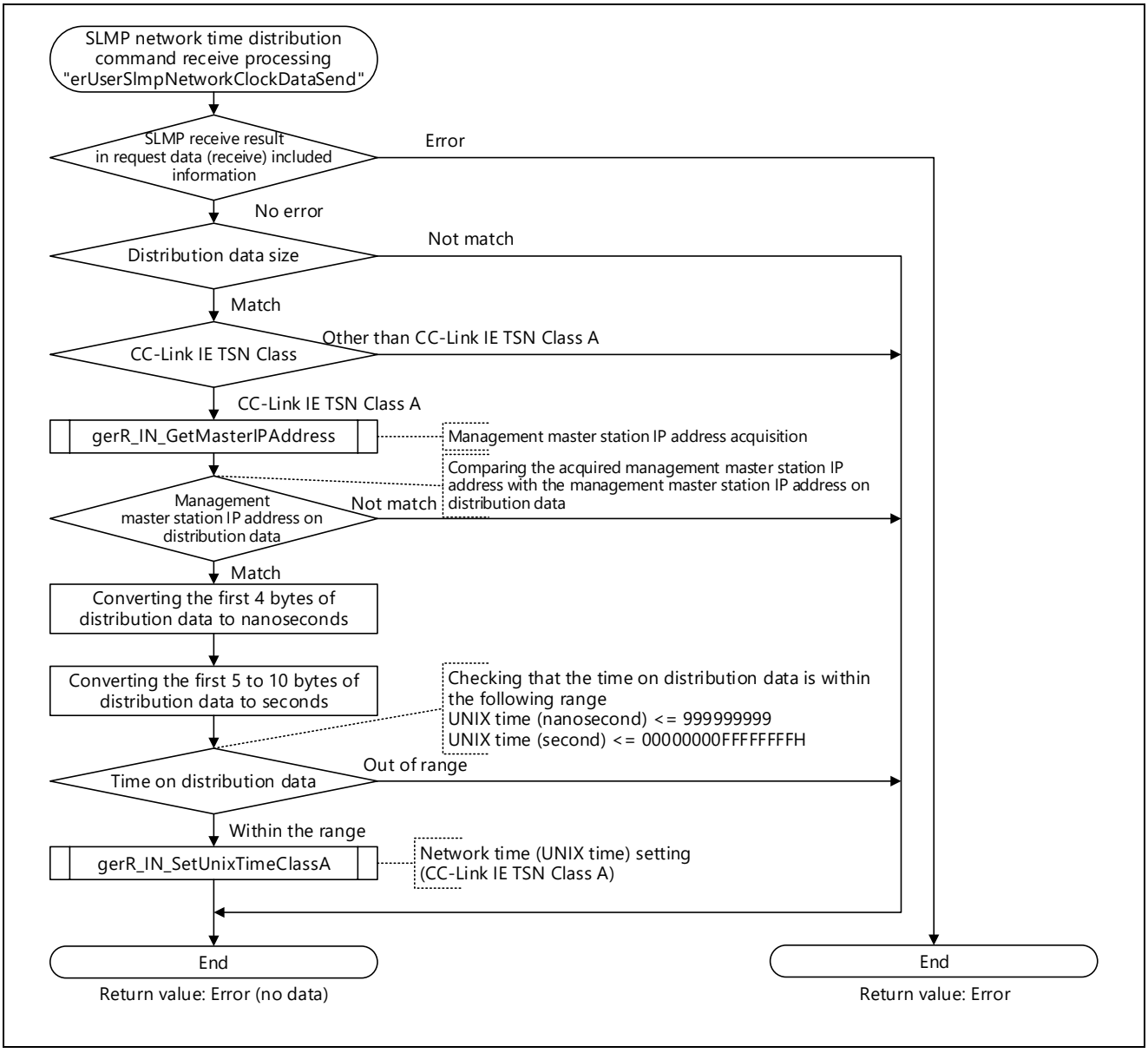


Figure 5.51 Flowchart for SLMP Network Time Distribution Command Receive Processing

5.6.11 SLMP Watchdog Counter Information Setting Request Command Reception Processing

This function sets the counter threshold for consecutive watchdog counter checking errors specified by the request message of the SLMP watchdog counter setting command (3210H) and returns offset information on the watchdog counter, etc. in the response message.

A frame of the LMT type is received through processing in response to reception of this command. If the received frame is of a different type from the specified one, an error response is sent and received data are discarded. If an error in the data length after the subcommand section of the received LMT frame is detected, the R-IN32M4-CL3 driver sets C05CH to the end code and sends a response. Therefore, this command receive processing is not performed.

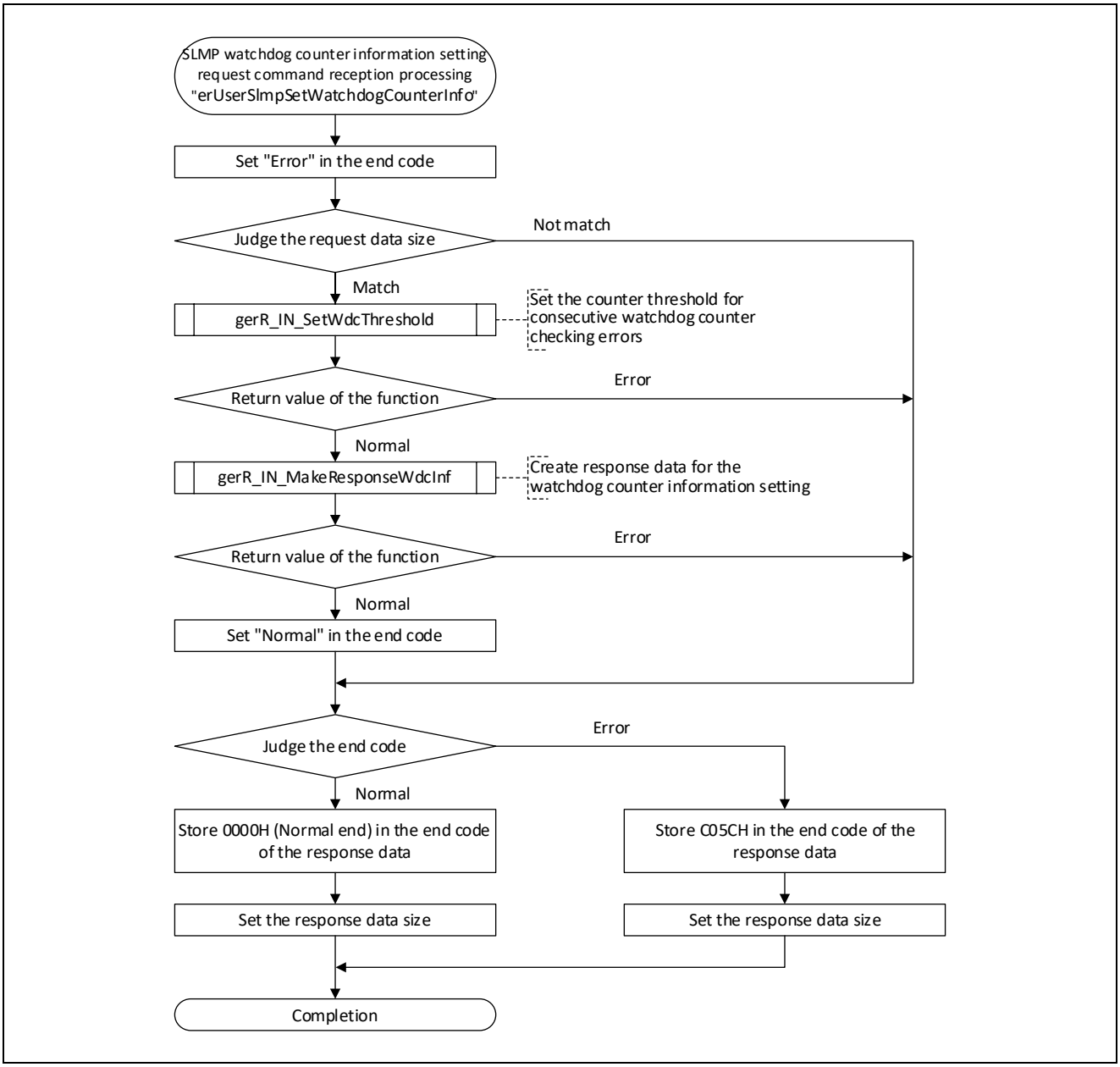


Figure 5.52 Flowchart for SLMP Watchdog Counter Information Setting Request Command Reception Processing

5.7 Details on Processing of User Programs (Slave Station Parameter Automatic Setting Related)

5.7.1 SLMP Communications Settings Acquisition Request Command Reception Processing

This function sets the communications port and the timeout time in the transmission buffer.
A frame of the LMT type is received through processing in response to reception of this command. If the received frame is of a different type from the specified one, an error response is sent and received data are discarded. If an error in the data length after the subcommand section of the received LMT frame is detected, the R-IN32M4-CL3 driver sets C05CH to the end code and sends a response. Therefore, this command receive processing is not performed.

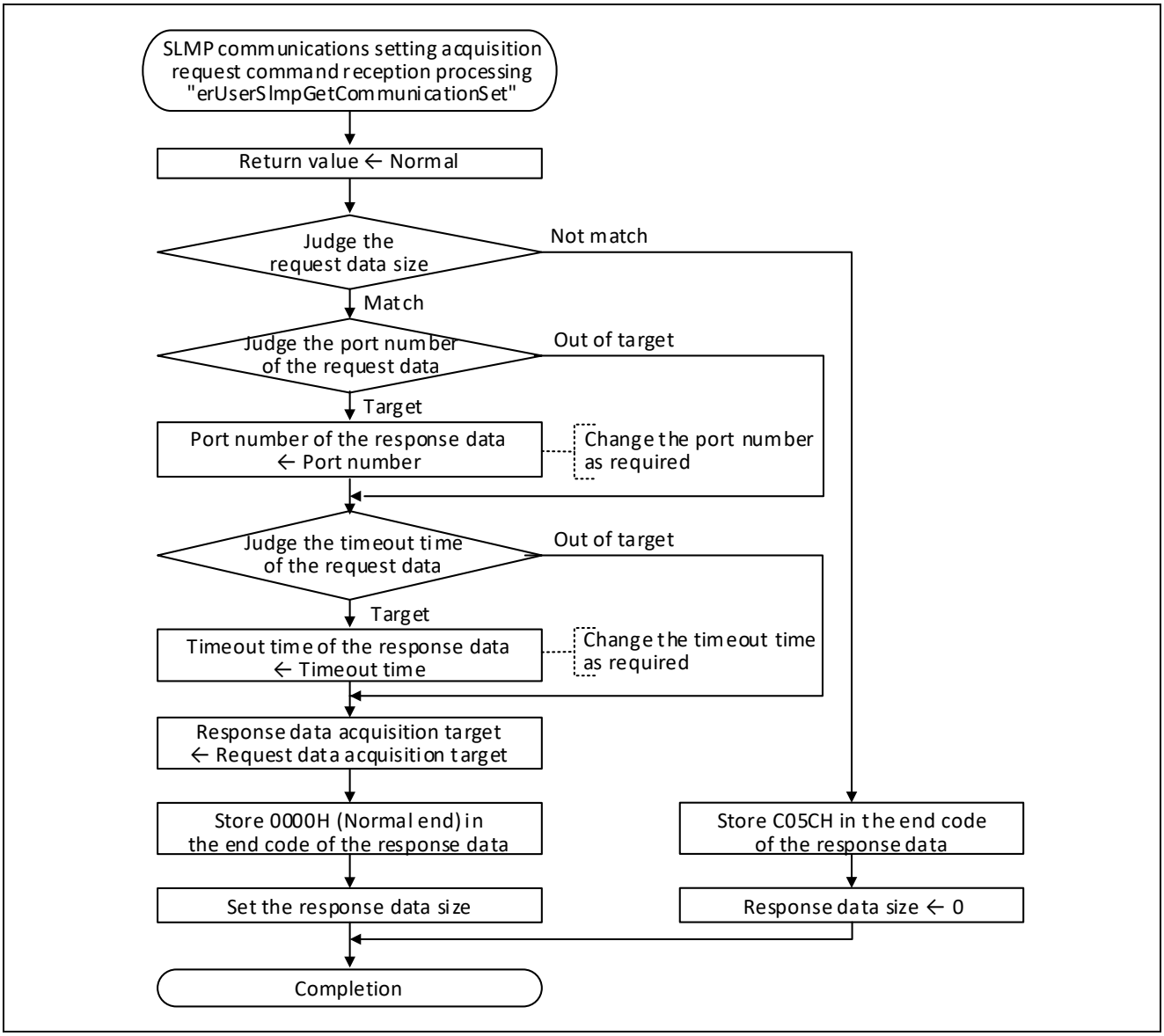


Figure 5.53 Flowchart for SLMP Communications Settings Acquisition Request Command Reception Processing

5.7.2 SLMP Parameter Distribution Necessity Check Request Command Reception Processing

This function checks whether parameter distribution is required or not.
A frame of the LMT type is received through processing in response to reception of this command. If the received frame is of a different type from the specified one, an error response is sent and received data are discarded. If an error in the data length after the subcommand section of the received LMT frame is detected, the R-IN32M4-CL3 driver sets C05CH to the end code and sends a response. Therefore, this command receive processing is not performed.

Figure 5.54 Flowchart for SLMP Parameter Distribution Necessity Check Request Command Reception Processing

5.7.3 SLMP Parameter Distribution Necessity Check Processing

This function checks whether parameter distribution is necessary every station sub-ID and creates response data.

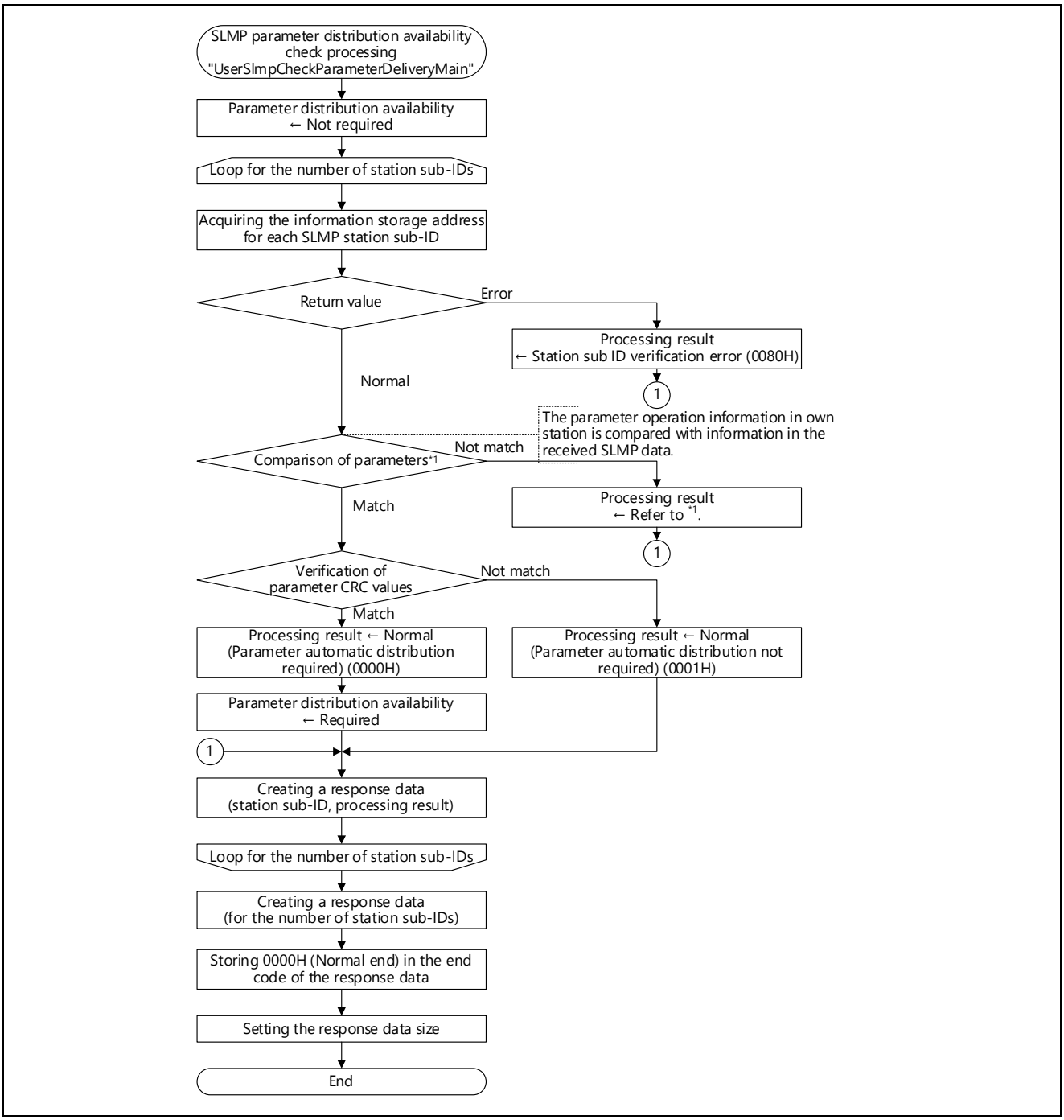


Figure 5.55 Flowchart for SLMP Parameter Distribution Necessity Check Processing

Note 1. The data to be compared and the result of judgment are as follows:

- Header version: Header version matching error (0010H)
- Vendor code: Vendor code matching error (0020H)
- Type name code: Type name code matching error (0030H)
- Device version: Device version matching error (0040H)
- IP address: IP address checking error (0060H)

5.7.4 SLMP Restoration Start Notification Request Command Reception Processing

This function performs pre-processing for restoration.
A frame of the LMT type is received through processing in response to reception of this command. If the received frame is of a different type from the specified one, an error response is sent and received data are discarded. If an error in the data length after the subcommand section of the received LMT frame is detected, the R-IN32M4-CL3 driver sets C05CH to the end code and sends a response. Therefore, this command receive processing is not performed.

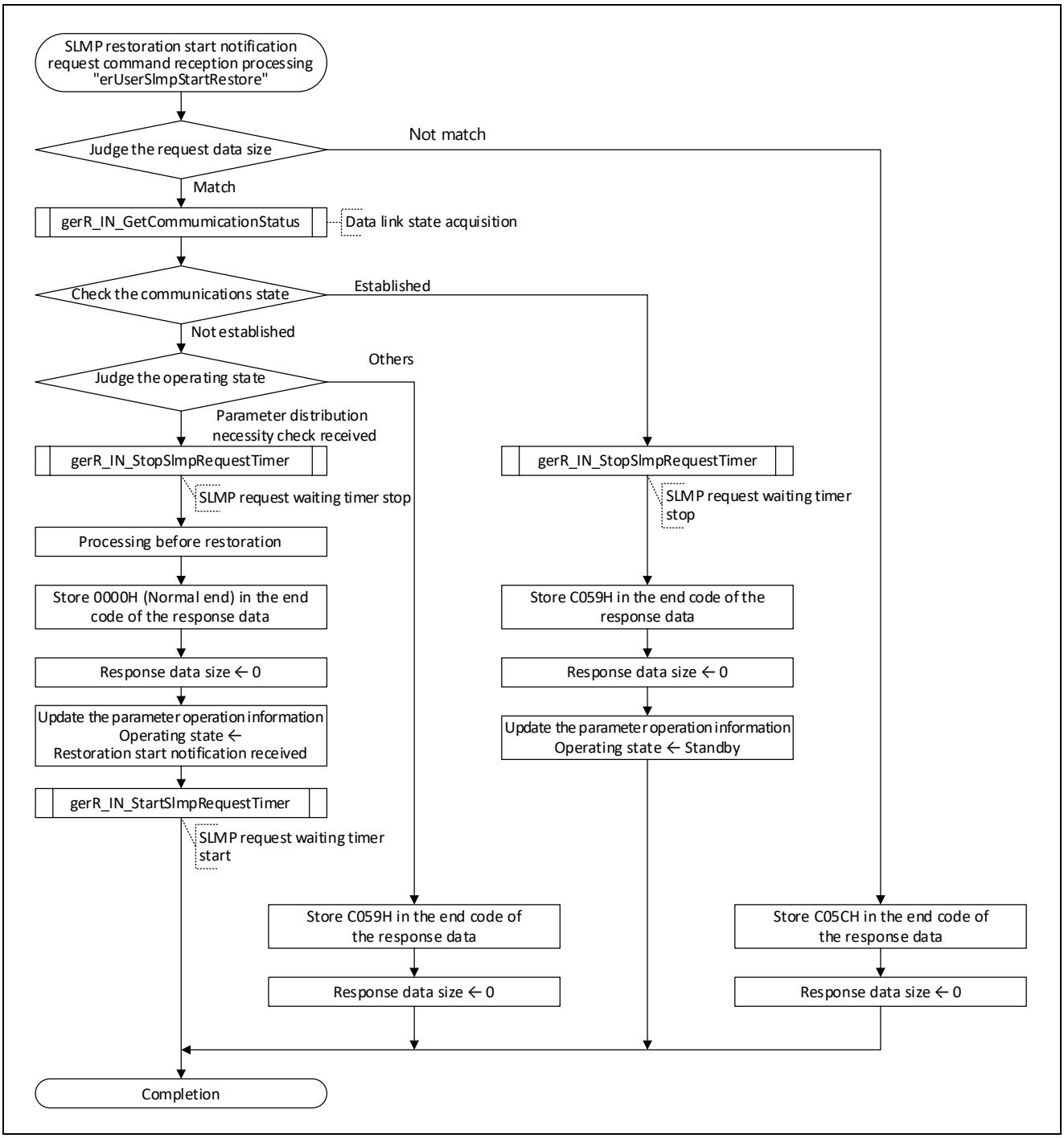


Figure 5.56 Flowchart for SLMP Restoration Start Notification Request Command Reception Processing

5.7.5 SLMP Restoration End Notification Request Command Reception Processing

This function performs post-processing for restoration.
A frame of the LMT type is received through processing in response to reception of this command. If the received frame is of a different type from the specified one, an error response is sent and received data are discarded. If an error in the data length after the subcommand section of the received LMT frame is detected, the R-IN32M4-CL3 driver sets C05CH to the end code and sends a response. Therefore, this command receive processing is not performed.

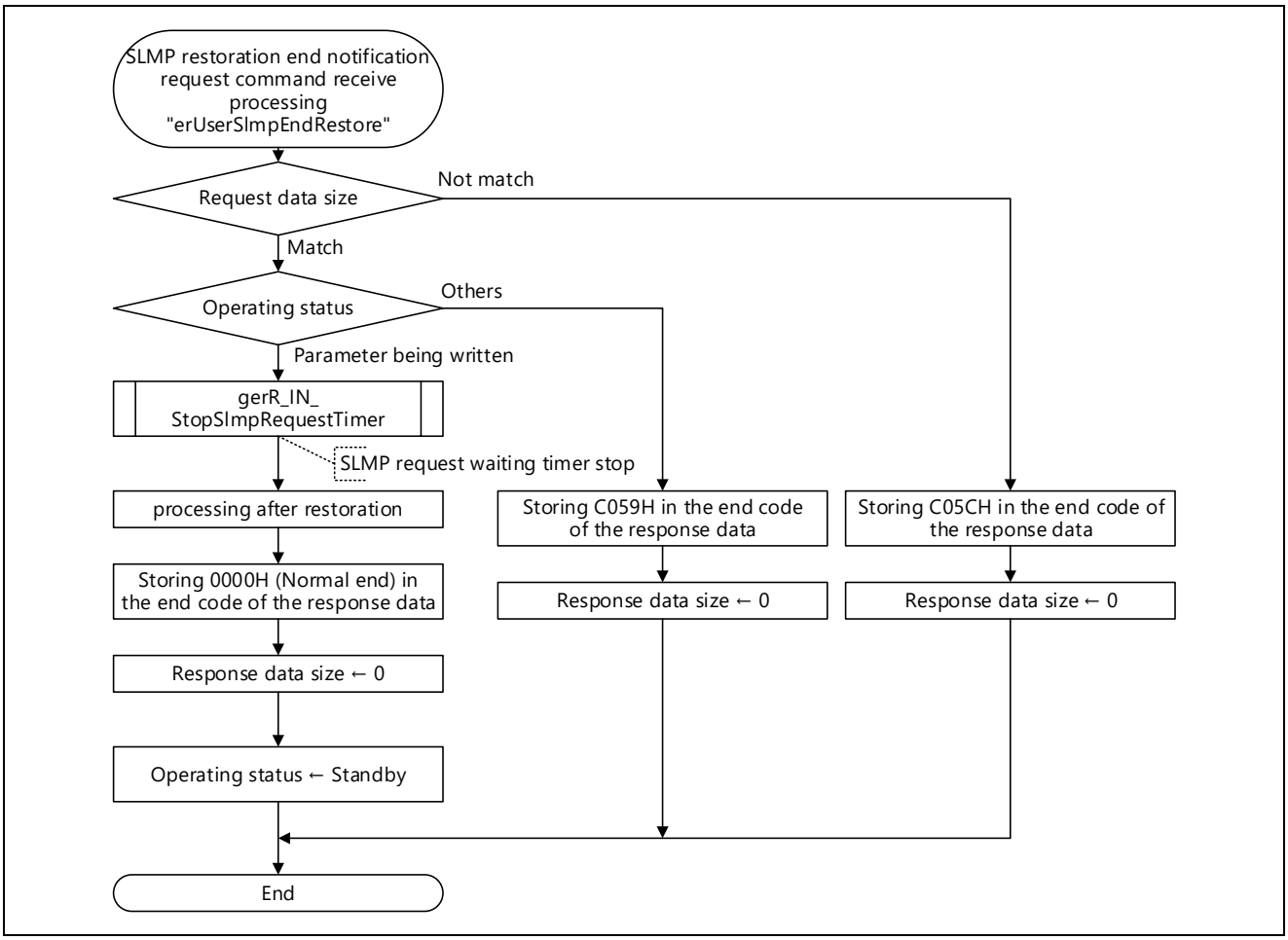


Figure 5.57 Flowchart for SLMP Restoration End Notification Request Command Receive Processing

5.7.6 SLMP Parameter Data Write Request Command Reception Processing

This function handles processing for writing parameters.
A frame of the LMT type is received through processing in response to reception of this command. If the received frame is of a different type from the specified one, an error response is sent and received data are discarded. If an error in the data length after the subcommand section of the received LMT frame is detected, the R-IN32M4-CL3 driver sets C05CH to the end code and sends a response. Therefore, this command receive processing is not performed.

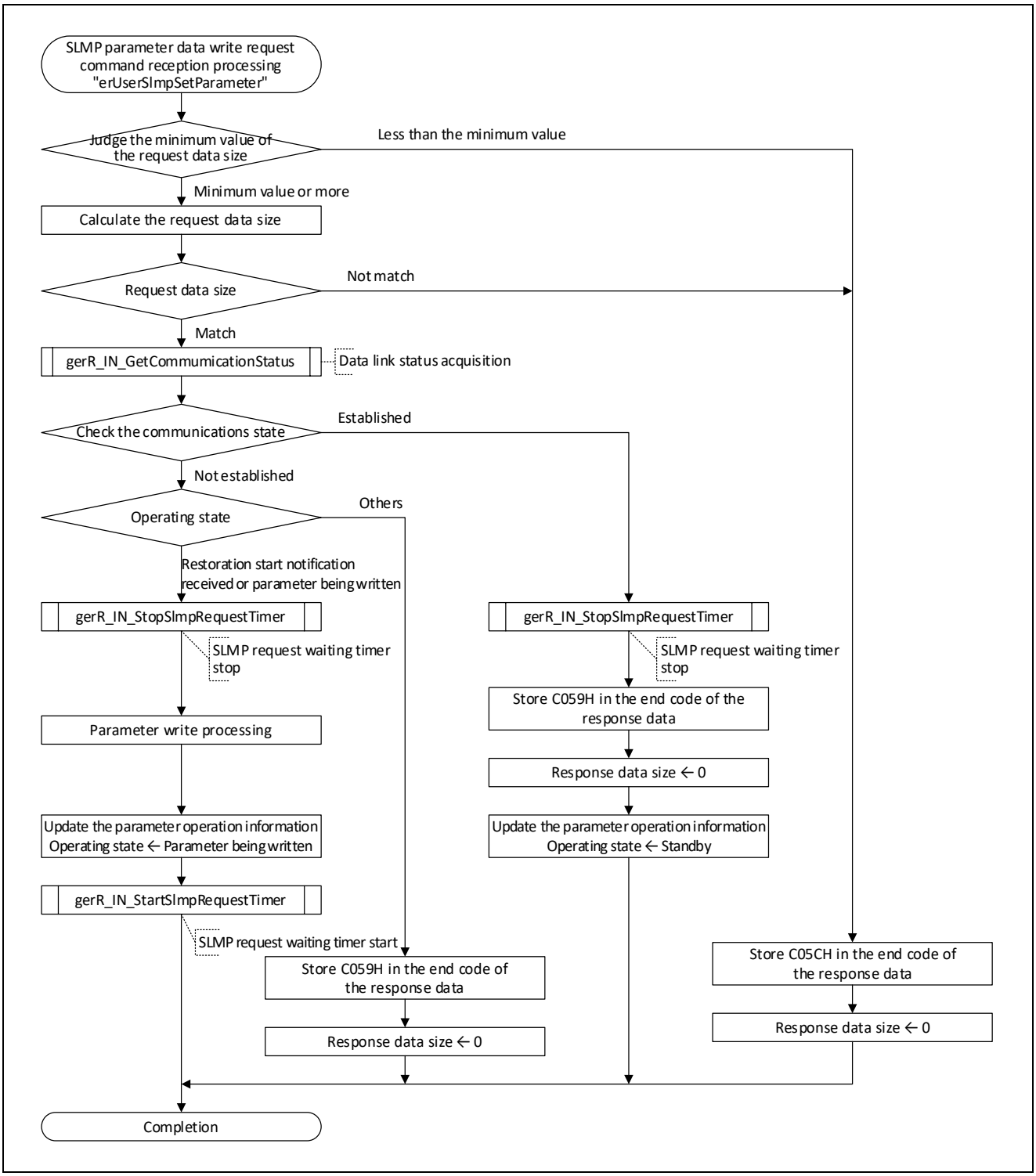


Figure 5.58 Flowchart for SLMP Parameter Data Write Request Command Reception Processing

5.7.7 SLMP Parameter Data Write Processing

This function stores received parameters in the parameter buffer of the station with the corresponding sub-ID. This processing is available when the amount of parameter data is no greater than 1438 bytes.

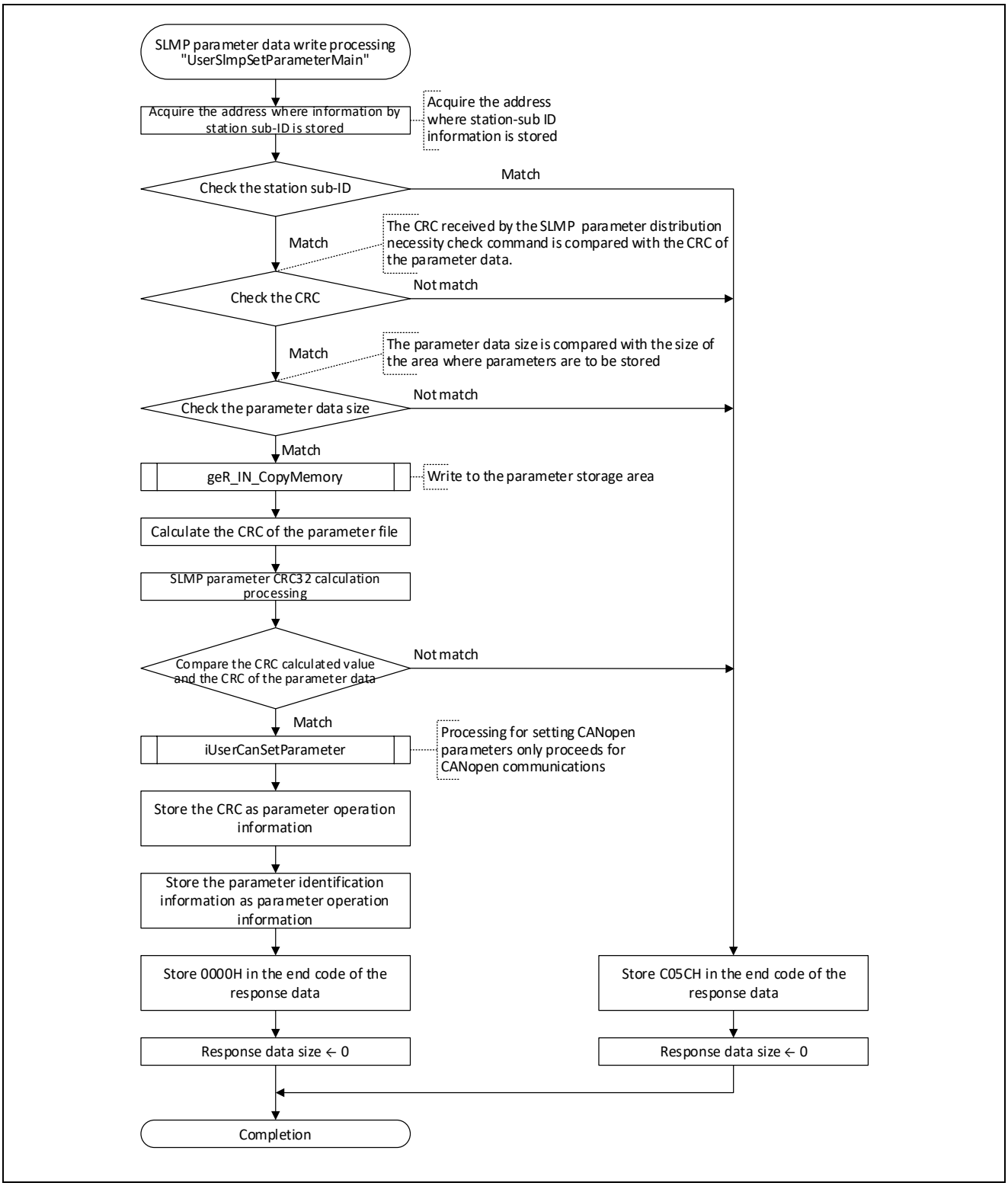


Figure 5.59 Flowchart for SLMP Parameter Data Write Processing

5.8 Details on Processing of User Programs (CC-Link IE TSN Device Parameter Setting)

This function stores the communication speed and CC-Link IE TSN Class of the own station to the non-volatile memory via SLMP and reflects them next time the power is turned on.

Use CC-Link IE TSN Configurator (CC-Link IE TSN configuration tool) as an SLMP client.

The following shows a setting image for setting the communication speed.

For details on how to operate CC-Link IE TSN Configurator, refer to the "CC-Link IE TSN Configuration Tool User's Manual" (BAP-C3009ENG-001).

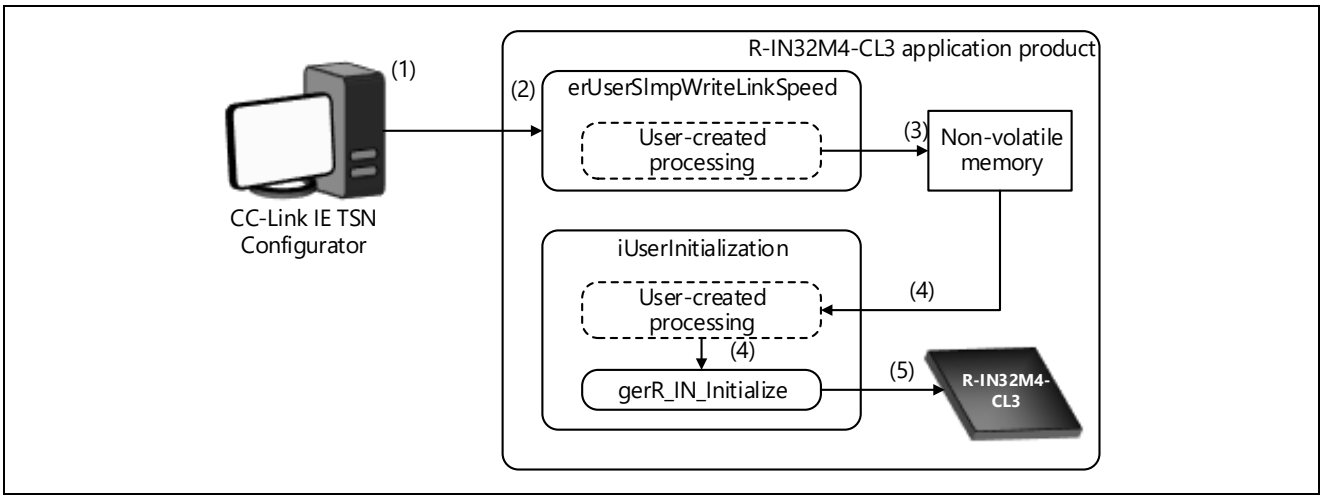


Figure 5.60 Communication speed setting image via SLMP

Procedure	contents
(1)	Connect a personal computer where CC-Link IE TSN Configurator is installed and the R-IN32M4-CL3 application product where the communication speed is set using an Ethernet cable. Select "Tool" →"Function setting of Remote station batch/individual execution function" on CC-Link IE TSN Configurator. Select "Communication speed write" from "Process to be executed", and execute it.
(2)	Send the SLMP command "3082H" (Writing the function setting (communication speed)) by CC-Link IE TSN Configurator. The R-IN32M4-CL3 application product performs receive processing in erUserSlmpWriteLinkSpeed.
(3)	In the user creation process in erUserSlmpWriteLinkSpeed, the communication speed setting value is written to the non-volatile memory. Turn off the power after writing is completed.
(4)	When the power is turned on next time, the data written in the non-volatile memory is read by the user creation process in iUserInitialization and set in the argument (stUnitInfo.ulLinkSpeed) of gerR_IN_Initialize.
(5)	gerR_IN_Initialize sets the communication speed to the built-in GbE-PHY of R-IN32M4-CL3. After the completion of gerR_IN_Initialize, the R-IN32M4-CL3 driver resets the built-in GbE-PHY and the communication speed change is applied.

-For erUserSlmpWriteLinkSpeed, see "5.8.4 SLMP function setting write (communication speed) request command receive processing".

-For iUserInitialization, see "5.3.1 Initialization Processing".

-For gerR_IN_Initialize, see "6.4.1(2) ".

5.8.1 SLMP connected device detection (extended) request command reception processing

This function responds to the connection device detection request.
This request command is sent when "Function setting of Remote station batch/individual execution function" or "Detection of connected/disconnected devices" is selected on CC-Link IE TSN Configurator.
The function receives LMT frames. When a frame other than above is received, the receive data is discarded.
If an error in the data length after the subcommand section of the received LMT frame is detected, the R-IN32M4-CL3 driver sets an error to the SLMP receive result in the request data (receive) included information.

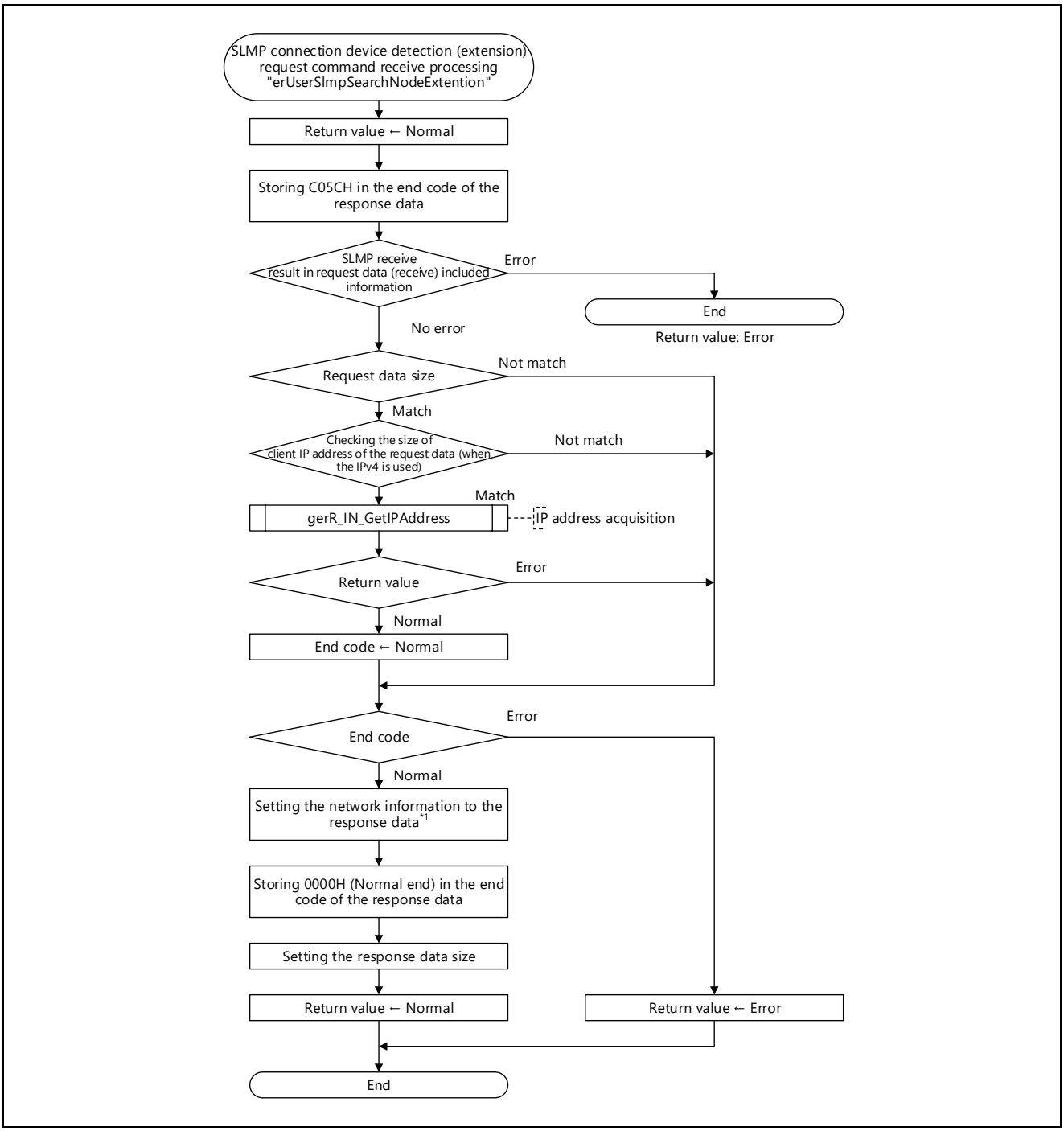


Figure 5.61 SLMP connection device detection (extended) request command reception processing flow diagram

Note 1. For details on the response data, see Table 5.23.

Table 5.23 USER_SLMP_COMMAND_SEARCH_NODE_EXTENTION_RESPONSE_T

No	Member		content
1	UCHAR	auchClientMACAddress[R_IN_MACADR_SZ]	Client MAC address (6 bytes)
2	UCHAR	uchClientIPAddressSize	Client IP address size
3	ULONG	ulClientIPAddress	Client IP address
4	UCHAR	auchServerMACAddress[R_IN_MACADR_SZ]	Server MAC address (6 bytes)
5	UCHAR	uchServerIPAddressSize	Server IP address size
6	ULONG	ulServerIPAddress	Server IP address
7	ULONG	ulServerSubnetmask	Server subnet mask
8	ULONG	ulServerDefaultGateway	Server default gateway IP address
9	UCHAR	uchServerHostNameSize	Server host name size (fixed to 0)
10	USHORT	usServerVendorCode	Server vendor code
11	ULONG	ulServerModelCode	Server model code
12	UCHAR	uchServerModelNameSize	Server model name size
13	UCHAR	auchServerModelName[R_IN_MODEL_NAME_LENGTH]	Server model name (up to 20 bytes)
14	USHORT	usServerMachineVersion	Server device version
15	UCHAR	uchTargetUnitIPAddressSize	Communication partner unit IP address size
16	ULONG	ulTargetUnitIPAddress	Communication partner unit IP address
17	USHORT	usTargetUnitPortNumber	Communication partner unit communication port number
18	USHORT	usServerStatus	Server status
19	USHORT	usServerPortNumber	Server communication port number
20	UCHAR	uchServerProtocol	Server communication protocol settings

5.8.2 SLMP function setting (support information acquisition) Request command reception processing

This function responds the function setting status information of the own station.
This request command is sent when "Function setting of Remote station batch/individual execution function" or "Detection of connected/disconnected devices" is selected on CC-Link IE TSN Configurator.

The function receives LMT frames. When a frame other than above is received, an error response is sent and the receive data is discarded. If an error in the data length after the subcommand section of the received LMT frame is detected, the R-IN32M4-CL3 driver sets C05CH to the end code and sends a response. Therefore, this command receive processing is not performed.

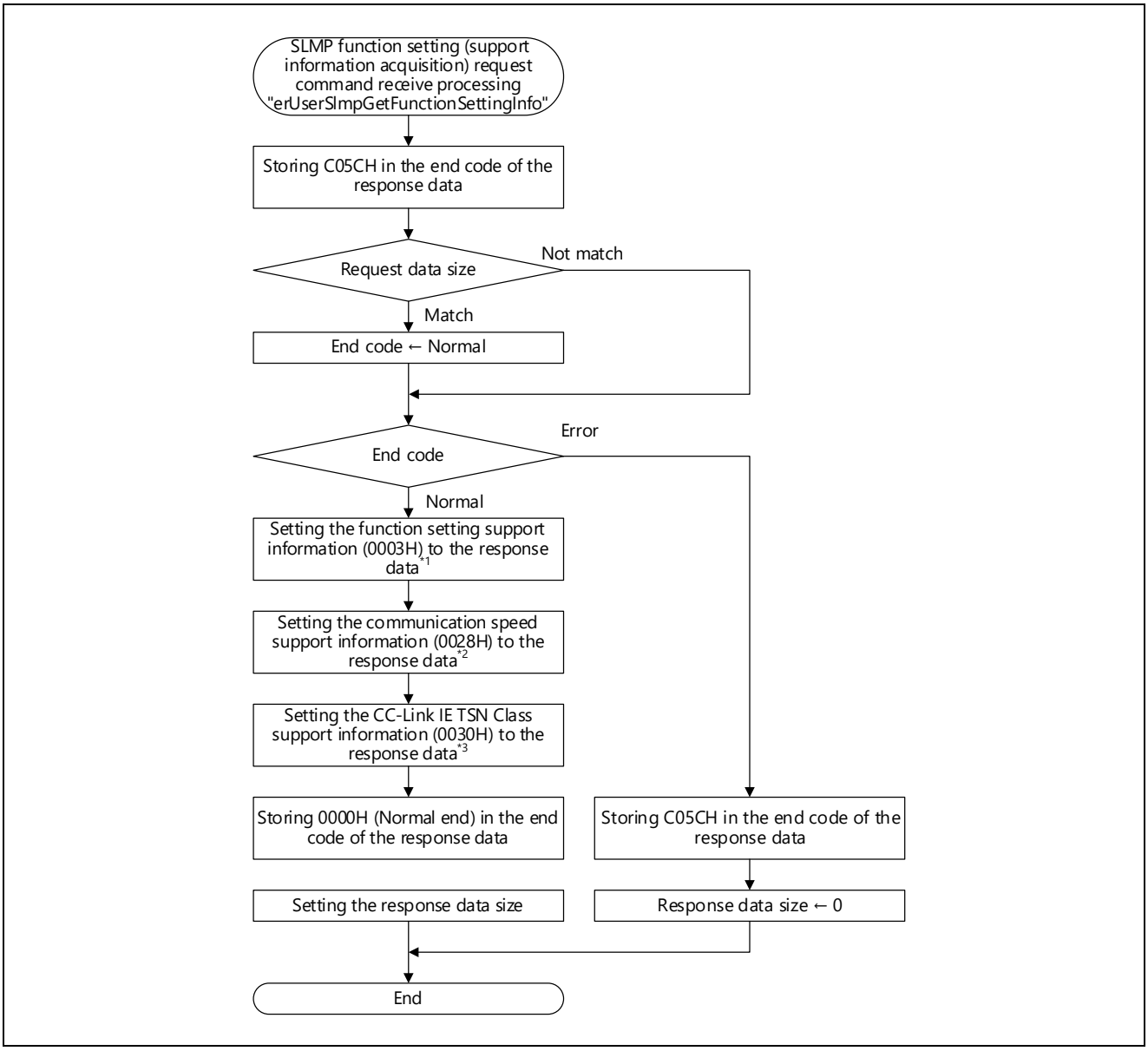


Figure 5.62 SLMP function setting (support information acquisition) request command reception processing flow diagram

Note 1. Refer to "Table 5.24" for details.

Note 2. Refer to "Table 5.25" for details.

Note 3. Refer to "Table 5.26" for details.

Table 5.24 USER_SLMP_FUNCTION_SETTING_INFO

Bit	Item	Content	Initial value
15-2	Reserve	0 fixed	0
1	CC-Link IE TSN Class	0: Not supported / 1: Supported	1b
0	Communication speed	0: Not supported / 1: Supported	1b

Table 5.25 USER_SLMP_FUNCTION_SETTING_INFO_LINKSPEED_100M_1G_FULL

Bit	Item	Content	Initial value
31-6	Reserve	0 fixed	0
5	1Gbps (full duplex)	0: Not supported / 1: Supported	1b
4	Reserve	0 fixed	0b
3	100Mbps (full duplex)	0: Not supported / 1: Supported	1b
2-0	Reserve	0 fixed	000b

Table 5.26 USER_SLMP_FUNCTION_SETTING_INFO_CCIETSN_CLASS

Bit	Item	Content	Initial value
15 to 6	Reserved	Fixed to 0	0
5	CC-Link IE TSN Class B ver.2.0	0: Not supported / 1: Supported	1b
4	CC-Link IE TSN Class A ver.2.0 (network time distribution supported)	0: Not supported / 1: Supported	1b
3	CC-Link IE TSN Class A ver.2.0 (network time distribution not supported)	0: Not supported / 1: Supported	0b
2	CC-Link IE TSN Class B ver.1.0	0: Not supported / 1: Supported	0b
1	Reserved	Fixed to 0	0b
0	CC-Link IE TSN Class A ver.1.0	0: Not supported / 1: Supported	0b

5.8.3 SLMP function setting read (communication speed) request command receive processing

This function responds the communication speed of the own station.
This request command is sent when "Communication speed read" of "Function setting of Remote station batch/individual execution function" is selected on CC-Link IE TSN Configurator.
The function receives LMT frames. When a frame other than above is received, an error response is sent and the receive data is discarded. If an error in the data length after the subcommand section of the received LMT frame is detected, the R-IN32M4-CL3 driver sets C05CH to the end code and sends a response. Therefore, this command receive processing is not performed.

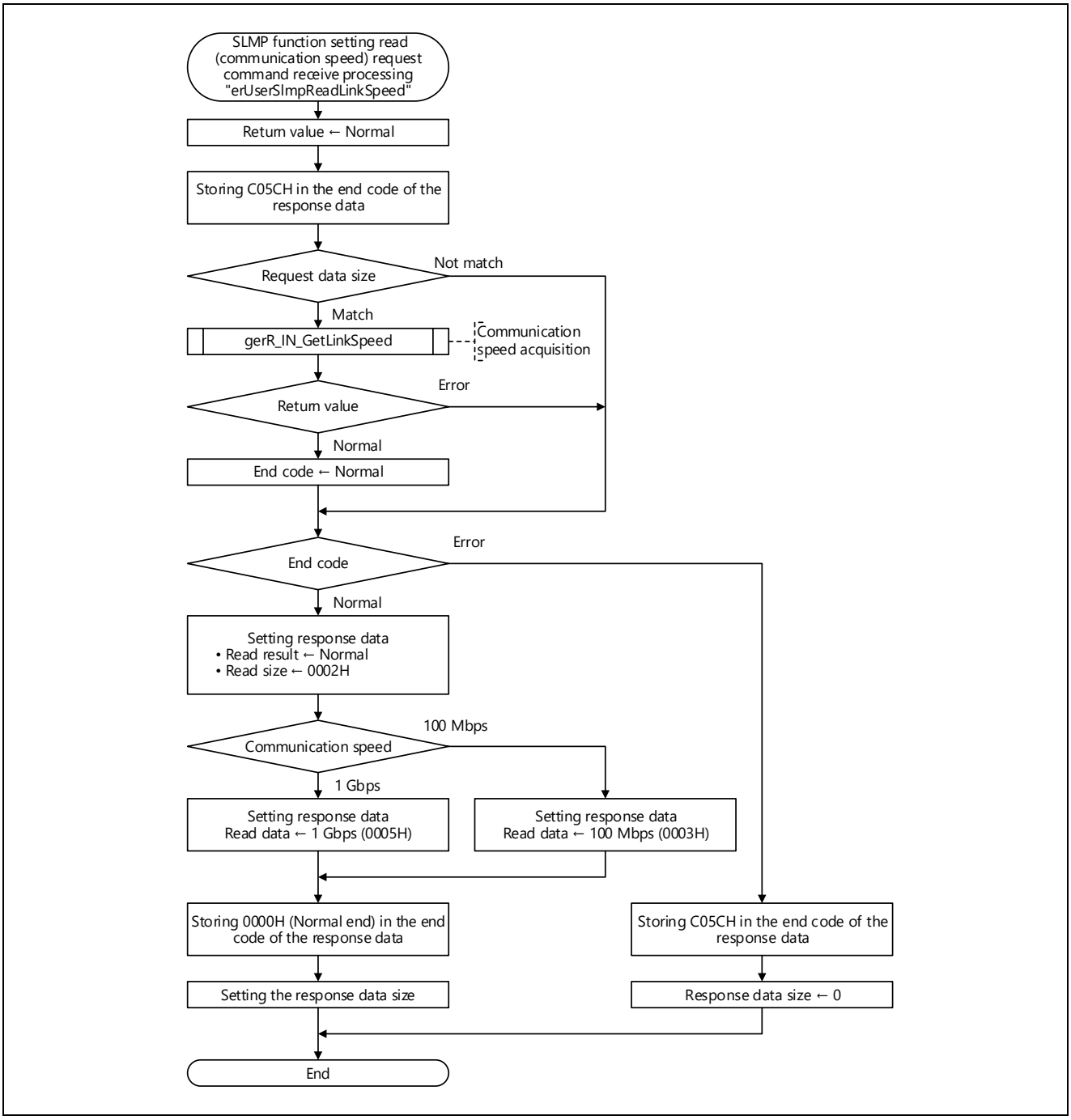


Figure 5.63 Flowchart for SLMP Function Setting Read (Communication Speed) Request Command Receive Processing

5.8.4 SLMP function setting write (communication speed) request command receive processing

This function acquires the specified communication speed setting from the SLMP frame and stores it to the non-volatile memory.
This request command is sent when "Communication speed write" of "Function setting of Remote station batch/individual execution function" is selected on CC-Link IE TSN Configurator.
The function receives LMT frames. When a frame other than above is received, an error response is sent and the receive data is discarded. If an error in the data length after the subcommand section of the received LMT frame is detected, the R-IN32M4-CL3 driver sets C05CH to the end code and sends a response. Therefore, this command receive processing is not performed.

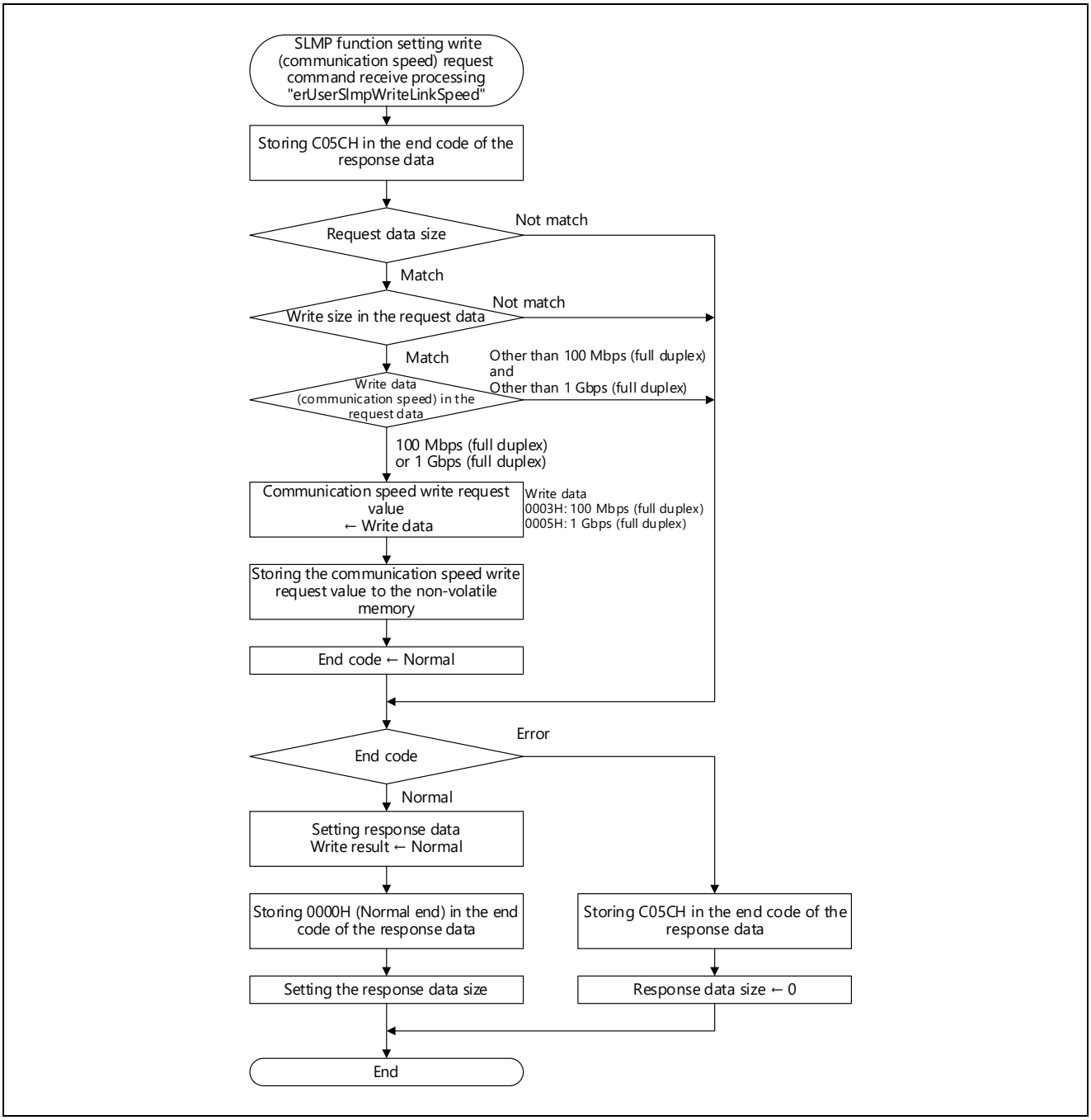


Figure 5.64 Flowchart for SLMP Function Setting Write (Communication Speed) Request Command Receive Processing

5.8.5 SLMP function setting read (CC-Link IE TSN Class) request command receive processing

This function responds the CC-Link IE TSN Class of the own station.
This request command is sent when "Authentication Class read" of "Function setting of Remote station batch/individual execution function" is selected on CC-Link IE TSN Configurator.
The function receives LMT frames. When a frame other than above is received, an error response is sent and the receive data is discarded. If an error in the data length after the subcommand section of the received LMT frame is detected, the R-IN32M4-CL3 driver sets C05CH to the end code and sends a response. Therefore, this command receive processing is not performed.

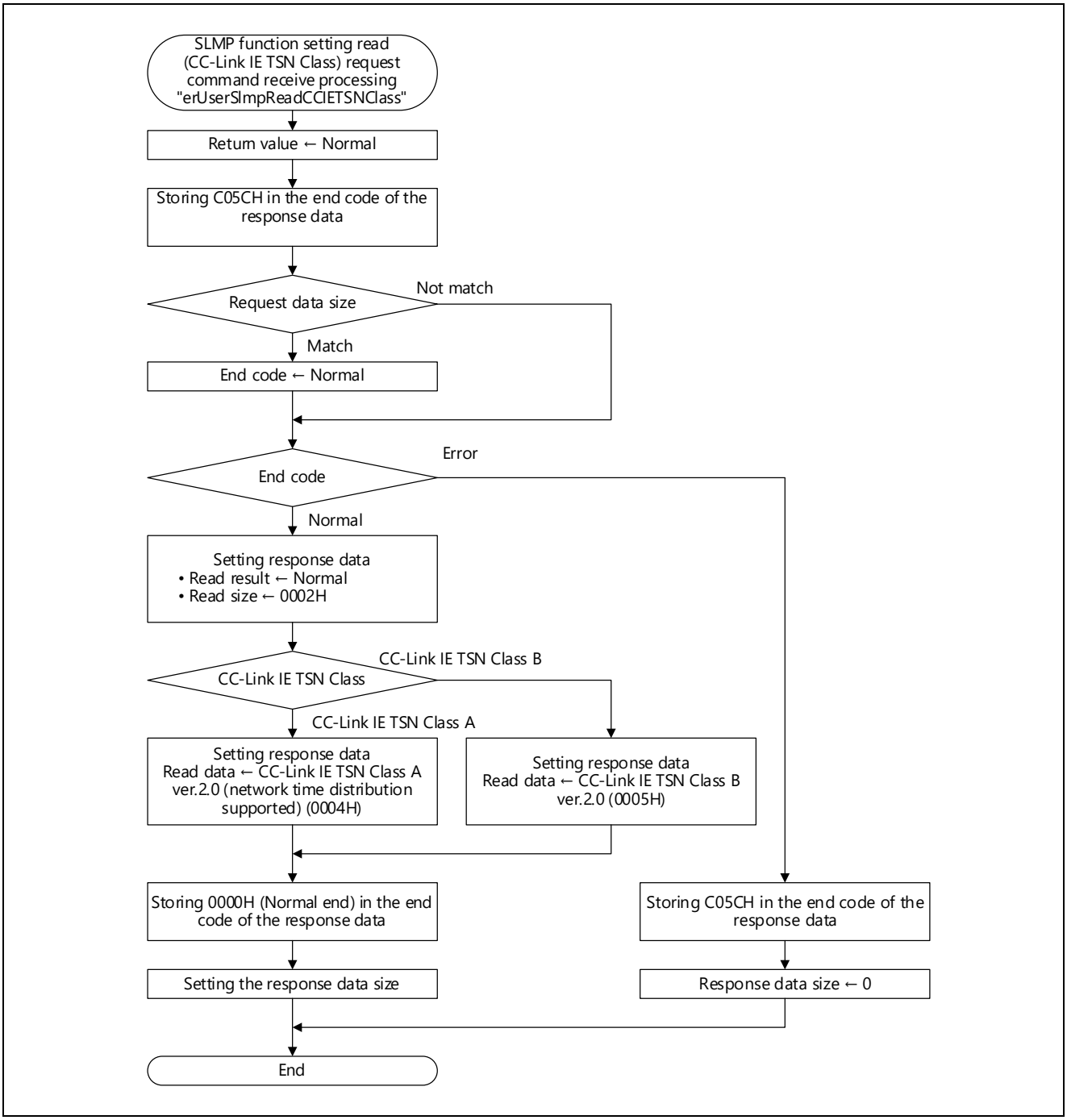


Figure 5.65 Flowchart for SLMP Function Setting Read (CC-Link IE TSN Class) Request Command Receive Processing

5.8.6 SLMP function setting write (CC-Link IE TSN Class) request command receive processing

This function acquires the specified CC-Link IE TSN Class from the SLMP frame and stores it to the non-volatile memory.
This request command is sent when "Authentication Class write" of "Function setting of Remote station batch/individual execution function" is selected on CC-Link IE TSN Configurator.
The function receives LMT frames. When a frame other than above is received, an error response is sent and the receive data is discarded. If an error in the data length after the subcommand section of the received LMT frame is detected, the R-IN32M4-CL3 driver sets C05CH to the end code and sends a response. Therefore, this command receive processing is not performed.

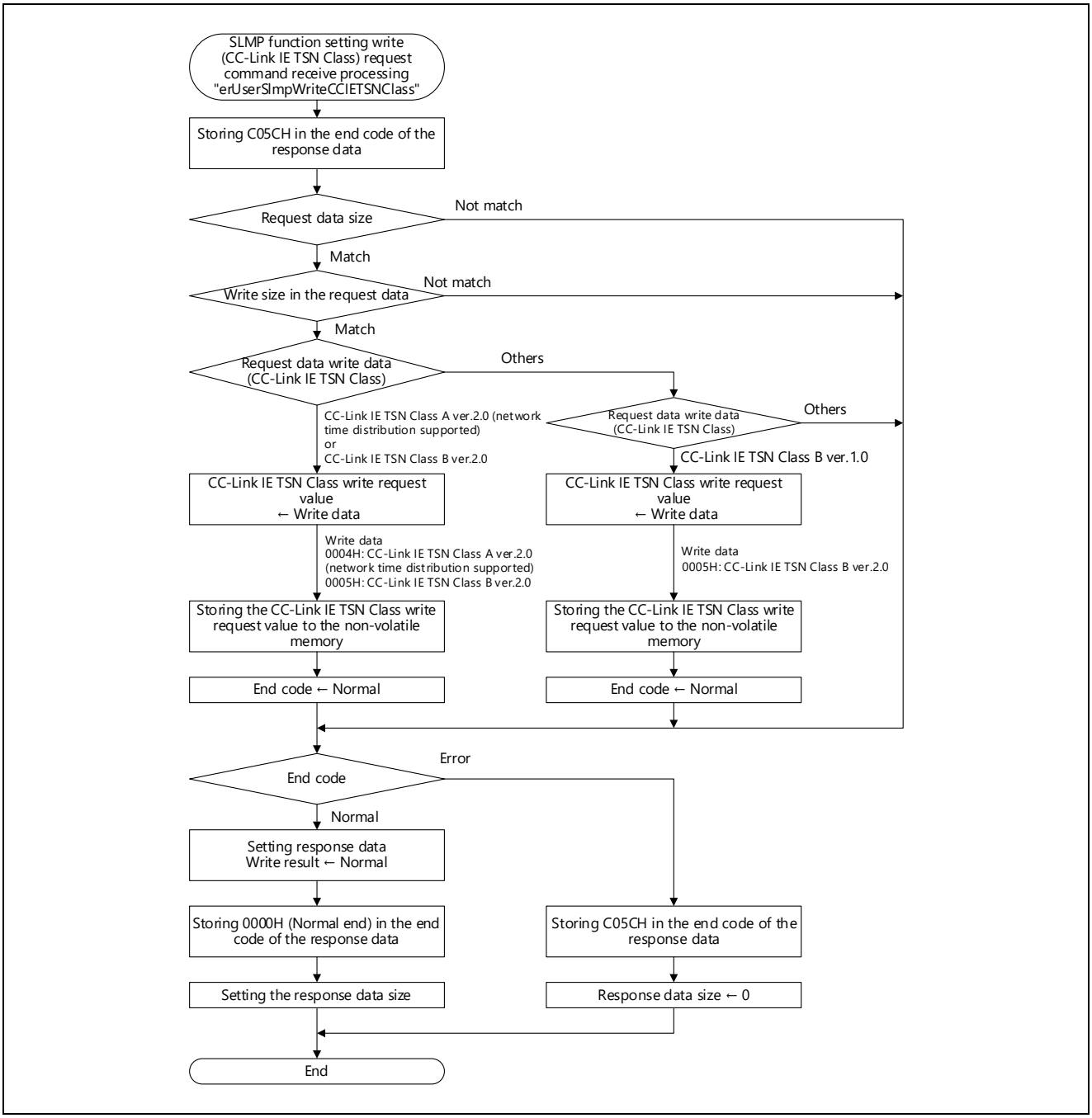


Figure 5.66 Flowchart for SLMP Function Setting Write (CC-Link IE TSN Class) Request Command Receive Processing

5.9 User Program Details (CANopen Communications Related)

(1) Object Dictionary

Since the object dictionary has data that are dependent on the device, such as parameters and command values, it is defined by the user and set in R-IN32M4-CL3. The object dictionary specifies the target for access by index or sub-index. For details of its structure, refer to the CC-Link IE TSN Specification (Overview) published by the CC-Link Partner Association.

Table 5.27 Structure of the Object Dictionary

Index	Object Dictionary Area	Description
0000H to 0FFFH	Data type area	Defines the data type.
1000H to 1FFFH	Communication profile area	Defines the communications-specific objects.
2000H to 5FFFH	Manufacturer specific profile area	Defines the manufacture-specific objects. Implementation in the object dictionary by the user is optional.
6000H to FFFFH	Device profile area	Defines the objects defined by the device profile. Implementation in the object dictionary by the user is optional.

(2) Data Structure of the Object Dictionary

The object dictionary is an array of R_IN_CAN_OD_T type structures, with the number of such elements corresponding to the required number of indices. Sort the arrays in ascending order of indices. The settings in the object dictionary should be the same as those defined by the profile.

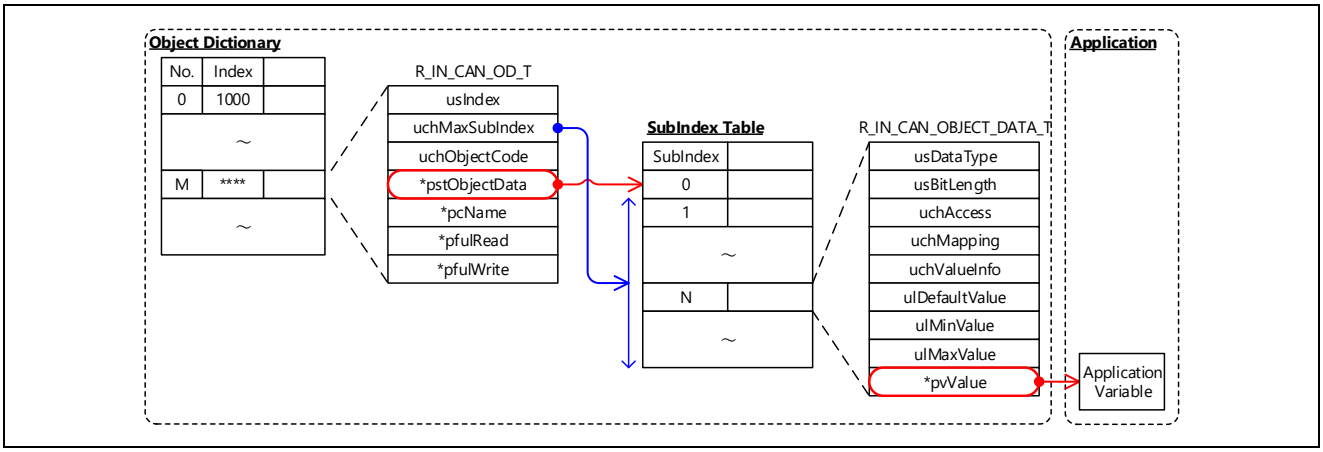


Figure 5.67 Schematic View of the Data Structure of the Object Dictionary

Table 5.28 Overview of the R_IN_CAN_OD_T Structure

No	Member		Description
1	USHORT	usIndex	Index
2	UCHAR	uchMaxSubIndex	Maximum sub-index
3	UCHAR	uchObjectCode	Object code
4	const R_IN_CAN_OBJECT_DATA_T	*pstObjectData	Object data
5	const CHAR	*pcName	Start address of the name array
6	ULONG	(*pfulRead) (USHORT usIndex, UCHAR uchSubindex, ULONG ulSize, UCHAR *puchData)	Read function
7	ULONG	(*pfulWrite) (USHORT usIndex, UCHAR uchSubindex, ULONG ulSize, UCHAR *puchData)	Write function

For details of the R_IN_CAN_OD_T structure, refer to Table 6.40, R_IN_CAN_OD_T List.

(3) Definitions of the Object Dictionary on PDOs

When defining the objects related to process data objects (PDOs) in the object dictionary, the following specifications must be satisfied.

Table 5.29 Definitions of the Object Dictionary on PDOs

<table>
<tr><th>No</th><th>Target Index</th><th>Item</th><th>Overview</th></tr>
<tr><td>1</td><td>All</td><td>String type PDO mapping</td><td>Objects of string type (R_IN_CAN_VISIBLESTRING) cannot be mapped to RPDOs and TPDOs.</td></tr>
<tr><td>2</td><td>1C00H to 1CFFH</td><td>Available indices</td><td>Use the indices in order from 1C00H.
Set a value that is at least equal to the number of indices to be used as definition No.1 below.
(Index: 1C00H to 1C01H)
<table>
<tr><th>No</th><th>Definition</th><th>Setting</th></tr>
<tr><td>1</td><td>R_IN_CAN_PDO_CONFIG_OBJECT_NUM</td><td>2</td></tr>
</table></td></tr>
<tr><td>3</td><td>1600H to 17FFH</td><td>Available indices</td><td>Use the indices in order from 1600H.
Set a value that is at least equal to the number of indices to be used as definition No.1 below.
Set the maximum sub-index value to be used as definition No.2 below.
(Index: 1600H to 1601H; Sub-index: 0 to 16)
<table>
<tr><th>No</th><th>Definition</th><th>Setting</th></tr>
<tr><td>1</td><td>R_IN_CAN_RPDO_MAPPING_OBJECT_NUM</td><td>2</td></tr>
<tr><td>2</td><td>R_IN_CAN_RPDO_APPLICATION_OBJECT_NUM</td><td>16</td></tr>
</table></td></tr>
<tr><td>4</td><td>1A00H to 1BFFH</td><td>Available indices</td><td>Use the indices in order from 1A00H.
Set a value that is at least equal to the number of indices to be used as definition No.1 below.
Set the maximum sub-index value to be used as definition No.2 below.
(Index: 1A00H to 1A01H; Sub-index: 0 to 16)
<table>
<tr><th>No</th><th>Definition</th><th>Setting</th></tr>
<tr><td>1</td><td>R_IN_CAN_TPDO_MAPPING_OBJECT_NUM</td><td>2</td></tr>
<tr><td>2</td><td>R_IN_CAN_TPDO_APPLICATION_OBJECT_NUM</td><td>16</td></tr>
</table></td></tr>
<tr><td>5</td><td>1600H to 17FFH
1A00H to 1BFFH</td><td>Bit length setting</td><td>For "Bit Length" of a PDO mapping object setting, set the bit length corresponding to the data type.
<table>
<tr><th>Data Type</th><th>Bit Length</th></tr>
<tr><td>R_IN_CAN_INTEGER8</td><td>8</td></tr>
<tr><td>R_IN_CAN_INTEGER16</td><td>16</td></tr>
<tr><td>R_IN_CAN_INTEGER32</td><td>32</td></tr>
<tr><td>R_IN_CAN_UNSIGNED8</td><td>8</td></tr>
<tr><td>R_IN_CAN_UNSIGNED16</td><td>16</td></tr>
<tr><td>R_IN_CAN_UNSIGNED32</td><td>32</td></tr>
<tr><td>R_IN_CAN_VISIBLESTRING</td><td>PDOs cannot be mapped.</td></tr>
</table></td></tr>
<tr><td>6</td><td>1600H to 17FFH
1A00H to 1BFFH</td><td>Maximum number of PDO mapping objects</td><td>With GX Works3, the maximum number of PDO mapping objects are 64 for RPDOs and TPDOs in total.</td></tr>
<tr><td>7</td><td>1600H to 17FFH
1A00H to 1BFFH</td><td>Padding object</td><td>With GX Works3, when a PDO object[*1] having one byte of data is mapped, one byte of a padding object will be inserted.
Considering the above, define the maximum number of PDO mapping objects in R_IN_CAN_RPDO_APPLICATION_OBJECT_NUM and R_IN_CAN_TPDO_APPLICATION_OBJECT_NUM.</td></tr>
</table>

Note 1. A PDO object with data type ("usDataType") of R_IN_CAN_INTEGER8 (8 bits) or R_IN_CAN_UNSIGNED8 (8 bits). The data size of a link device is two bytes. When a PDO object having one byte of data is mapped to a link device, another one byte needs to be padded.

(4) Extension module(s) used in CANopen communications

If a R-IN32M4-CL3 application product is, for example, a multi-axis servo amplifier, which consists of a main module (axis 1) that performs data communications and extension modules (axis 2 and later) that do not perform data communications, set the following for the extension modules.

Table 5.30 Setting Items for the Extension Modules

No	Item	Item
1	Number of axes	Set the number of axes to be used (1 to 8) in the macro definition "R_IN_CAN_MAX_ODTABLE_NUM" (R_IN32M4_CL3CanConst.h). Set the number equal to or bigger than the total number of main module and extension modules.
2	Address where PDO is stored	Define the address where PDO is stored in the R_IN32M4_CL3MemoryAddress.h file for each axis.*1 R_IN_MEMORY_ADDRESS_RWW_EXT1 to R_IN_MEMORY_ADDRESS_RWW_EXT7 R_IN_MEMORY_ADDRESS_RWR_EXT1 to R_IN_MEMORY_ADDRESS_RWR_EXT7 Define the addresses sequentially from axis 1.
3	User program	Define the object dictionaries for the number of axes to be used, and set them to the arguments of the gerR_IN_CanInit function (CANopen communication function initialization). Add processing for each axis as necessary in the SLMP command receive processing (Section 5.7.1 to Section 5.7.7) related to the slave station parameter automatic setting.
4	Axis specification	Specify an axis in NMT states and SDOs using the request destination station processor subnumber in an SLMP frame. Axes 1 to 8 are allocated to the request destination station processor subnumbers 0 to 7. Axis 1: Request destination station processor subnumber 0 Axis 2: Request destination station processor subnumber 1 Axis 3: Request destination station processor subnumber 2
5	Option information	Change the following definitions in accordance with the number of extension modules used. For details, refer to "Table 6.9 R_IN_UNITINFO_T List". ・ USER_OPTN_INFOFLG ・ USER_NUMBER_OF_OPTION ・ R_IN_OPTIONTABLE_ENTRY_SIZE

Note 1. The same address must be set in the object dictionary, sample code, and CSP+ file.

Table 5.31 Address Settings

Axis	PDO	Setting value in the object dictionary	Setting value in the sample code (R_IN32M4_CL3MemoryAddress.h)	Setting value in the CSP+ file (0x1234_RemoteSample_CAN_Base_1_en.CSPP)
1	RPDO	OD (Axis 1): Index 1C00H, subindex 4	R_IN_MEMORY_ADDRESS_RWW	PDOConfigMemoryAddress1
	TPDO	OD (Axis 1): Index 1C01H, subindex 4	R_IN_MEMORY_ADDRESS_RWR	PDOConfigMemoryAddress2
2	RPDO	OD (Axis 2): Index 1C00H, subindex 4	R_IN_MEMORY_ADDRESS_RWW_EXT1	EXT1_RPDOConfigMemoryAddress
	TPDO	OD (Axis 2): Index 1C01H, subindex 4	R_IN_MEMORY_ADDRESS_RWR_EXT1	EXT1_TPDOConfigMemoryAddress
…	…	…	…	…
8	RPDO	OD (Axis 8): Index 1C00H, subindex 4	R_IN_MEMORY_ADDRESS_RWW_EXT7	EXT7_RPDOConfigMemoryAddress
	TPDO	OD (Axis 8): Index 1C01H, subindex 4	R_IN_MEMORY_ADDRESS_RWR_EXT7	EXT7_TPDOConfigMemoryAddress

5.9.1 CANopen Communications Function Initialization Processing

This function initializes the CANopen communications function.
The start address of the object dictionary and the number of elements is conveyed to the R-IN32M4-CL3 driver.

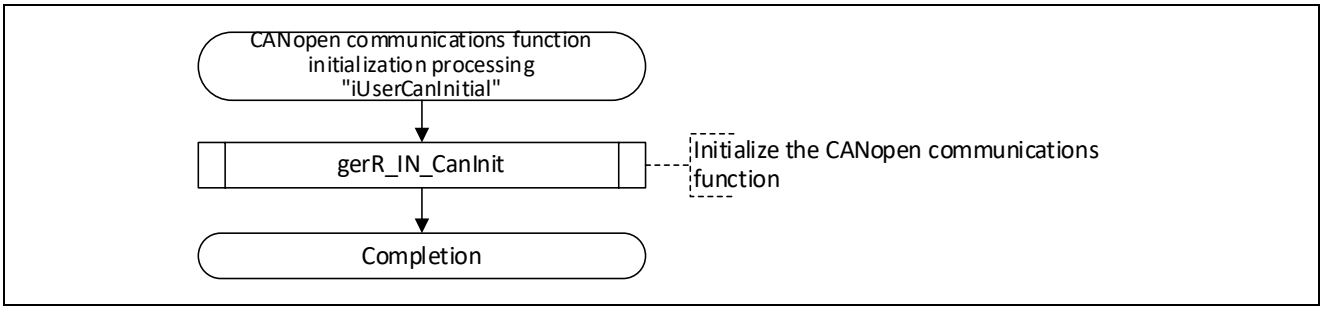


Figure 5.68 Flowchart for CANopen Communications Function Initialization Processing

(1) Initialization Sequence
The following shows the flow from initialization processing to the start of cyclic transfer.

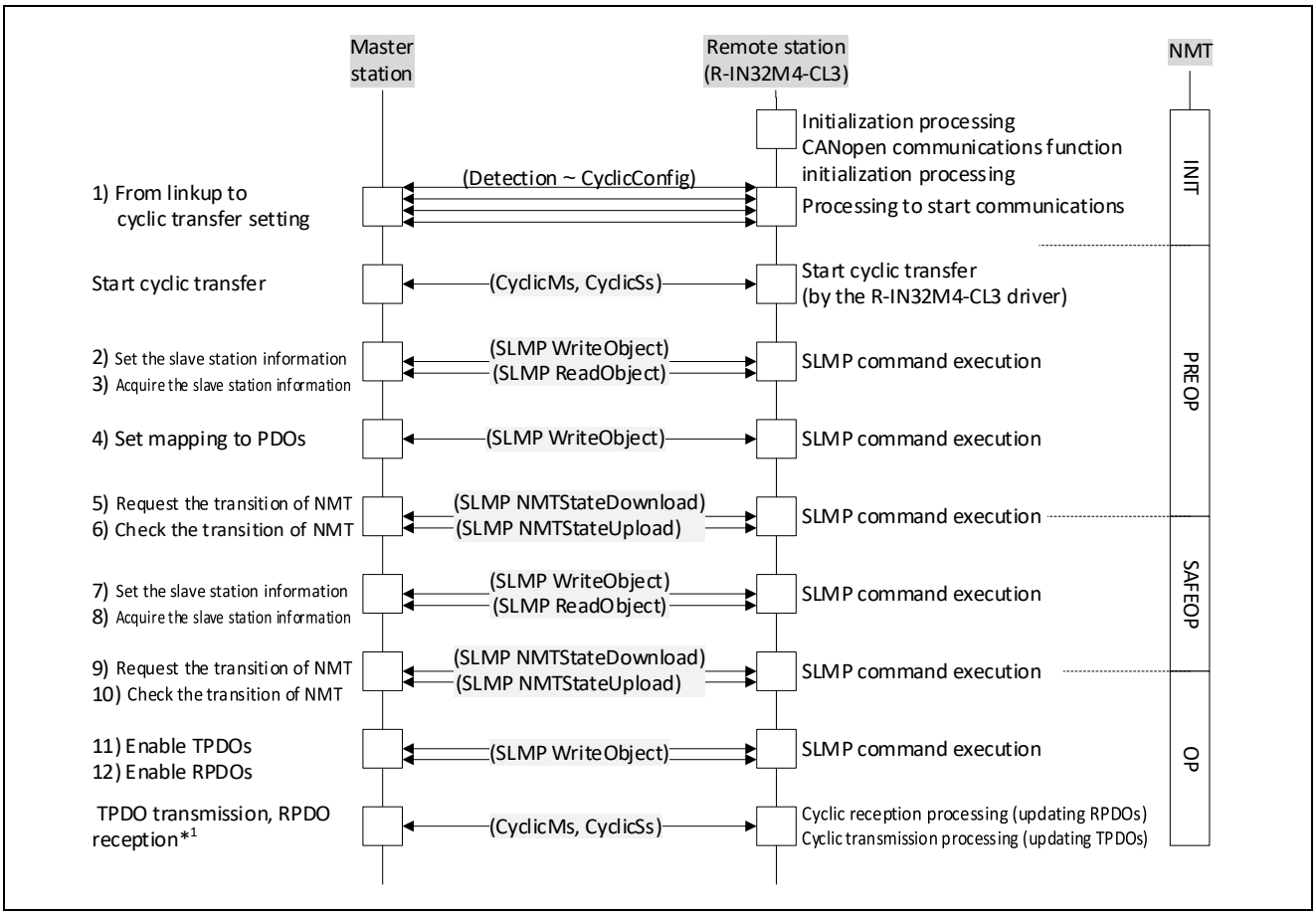


Figure 5.69 Initialization Sequence for CANopen Communications (Overview)

No	Member	Refer to
1	The master station makes settings to launch the network. The procedure is the same as that for CC-Link IE TSN communications from linkup to the cyclic transfer setting.	5.3.1 5.3.2
2	The master station writes the initial settings to the remote station (R-IN32M4-CL3) application as required.	5.9.5
3	The master station reads the setting information of the remote station (R-IN32M4-CL3) application as required.	5.9.4
4	The master station writes the allocation of PDOs set by the user to the remote station (R-IN32M4-CL3).	5.9.5
5	The master station changes the communications state (NMT) of the remote station (R-IN32M4-CL3) to SAFEOP.	5.9.9
6	The master station checks that the communications state (NMT) of the remote station (R-IN32M4-CL3) has been changed.	5.9.8
7	Same as in step 2.	—
8	Same as in step 3.	—
9	The master station changes the communications state (NMT) of the remote station (R-IN32M4-CL3) to OP.	5.9.9
10	The master station checks that the communications state (NMT) of the remote station (R-IN32M4-CL3) has been changed.	5.9.8
11	The master station enables TPDOs of the remote station (R-IN32M4-CL3). After the completion, the remote station (R-IN32M4-CL3) sends PDOs to the master station.	5.9.5
12	The master station enables RPDOs of the remote station (R-IN32M4-CL3). After the completion, the remote station (R-IN32M4-CL3) receives PDOs from the master station.	5.9.5

Note 1. The data in TPDOs and RPDOs will be enabled in the SAFEOP and OP states.

(2) NMT transition when extension modules are used

If a R-IN32M4-CL3 application product is, for example, a multi-axis servo amplifier, which consists of a main module (axis 1) that performs data communications and extension modules (axis 2 and later) that do not perform data communications, the NMT state transition needs to be processed for the number of axes. The NMT state transition of each axis is sequentially performed by the R-IN32M4-CL3 driver in the remote station. The following figure shows the NMT state transition image when the R-IN32M4-CL3 application product consists of one main module (axis 1) and two extension modules (axis 2 and axis 3).

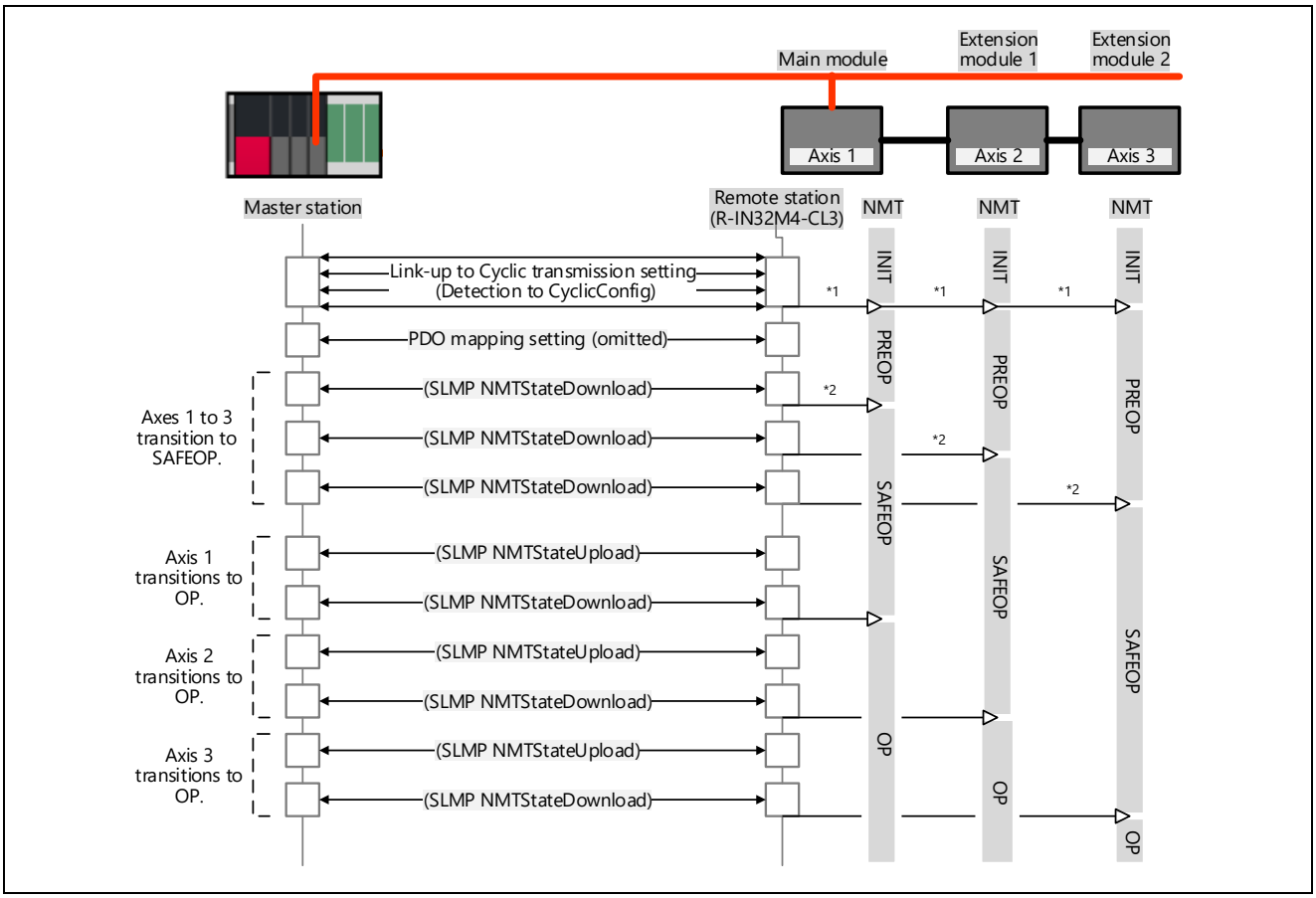


Figure 5.70 NMT State Transition Image (When Extension Modules are Used)

Specify an axis in NMT states and SDOs using the request destination station processor subnumber in an SLMP frame. In the figure above, the request destination station processor subnumbers are set as follows:
Axis 1: Request destination station processor subnumber 0
Axis 2: Request destination station processor subnumber 1
Axis 3: Request destination station processor subnumber 2

*1: When data link is established, the NMT state of all axes transitions from INIT to PREOP.
*2: When the NMT state of all axes is in the SAFEOP or OP state, the remote station receives RPDOs and sends TPDOs.

The PDO received in the PREOP state is mapped when the NMT state of the axes last transitions from PREOP to SAFEOP or from PREOP to OP state. If an error is detected during the PDO mapping, the PDO mapping is interrupted and the NMT state of the axis where an error was detected first transitions to PREOP state.

5.9.2 Cyclic Reception Processing (Updating RPDOs)

This function reflects received RWw data in the object dictionary in accord with the PDO mapping setting.

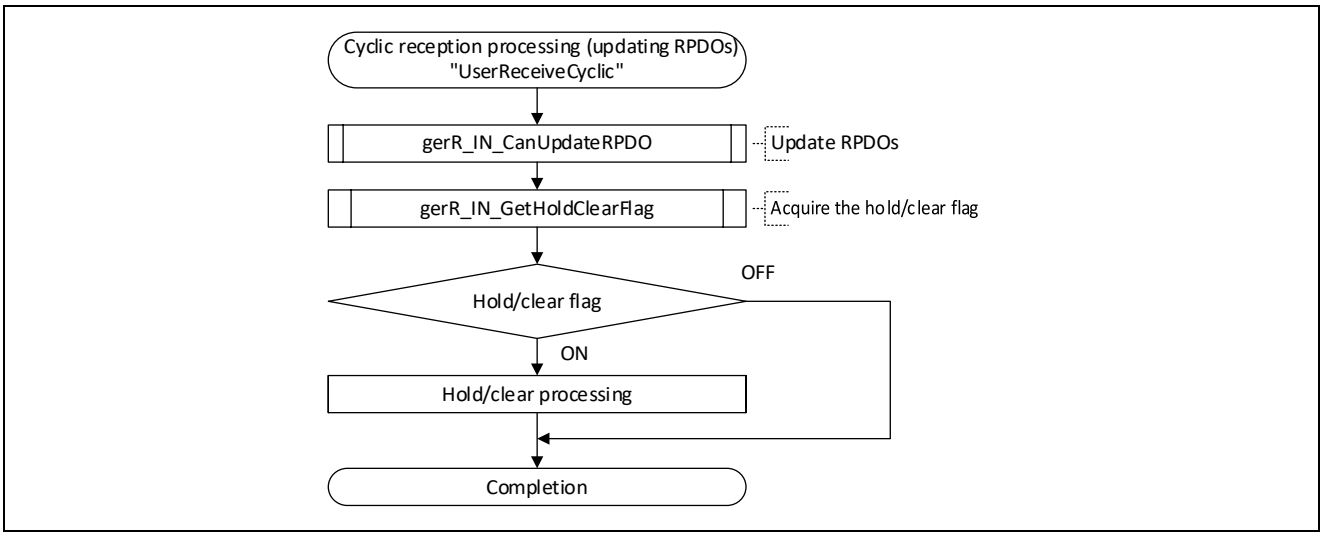


Figure 5.71 Flowchart for Cyclic Reception Processing (Updating RPDOs)

When RWw data is received multiple times under the condition that the processing interval of the fixed scan processing task (TSKID_PERIODIC) is longer than the cyclic transmission cycle, the R-IN32M4-CL3 driver acquires only the RWw data received immediately before the fixed scan processing task is performed and updates the data to the object dictionary.

Point

In CANopen communications, the maximum size of TPDOs/RPDOs that can be sent and received using the cyclic transmission function varies depending on application loads. The time required to update TPDOs/RPDOs differs depending on the data size of the TPDOs/RPDOs.

1) Size of data that can be sent/received
The following are the approximate maximum data sizes (bytes) when the fixed scan processing task is performed in 200 µs intervals.(For all data, 2-byte objects are assigned.)

User program	CC-Link IE TSN Class B	CC-Link IE TSN Class A	
	1 Gbps/100 Mbps	1 Gbps	100 Mbps
Cyclic send processing (TPDO update) and cyclic receive processing (RPDO update)	128	108	92

2) TPDO/RPDO update time
The following table lists the time required to update TPDOs/RPDOs when no external MCU is used.
(For all data, 2-byte objects are assigned.)

No	TPDO/RPDO data size (byte)	Time required to update TPDO/RPDO (µs)
1	128	171
2	192	226
3	256	281
4	384	391

5.9.3 Cyclic Transmission Processing (Updating TPDOs)

This function reflects the object dictionary data in RWr for transmission in accord with the PDO mapping setting.

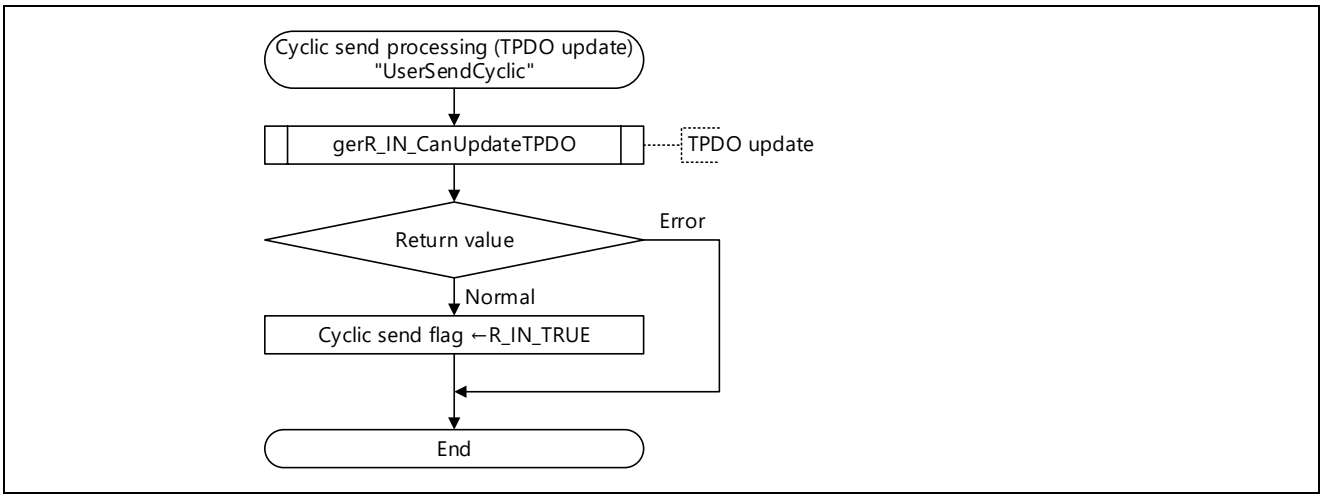


Figure 5.72 Flowchart for Cyclic Transmission Processing (Updating TPDOs)

When data in TPDO is set to RWr under the condition that the processing interval of the fixed scan processing task (TSKID_PERIODIC) is longer than the cyclic transmission cycle, the same data may be sent to the master station multiple times depending on the timing.

Point

In CANopen communications, the maximum size of TPDOs/RPDOs that can be sent and received using the cyclic transmission function varies depending on application loads. The time required to update TPDOs/RPDOs differs depending on the data size of the TPDOs/RPDOs.

1) Size of data that can be sent/received
The following are the approximate maximum data sizes (bytes) when the fixed scan processing task is performed in 200 µs intervals.
(For all data, 2-byte objects are assigned.)

User program	CC-Link IE TSN Class B	CC-Link IE TSN Class A	
	1 Gbps/100 Mbps	1 Gbps	100 Mbps
Cyclic send processing (TPDO update) and cyclic receive processing (RPDO update)	128	108	92

2) TPDO/RPDO update time
The following table lists the time required to update TPDOs/RPDOs when no external MCU is used.
(For all data, 2-byte objects are assigned.)

No	TPDO/RPDO data size (byte)	Time required to update TPDO/RPDO (µs)
1	128	171
2	192	226
3	256	281
4	384	391

5.9.4 SLMP ReadObject Request Command Reception Processing

This function receives SDO read requests from the master station and sends responses.
The data stored in the corresponding object of the object dictionary are read based on the specified index and sub-index.
The function receives LMT frames. When a frame other than above is received, an error response is sent and the receive data is discarded. If an error in the data length after the subcommand section of the received LMT frame is detected, the R-IN32M4-CL3 driver sets C05CH to the end code and sends a response. Therefore, this command receive processing is not performed.

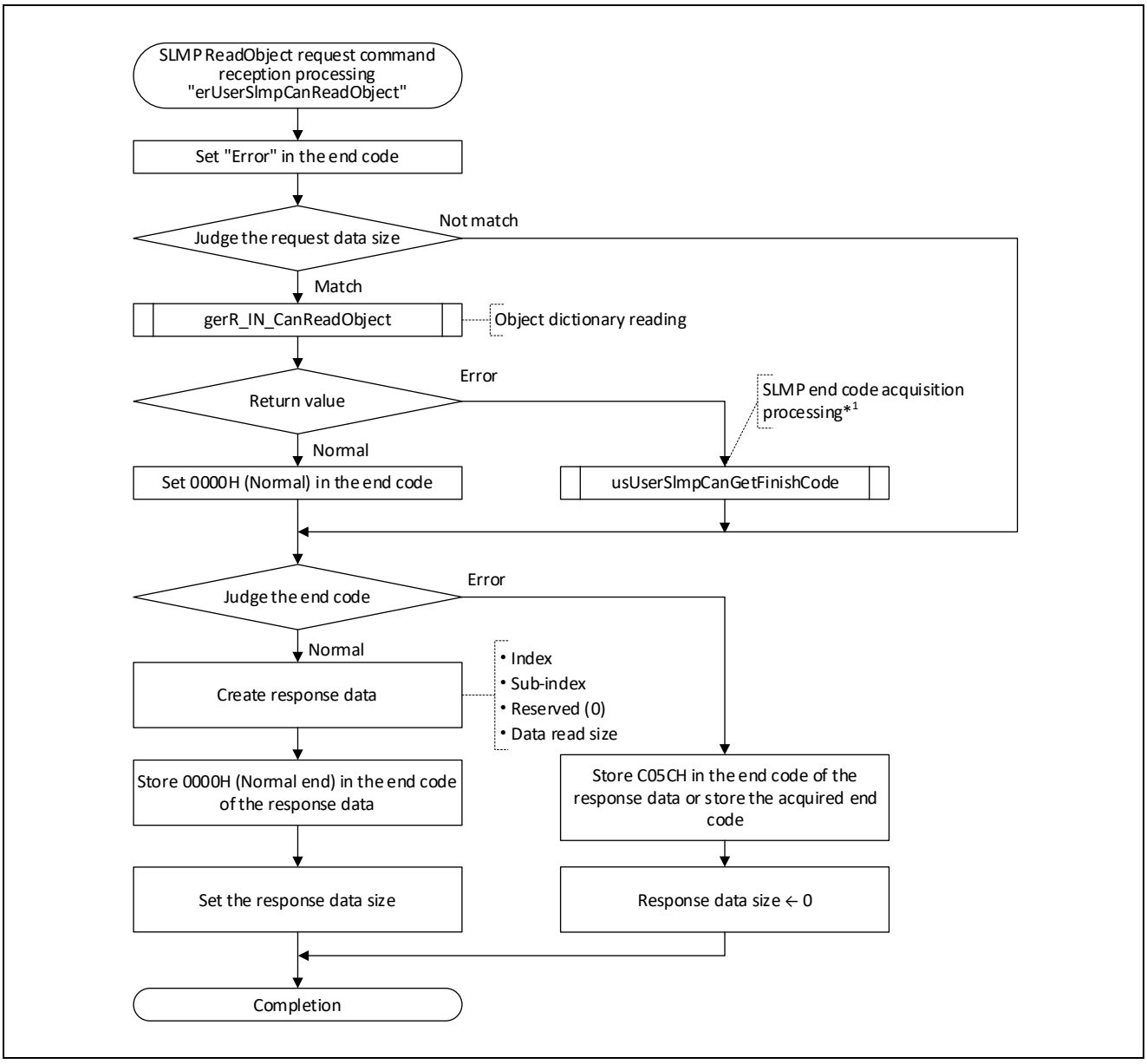


Figure 5.73 Flowchart for SLMP ReadObject Request Command Reception Processing

Note 1. Refer to "usUserSlmpCanGetFinishCode" (5.9.10, SLMP End Code Acquisition Processing).

5.9.5 SLMP WriteObject Request Command Reception Processing

This function receives SDO write requests from the master station and sends responses.
The specified data are written to the corresponding object in the object dictionary based on the specified index and sub-index.
The function receives LMT frames. When a frame other than above is received, an error response is sent and the receive data is discarded. If an error in the data length after the subcommand section of the received LMT frame is detected, the R-IN32M4-CL3 driver sets C05CH to the end code and sends a response. Therefore, this command receive processing is not performed.

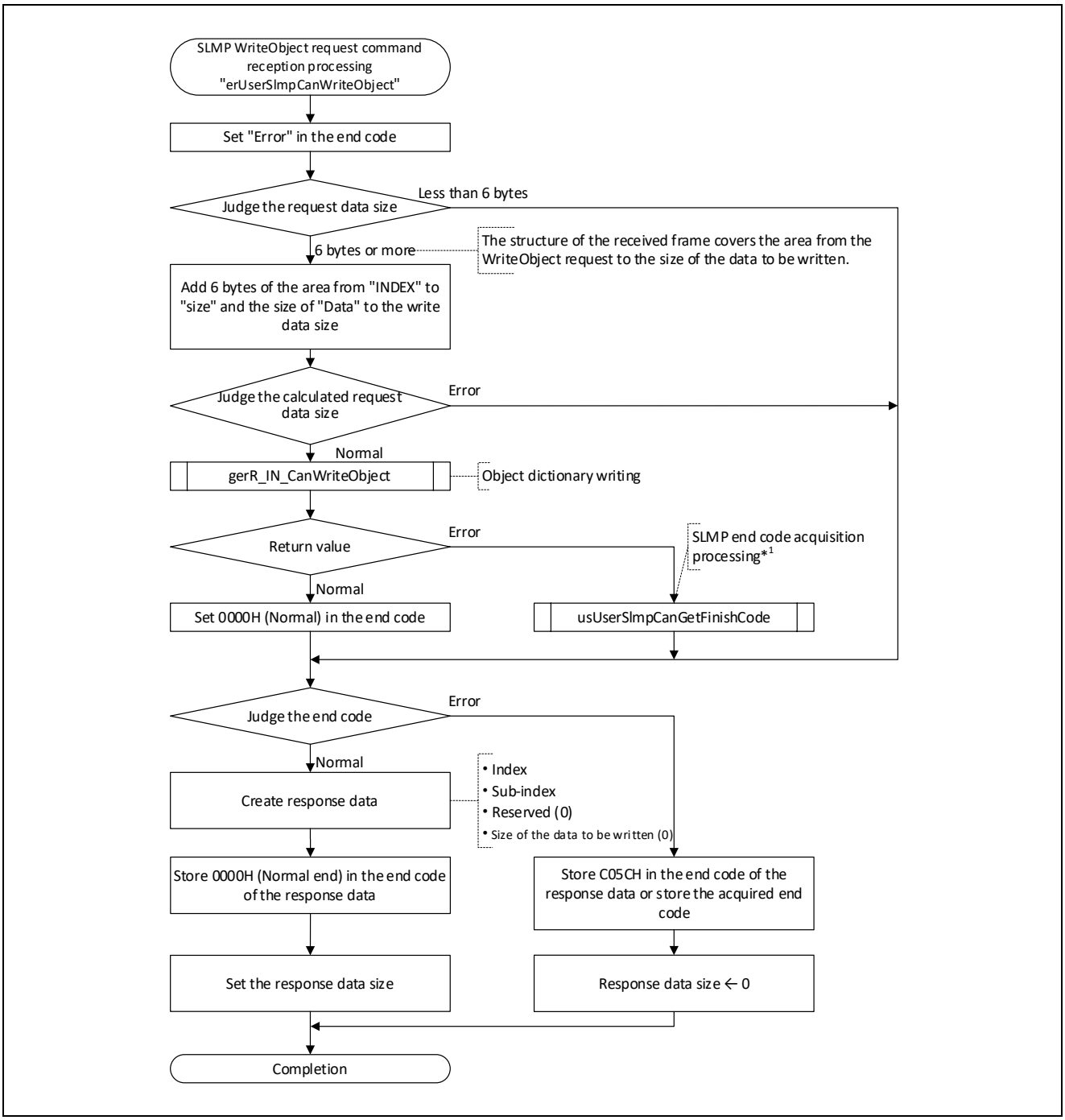


Figure 5.74 SLMP WriteObject Flowchart for SLMP WriteObject Request Command Reception Processing

Note 1. Refer to "usUserSlmpCanGetFinishCode" (5.9.10, SLMP End Code Acquisition Processing).

5.9.6 SLMP ObjectSubIDReadBlock Request Command Reception Processing

This function receives SDO read requests from the master station and sends responses.
The specified amount of data is read from the corresponding object in the object dictionary based on the specified index and sub-index.
The function receives LMT frames. When a frame other than above is received, an error response is sent and the receive data is discarded. If an error in the data length after the subcommand section of the received LMT frame is detected, the R-IN32M4-CL3 driver sets C05CH to the end code and sends a response. Therefore, this command receive processing is not performed.

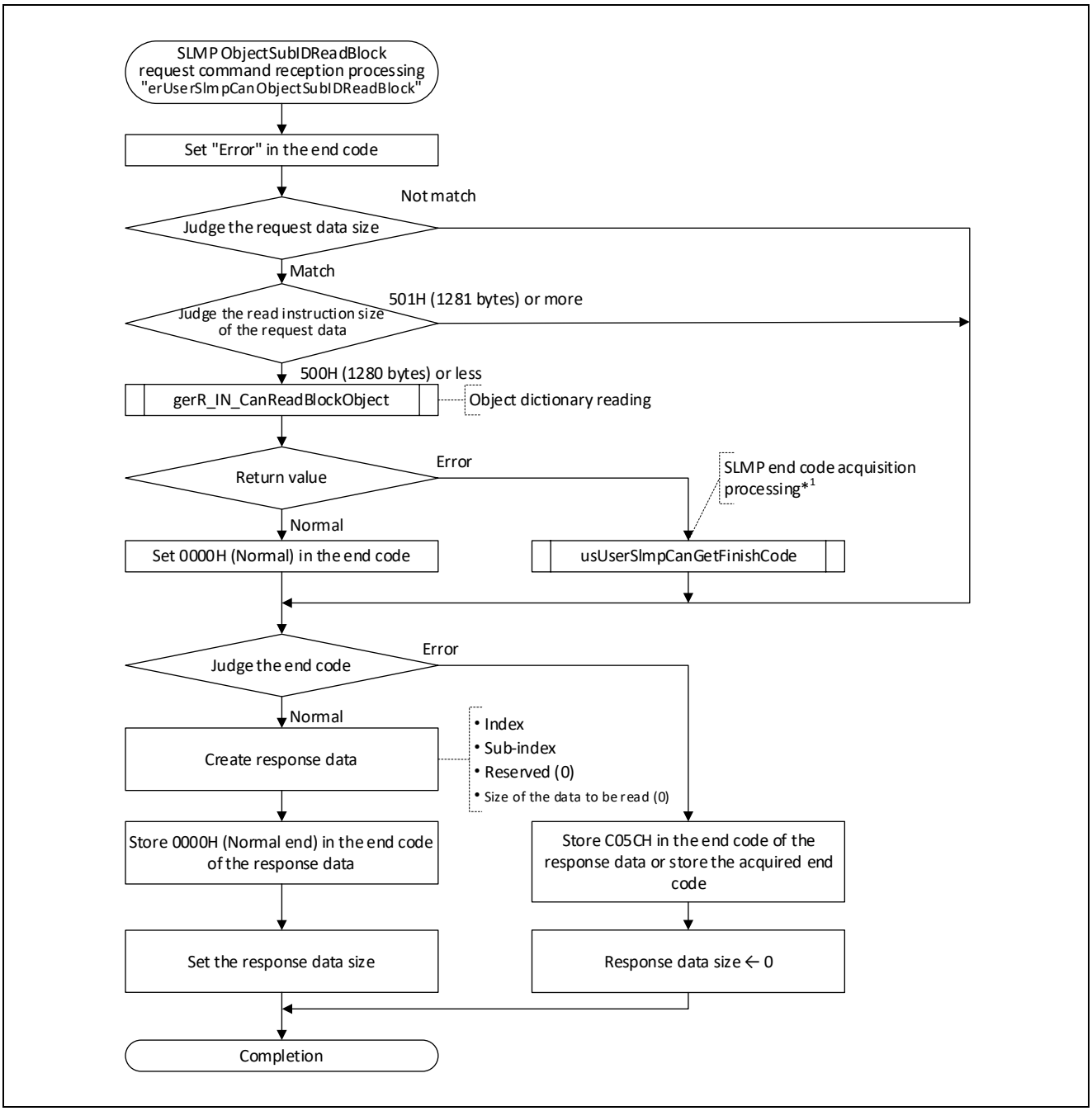


Figure 5.75 Flowchart for SLMP ObjectSubIDReadBlock Request Command Reception Processing

Note 1. Refer to "usUserSlmpCanGetFinishCode" (5.9.10, SLMP End Code Acquisition Processing).

5.9.7 SLMP ObjectSubIDWriteBlock Request Command Reception Processing

This function receives SDO write requests from the master station and sends responses.
The specified amount of data is written from the corresponding object in the object dictionary based on the specified index and sub-index.
The function receives LMT frames. When a frame other than above is received, an error response is sent and the receive data is discarded. If an error in the data length after the subcommand section of the received LMT frame is detected, the R-IN32M4-CL3 driver sets C05CH to the end code and sends a response. Therefore, this command receive processing is not performed.

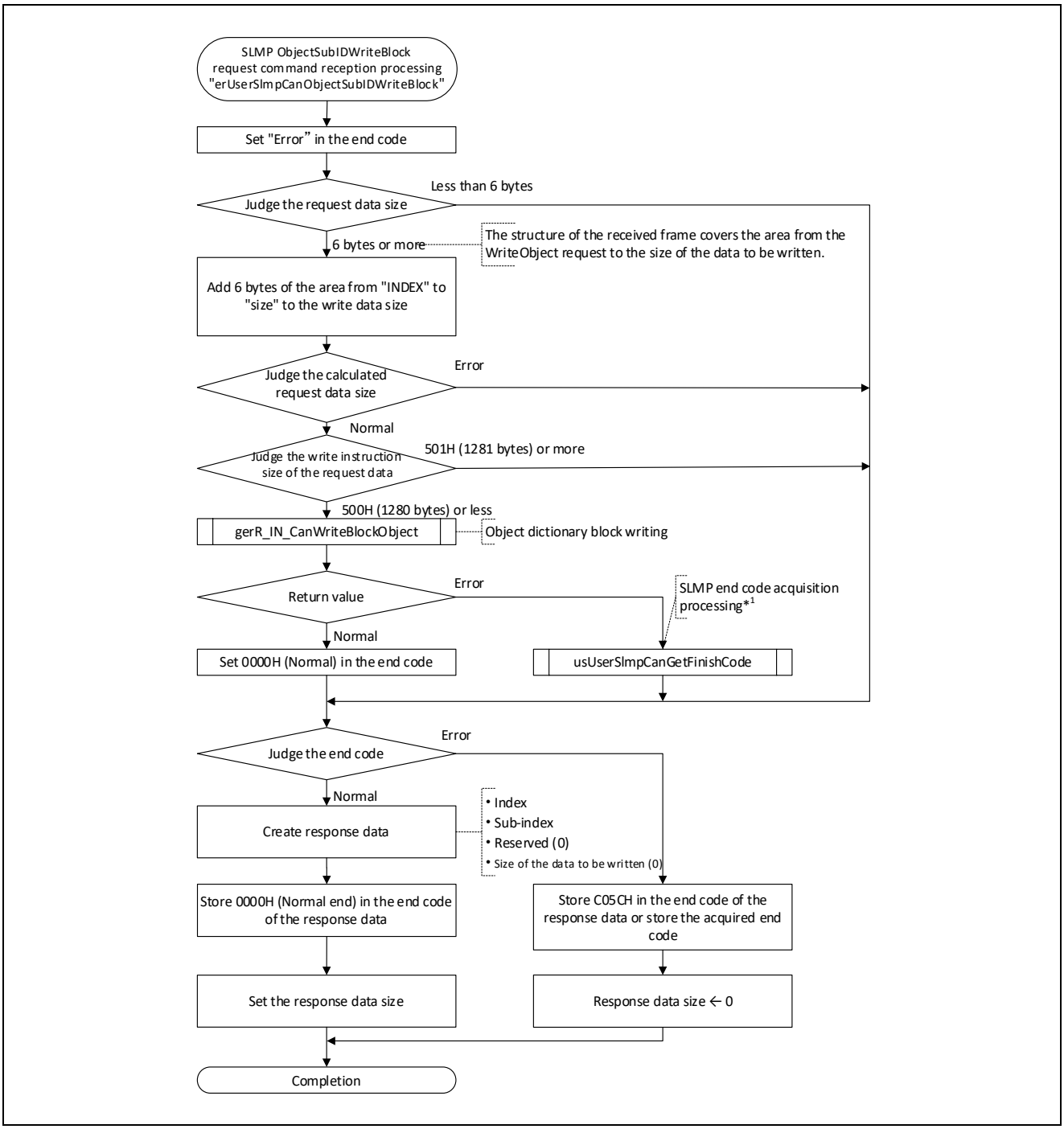


Figure 5.76 Flowchart for SLMP ObjectSubIDWriteBlock Request Command Reception Processing

Note 1. Refer to "usUserSlmpCanGetFinishCode" (5.9.10, SLMP End Code Acquisition Processing).

5.9.8 SLMP NMTStateUpload Request Command Reception Processing

This function acquires the NMT state of the home station and the latest NMT state to have been specified by the master station.
If the master has not specified an NMT state, INIT is acquired.

The function receives LMT frames. When a frame other than above is received, an error response is sent and the receive data is discarded. If an error in the data length after the subcommand section of the received LMT frame is detected, the R-IN32M4-CL3 driver sets C05CH to the end code and sends a response. Therefore, this command receive processing is not performed.

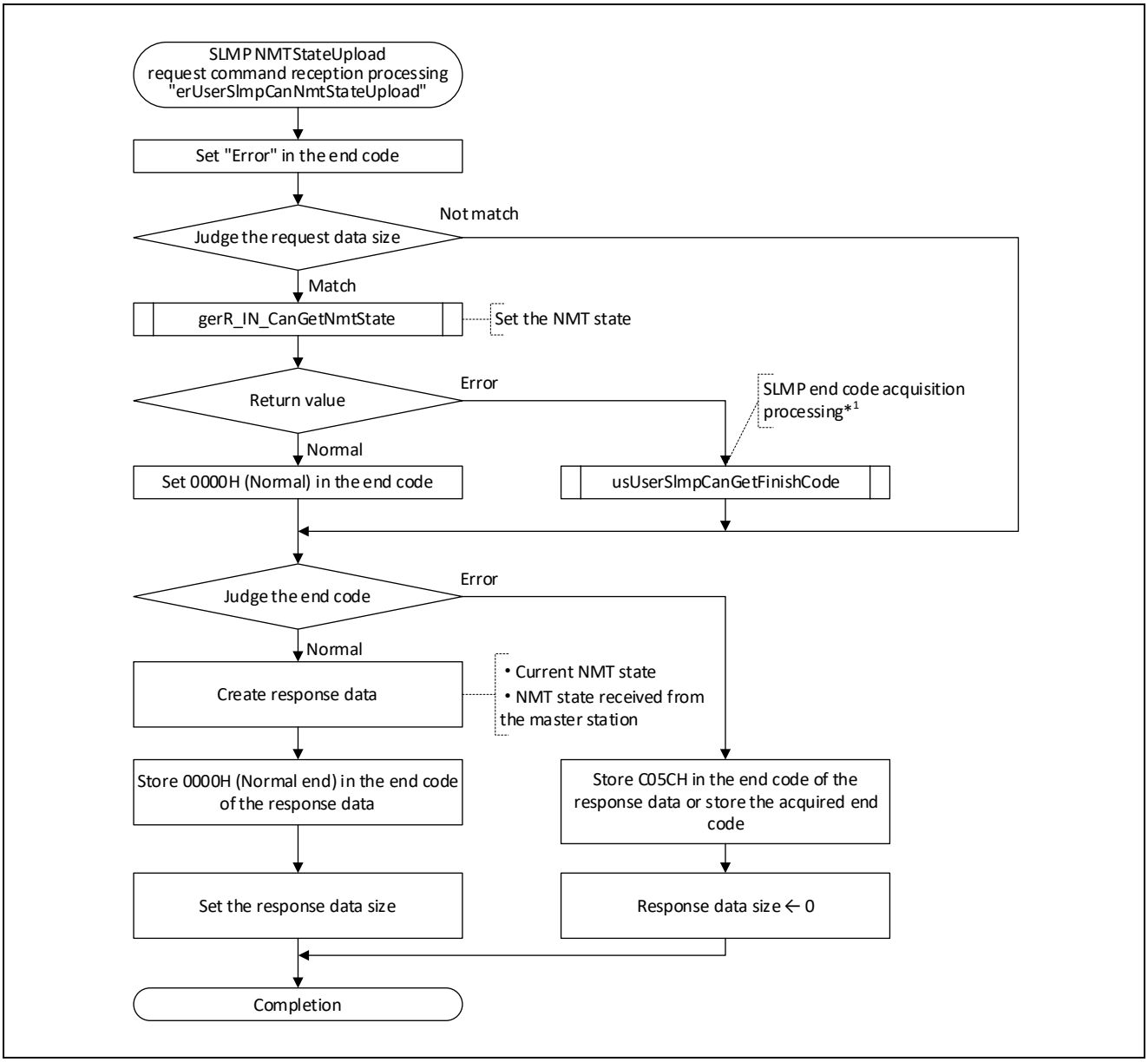


Figure 5.77 Flowchart for SLMP NMTStateUpload Request Command Reception Processing

Note 1. Refer to "usUserSlmpCanGetFinishCode" (5.9.10, SLMP End Code Acquisition Processing).

5.9.9 SLMP NMTStateDownload Request Command Reception Processing

This function sets the specified NMT state in the R-IN32M4-CL3 driver.

The function receives LMT frames. When a frame other than above is received, an error response is sent and the receive data is discarded. If an error in the data length after the subcommand section of the received LMT frame is detected, the R-IN32M4-CL3 driver sets C05CH to the end code and sends a response. Therefore, this command receive processing is not performed.

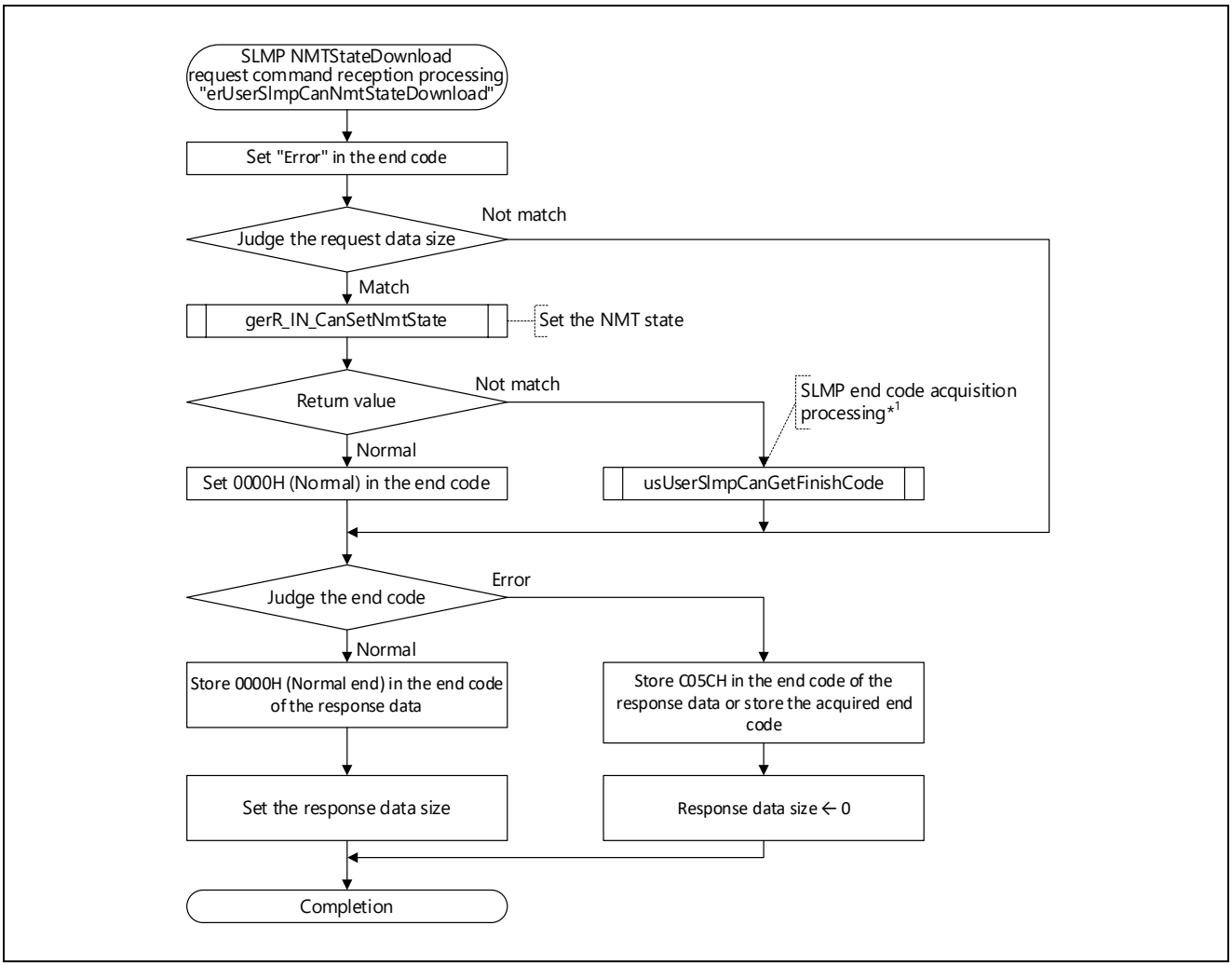


Figure 5.78 Flowchart for SLMP NMTStateDownload Request Command Reception Processing

Note 1. Refer to "usUserSlmpCanGetFinishCode" (5.9.10, SLMP End Code Acquisition Processing).

5.9.10 SLMP End Code Acquisition Processing

This function converts the specified SDO abort code into the end code of the SLMP.

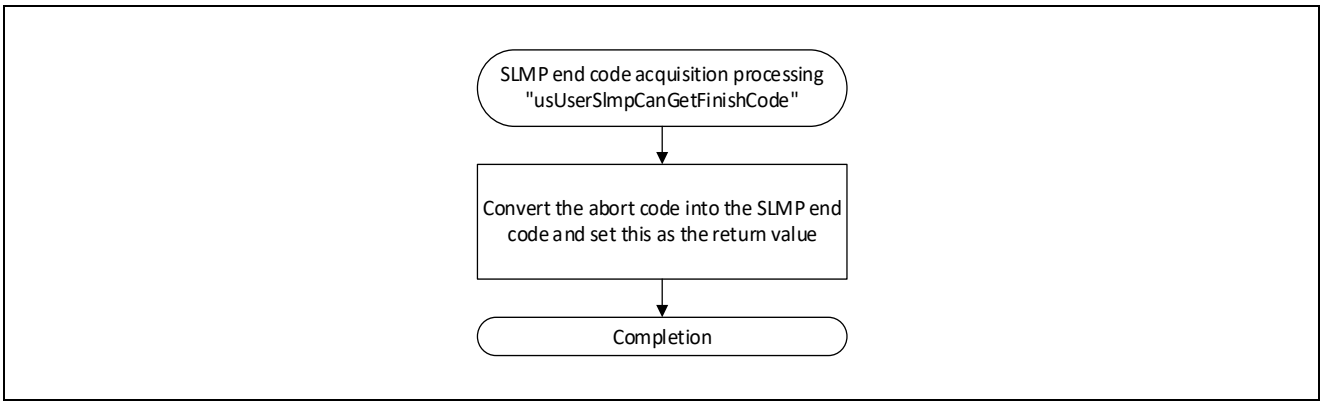


Figure 5.79 Flowchart for SLMP End Code Acquisition Processing

Table 5.32 Corresponding Table between SDO Abort Code and SLMP End Code

No	Abort Code	Description	SLMP End Code	Description
1	0000 0000H	No error	CCFFH	Object access error (although an error was found, the abort code has not been set)
2	0503 0000H	Toggle bit unchangeable	CCFFH	Object access error
3	0504 0000H	SDO protocol timeout	CCFFH	(Same as the above)
4	0504 0001H	Invalid or unknown client/server command specifier	CCFFH	(Same as the above)
5	0504 0005H	Out of the memory range	CCFFH	(Same as the above)
6	0601 0000H	Access to a non-supported object	CCC7H	An object was accessed under the condition where access to objects is not enabled.
7	0601 0001H	Read access to a write-only object	CCC8H	A write-only object was read-accessed.
8	0601 0002H	Write access to a read-only object	CCC9H	A read-only object was write-accessed
9	0602 0000H	Access to a non-present object	CCCAH	An index not defined in the object dictionary was specified.
10	0604 0041H	PDO mapping is not possible. An object cannot be mapped to PDO	CCCBH	An object for which PDO mapping is not enabled was mapped.
11	0604 0042H	The number of PDO mapping objects and the data length exceed the data length of PDO.	CCCCH	The total of the amount of data to be mapped to PDOs and the length of the data exceeded the value defined by the application, etc.
12	0604 0043H	Mismatch in general parameters	CCFFH	Object access error
13	0604 0047H	General internal mismatch in the device	CCFFH	(Same as the above)
14	0606 0000H	Hardware error	CCFFH	(Same as the above)
15	0607 0010H	Mismatch in the data length of an SDO	CCFFH	(Same as the above)
16	0607 0012H	The data of SDO are too long.	CCFFH	(Same as the above)
17	0607 0013H	The data of SDO are too short.	CCFFH	(Same as the above)
18	0609 0011H	A sub-index is not present.	CCD3H	A sub-index not defined in the object dictionary was specified.
19	0609 0030H	Invalid parameter value (written value only)	CCD4H	A requested parameter is out of range.
20	0609 0031H	The written parameter value is too large.	CCD5H	A larger value than the parameter range was set.
21	0609 0032H	The written parameter value is too small.	CCD6H	A smaller value than the parameter range was set.
22	0609 0036H	The maximum value is smaller than the minimum value.	CCFFH	Object access error
23	0800 0000H	General error	CCFFH	(Same as the above)
24	0800 0020H	Reading from and writing to the application is not possible.	CCDAH	The application can neither transfer nor store data.
25	0800 0021H	Reading from and writing to the application is not possible (by local control).	CCFFH	Object access error
26	0800 0022H	Reading from and writing to the application is not possible (by the current device state).	CCFFH	(Same as the above)
27	0800 0023H	An object dictionary is not present.	CCFFH	(Same as the above)
28	—	Other than the above	CCFFH	(Same as the above)

5.9.11 CANopen Parameter Setting Processing

This function sets parameters received through the parameter automatic setting function in the CANopen parameter variables.

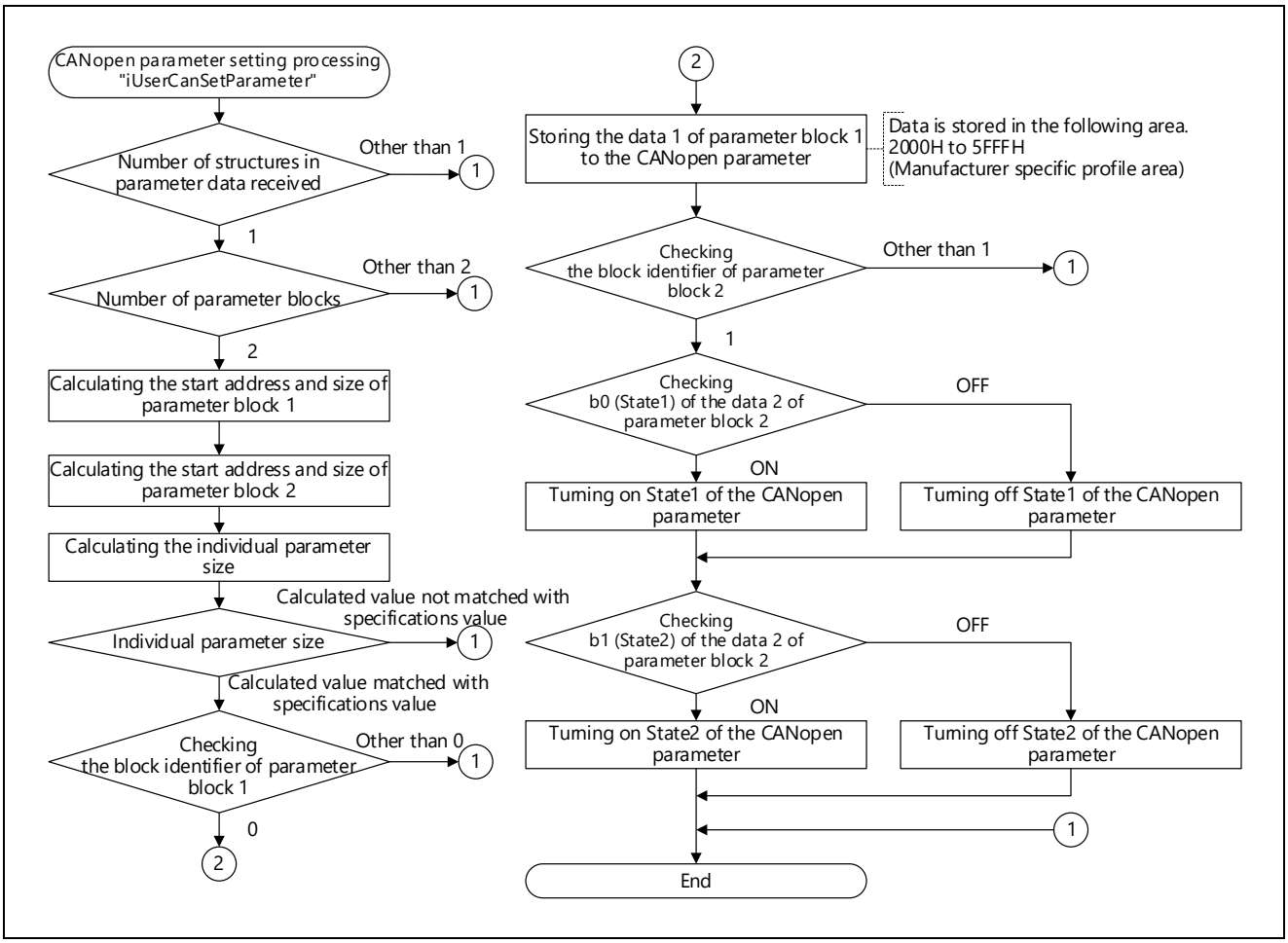


Figure 5.80 Flowchart for CANopen Parameter Setting Processing

The table below shows the structure of the areas passed as arguments of this processing when the sample CSP+ file included in the "CSPP" folder is used.

Table 5.33 CANopen Parameter Structure

No	Item		Size (Byte)	Setting	Description
1	Offset		2	24 (byte)	Data size of the file data main body section
2	Parameter configuration		2	0001H	Individual parameters are only used.
3	Individual parameter size		2	18 (byte)	Size of "the number of parameter blocks" + size of "a block"
4	Number of parameter blocks		2	2	Number of parameter blocks
5	Block 1	Block identification ID	4	00000000H	ID for identifying the blocks
6		Data length	2	1 (word)	Size (word) of "data 1" to "data n"
7		Data 1	2	Parameter 1	The setting of "Parameter 1" to be set by GX Works 3 is stored. The setting will be stored in CANopen parameters.
8	Block 2	Block identification ID	4	00000000H	ID for identifying the blocks
9		Data length	2	1 (word)	Size (word) of "data 1" to "data n"
10		Data 2	2	Parameter 2 b0: State 1 b1: State 2 b2-15: Not used	The setting of "Parameter 2" to be set by GX Works 3 is stored. The setting will be stored in CANopen parameters.

5.9.12 Index1010 Write Processing

Processing by this function proceeds when it is registered in the object dictionary and data are written to Index1010.
In the specification for CANopen communications, when "save (65766173H)" is written to Index1010/SubIndex1, all parameters in the device are stored in a non-volatile memory (the storage of parameters itself is to be implemented by the user).

If a R-IN32M4-CL3 application product is, for example, a multi-axis servo amplifier, which consists of a main module (axis 1) that performs data communications and extension modules (axis 2 and later) that do not perform data communications, add this function for the number of axes used (up to eight functions).

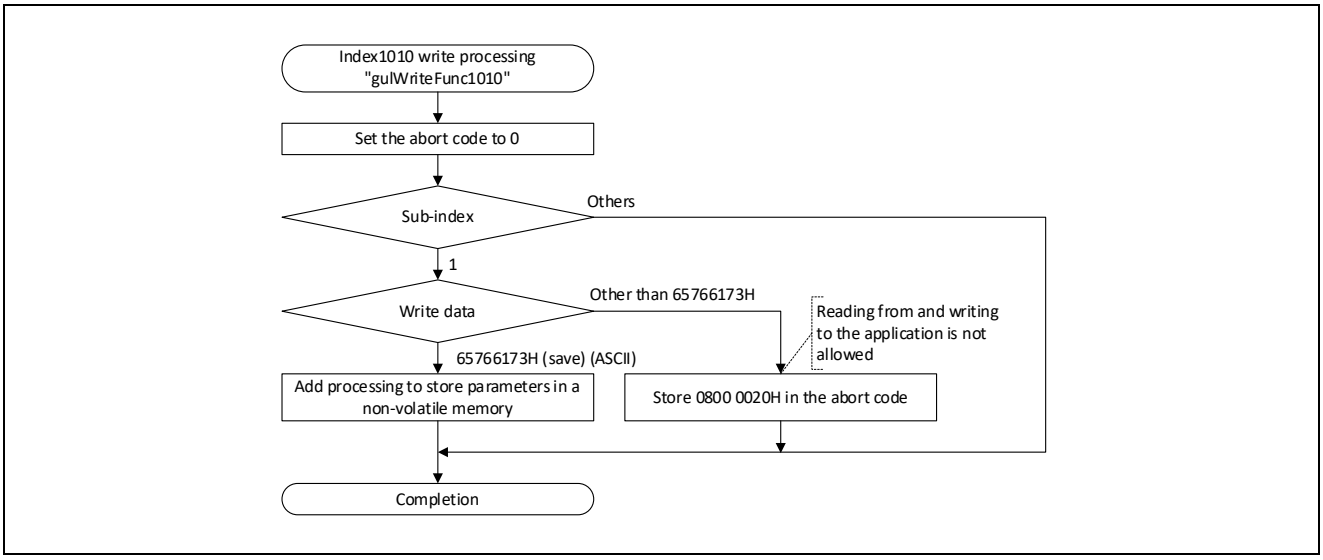


Figure 5.81 Flowchart for Index1010 Write Processing

5.9.13 Index1011 Write Processing

Processing by this function proceeds when it is registered in the object dictionary and data are written to Index1011.
In the specification for CANopen communications, when "load (64616F6CH)" is written to Index1011/SubIndex1, all parameters in the device are restored to their default values (the storage of parameters itself is to be implemented by the user).

If a R-IN32M4-CL3 application product is, for example, a multi-axis servo amplifier, which consists of a main module (axis 1) that performs data communications and extension modules (axis 2 and later) that do not perform data communications, add this function for the number of axes used (up to eight functions).

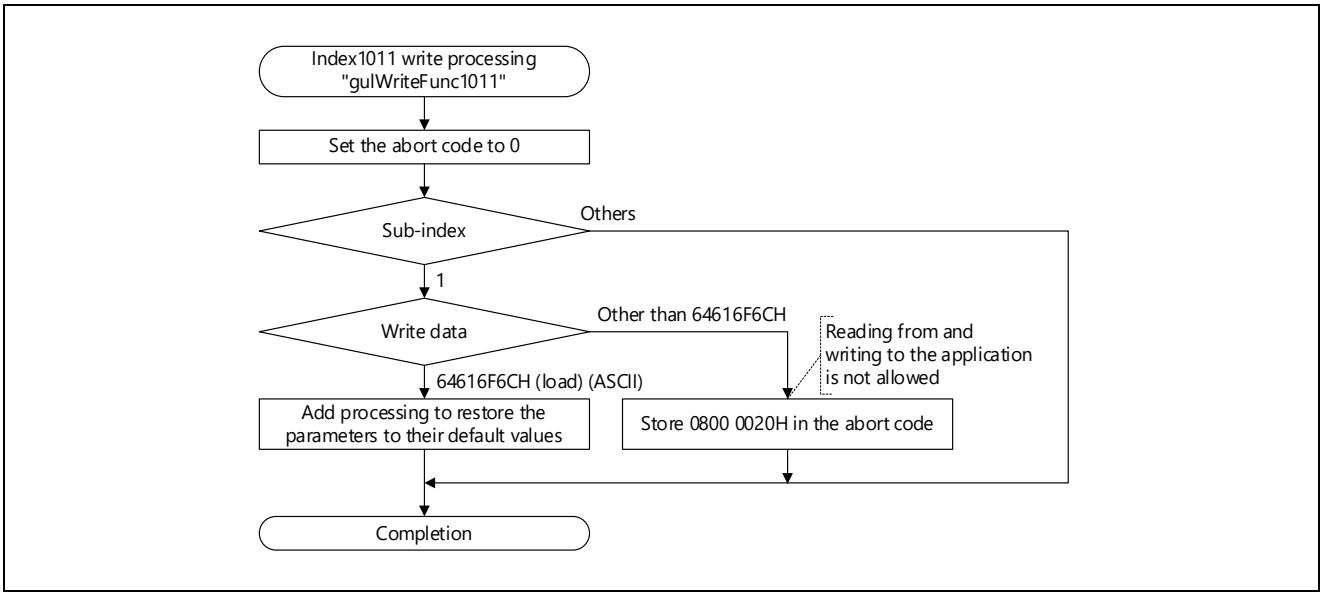


Figure 5.82 Flowchart for Index1011 Write Processing

5.10 Details on Processing of User Programs (MCU-MCU Interface Related)

(1) Overview of the MCU-MCU interface function

This function sends/receives safety PDUs between the MCU for communications (internal R-IN32M4-CL3) and the MCU for safety processing (external).

Each MCU consists of GPIO communications (for receive), GPIO communications (for send), and serial communication (RT DMA transfer).

The following figure shows a communication image and the table lists the purpose of use.

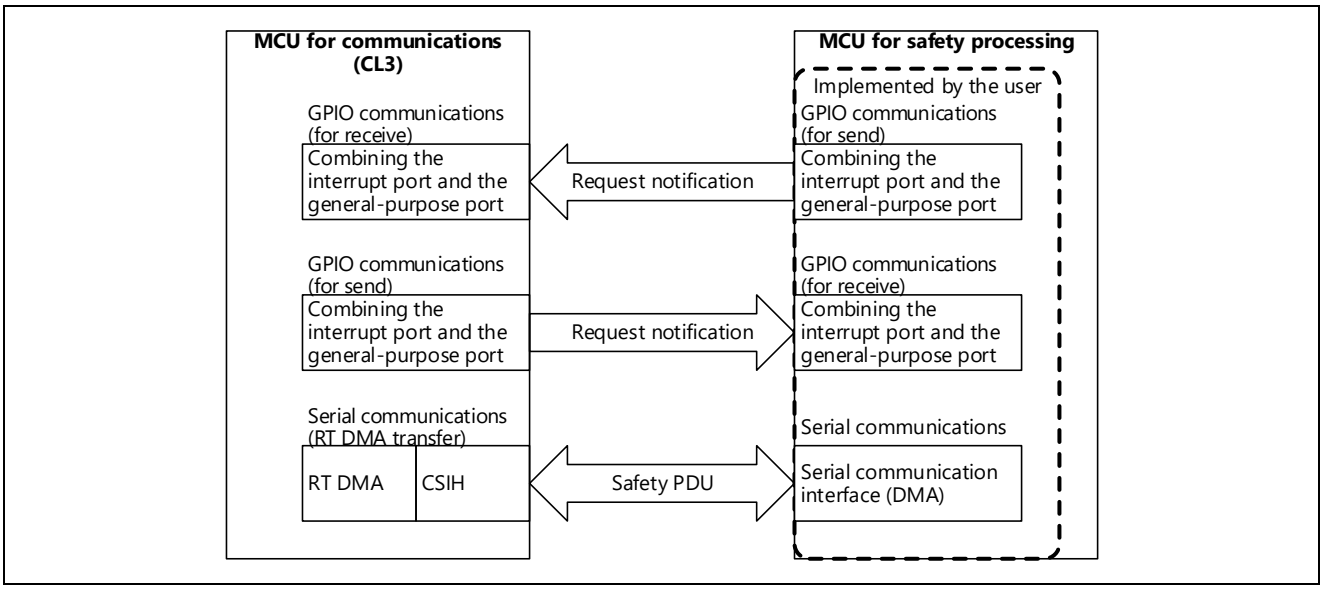


Figure 5.83 Communication Image

Table 5.34 Purpose of Use

No.	Function	Description
1	GPIO communications (for receive)	To notify a request from the MCU for safety processing to the MCU for communications.
2	GPIO communications (for send)	To notify a request from the MCU for communications to the MCU for safety processing.
3	Serial communication (RT DMA transfer)	To send or receive safety PDU values in serial communication.

(2) GPIO communications

GPIO communications are performed with one interrupt port and 11 general-purpose ports.

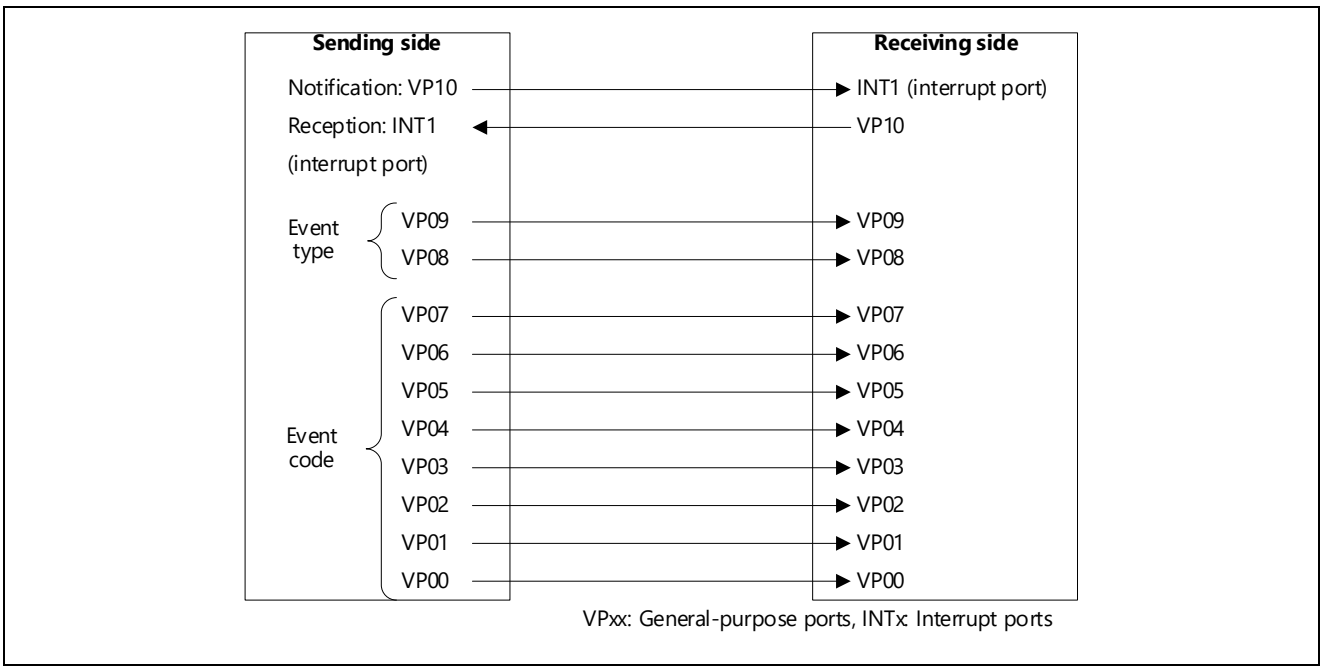


Figure 5.84 Configuration Image of GPIO Communications

The following lists the port assignment for GPIO communications.

Table 5.35 Port Assignment for GPIO Communications

Signal type		Signal assignment					
		MCU for communications (R-IN32M4-CL3)	Direction	MCU for safety processing	MCU for safety processing	Direction	MCU for communications (R-IN32M4-CL3)
Notification	VP10	Any general-purpose port	→	Any interrupt port	Any general-purpose port	→	Interrupt port INTPZ15*1
Reception	INT1	Interrupt port INTPZ14*1	←	Any general-purpose port	Any interrupt port	←	Any general-purpose port
Event type	VP09 VP08	Any general-purpose port	→	Any general-purpose port	Any general-purpose port	→	Any general-purpose port
Event code	VP07 VP06 VP05 VP04 VP03 VP02 VP01 VP00	Any general-purpose port	→	Any general-purpose port	Any general-purpose port	→	Any general-purpose port

Note 1. When an interrupt occurs, the GPIO communication response receive task starts.

Note 2. When an interrupt occurs, the GPIO communication receive task starts.

The following tables list the event types and event codes used in this sequence.

Table 5.36 Event Types and Event Codes Received by the Internal MCU (MCU for Communications)

Event type	Event code	Command
01b	0000 0001b	Safety PDU transfer request notification (internal → external MCU)
01b	0000 0010b	Safety PDU transfer request notification (internal ← external MCU)
Other than the above		Not used

Table 5.37 Event Types and Event Codes Sent from the Internal MCU (MCU for Communications)

Event type	Event code	Command
01b	0000 0001b	Safety PDU transfer ready notification (internal → external MCU)
01b	0000 0010b	Safety PDU transfer ready notification (internal ← external MCU)
Other than the above		Not used

The following is the timing chart of GPIO communications.

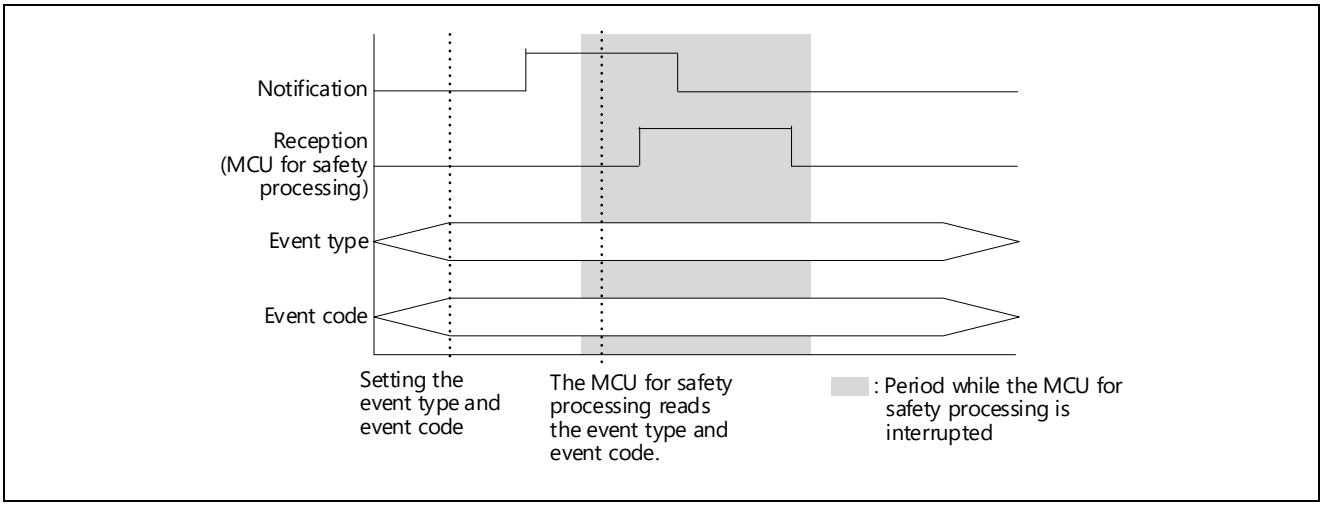


Figure 5.85 Timing Chart of GPIO Communications

No.	Overview of send processing of the MCU for communications (To send a request notification)
1	Turns on or off the general-purpose ports for the event type and event code.
2	Turns off, on, and then off the general-purpose port for the notification signal. Turns off and on the interrupt port on the receiving side to detect the request notification.
3	Detects that the interrupt port for the reception signal is turned off and on.
4	Performs the GPIO communication end processing.

No.	Overview of receive processing of the MCU for communications (To receive a request notification)
1	Detects that the interrupt port for the notification signal is turned off and on.
2	Reads the event type and event code from the general-purpose ports.
3	Turns off and on the general-purpose port for the reception signal.
4	Performs command processing corresponding to the event type and event code read.
5	Turns on and off the general-purpose port for the reception signal.

(3) Serial communication (RT DMA transfer)

Serial communication is performed using the clocked serial interface H of R-IN32M4-CL3.

Table 5.38 Port Assignment for GPIO Communications

Communication method	Transmission speed	Data length	DMA	Interrupt
Synchronous	7.14 Mbps	• 8bits • No parity • Stopbit = 1	RT DMA	DMA transfer completion interrupt

The following shows the structure of data sent/received in serial communication.
For details on the safety PDU structure, refer to the "CC-Link IE Safety Communication Function Specification" published by CC-Link Partner Association

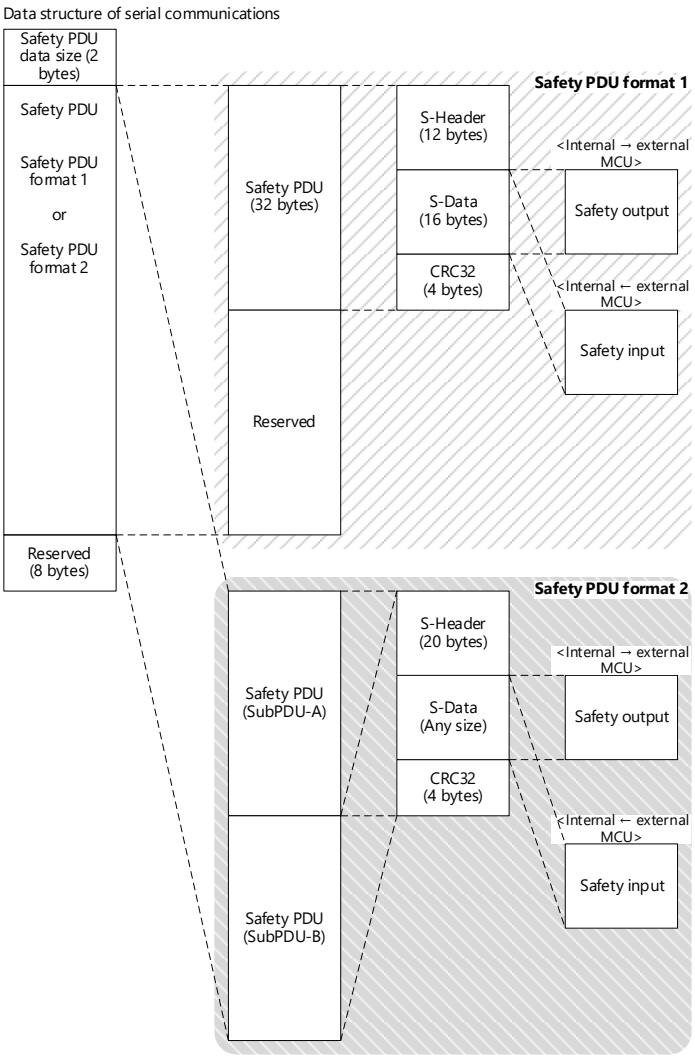


Figure 5.86 Safety PDU Structure of Serial Communication (RT DMA Transfer) (Overview)

To use the safety PDU in the R-IN32M4-CL3 driver, set the all safety PDU elements*1. To acquire the safety PDU using the R-IN32M4-CL3 driver, read the all safety PDU elements*1.

Note 1. For the safety PDU format 1, S-Header to CRC32 are required.
For the safety PDU format 2, the start of SubPDU-A to the end of SubPDU-B are required.

(4) Sequence between MCUs

The following shows a sequence when the safety PDU is sent/received between MCUs.

(a) Transferring the safety PDU from the internal MCU to the external MCU

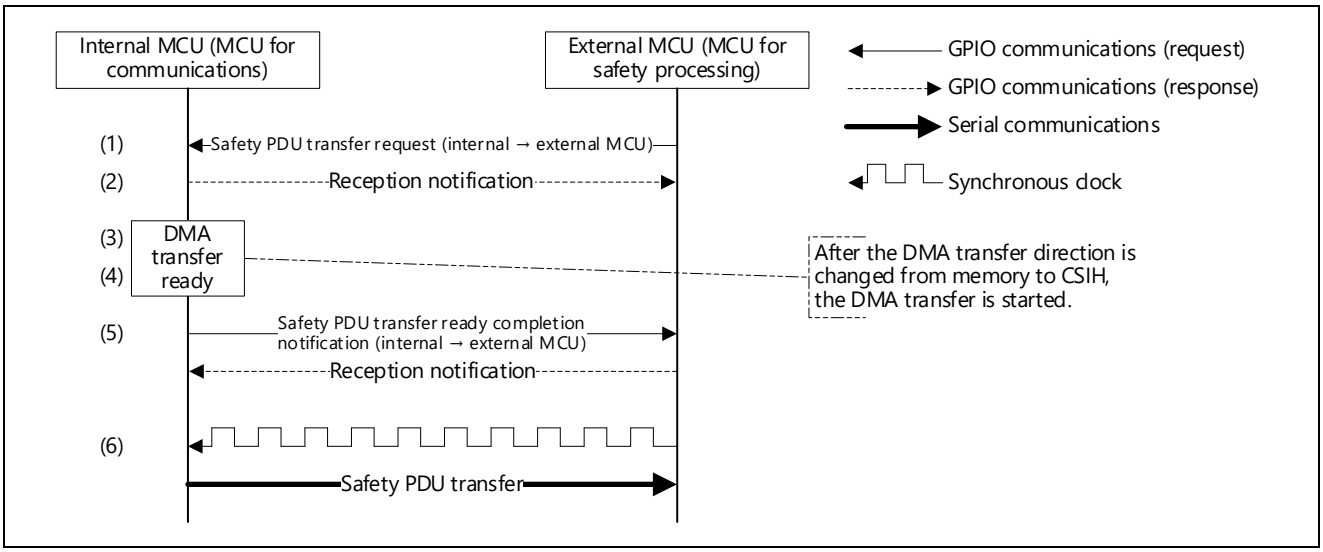


Figure 5.87 Transferring the Safety PDU from the Internal MCU to the External MCU

No.	Processing in the internal MCU	Reference
1	Detects that the notification signal of GPIO communications (for receive) is turned on.	
2	Reads the event type and the event code received in the GPIO communications (for receive) and turns on the reception signal.	5.10.2
3	Executes DMA transfer ready processing based on the event type and the event code read.	5.10.5
4	After the processing is completed, turns off the reception signal.	5.10.3
5	Notifies that the safety PDU transfer is ready in GPIO communications (for send). (Turns off, on and then off the notification signal after the event type and the event code are set.)	5.10.4 5.10.7
6	Sends the memory value set in the DMA transfer ready processing (No.3) upon receiving the synchronous clocks.	

(b) Transferring the safety PDU from the external MCU to the internal MCU

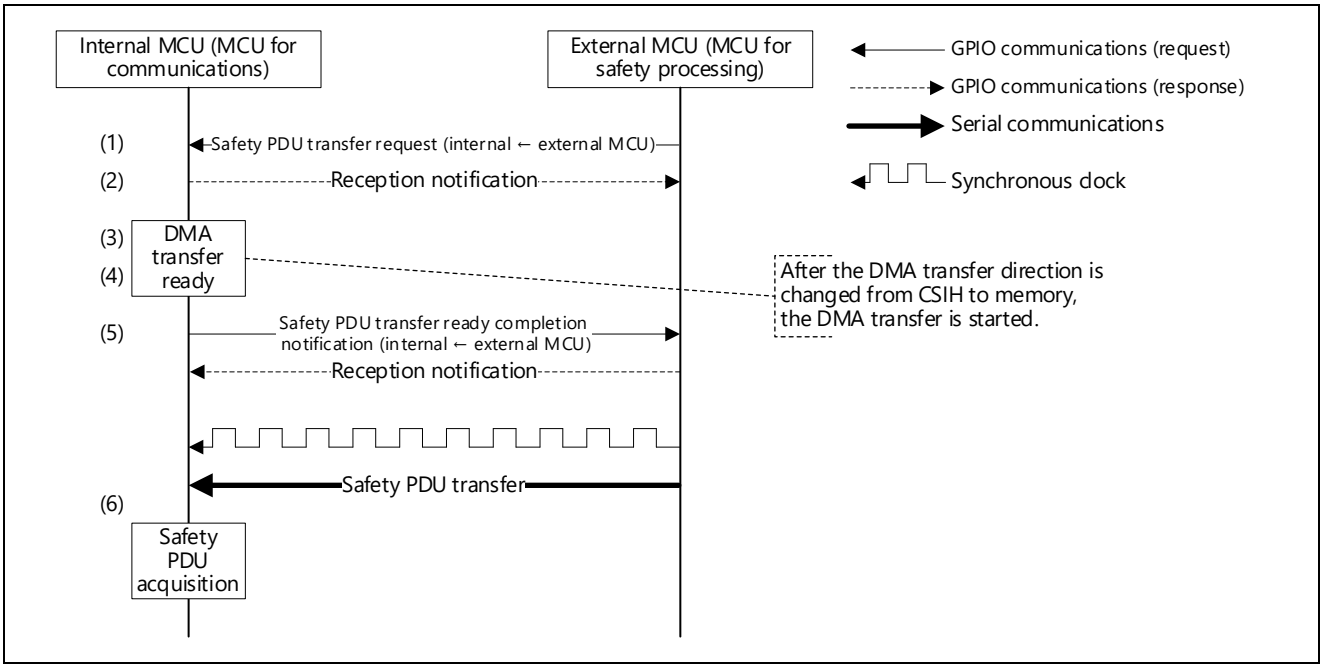


Figure 5.88 Transferring the Safety PDU from the External MCU to the Internal MCU

No.	Processing in the internal MCU	Reference
1	Detects that the notification signal of GPIO communications (for receive) is turned on.	
2	Reads the event type and the event code received in the GPIO communications (for receive) and turns on the reception signal.	5.10.2
3	Executes DMA transfer ready processing based on the event type and the event code read.	5.10.6
4	After the processing is completed, turns off the reception signal.	5.10.3
5	Notifies that the safety PDU transfer is ready in GPIO communications (for send). (Turns off, on and then off the notification signal after the event type and the event code are set.)	5.10.4 5.10.8
6	Acquires the safety PDU from the memory set in the DMA transfer ready processing (No.3) after the safety PDU receive processing is completed.	5.10.9

5.10.1 MCU-MCU interface initialization processing

This function notifies the R-IN32M4-CL3 driver of the function pointers of the GPIO communication functions and the functions executed when the serial communication is completed.

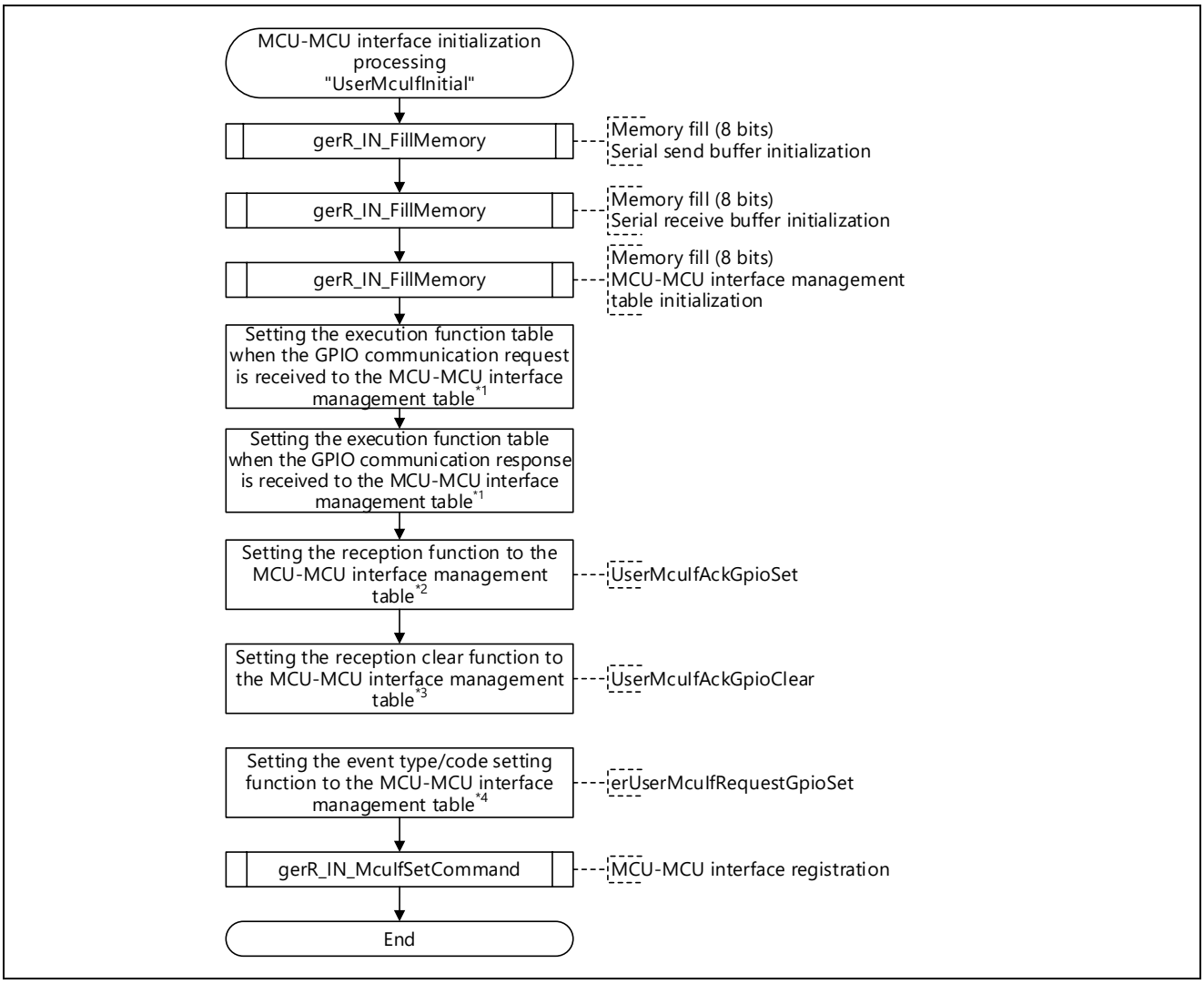


Figure 5.89 Flowchart for MCU-MCU Interface Initialization Processing

Note 1. For details, refer to "Table 6.42 R_IN_MCU_IF_GPIO_MANAGEMENT_TBL_T List".

Note 2. For UserMcuIfAckGpioSet, refer to Section 5.10.2 "GPIO communication acceptance processing".

Note 3. For UserMcuIfAckGpioClear, refer to Section 5.10.3 "GPIO communication acceptance clear processing".

Note 4. For erUserMcuIfRequestGpioSet, refer to Section 5.10.4 "GPIO communication event type/code setting processing".

5.10.2 GPIO communication acceptance processing

This function reads the event type and the event code from any GPIO assigned by a user and turns on the reception signal.
Implement the function in accordance with the hardware configuration by the user.

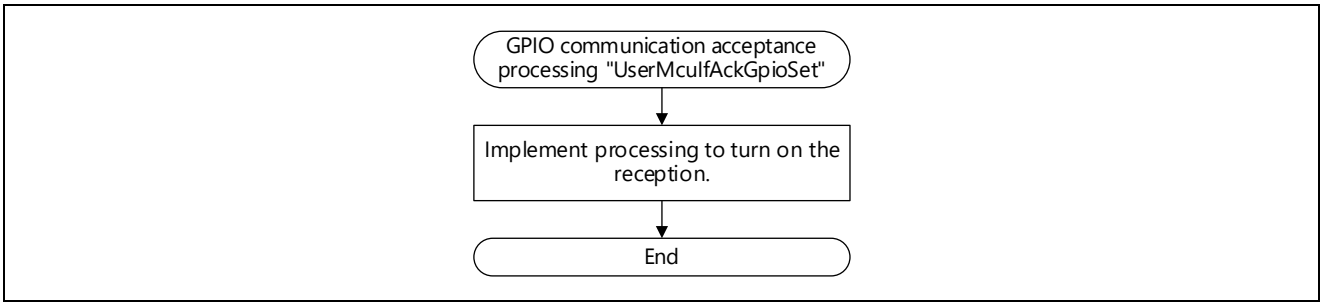


Figure 5.90 Flowchart for GPIO Communication Acceptance Processing

5.10.3 GPIO communication acceptance clear processing

This function turns off the reception signal using any GPIO assigned by the user.
Implement the function in accordance with the hardware configuration by the user.

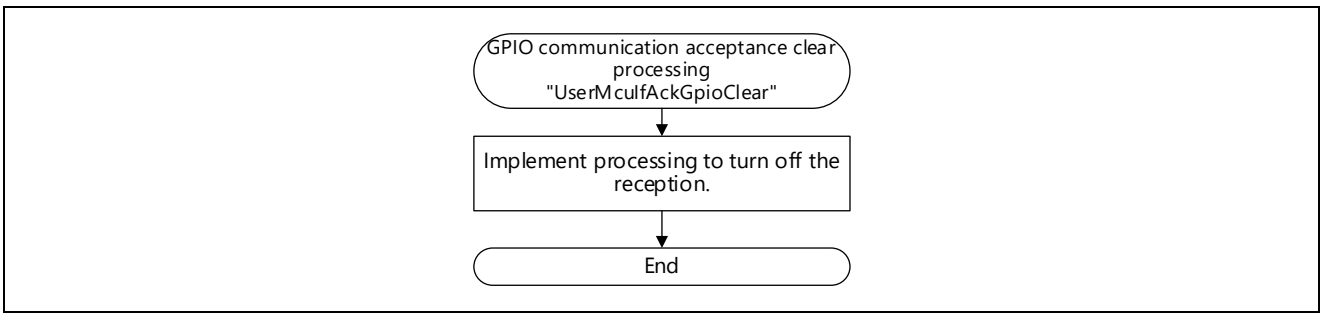


Figure 5.91 Flowchart for GPIO Communication Acceptance Clear Processing

5.10.4 GPIO communication event type/code setting processing

This function sets the event type and event code using any GPIO assigned by a user and turns off, on and then off the notification signal.
Implement the function in accordance with the hardware configuration by the user.

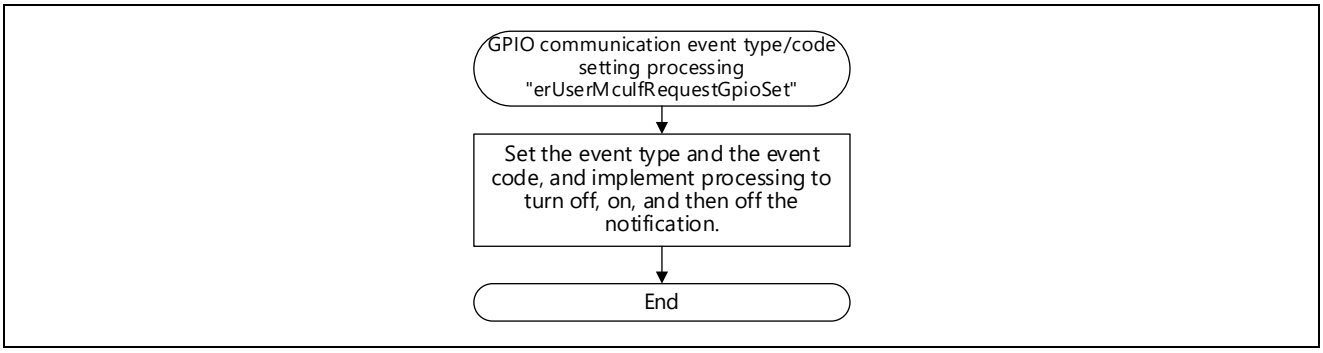


Figure 5.92 Flowchart for GPIO Communication Event Type/Code Setting Processing

Change the on/off timing of the notification signal in accordance with the processing speed of the MCU for safety processing implemented by the user.

5.10.5 Safety PDU transfer processing (internal → external MCU)

This function writes the safety PDU to a send buffer for serial communication.

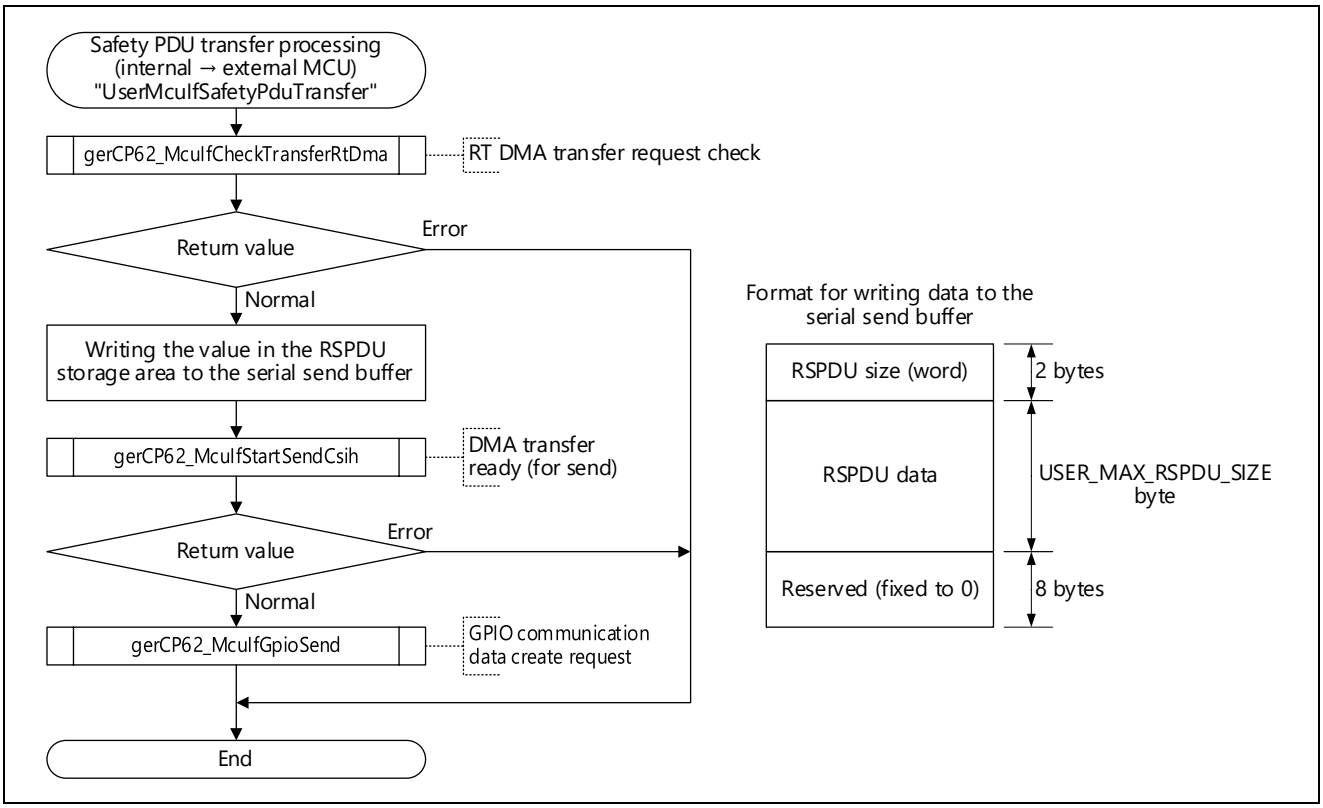


Figure 5.93 Flowchart for Safety PDU Transfer Processing (Internal → External MCU)

5.10.6 Safety PDU transfer processing (internal ← external MCU)

This function performs the DMA transfer ready processing (for receive) and GPIO communication data create request processing.

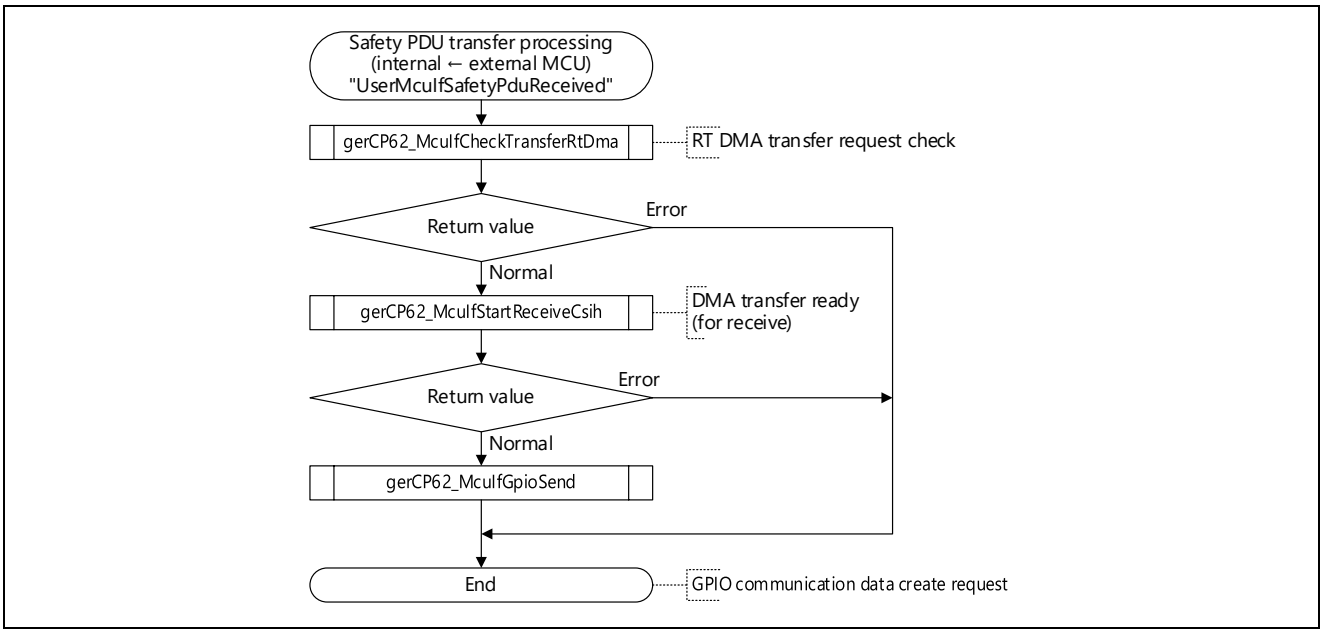


Figure 5.94 Flowchart for Safety PDU Transfer Processing (Internal ← External MCU)

5.10.7 Safety PDU transfer ready (internal → external MCU) acceptance receive processing

This function accepts the execution result in an argument after the safety PDU transfer ready (internal → external MCU) is notified to the external MCU. Execute any processing.

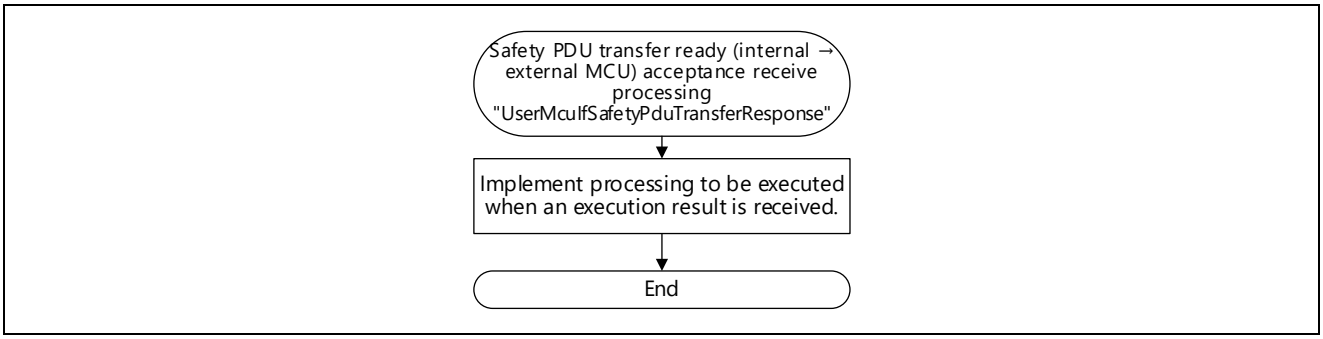


Figure 5.95 Flowchart for Safety PDU Transfer Ready (Internal → External MCU) Acceptance Receive Processing

Set the execution result of GPIO communication send processing as shown below.

Argument	Result	Description
R_IN_OK	Normal end	Detects the reception signal from the external MCU.
R_IN_ERR	Abnormal end	Detects that the GPIO communication data create request buffer is full (16 or more areas are used) or detects the timeout[*1] while waiting for the reception signal from the external MCU.

Note 1. Defined by R_IN_MCU_IF_RESPONSE_TIME.

5.10.8 Safety PDU transfer ready (internal ← external MCU) acceptance receive processing

This function accepts the execution result in an argument after the safety PDU transfer ready (internal ← external MCU) is notified to the external MCU. Execute any processing.

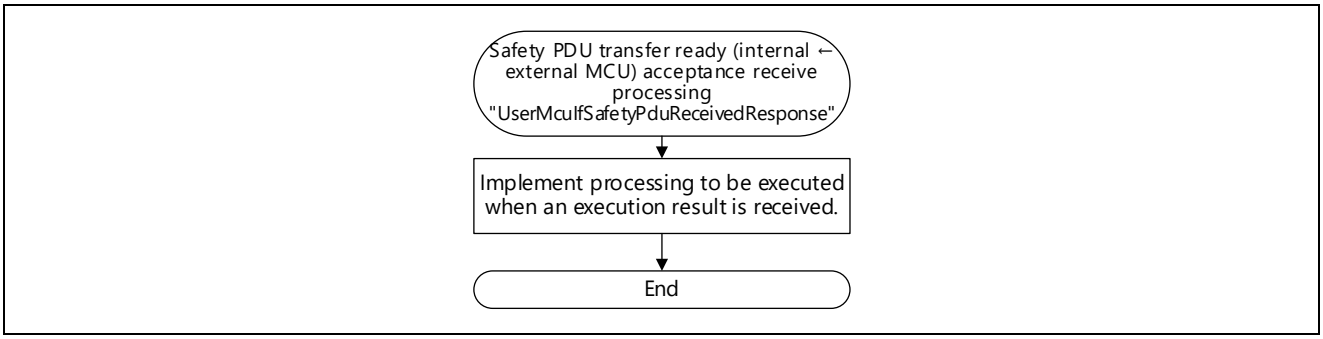


Figure 5.96 Flowchart for Safety PDU Transfer Ready (Internal ← External MCU) Acceptance Receive Processing

Set the execution result of GPIO communication send processing as shown below.

Argument	Result	Description
R_IN_OK	Normal end	Detects the reception signal from the external MCU.
R_IN_ERR	Abnormal end	Detects that the GPIO communication data create request buffer is full (16 or more areas are used) or detects the timeout*1 while waiting for the reception signal from the external MCU.

Note 1. Defined by R_IN_MCU_IF_RESPONSE_TIME.

5.10.9 Safety PDU transfer completion processing (internal ← external MCU)

This function stops serial communication and reads the safety PDU from the receive buffer for serial communication.

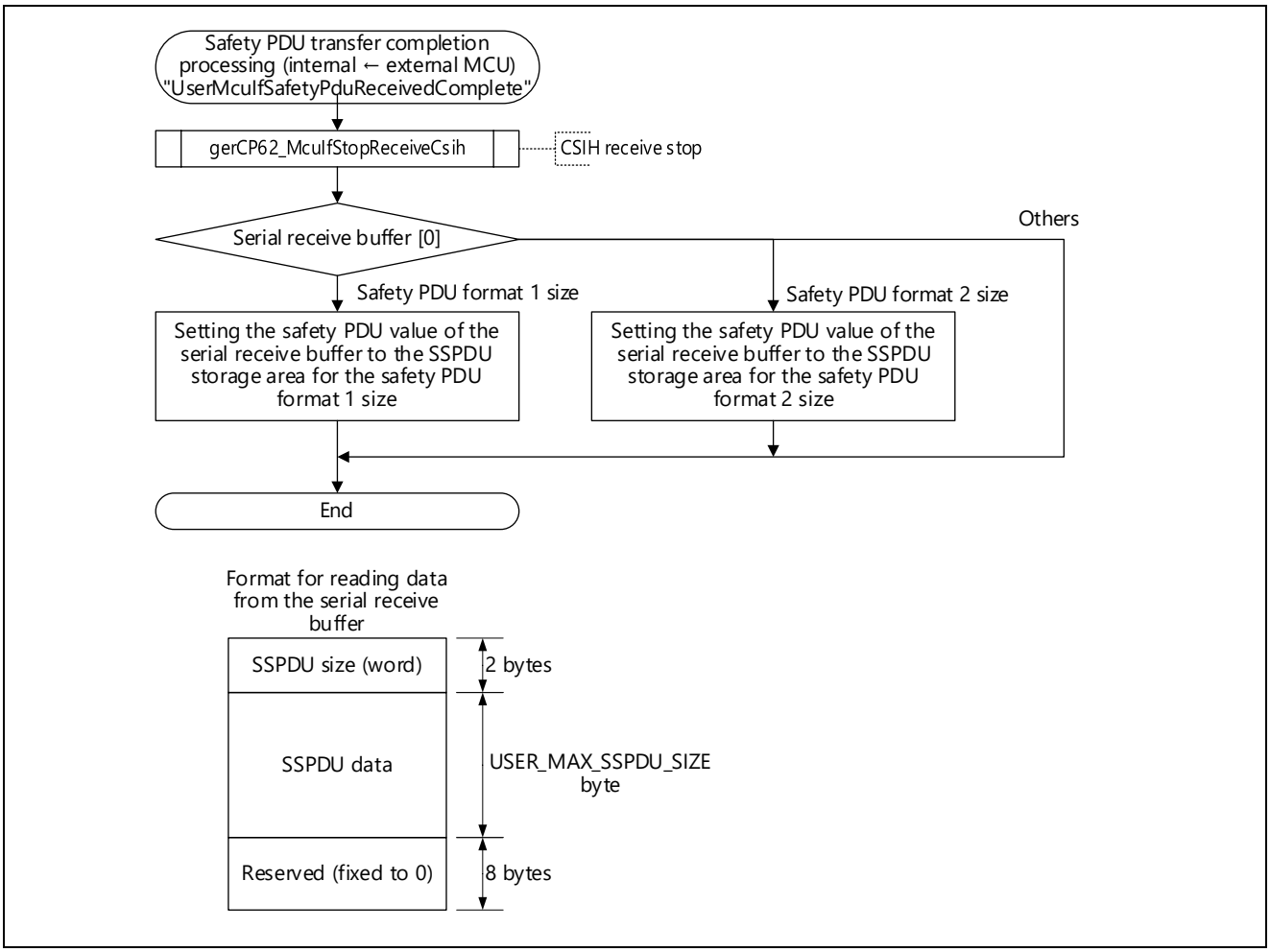


Figure 5.97 Flowchart for Safety PDU Transfer Completion Processing (Internal ← External MCU)

5.11 User Program Details (Hardware Test Related)

Implement the hardware test as processing that is performed offline, not as processing that is performed online (during data link).

5.11.1 Hardware Test Processing (IEEE802.3ab Compliance Test)

This function outputs waveforms for the IEEE802.3ab compliance test.
The gerR_IN_IEEETest function (6.4.13(1)) is executed in test modes (test modes 1 to 4, test end) as arguments.
After having saved the data in the PHY registers, gerR_IN_IEEETest writes test mode data to the PHY registers according to the test mode, so check the waveform output by using the measurement unit after writing has proceeded.

Table 5.39 Test Mode List

Test Mode	Constant Name (Value)	Description
Test mode 1	R_IN_IEEE_MODE1 (0)	Template, peak voltage, droop
Test mode 2	R_IN_IEEE_MODE2 (1)	Jitter (PHY is set as the master at the time of negotiation)
Test mode 3	R_IN_IEEE_MODE3 (2)	Jitter (PHY is set as a slave at the time of negotiation)
Test mode 4	R_IN_IEEE_MODE4 (3)	Distortion, return loss, common-mode voltage
Test end	R_IN_IEEE_END (4)	—

<Points to note>

- Do not start tasks other than "Initial Task" (5.2.1) and "Idle Task" (5.2.2).
- Execute this function after the completion of idle task initialization processing (iUserInitializationRoutine).
- Implement this processing as independent processing, not as processing that is performed within state management and transient main processing (iUserMainRoutin) in the idle task.

When moving to the end of a test or to a next test, the user must set the next test implementation flag to true. This allows moving to the end of a test end or to a next text after the saved PHY register data have been restored.

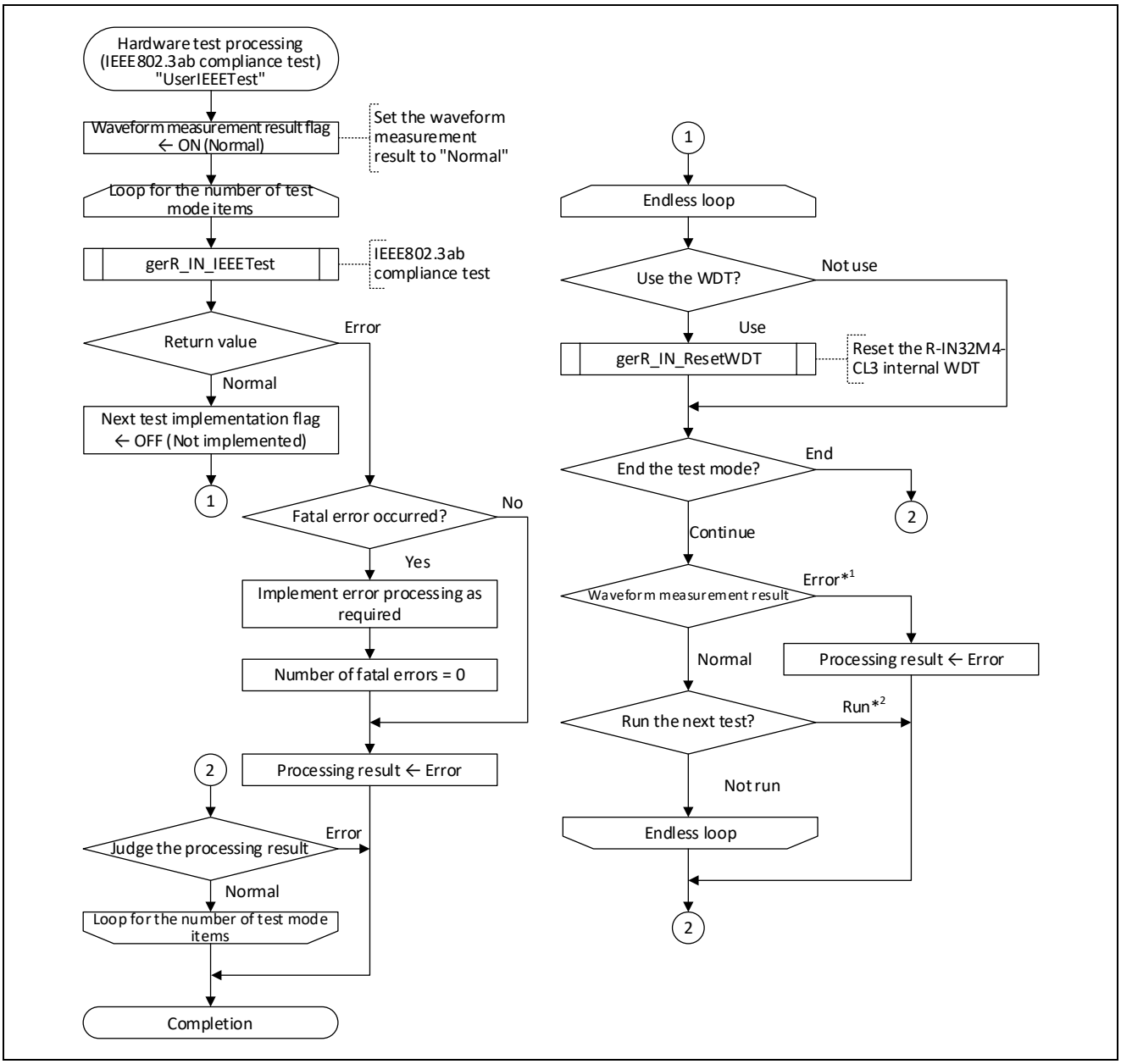


Figure 5.98 Flowchart for Hardware Test Processing (IEEE802.3ab Compliance Test)

Note 1. If the result of measurement is abnormal, processing to set the waveform measurement result flag to indicate the abnormality is required.

Note 2. Running a next test also requires processing to set the implementation flag for the next test.

5.11.2 Hardware Test Processing (Loopback Communications Test)

This function performs a loopback communications test.
When running this test, connect an Ethernet cable between port 1 and port 2.
This processing should be performed after initialization processing and communications start processing.

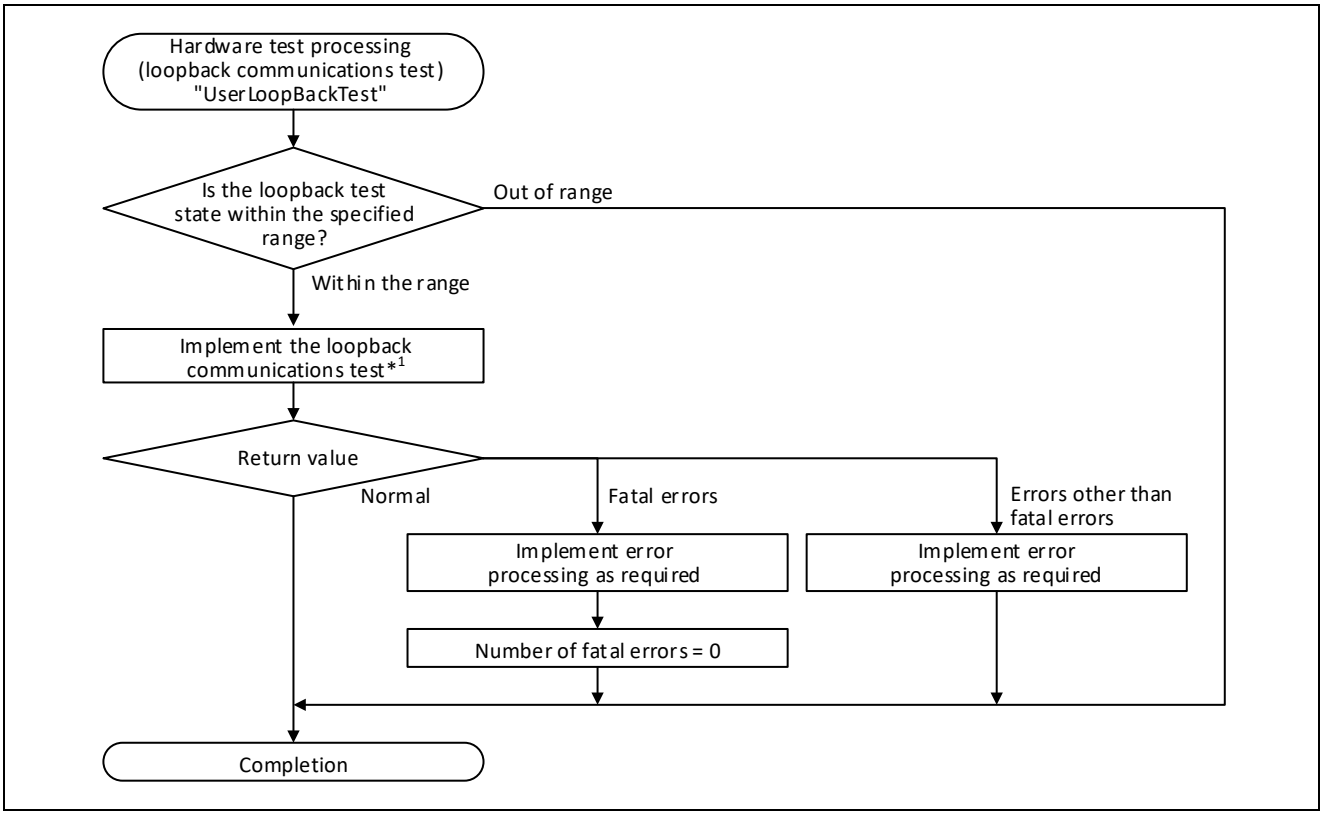


Figure 5.99 Flowchart for Hardware Test Processing (Loopback Communications Test)

Note 1. For the list of the loopback communications test states and the corresponding processing, refer to the following table.

Table 5.40 List of Test States

No	Loopback Communications Test State	Value	Processing Details
1	USER_LOOP_BACK_TEST_INITIAL	0	section 5.11.3, Loopback Communications Test Preparation Processing.
2	USER_LOOP_BACK_TEST_LINK_CHECK	1	section 5.11.4, Loopback Communications Test Link State Check Processing.
3	USER_LOOP_BACK_TEST_SEND	2	section 5.11.5, Loopback Communications Test Data Transmission Processing.
4	USER_LOOP_BACK_TEST_RECEIVE	3	section 5.11.6, Loopback Communications Test Reception Result Determination Processing.
5	USER_LOOP_BACK_TEST_EXIT	4	section 5.11.7, Loopback Communications Test Completion Processing.

In the loopback communications test state, the test proceeds from the No. 1 through to the No. 4 state of port 1 and then applies the same sequence to port 2. When all processing is completed or an error is found in any of these states for either port, the test proceeds to the No. 5 state.

5.11.3 Loopback Communications Test Preparation Processing

This function initializes the registers for detecting an error and the PHY registers during the loopback communications test.

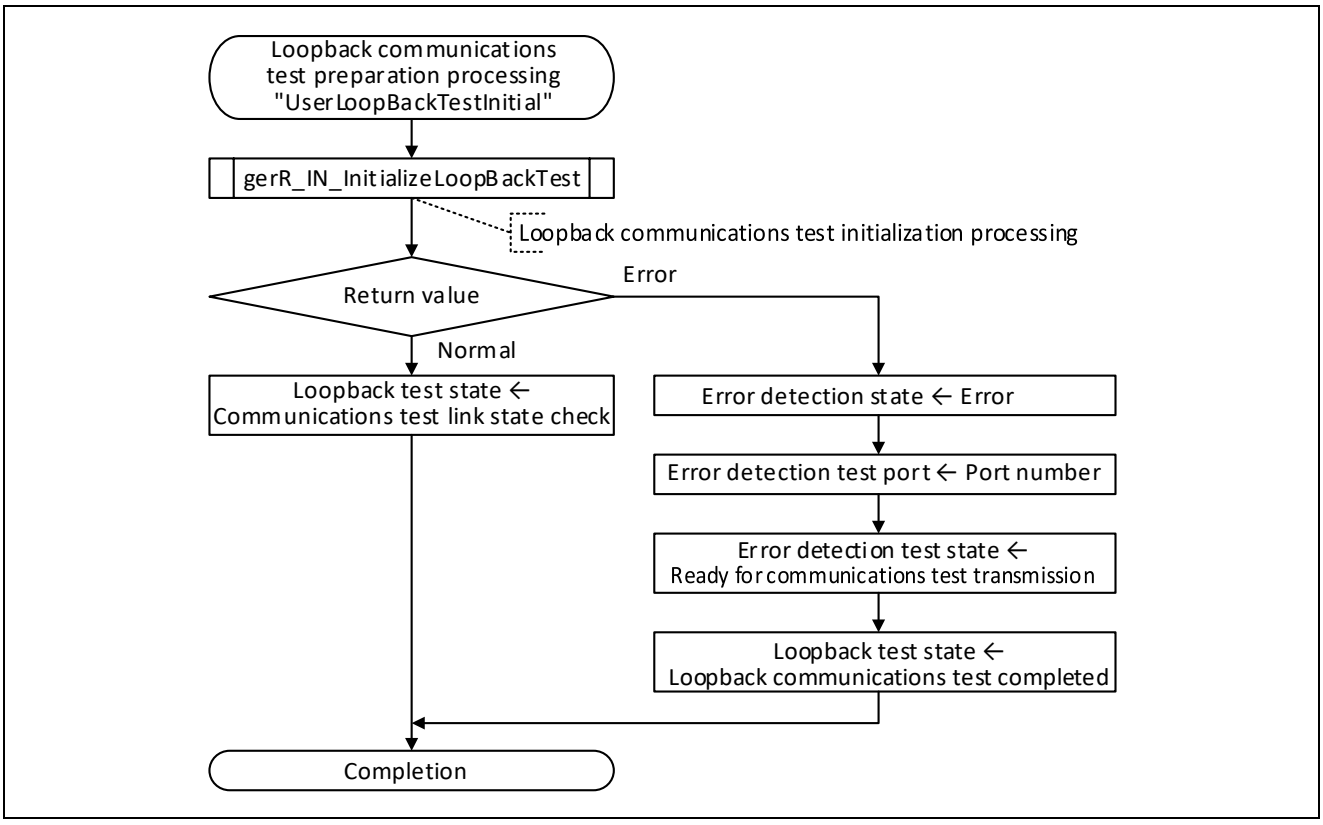


Figure 5.100 Flowchart for Loopback Communications Test Preparation Processing

5.11.4 Loopback Communications Test Link State Check Processing

This function sets the communications port state for the loopback communications test.

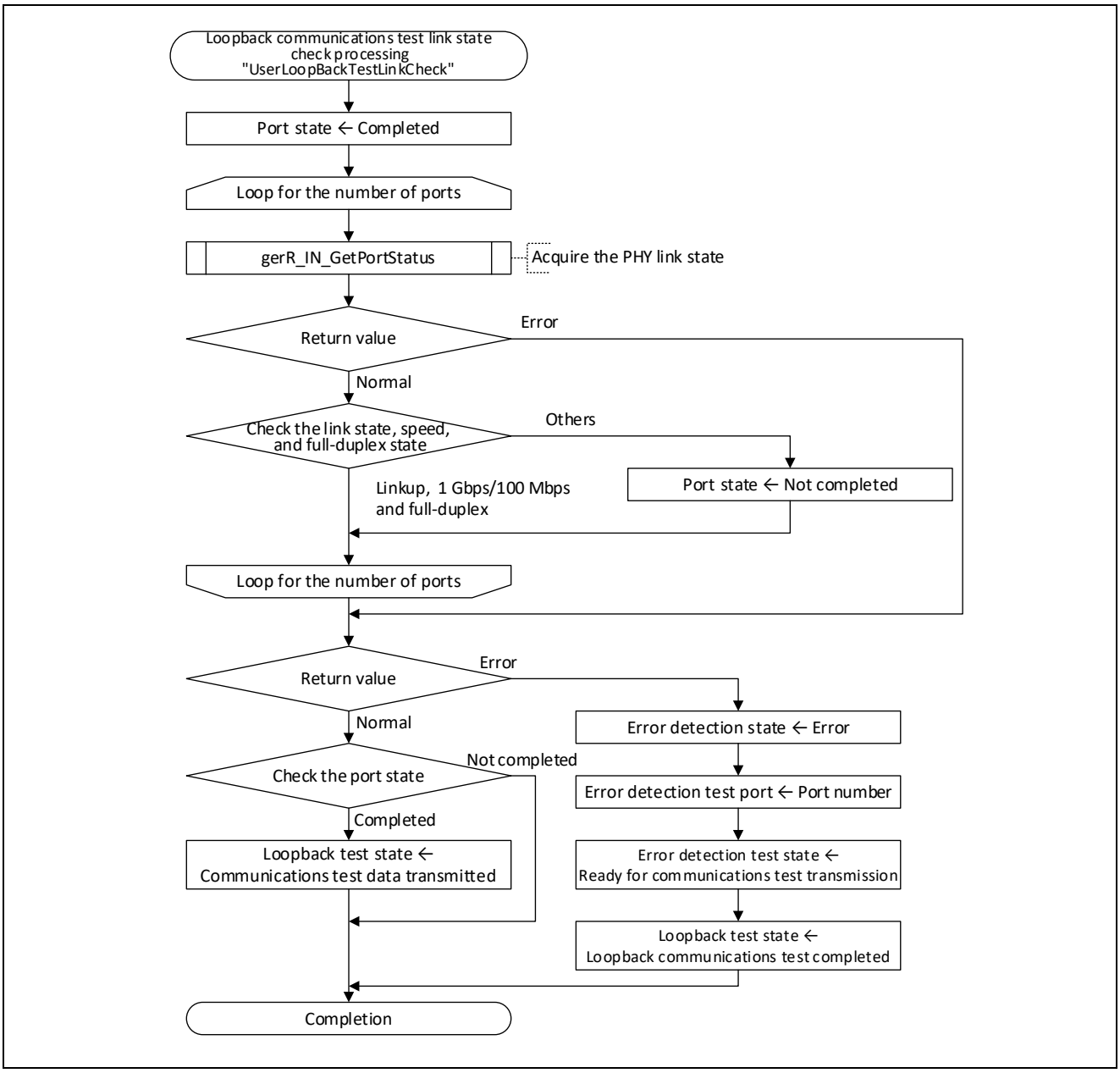


Figure 5.101 Flowchart for Loopback Communications Test Link State Check Processing

5.11.5 Loopback Communications Test Data Transmission Processing

This function creates and sends test data for use in the loopback communications test.

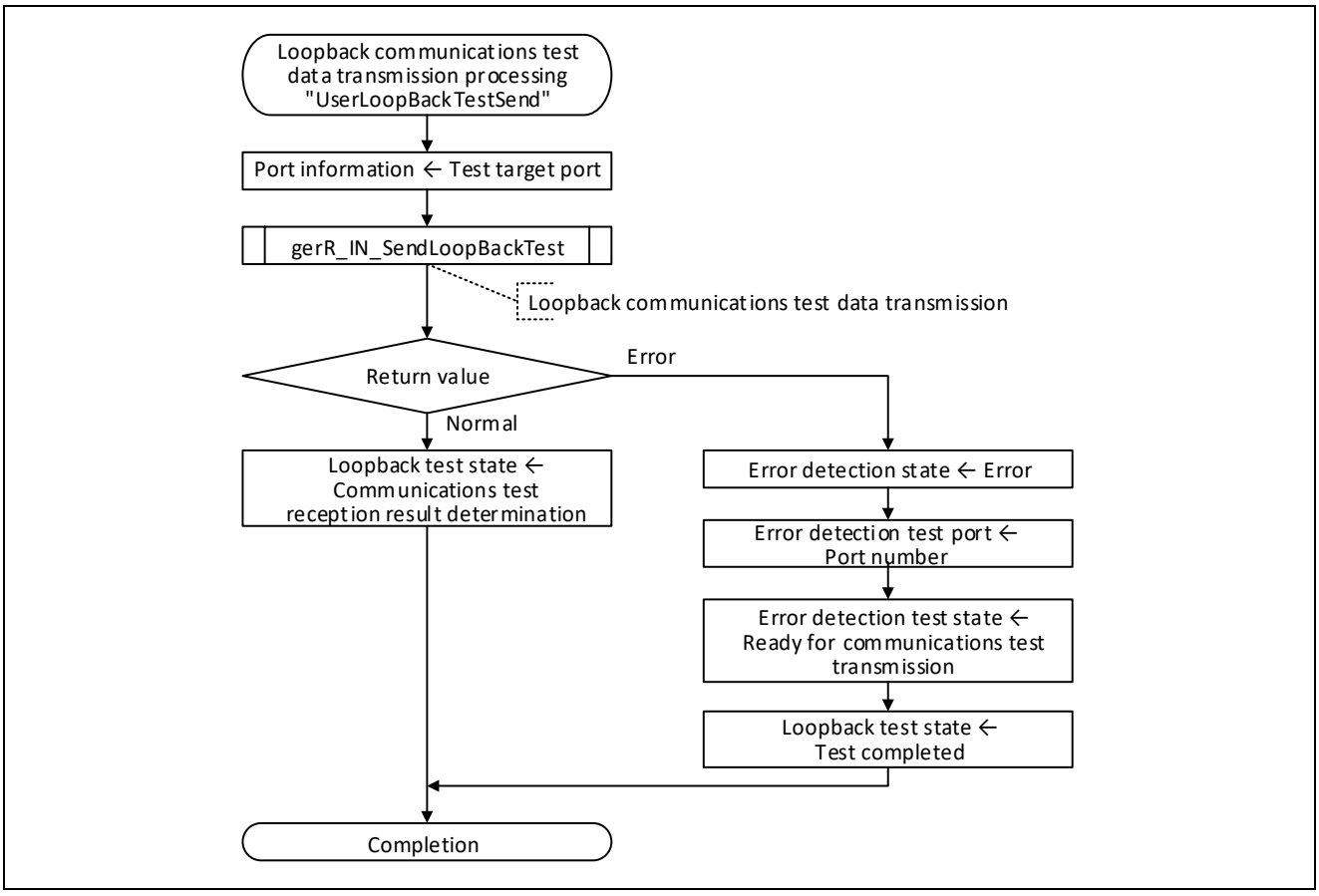


Figure 5.102 Flowchart for Loopback Communications Test Data Transmission Processing

5.11.6 Loopback Communications Test Reception Result Determination Processing

This function determines the validity of the test result from the received test data and register values.

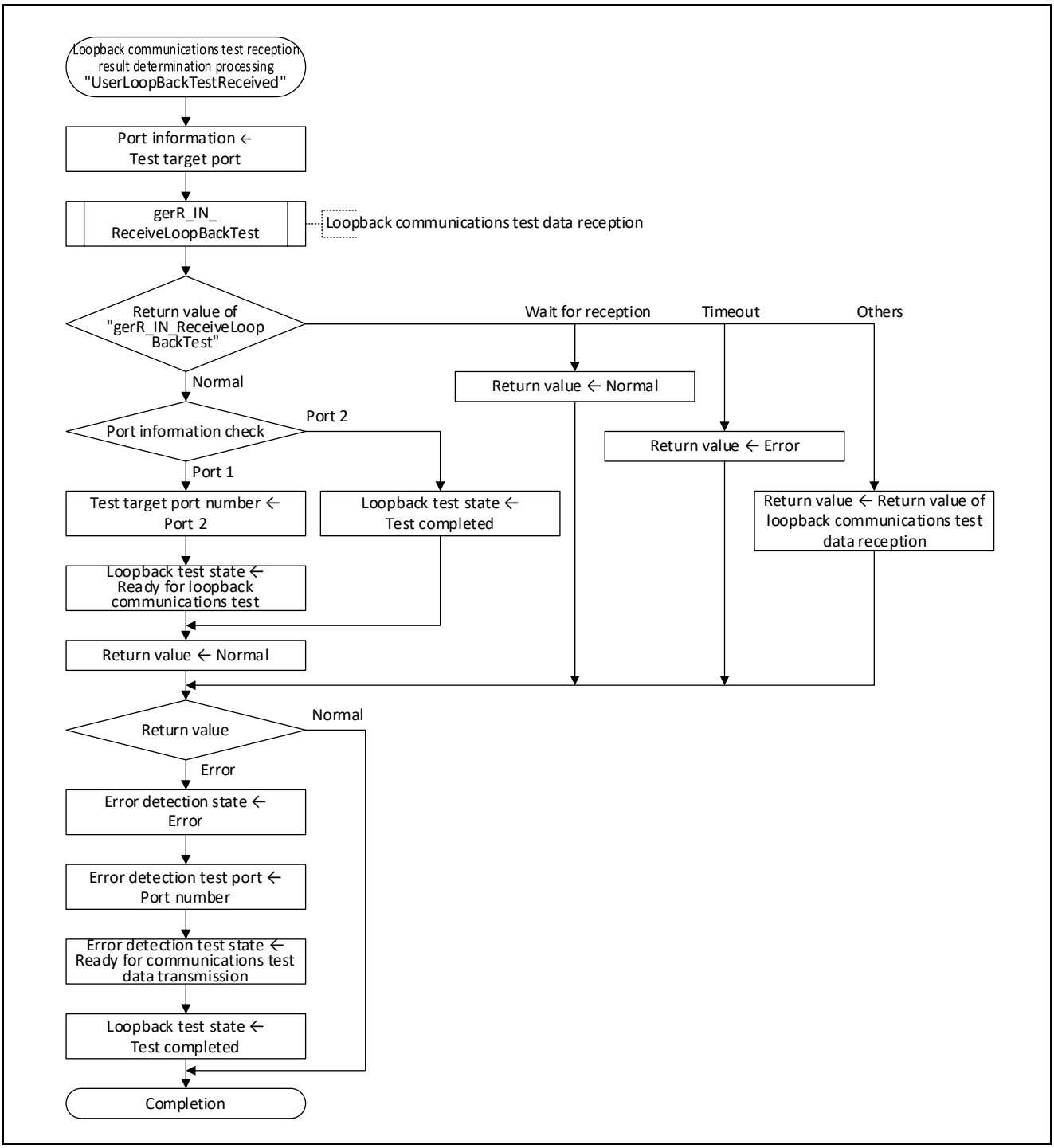


Figure 5.103 Flowchart for Loopback Communications Test Reception Result Determination Processing

5.11.7 Loopback Communications Test Completion Processing

This function notifies the user of the completion of the hardware test processing (loopback communications test).

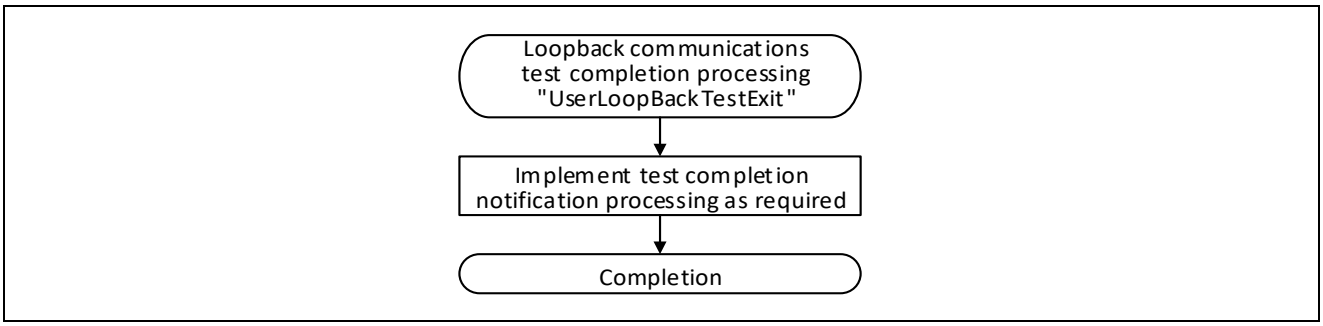


Figure 5.104 Flowchart for Loopback Communications Test Completion Processing

6. Specifications of the R-IN32M4-CL3 Driver Functions

This section describes the specifications of the R-IN32M4-CL3 driver interface functions and R-IN32M4-CL3 driver callback functions which constitute the R-IN32M4-CL3 driver.

6.1 Overview of the Functions

(1) Overview

The table below gives an overview of the functions and indicates whether or not changes are required.

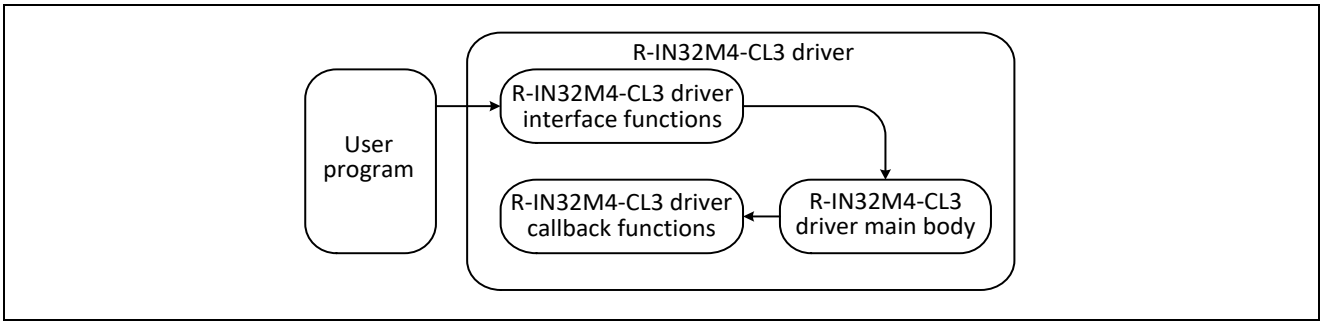


Figure 6.1 Relation of the Functions

Table 6.1 Overview of the Functions

Program Part Name	Overview	Need for Change
R-IN32M4-CL3 driver interface function	A function that is called when the functions of the R-IN32M4-CL3 driver are used from the user program. (File: R_IN_Interface.c)	—
R-IN32M4-CL3 driver callback function	A function that is used when the user program requests a callback to the R-IN32M4-CL3 driver. Describes processing of the user program for events which would occur in the R-IN32M4-CL3 driver. (File: R_IN32M4_CL3_Callback.c)	✓
R-IN32M4-CL3 driver main body	The main body of the driver that is called by R-IN32M4-CL3 driver interface function and controls R-IN32M4-CL3. (Files: Files below the driver folder excluding R_IN_Interface.c)	—

(2) Description Specifications

The table below lists the description specifications of the functions.

Table 6.2 Source Code Description Specifications

Item	Description	Remarks
C language standard	ANSI C compliant	The extended specifications of the compiler maker are partly used.
Character encoding	ASCII, Japanese (Shift_JIS)	—
Tab length	4	—
Return code	CR+LF	—

(3) Type Definition and Error Code

The table below lists the types and error codes defined by the R-IN32M4-CL3 driver.

Table 6.3 R-IN32M4-CL3 Driver Type List

No.	Defined Type	Implementation
1	VOID	void
2	CHAR	char
3	UCHAR	unsigned char
4	SHORT	short
5	USHORT	unsigned short
6	INT	int
7	UINT	unsigned int
8	LONG	long
9	ULONG	unsigned long
10	ERRCODE	int
11	BOOL	int

Table 6.4 R-IN32M4-CL3 Driver Error Code List

No.	Symbol	Value	Description
1	R_IN_OK	0	Normal
2	R_IN_BUSY	1	Operating
3	R_IN_ERR	-1	Abnormal completion (state error/mismatch)
4	R_IN_ERR_OTHER	-2	(Error occurred in driver inside library)
5	R_IN_ERR_OUTOFRANGE	-3	Out of range
6	R_IN_ERR_EMPTY	-4	Empty
7	R_IN_ERR_OVERFLOW	-5	Overflow
8	R_IN_ERR_NOENTRY	-6	No entry
9	R_IN_ERR_NOPERMIT	-7	Not permitted
10	R_IN_ERR_NODATA	-8	No data
11	R_IN_ERR_STSW	-9	No valid information of the home station state
12	R_IN_ERR_BOUNDARY	-10	Boundary specification error
13	R_IN_ERR_SEMAPHORE	-11	Semaphore acquisition failure
14	R_IN_ERR_FATAL	-12	Fatal error occurred
15	R_IN_ERR_TIMEOUT	-13	Timeout

6.2 R-IN32M4-CL3 Driver Interface Function List

The table below lists the R-IN32M4-CL3 driver interface functions.

Table 6.5 R-IN32M4-CL3 Driver Interface Function List

Function Category (Ref. Section)	Function Name	Function Type	Overview
Initial setup (section 6.4.1)	gulR_IN_GetResetStatus	ULONG	Acquiring the reset state
	gerR_IN_Initialize	ERRCODE	Initializing R-IN32M4-CL3
	gerR_IN_SetIPAddress	ERRCODE	Setting the IP address
	gerR_IN_Start	ERRCODE	Starting R-IN32M4-CL3 communications
	gerR_IN_MIBLedTableDefine	ERRCODE	LED state information MIB definition
	gerR_IN_RegistIPAddressFilteringFunction	VOID	Registering IP address filtering processing
	gerR_IN_RegistCallback	ERRCODE	Registering callback processing
Watchdog timer (section 6.4.2)	gerR_IN_ResetWDT	ERRCODE	Resetting the R-IN32M4-CL3 internal WDT
	gerR_IN_SetWDT	ERRCODE	Setting the R-IN32M4-CL3 internal WDT time limit
	gerR_IN_DisableWDT	ERRCODE	Disabling the R-IN32M4-CL3 internal WDT
	gerR_IN_EnableWDT	ERRCODE	Enabling the R-IN32M4-CL3 internal WDT
Event (section 6.4.3)	gerR_IN_GetEvent	ERRCODE	R-IN32M4-CL3 event detection
	gerR_IN_Main	ERRCODE	R-IN32M4-CL3 event detection main processing
	gerR_IN_RestartEvent	ERRCODE	Restarting R-IN32M4-CL3 event detection
Cyclic transfer (section 6.4.4)	gerR_IN_GetMasterNodeStatus	ERRCODE	Acquiring the master station state
	gerR_IN_GetHoldClearFlag	ERRCODE	Acquiring the hold/clear flag
	gerR_IN_GetReceivedCyclicData	ERRCODE	Acquiring cyclic received data (batch)
	gerR_IN_UpdateReceivedCyclicData	ERRCODE	Updating cyclic received data
	gerR_IN_FinishReceivedCyclicDataAcquisition	ERRCODE	Completion of cyclic received data acquisition
	gerR_IN_GetReceivedCyclicRY	ERRCODE	Acquiring cyclic received data (RY)
	gerR_IN_GetReceivedCyclicRWw	ERRCODE	Acquiring cyclic received data (RWw)
	gerR_IN_GetReceivedCyclicRspdu	ERRCODE	Acquiring cyclic received data (RSPDU)
	gerR_IN_SetSendCyclicData	ERRCODE	Setting data for cyclic transmission (batch)
	gerR_IN_FinishSendCyclicDataSetting	ERRCODE	Completion of setting data for cyclic transmission
	gerR_IN_SetSendCyclicRX	ERRCODE	Setting data for cyclic transmission (RX)
	gerR_IN_SetSendCyclicRWr	ERRCODE	Setting data for cyclic transmission (RWr)
	gerR_IN_SetSendCyclicSspdu	ERRCODE	Setting data for cyclic transmission (SSPDU)
	gerR_IN_GetReceivedCyclicBuffer	ERRCODE	Acquiring the cyclic reception buffer
	gerR_IN_FinishReceivedCyclicBuffer	ERRCODE	Completion of cyclic reception buffer acquisition
	gerR_IN_GetSendCyclicBuffer	ERRCODE	Acquiring the cyclic transmission buffer
	gerR_IN_GetSplitSendCyclicBuffer	ERRCODE	Acquiring the cyclic split transmission buffer
	gerR_IN_FinishSendCyclicBuffer	ERRCODE	Completion of setting the cyclic transmission buffer
	gvR_IN_SendCyclicFrameClassA	VOID	Cyclic frame send (CC-Link IE TSN Class A)
Home station state setup (section 6.4.5)	gerR_IN_SetNodeStatus	ERRCODE	Setting the home station state
Home station state acquisition (section 6.4.6)	gerR_IN_GetIPAddress	ERRCODE	Acquiring the IP address
	gerR_IN_GetCurrentCyclicSize	ERRCODE	Acquiring the cyclic transfer size specified by the master station
	gerR_IN_GetCyclicStatus	ERRCODE	Acquiring the cyclic transfer state

Function Category (Ref. Section)	Function Name	Function Type	Overview
Home station state acquisition (section 6.4.6)	gerR_IN_GetCommumicationStatus	ERRCODE	Acquiring the data link state
	gerR_IN_GetPortStatus	ERRCODE	Acquiring the PHY link state
	gerR_IN_GetStatisticalInformation	ERRCODE	Acquiring statistical information
	gerR_IN_ClearStatisticalInformation	ERRCODE	Clearing statistical information
	gulR_IN_GetNetworkTopology	ULONG	Acquiring network topology information
	gerR_IN_GetNodeOperationMode	ERRCODE	Acquiring the home station's operating mode
	gerR_IN_GetLinkSpeed	ERRCODE	Communication speed acquisition
LED control (section 6.4.7)	gerR_IN_SetUSER1LED	ERRCODE	LED lighting control (User LED 1)
	gerR_IN_SetUSER2LED	ERRCODE	LED lighting control (User LED 2)
	gerR_IN_SetRUNLED	ERRCODE	LED lighting control (RUN)
	gerR_IN_SetERRLED	ERRCODE	LED lighting control (ERR)
	gerR_IN_SetLERR1LED	ERRCODE	LED lighting control (L ER1 LED)
	gerR_IN_SetLERR2LED	ERRCODE	LED lighting control (L ER2 LED)
	gerR_IN_DisableLED	ERRCODE	Disabling the LED lighting function
	gerR_IN_EnableLED	ERRCODE	Enabling the LED lighting function
	gerR_IN_UpdateLedStatus	ERRCODE	Updating the communications state display LED
	gerR_IN_SetSDRDLEDMode	ERRCODE	Setting SD/RD lightening mode
	gerR_IN_StartTestLED	ERRCODE	Starting the LED test
	gerR_IN_ExecuteTestLED	ERRCODE	Executing the LED test
	gerR_IN_StopTestLED	ERRCODE	Ending the LED test
Network time (section 6.4.8)	gerR_IN_GetNetworkTime	ERRCODE	Acquiring the network time (serial value)
	gerR_IN_NetworkTimeToDate	ERRCODE	Network time (serial value) to clock information conversion
	gerR_IN_DateToNetworkTime	ERRCODE	Clock information to network time (serial value) conversion
	gerR_IN_GetUnixTime	ERRCODE	Acquiring the network time (UNIX time)
	gerR_IN_UnixTimeToDate	ERRCODE	Network time (UNIX time) to clock information conversion
	gerR_IN_DateToUnixTime	ERRCODE	Clock information to network time (UNIX time) conversion
	gerR_IN_SetUnixTimeClassA	ERRCODE	Network time (UNIX time) setting (CC-Link IE TSN Class A)
	gvR_IN_SetUnixOffsetTime	VOID	Setting the network time offset
	gerR_IN_GetUnixOffsetTime	ERRCODE	Acquiring the network time offset
MDIO access (section 6.4.9)	gerR_IN_EnableMACIPAccess	ERRCODE	Enabling MAC IP access
	gerR_IN_DisableMACIPAccess	ERRCODE	Disabling MAC IP access
	gerR_IN_WritePhy	ERRCODE	PHY internal register writing
	gerR_IN_ReadPhy	ERRCODE	PHY internal register reading
SLMP transmission/ reception (section 6.4.10)	gerR_IN_ReceivedSlmpMain	ERRCODE	SLMP reception main processing
	gvR_IN_ReceivedSlmpExecution	VOID	SLMP reception command execution
	gblR_IN_GetReceiveSlmpStatus	BOOL	Acquiring the SLMP reception enabled state for user reasons
	gerR_IN_SetReceiveSlmpStatus	ERRCODE	Setting to enable SLMP reception for user reasons
	gerR_IN_SetSlmpCommand	ERRCODE	SLMP command determination registration

Function Category (Ref. Section)	Function Name	Function Type	Overview
SLMP transmission/ reception (section 6.4.10)	gerR_IN_SendSlmpFrame	ERRCODE	SLMP frame transmission
	gerR_IN_WriteSlmpSendBuffer	ERRCODE	SLMP transmission buffer writing
	gblR_IN_GetSlmpSendBufferUsed	BOOL	SLMP transmission buffer usage acquisition
	gerR_IN_SetSlmpResponseFrameSendRequest	ERRCODE	SLMP response frame transmission request
	gvR_IN_ReleaseSlmpReceiveFrame	VOID	SLMP reception buffer release request
SLMP command execution (section 6.4.11)	gvR_IN_ExecuteReset	VOID	Resetting the home station
	gerR_IN_StartSlmpRequestTimer	ERRCODE	Starting the SLMP request waiting timer
	gerR_IN_StopSlmpRequestTimer	ERRCODE	Stopping the SLMP request waiting timer
	gerR_IN_GetMasterIPAddress	ERRCODE	Acquiring the management master station's IP address
	gerR_IN_CheckIpAddressSlmp	ERRCODE	Checking the IP address
Error history (section 6.4.12)	gerR_IN_SetErrorHistory	ERRCODE	Registering the error history
	gvR_IN_ClearErrorHistory	VOID	Clearing the error history
	gvR_IN_ClearErrorHistoryController	VOID	Error history clear (controller information)
	gerR_IN_SetErrorHistoryOption	ERRCODE	Error history registration (option information)
	gerR_IN_ClearErrorHistoryOption	ERRCODE	Error history clear (option information)
Hardware test (section 6.4.13)	gerR_IN_IEEETest	ERRCODE	IEEE802.3ab compliance test
	gerR_IN_InitializeLoopBackTest	VOID	Loopback communications test initialization
	gerR_IN_SendLoopBackTest	ERRCODE	Loopback communications test data transmission
	gerR_IN_ReceiveLoopBackTest	ERRCODE	Loopback communications test data reception
General-purpose common (section 6.4.14)	gverR_IN_CopyMemory	ERRCODE	Memory copy
	gerR_IN_FillMemory	ERRCODE	Memory fill (8-bit)
	gerR_IN_FillMemory16	ERRCODE	Memory fill (16-bit)
	gerR_IN_FillMemory32	ERRCODE	Memory fill (32-bit)
	gerR_IN_EndianShort	ERRCODE	Endian conversion (USHORT)
	gerR_IN_EndianLong	ERRCODE	Endian conversion (ULONG)
	gerR_IN_EndianLongLong	ERRCODE	Endian conversion (ULONGULONG)
	gvR_IN_DisableInt	VOID	Disabling interrupts
	gvR_IN_EnableInt	VOID	Enabling interrupts
	gvR_IN_DisableDispatch	VOID	Disabling dispatching
	gvR_IN_EnableDispatch	VOID	Enabling dispatching
Network-synchronized communications (section 6.4.15)	gerR_IN_GetSyncDeviationFlag	ERRCODE	Time synchronization deviation detection
	gerR_IN_StopAppSyncSignal	ERRCODE	Stopping the output of the synchronization signal
	gerR_IN_SetWdcThreshold	ERRCODE	Setting the counter threshold for consecutive watchdog counter checking errors
	gerR_IN_MakeResponseWdcInfomation	ERRCODE	Creating watchdog counter information setting response data
	gvR_IN_IncrementWdcUL	VOID	Incrementing the watchdog counter (for transmission)
	gvR_IN_LatchWdcUL	VOID	Updating the watchdog counter latched value (for transmission)
	gerR_IN_GetWdcUL	ERRCODE	Watchdog counter UL acquisition
	gerR_IN_CheckWdcDL	ERRCODE	Checking the watchdog counter (for reception)

Function Category (Ref. Section)	Function Name	Function Type	Overview
CANopen communications (section 6.4.16)	gerR_IN_CanInit	ERRCODE	CANopen communications function initialization
	gerR_IN_CanGetValidObjectDictionaryNumber	ERRCODE	Valid object dictionary number acquisition
	gerR_IN_CanGetObjectHandle	ERRCODE	Object dictionary handling acquisition
	gerR_IN_CanReadObject	ERRCODE	Object dictionary reading
	gerR_IN_CanWriteObject	ERRCODE	Object dictionary writing
	gerR_IN_CanReadBlockObject	ERRCODE	Object dictionary block reading
	gerR_IN_CanWriteBlockObject	ERRCODE	Object dictionary block writing
	gerR_IN_CanGetNmtState	ERRCODE	Acquiring the NMT state
	gerR_IN_CanSetNmtState	ERRCODE	Setting the NMT state
	gerR_IN_CanUpdateRPDO	ERRCODE	Updating RPDOs
	gerR_IN_CanUpdateTPDO	ERRCODE	Updating TPDOs
MCU-MCU interface (section 6.4.17)	gvR_IN_SchedulerMain	VOID	Scheduler task main
	gvR_IN_McuIfGpioSendMain	VOID	GPIO communication send task main
	gvR_IN_McuIfGpioResponseMain	VOID	GPIO communication response receive task main
	gvR_IN_McuIfGpioReceivedMain	VOID	GPIO communication receive task main
	gvR_IN_McuIfRtDmaCompleteMain	VOID	RT DMA transfer completion task main
	gvR_IN_McuIfInitial	VOID	MCU-MCU interface initial
	gerR_IN_McuIfSetCommand	ERRCODE	MCU-MCU interface registration processing
	gerR_IN_McuIfCheckTransferRtDma	ERRCODE	RT DMA transfer request check
	gerR_IN_McuIfStartSendCsih	ERRCODE	DMA transfer ready (for send)
	gerR_IN_McuIfStartReceiveCsih	ERRCODE	DMA transfer ready (for receive)
	gerR_IN_McuIfStopReceiveCsih	ERRCODE	CSIH receive stop
	gerR_IN_McuIfStopSendCsih	ERRCODE	CSIH send stop
	gerR_IN_McuIfCheckGpioState	ERRCODE	GPIO communication status check
	gerR_IN_McuIfGetGpioState	ERRCODE	GPIO communication status acquisition
	gvR_IN_McuIfClearGpioState	VOID	GPIO communication status clear
	gerR_IN_McuIfGpioSend	ERRCODE	GPIO communication data create request

6.3 R-IN32M4-CL3 Driver Tasks

The following table lists generation information to be stated in the RTOS configuration file. For details of the settings, refer to "R-IN32M4-CL3 Programming Manual: OS edition".
The settings should not be changed unless there is a specific reason.

Before creating a new task, check the generation information described in Section 5.2 "User Program Tasks".

(1) Task List

Table 6.6 R-IN32M4-CL3 Driver Task List

No.	Task Name	Task ID	T_CTSK Structure Member					
			Task Attribute	Extension Info.	Start Address (Described by function name)	Start Priority	Stack Size	Start Address of Stack Area
1	Low-priority interrupt task	TSKID_NX_LOW_INT (33)	TA_HLNG	0000H	vNX_Task_CciefNx_Low	2	800H	NULL
2	High-priority interrupt task	TSKID_NX_HIGH_INT (34)	TA_HLNG	0000H	vNX_Task_CciefNx_High	1	400H	NULL
3	Non-cyclic frame reception task	TSKID_NX_NCYC_RX (35)	TA_HLNG	0000H	vNX_Task_RxNonCycFrame	5	1000H	NULL
4	Non-cyclic frame transmission task	TSKID_NX_NCYC_TX (36)	TA_HLNG	0000H	vNX_Task_TxNonCycFrame	4	1800H	NULL
5	Communications driver periodic processing task	TSKID_NX_PERIODIC (37)	TA_HLNG	0000H	vNX_Task_ComDriverPeriodic	8	1000H	NULL
6	USnet periodic processing task	TSKID_NX_IPCOMM (38)	TA_HLNG	0000H	vNX_Task_IpFrameComm	6	400H	NULL
7	TSN periodic processing task	TSKID_NX_TSN (39)	TA_HLNG	0000H	vNX_Task_TSNPeriodic	7	1000H	NULL
8	Relay RAM clear task	TSKID_NX_RLYRAMCLR (40)	TA_TPRI	0000H	vNX_Task_RelayRamClr	11	400H	NULL

(2) Semaphore

Table 6.7 Semaphore Generation Information

No.	Name	ID	Semaphore Attribute*1	Initial Value of Number of Resources	Max. Number of Resources
1	Semaphore 1 for PTP provision	SEMID_NX_TSN_1(65)	TA_TFIFO	1	1
2	Semaphore 2 for PTP provision	SEMID_NX_TSN_2(66)	TA_TFIFO	1	1
3	Semaphore for PHY access	SEMID_NX_PHYACCESS(67)	TA_TFIFO	1	1
4	Relay RAM clear semaphore	SEMID_NX_RLYRAMCLR(68)	TA_TFIFO	1	1

Note 1. µITRON defines semaphore attributes as follows (the µITRON specification, ver 4.03.03, P.75).
TA_TFIFO (00H): Task queues are managed in FIFO order.
TA_TPRI (01H): Task queues are managed in the priority order of the tasks.

(3) Hardware ISR Settings

Table 6.8 Hardware ISR Settings

No.	Hardware ISR Name	Target Interrupt Number	Exception No.	Service Call to be Automatically Executed in Response to an Interrupt	Target Object ID of the Service Call to be Automatically Executed
1	Non-cyclic reception DMA transfer completion	24 (General-purpose DMAC channel 2 transfer complete interrupt)	40	HWISR_SET_FLG	EVFLGID_NX_DMACOMP02
2	Non-cyclic transmission DMA transfer completion	25 (General-purpose DMAC channel 3 transfer complete interrupt)	41	HWISR_SET_FLG	EVFLGID_NX_DMACOMP03
3	Communications driver periodic processing period timer	29 (TAUD channel 2 interrupt)	45	HWISR_WUP_TSK	TSKID_NX_PERIODIC
4	IP frame communication periodic timer	30 (TAUD channel 3 interrupt)	46	HWISR_WUP_TSK	TSKID_NX_IPCOMM
5	Non-cyclic frame transmission periodic timer	31 (TAUD channel 4 interrupt)	47	HWISR_WUP_TSK	TSKID_NX_NCYC_TX
6	Non-cyclic frame reception periodic timer	58 (INTPZ11 input/TAUD channel 5 interrupt)	74	HWISR_WUP_TSK	TSKID_NX_NCYC_RX
7	Time synchronization control periodic timer	59 (INTPZ12 input/TAUD channel 6 interrupt)	75	HWISR_WUP_TSK	TSKID_NX_TSN
8	TSN high-priority interrupt periodic timer	111 (TSN high-priority interrupt)	127	HWISR_WUP_TSK	TSKID_NX_HIGH_INT
9	TSN low-priority interrupt periodic timer	112 (TSN low-priority interrupt)	128	HWISR_WUP_TSK	TSKID_NX_LOW_INT

6.4 R-IN32M4-CL3 Driver Interface Function Details

This section describes how to use the R-IN32M4-CL3 driver interface functions and the details of related functions

6.4.1 Initial Setup

(1) gulR_IN_GetResetStatus

Function	Acquiring the reset state			
Call Format	ULONG gulR_IN_GetResetStatus (VOID)			
Arguments	Type Name	Variable Name	Description	I/O
	—	—	—	—
Return Value	R_IN_RESET_PWRON (1): Power-on reset R_IN_RESET_SYSTEM (2): System reset			
Description	This function acquires the reset state. Call this function before calling the gerR_IN_Initialize function (6.4.1(2)).			

(2) gerR_IN_Initialize

Function	Initializing R-IN32M4-CL3			
Call Format	ERRCODE gerR_IN_Initialize (const UCHAR *puchMACAddress, const R_IN_UNITINFO_T *pstUnitInfo, const R_IN_UNITINIT_T *pstUnitInit)			
Arguments	Type Name	Variable Name	Description	I/O
	const UCHAR*	puchMACAddress	Home station MAC address Set as follows for 12-34-56-78-90-AB: puchMACAddress [0]: 12H puchMACAddress [1]: 34H puchMACAddress [2]: 56H puchMACAddress [3]: 78H puchMACAddress [4]: 90H puchMACAddress [5]: ABH	Input
	const R_IN_UNITINFO_T*	pstUnitInfo	R-IN32M4-CL3 unit information For details, refer to Table 6.9, R_IN_UNITINFO_T List.	Input
	const R_IN_UNITINIT_T*	pstUnitInit	R-IN32M4-CL3 initial setting information For details, refer to Table 6.17, R_IN_UNITINIT_T List.	Input
Return Value	R_IN_OK: Normal completion R_IN_ERR: Abnormal completion (parameter error) R_IN_ERR_OUTOFRANGE: Abnormal completion (out of range)			
Description	This function initializes R-IN32M4-CL3.			

The following shows the configuration of the argument "R_IN_UNITINFO_T" of gerR_IN_Initialize.

Table 6.9 R_IN_UNITINFO_T List

No.	Member		Overview	Setting
1	USHORT	usMaxRySize	RY size	Specifies the RY size (bytes) transferable by the home station in 1-byte units.
2	USHORT	usMaxRWwSize	RWw size	Specifies the RWw size (words) transferable by the home station in 2-word units.
3	USHORT	usMaxRspduSize	RSPDU size	Specifies the RSPDU size (bytes) communicable by the own station in increments of 1 byte. This is used when the safety PDU send/receive is performed (the compiler switch "SAFETY_PDU_ENABLE" is enabled).
4	USHORT	usMaxRxSize	RX size	Specifies the RX size (bytes) transferable by the home station in 1-byte units.
5	USHORT	usMaxRWrSize	RWr size	Specifies the RWr size (words) transferable by the home station in 2-word units.
6	USHORT	usMaxSspduSize	SSPDU size	Specifies the SSPDU size (bytes) communicable by the own station in increments of 1 byte. This is used when the safety PDU send/receive is performed (the compiler switch "SAFETY_PDU_ENABLE" is enabled).
7	UCHAR	uchMyStationPortTotalNumber	Number of home station ports	Specifies the number of physical CC-Link IE TSN ports of the home station. Set this to 2 or 1.
8	USHORT	usNetVersion	Network firmware version	The firmware version can be defined arbitrarily by the user.
9	USHORT	usNetModelType	Network model type	Specifies the model type (deviceType) specified by the CC-Link Partner Association. Refer to section 3.2, Acquiring a Vendor Code and Selecting a Device Type.
10	ULONG	UlNetUnitModelCode	Network model code	The model code is a code defined arbitrarily by the user. Manage the code so that it is unique within the same vendor code.
11	USHORT	usNetVendorCode	Network vendor code	Specifies the vendor code (vendorCode) acquired when the vendor became a member of the CC-Link Partner Association, in BCD. (If the vendor code is 5678, 5678h is specified.) Refer to section 3.2, Acquiring a Vendor Code and Selecting a Device Type.
12	UCHAR	auchNetUnitModelName [20]	Network model name	The model name is a name defined arbitrarily by the user. Manage the name so that it is unique within the same vendor code. (20-byte character string (ASCII code))
13	UCHAR	auchNetVendorName [32]	Network vendor name	The vendor name (such as company name or brand name) can be arbitrarily defined by the user. (32-byte character string (ASCII code))

No.	Member		Overview	Setting Description
14	UCHAR	uchNetHwVersion	Network hardware version	The hardware version can be defined arbitrarily by the user.
15	USHORT	usNetDeviceVersion	Network device version	The device version indicates the version of the function of the developed device. Used for mapping the developed device and the CSP+ file.
16	BOOL	blInformationFlag	Controller information presence flag	Enables/disables the arguments No.17 to No.22 and No.25 to No.28 in this table. R_IN_FALSE indicates disable, and R_IN_TRUE indicates enable. The flag is disabled when the device only has the communication function.
17	UCHAR	uchCtrlVersion	Controller firmware version	The firmware version can be defined arbitrarily by the user.
18	USHORT	usCtrlModelType	Controller model type	Specifies the model type (deviceType) specified by the CC-Link Partner Association. Refer to section 3.2, Acquiring a Vendor Code and Selecting a Device Type.
19	ULONG	ulCtrlUnitModelCode	Controller model code	The model code is a code defined arbitrarily by the user. Manage the code so that it is unique within the same vendor code.
20	USHORT	usCtrlVendorCode	Controller vendor code	Specifies the vendor code (vendorCode) acquired when the vendor became a member of the CC-Link Partner Association, in BCD. Refer to section 3.2, Acquiring a Vendor Code and Selecting a Device Type.
21	UCHAR	auchCtrlUnitModelName [20]	Controller model name	The model name is a name defined arbitrarily by the user. Manage the name so that it is unique within the vendor code. (20-byte character string (ASCII code))
22	UCHAR	auchCtrlVendorName [32]	Controller vendor name	The vendor name can be defined arbitrarily by the user. (32-byte character string (ASCII code))
23	USHORT	usNetExUnitModelCode	Network model extension code	The network extension code can be defined arbitrarily by the user.
24	UCHAR	auchNetSerialNumber [32]	Network serial number	The serial number can be defined arbitrarily by the user. (32-byte character string (ASCII code))
25	UCHAR	uchCtrlDeviceVersion	Controller device version	The device version indicates the version of the function of the developed device. Used for mapping the developed device and the CSP+ file.
26	UCHAR	uchCtrlHwVersion	Controller hardware version	The hardware version can be defined arbitrarily by the user.
27	UCHAR	auchCtrlSerialNumber [32]	Controller serial number	The serial number can be defined arbitrarily by the user. (32-byte character string (ASCII code))
28	USHORT	usCtrlExUnitModelCode	Controller model extension code	The model extension code can be defined arbitrarily by the user.
29	USHORT	usStationMode	Station mode	Sets the information to identify COMM_IF of CSP+. (0000h to FFFFh)

No.	Member		Overview	Setting Description
30	BOOL	blOptionInfoFlag	Option information presence flag	Enables/disables "Number of option information entries" and "Option information table". R_IN_FALSE indicates disable, and R_IN_TRUE indicates enable. Set this flag to R_IN_FALSE when a slice remote I/O module or an extension module is not used. Set this flag to R_IN_TRUE when a slice remote I/O module or an extension module is used.
31	UCHAR	uchNumberOfOption	Number of option information entries	Specifies the number of slice remote I/O modules and extension modules used.
32	R_IN_OPTIONINFO_T	astOptionInfo[R_IN_OPTION TABLE_ENTRY_SIZE]	Option information table	For details, refer to Table 6.10, R_IN_OPTIONINFO_T List.
33	R_IN_PTPINFO_T	stPtpInfo	Time synchronization related information	For details, refer to Table 6.11, R_IN_PTPINFO_T List.
34	ULONG	ulCorrespondingFunction	Corresponding function	Specify the support status of the function. For details, refer to the following. CC-Link IE TSN Class A: "Table 6.12" CC-Link IE TSN Class B: "Table 6.13"
35	USHORT	usNumberOfStationSubID	Number of station sub-IDs	Set the number of station sub-ID entries. (0001F to 1000H)
36	R_IN_WDCINFO_T	stWdcInfo	Watchdog counter information	Set for network synchronous communications. For details, refer to Table 6.14, R_IN_WDCINFO_T List.
37	ULONG	ulLinkSpeed	Communication speed	Specify the speed of communications. The setting is common to two ports. 1 Gbps: R_IN_SPEED_1G (0) 100 Mbps: R_IN_SPEED_100M (1)
38	USHORT	usCCIETSNClass	CC-Link IE TSN Class	Set the CC-Link IE TSN Class. CC-Link IE TSN Class A: 0 CC-Link IE TSN Class B: 1
39	USHORT	usCyclicDataUpdatePeriodic	Cyclic data update cycle	Set the execution interval of the fixed scan processing task. Initial value: 200 (μs)
40	ULONG	ullCommunicationCycleMin	Communication cycle minimum value (1 Gbps)	Set the minimum value of the communication cycle. When changing the initial value, be sure to check that there is no problem with the user application control. (Initial value) CC-Link IE TSN Class A: 1000000 (1000 μs) CC-Link IE TSN Class B: 31250 (31.25 μs) For CC-Link IE TSN Class B, do not set a value smaller than the initial value.

No.	Member		Overview	Setting Description
41	ULONG	ullCommunicationCycleMax 1G	Communication cycle maximum value (1 Gbps)	Set the maximum value of the communication cycle. When changing the initial value, be sure to check that there is no problem with the user application control. (Initial value) CC-Link IE TSN Class A: 6400000000 (6400 ms) CC-Link IE TSN Class B: 10000000 (10 ms)
42	ULONG	ullCommunicationCycleMin 100M	Communication cycle minimum value (100 Mbps)	Set the minimum value of the communication cycle. When changing the initial value, be sure to check that there is no problem with the user application control. (Initial value) CC-Link IE TSN Class A: 1000000 (1000 µs) CC-Link IE TSN Class B: 500000 (500 µs) For CC-Link IE TSN Class B, do not set a value smaller than the initial value.
43	ULONG	ullCommunicationCycleMax 100M	Communication cycle maximum value (100 Mbps)	Set the maximum value of the communication cycle. When changing the initial value, be sure to check that there is no problem with the user application control. (Initial value) CC-Link IE TSN Class A: 6400000000 (6400 ms) CC-Link IE TSN Class B: 10000000 (10 ms) (Maximum value) 15000000000 (15 s) or less
44	USHORT	usCyclicMaxResponseTime	Time managed polling method response time	For CC-Link IE TSN Class A, set the interval in which the own station can respond to periodic polling from the master station with 2^ (this value). Use a value twice the execution interval of the fixed scan processing task (vTask_Periodic) as a guide. Initial value: 9 (29 = 512 µs) For CC-Link IE TSN Class B, set 0.

Table 6.10 R_IN_OPTIONINFO_T List

No.	Member		Overview	Setting
1	ULONG	ulOptionUnitModelCode	Option model code	The model code is a code defined arbitrarily by the user. Manage the code so that it is unique within the same vendor code.
2	USHORT	usOptionExUnitModelCode	Option model extension code	The model extension code can be defined arbitrarily by the user.
3	USHORT	usOptionVendorCode	Option vendor code	Specifies the vendor code (vendorCode) acquired when the vendor became a member of the CC-Link Partner Association, in BCD. (If the vendor code is 5678, 5678h is specified.) Refer to section 3.2, Acquiring a Vendor Code and Selecting a Device Type.
4	UCHAR	auchOptionSerialNumber[32]	Option serial number	The serial number can be defined arbitrarily by the user. (32-byte character string (ASCII code))
5	UCHAR	uchOptionDeviceVersion	Option device version	The option device version indicates the version of the function of the developed device. Used for mapping the developed device and the CSP+ file.
6	UCHAR	auchOptionModelName[20]	Option model name	The model name is a name defined arbitrarily by the user. Manage the name so that it is unique within the vendor code.
7	UCHAR	auchOptionVendorName[32]	Option vendor name	The vendor name can be defined arbitrarily by the user.

Table 6.11 R_IN_PTPINFO_T List

No.	Member		Overview	Setting
1	UCHAR	uchPriority1	Grandmaster priority 1	Priority information for determining the grandmaster station. The value is between 0 and 255, and a smaller value indicates a higher priority. If the value is 255, the device cannot be the grandmaster station. The value 0 is reserved for management and cannot be specified. CC-Link IE TSN Class A: 255 CC-Link IE TSN Class B: 127
2	ULONG	ulClockQuality	Clock quality	Sets the clock quality. For details on the possible values of each information, refer to "IEEE Std 1588-2008". First octet: clockClass (indicates the performance value of the clock source) Second octet: clockAccuracy (indicates the accuracy of the clock) Third and fourth octets: offsetScaledLogVariance (indicates the distributed value of the clock) CC-Link IE TSN Class A: FFFFFFFFH CC-Link IE TSN Class B: F8FE436AH
3	UCHAR	uchPriority2	Grandmaster priority 2	Second priority information for determining the grandmaster station. The value is between 0 and 255, and a smaller value indicates a higher priority, same as priority 1. The value 0 is reserved for management and cannot be specified. CC-Link IE TSN Class A: 255 CC-Link IE TSN Class B: 128

Table 6.12 USER_CORRESPONDING_FUNCTION_CLASS_A (CC-Link IE TSN Class A)

Bit	Overview	Setting	Initial Value	Remark
0	Local function	0b: Not supported 1b: Supported	0b	Changing the initial value is prohibited.
1	Setting overwrite	0b: Disabled 1b: Enabled	1b	Changing the initial value is prohibited.
2	Watchdog counter	0b: Not supported 1b: Supported	0b	Changing the initial value is prohibited.
3	Backup/restoration function	0b: Not supported 1b: Supported	0b	Changing the initial value is prohibited.
4	Safety communications	0b: Not supported 1b: Supported	0b	Set "1b" for safety communications.
5	Relay filter setting function	0b: Not supported 1b: Supported	1b	-
6	Network synchronous communication function	0b: Not supported 1b: Supported	0b	Changing the initial value is prohibited.
7 to 15	Reserved	-	-	-
16	Error history	0b: Not supported 1b: Supported	1b	-
17	Event history	0b: Not supported 1b: Supported	0b	Changing the initial value is prohibited.
18	Relay send prohibition function	0b: Not supported 1b: Supported	0b	Changing the initial value is prohibited.
19	Own station information storage function during relay	0b: Not supported 1b: Supported	1b	Changing the initial value is prohibited.
20	IEEE 802.1Qbv (IEEE 802.1AS)	0b: Not supported 1b: Supported	0b	Changing the initial value is prohibited.
21	IEEE 802.1Qbv (IEEE 1588)	0b: Not supported 1b: Supported	0b	Changing the initial value is prohibited.
22	Mesh topology	0b: Not supported 1b: Supported	0b	Changing the initial value is prohibited.
23	Ring topology	0b: Not supported 1b: Supported	0b	Changing the initial value is prohibited.
24	VLAN	0b: Not supported 1b: Supported	1b	-
25	Time managed polling method	0b: Not supported 1b: Supported	1b	Changing the initial value is prohibited.
26 to 31	Reserved	—		

Table 6.13 USER_CORRESPONDING_FUNCTION (CC-Link IE TSN Class B)

Bit	Overview	Setting	Initial Value	Remark
0	Local function	0b: Not supported 1b: Supported	0b	Changing the initial value is prohibited.
1	Setting overwrite	0b: Disabled 1b: Enabled	1b	Changing the initial value is prohibited.
2	Watchdog counter	0b: Not supported 1b: Supported	0b	Set "1b" for network synchronous communications.
3	Backup/restoration function	0b: Not supported 1b: Supported	0b	Changing the initial value is prohibited.
4	Safety communications	0b: Not supported 1b: Supported	0b	Set "1b" for safety communications.
5	Relay filter setting function	0b: Not supported 1b: Supported	1b	-
6	Network synchronous communication function	0b: Not supported 1b: Supported	0b	Set "1b" for network synchronous communications.
7 to 15	Reserved	-	-	-
16	Error history	0b: Not supported 1b: Supported	1b	-
17	Event history	0b: Not supported 1b: Supported	0b	Changing the initial value is prohibited.
18	Relay send prohibition function	0b: Not supported 1b: Supported	1b	-
19	Own station information storage function during relay	0b: Not supported 1b: Supported	1b	Changing the initial value is prohibited.
20	IEEE 802.1Qbv (IEEE 802.1AS)	0b: Not supported 1b: Supported	1b	-
21	IEEE 802.1Qbv (IEEE 1588)	0b: Not supported 1b: Supported	1b	-
22	Mesh topology	0b: Not supported 1b: Supported	0b	Changing the initial value is prohibited.
23	Ring topology	0b: Not supported 1b: Supported	1b	-
24	VLAN	0b: Not supported 1b: Supported	1b	-
25	Time managed polling method	0b: Not supported 1b: Supported	0b	Changing the initial value is prohibited.
26 to 31	Reserved	-	-	-

Table 6.14 R_IN_WDCINFO_T List

No.	Member		Description
1	R_IN_TRN_SPLD_WDC_INFO_T	stTrnSpldWdcInfo	WDC information (transmitted sub-payload) For details, refer to Table 6.15.
2	R_IN_RCV_SPLD_WDC_INFO_T	stRcvSpldWdcInfo	WDC information (received sub-payload) For details, refer to Table 6.16.
3	USHORT	usWdcThresholdDef	Counter threshold for consecutive WDC checking errors (default value)
4	USHORT	usWdcIncrement	WDC increment

Table 6.15 R_IN_TRN_SPLD_WDC_INFO_T List

No.	Member		Description
1	USHORT	usWdcOffset	WDC UL offset Specifies an offset from the start of RWr in byte units. Checking is disabled while FFFFH is specified.

Table 6.16 R_IN_RCV_SPLD_WDC_INFO_T List

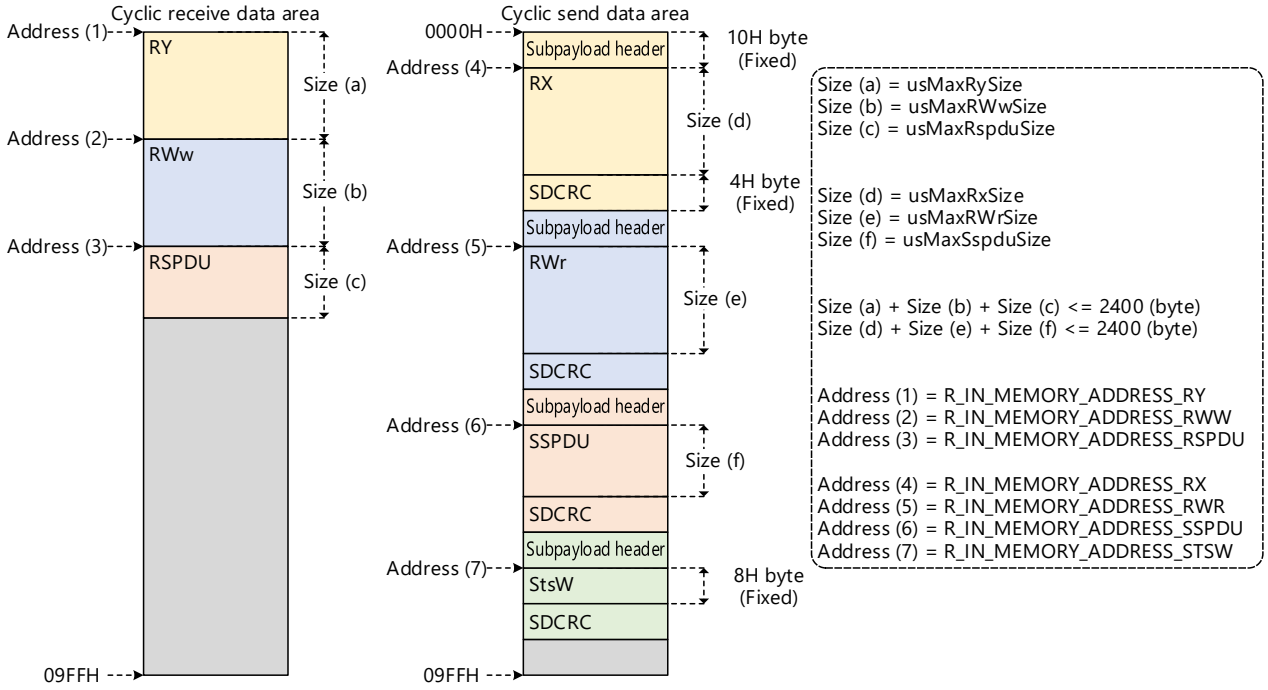
No.	Member		Description
1	USHORT	usWdcOffset	WDC UL offset Specifies an offset from the start of RWw in byte units. Checking is disabled while FFFFH is specified.

Supplementary notes on RY, RWw, RX, RWr, RSPDU, and SSPDU sizes

Modify the offset address from the start of the cyclic receive/transmit data area in accord with the setting value of RY, RWw, RX, RWr, RSPDU, and SSPDU sizes (usMaxRySize, usMaxRWwSize, usMaxRxSize, usMaxRWrSize, usMaxRspduSize, and usMaxSspduSize). The offset address is defined in the R_IN32M4_CL3MemoryAddress.h file.

R_IN_MEMORY_ADDRESS_RY (initial value: 0000 0000H)
R_IN_MEMORY_ADDRESS_RWW (initial value: 0000 0200H)
R_IN_MEMORY_ADDRESS_RSPDU (initial value: 0000 0700H)
R_IN_MEMORY_ADDRESS_RX (initial value: 0000 0010H)
R_IN_MEMORY_ADDRESS_RWR (initial value: 0000 0210H)
R_IN_MEMORY_ADDRESS_SSPDU (initial value: 0000 0710H)
R_IN_MEMORY_ADDRESS_STSW (initial value: 0000 0810H)

The following is an example of the offset address change.



Change the sizes and addresses described in the CSP+ file in accordance with those in the figure above.

Supplementary notes on network and controller

1) Definition of network and controller
Network: A communication section comprising R-IN32M4-CL3 and the peripheral circuit in the home station
Controller: A functional section which is unique to the user (such as I/O section, temperature adjustment section and robot section) in the home station

2) Setting of network
Network setting is required. The following items are checked in the conformance test.
No.6 Network firmware version
No.8 Network model code
No.7 Network model type
No.9 Network vendor code

3) Setting of controller
Controller setting is optional.
Set the controller in the following cases. (In other cases, controller setting is not required.)
When performing the parameter processing/command execution of slave station after verifying the vendor code/model code described in the CSP+ file against the controller information of the connected slave stations.
When the R-IN32M4-CL3 application product (network) is a communication optional item for a product (controller) such as series products.
When the manufacturer of controller and network is different.

Supplementary notes on the device version

[Background]
When updating the software version of an R-IN32M4-CL3 application product, changes to the specifications, such as the addition of slave station parameter processing or command execution, may be made. When the specifications of an R-IN32M4-CL3 application product are changed, the CSP+ file also needs to be updated in accord with the specification changes.

[Purpose of the device version]
The information that identifies the specifications before and after change is the device version. The device version is used to indicate to which R-IN32M4-CL3 application product specifications each CSP+ file corresponds.

(a) Purpose of use by the engineering tool
The engineering tool manages all CSP+ files having different device versions, making it possible to provide optimum functions and UI in accordance with the version of the R-IN32M4-CL3 application product to be used.

(b) Purpose of use by the end user
The end user can select the CSP+ file for the device to be actually used by comparing the device version described in the CSP+ file and the version of the R-IN32M4-CL3 application product to be used.

For details, refer to "DEVICE_INFO Part" in "Control & Communication System Profile (CSP+) Specification".

Option information: Supplemental information

When any slice remote I/O module or extension module is connected to the R-IN32M4-CL3 application product, the option information related definitions need to be changed depending on the number of modules connected.
(When no slice remote I/O module or extension module is connected, the option information related definitions do not need to be changed.)

The following is a definition change example when the number of connected modules is 64.

No.	Definition to be changed	Overview	Member	Initial value	Settable range	Changed value
1	USER_OPTN_INFOFLG	Option information status flag	blOptionInfoFlag	R_IN_FALSE	R_IN_TRUE / R_IN_FALSE	R_IN_TRUE
2	USER_NUMBER_OF_OPTION	Number of option information entries	uchNumberOfOption	0	0 to 64	64
3	R_IN_OPTIONTABLE_ENTRY_SIZE	Number of entries in the option information table	astOptionInfo[]	8	0 to 64	64

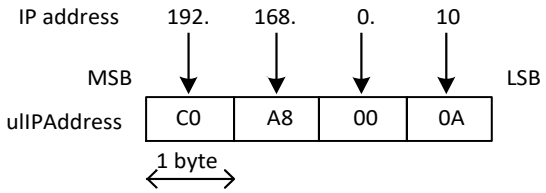
The option information defined here is reflected to the other module information (option information) of MIB.
In addition, when the [Connected/Disconnected Module Detection] button is clicked on the "CC-Link IE TSN Configuration" window of GX Works3, the slice remote I/O modules and extension modules defined in the option information are detected automatically.

The following shows the configuration of the argument “R_IN_UNITINIT_T” of gerR_IN_Initialize.

Table 6.17 R_IN_UNITINIT_T List

No.	Member		Overview	Setting Description
1	BOOL	blHighInterruptUse	High priority interrupt use	Set "R_IN_TRUE" when the high priority interrupt function is used, and "R_IN_FALSE" when it is not used. Setting "R_IN_TRUE" changes the RP05 pin to "High" when an R-IN32M4-CL3 high priority interrupt occurs. When the high priority interrupt function is used, the RP05 multiplexed functions cannot be used. The high priority interrupt occurs when a WDT timeout error or a send error occurs. To determine whether the high priority interrupt has occurred due to a WDT error or not, check the "home station WDT error interrupt" state in the high priority interrupt cause register.
2	UCHAR	uchNodeType	Station type	Specifies the station type of the home station. Set "R_IN_NODE_SLAVE" (01H) for a remote station.

(3) gerR_IN_SetIPAddress

Function	Setting the IP address			
Call Format	ERRCODE gerR_IN_SetIPAddress (ULONG ulIPAddress, ULONG ulSubnetmask, ULONG ulDefaultGateway)			
Arguments	Type Name	Variable Name	Description	I/O
	ULONG	ulIPAddress	IP address	Input
	ULONG	ulSubnetmask	Subnet mask	Input
	ULONG	ulDefaultGateway	Default gateway	Input
Return Value	R_IN_OK: Normal completion R_IN_ERR: Abnormal completion (state error) R_IN_ERR_OUTOFRANGE: Abnormal completion (out of range)			
Description	This function sets initial values related to the IP address in the R-IN32M4-CL3 driver. Specify the IP address in the following format using an ULONG type value.  Specify the subnet mask and the default gateway in the same manner, by setting the first to fourth octets in an ULONG type value from its upper byte. Execute this function after the gerR_IN_Initialize function(6.4.1(2)) and before the gerR_IN_Start function(6.4.1(4)). An IP address can be set in the range from 0.0.0.1 to 223.255.255.254. Otherwise, the function returns R_IN_ERR.			

(4) gerR_IN_Start

Function	Starting R-IN32M4-CL3 communications			
Call Format	ERRCODE gerR_IN_Start (VOID)			
Arguments	Type Name	Variable Name	Description	I/O
	—	—	—	—
Return Value	R_IN_OK: Normal completion R_IN_ERR: Abnormal completion			
Description	This function starts the data link. Calling this function disables the internal WDT of the R-IN32M4-CL3. To use the internal WDT of the R-IN32M4-CL3, call the gerR_IN_EnableWDT function (6.4.2(4)). Execute the function only once after the gerR_IN_Initialize function(6.4.1(2)) has completed. When an IP address has not been specified, specify it from the master station. Otherwise, the function returns R_IN_ERR.			

(5) gvR_IN_RegistIPAddressFilteringFunction

Function	Registering IP address filtering processing			
Call Format	VOID gvR_IN_RegistIPAddressFilteringFunction (R_IN_IP_FILTERING_FUNCTION fpulFunction)			
Arguments	Type Name	Variable Name	Description	I/O
	R_IN_IP_FILTERING_FUNCTION	fpulFunction	Filtering determination processing*1 NULL: IP address filtering disabled Other than NULL: IP address filtering enabled	—
Return Value	—			
Description	This function registers processing that determines whether to perform the IP address filtering. When the IP address filtering is not required, there is no need to execute the function. The processing registers the function that is passed as the argument to USNetPlus using the USNetPlus function "void regist_ip_filr_func (IPFLTRFUNC pfunc)" (IP address filtering processing registration). The registered function is called by USNetPlus when an IP frame is received and used to determine whether to perform reception processing. Execute the function only once after the "gerR_IN_Initialize"(6.4.1(2)) function has completed.			

Note 1. This processing determines whether to perform reception processing or discard the frame based on the IP address of the received frame. The details are described below.

Call Format	ULONG ulFunction (ULONG ulIPAddress)			
Arguments	Type Name	Variable Name	Description	I/O
	ULONG	ulIPAddress	IP address of the received frame	—
Return Value	1: Perform reception processing Other than 1: Discard the frame			

(6) gerR_IN_MIBLedTableDefine

Function	LED state information MIB definition			
Call Format	ERRCODE gerR_IN_MIBLedTableDefine (UCHAR uchNumberOfLED, const R_IN_MIBLEDDEFINE_T *pstMIBLedDefine)			
Arguments	Type Name	Variable Name	Description	I/O
	UCHAR	uchNumberOfLED	Number of LED entries	Input
	const R_IN_MIBLEDDEFINE_T*	pstMIBLedDefine	LED definition information For details, refer to Table 6.18.	Input
Return Value	R_IN_OK: Normal completion R_IN_ERR: Abnormal completion R_IN_ERR_OUTOFRANGE: Abnormal completion (out of range)			
Description	This function defines entries of LED information managed in MIB. Executing this function periodically stores the latest LED state in MIB after the R-IN32M4-CL3 communications start gerR_IN_Start(6.4.1(4)) function has completed.			

Table 6.18 R_IN_MIBLEDDEFINE_T List

No.	Member		Description
1	UCHAR	uchIdentification	LED type R_IN_LED_ID_NOTUSE: Not used R_IN_LED_ID_RUN: RUN LED R_IN_LED_ID_USER1: User LED 1 R_IN_LED_ID_USER2: User LED 2 R_IN_LED_ID_DLINK: DLINK LED R_IN_LED_ID_ERR: ERR LED R_IN_LED_ID_SD_SDRD1: SD/SDRD1 LED R_IN_LED_ID_RD_SDRD2: RD/SDRD2 LED R_IN_LED_ID_LER1: LER 1 LED R_IN_LED_ID_LER2: LER 2 LED R_IN_LED_ID_LINK1: LINK 1 LED R_IN_LED_ID_LINK2: LINK 2 LED R_IN_LED_ID_LINK: LINK LED
3	UCHAR	uchColor	LED color R_IN_LED_NOTUSE: Not used R_IN_LED_COLOR_GREEN: Green R_IN_LED_COLOR_RED: Red R_IN_LED_COLOR_ORANGE: Orange R_IN_LED_COLOR_OTHER: Others

(7) gerR_IN_RegistCallback

Function	Registering callback processing			
Call Format	ERRCODE gerR_IN_RegistCallback (ULONG ulFunctionType, R_IN_CALLBACK_FUNCTION fpulFunction)			
Arguments	Type Name	Variable Name	Description	I/O
	ULONG	ulFunctionType	Callback processing type For details, refer to Table 6.19.	Input
	R_IN_CALLBAC_FUNCTON	fpulFunction	Callback processing*1 NULL: Disable processing specified by ulFunctionType. Other than NULL: Enable processing specified by ulFunctionType. typedef ULONG(*R_IN_CALLBACK_FUNCTION) (ULONG, ULONG)	Input
Return Value	R_IN_OK: Normal completion R_IN_ERR_OUTOFRANGE: Abnormal completion (out of range)			
Description	This function registers the processing for the callback from the R-IN32M4-CL3 driver.			

Note 1. The details are given below.

Call Format	ULONG ulFunction (ULONG ulParam1, ULONG ulParam2)
Arguments	Additional information from the R-IN32M4-CL3 driver at the time of callback. For details, refer to Table 6.19.
Return Value	Notifies the R-IN32M4-CL3 driver of the result of the callback processing. For details, refer to Table 6.19.
Description	This function is called back with the timing that suits the processing specified by ulFunctionType. Proceed with the appropriate processing according to the timing of notification.

Table 6.19 Callback Processing Specifications

No	Callback Processing Type (Processing Name) <Section for reference>	Dispatching Enabled/Disabled State at Run Time	Argument		Return Value
			ulParam1	ulParam2	
1	R_IN_FUNCTIONTYPE_TIMESYNC_COMPLETE (Data link acceleration (at the completion of time synchronization)) <6.6(5)>	Dispatching disabled	Fixed to 0	Fixed to 0	Fixed to 0
2	R_IN_FUNCTIONTYPE_CYCLICDATA_INITIALIZE (Initializing data for cyclic transmission) <6.6(4)>	Dispatching disabled	Fixed to 0	Fixed to 0	Fixed to 0
3	R_IN_FUNCTIONTYPE_CYCLIC_START (Data link acceleration (at the start of cyclic transfer)) <6.6(6)>	Dispatching disabled	Fixed to 0	Fixed to 0	Fixed to 0
4	R_IN_FUNCTIONTYPE_DISCONNECT_PARTWAY_THROUGH (Data link acceleration (in the case of disconnection partway through)) <6.6(7)>	Dispatching disabled	Fixed to 0	Fixed to 0	Fixed to 0
5	R_IN_FUNCTIONTYPE_COMCYLCE_DEFINITION (Communications cycle check processing) <6.6(8)>	Dispatching disabled	Start address of communications cycle information*2	Fixed to 0	0: Normal Other than 0: Error
6	R_IN_FUNCTIONTYPE_SYNC_CYCLIC_COM (Synchronous cyclic communications processing) <6.6(9)>	Dispatching disabled	Fixed to 0	Fixed to 0	Fixed to 0

Note 2. For information on the communications cycle, refer to Table 6.47.

6.4.2 Watchdog Timer

(1) gerR_IN_ResetWDT

Function	Resetting the R-IN32M4-CL3 internal WDT			
Call Format	ERRCODE gerR_IN_ResetWDT (VOID)			
Arguments	Type Name	Variable Name	Description	I/O
	—	—	—	—
Return Value	R_IN_OK: Normal completion			
Description	This function resets the internal WDT of R-IN32M4-CL3.			

(2) gerR_IN_SetWDT

Function	Setting the R-IN32M4-CL3 internal WDT time limit			
Call Format	ERRCODE gerR_IN_SetWDT (USHORT usWDTCOUNT)			
Arguments	Type Name	Variable Name	Description	I/O
	USHORT	usWDTCOUNT	WDT time limit setting	Input
Return Value	R_IN_OK: Normal completion			
Description	This function sets the time limit for the internal WDT of R-IN32M4-CL3. Setting range: 0000H: 100 ms 0001H: 200 ms 0002H: 300 ms … 001FH: 3200 ms			

(3) gerR_IN_DisableWDT

Function	Disabling the R-IN32M4-CL3 internal WDT			
Call Format	ERRCODE gerR_IN_DisableWDT (VOID)			
Arguments	Type Name	Variable Name	Description	I/O
	—	—	—	—
Return Value	R_IN_OK: Normal completion			
Description	This function disables the internal WDT of R-IN32M4-CL3. The internal WDT of R-IN32M4-CL3 is enabled immediately after reset. In the sample code, disable the WDT by the gerR_IN_Start function (6.4.1(4)). Implement either of the following when it takes time until the gerR_IN_Start function is called after power-on (a WDT error occurs): • Call this function to disable the WDT after release from the reset state. • Call the gerR_IN_ResetWDT function (6.4.2(1)) to reset the internal WDT after release from the reset state.			

(4) gerR_IN_EnableWDT

Function	Enabling the R-IN32M4-CL3 internal WDT			
Call Format	ERRCODE gerR_IN_EnableWDT (VOID)			
Arguments	Type Name	Variable Name	Description	I/O
	—	—	—	—
Return Value	R_IN_OK: Normal completion			
Description	This function enables the internal WDT of R-IN32M4-CL3.			

6.4.3 Event

(1) gerR_IN_GetEvent

Function	R-IN32M4-CL3 event detection			
Call Format	ERRCODE gerR_IN_GetEvent (R_IN_EVTPRM_INTERRUPT_T* pstEvent)			
Arguments	Type Name	Variable Name	Description	I/O
	R_IN_EVTPRM_INTERRUPT_T*	pstEvent	Interrupt cause For details, refer to "Table 6.20".	Output
Return Value	R_IN_OK: Normal completion R_IN_ERR: Abnormal completion			
Description	This function detects an R-IN32M4-CL3 interrupt event and outputs the result. If a null pointer is specified as the argument, the function returns R_IN_ERR (abnormal completion). The function masks the interrupt cause after detecting any interrupt event.			

(2) gerR_IN_Main

Function	R-IN32M4-CL3 event detection main processing			
Call Format	ERRCODE gerR_IN_Main (const R_IN_EVTPRM_INTERRUPT_T* pstEvent)			
Arguments	Type Name	Variable Name	Description	I/O
	const R_IN_EVTPRM_INTERRUPT_T*	pstEvent	Interrupt cause For details, refer to "Table 6.20".	Input
Return Value	R_IN_OK: Normal completion R_IN_ERR: Abnormal completion			
Description	This function performs the R-IN32M4-CL3 event processing in response to the detected event information. If a null pointer is specified as the argument, the function returns R_IN_ERR (abnormal completion).			

Table 6.20 R_IN_EVTPRM_INTERRUPT_T List

Union	Member (1)		Member (2)		Overview
uniFlag	ULONG	usAll	-	-	-
	Struct	stBit	ULONG	b1ZCommConnect:1	(bit 0) Connect communications
			ULONG	b1ZCommDisconnect:1	(bit 1) Disconnect communications
			ULONG	b5ZReserve1:5	(bits 2 to 6) Reserved
			ULONG	b1ZChangeIPAddress:1	(bit 7) IP address update
			ULONG	b24ZReserve2:24	(bits 8 to 31) Reserved

(3) gerR_IN_RestartEvent

Function	Restarting R-IN32M4-CL3 event detection			
Call Format	ERRCODE gerR_IN_RestartEvent (VOID)			
Arguments	Type Name	Variable Name	Description	I/O
	—	—	—	—
Return Value	R_IN_OK: Normal completion			
Description	This function releases masking of masked interrupt sources.			

6.4.4 Cyclic Transfer

(1) gerR_IN_GetMasterNodeStatus

Function	Acquiring the master station state			
Call Format	ERRCODE gerR_IN_GetMasterNodeStatus (BOOL *pblRunSts, BOOL *pblErrSts)			
Arguments	Type Name	Variable Name	Description	I/O
	BOOL*	pblRunSts	Master station application operation state R_IN_TRUE: Running R_IN_FALSE: Stopped	Output
	BOOL*	pblErrSts	Master station application error state R_IN_TRUE: Error R_IN_FALSE: No error	Output
Return Value	R_IN_OK: Normal completion (a cyclic frame from the master station has been received.) R_IN_ERR: Abnormal completion (a cyclic frame from the master station has not received because the data link is currently disconnected.)			
Description	This function acquires the state of the master station from the cyclic frame received from the master station. When a cyclic frame from the master station is not received because the data link is currently disconnected, the arguments are as follows: pblRunSts R_IN_FALSE pblErrSts R_IN_FALSE			

(2) gerR_IN_GetHoldClearFlag

Function	Acquiring the hold/clear flag			
Call Format	ERRCODE gerR_IN_GetHoldClearFlag (BOOL *pblHoldClearFlag)			
Arguments	Type Name	Variable Name	Description	I/O
	BOOL*	pblHoldClearFlag	Hold/clear flag R_IN_TRUE: Holding/clearing is in progress. R_IN_FALSE: Holding/clearing has not occurred.	Output
Return Value	R_IN_OK: Normal completion R_IN_ERR: Abnormal completion			
Description	Acquires the hold/clear flag. Execute the gerR_IN_UpdateReceivedCyclicData function (6.4.4(4)) to update information to the latest information.			

(3) gerR_IN_GetReceivedCyclicData

Function	Acquiring cyclic received data (batch)			
Call Format	Standard communications	ERRCODE gerR_IN_GetReceivedCyclicData (VOID* pRyDst, VOID* pRWwDst, BOOL blEnable)		
	Safety communications	ERRCODE gerR_IN_GetReceivedCyclicData (VOID* pRyDst, VOID* pRWwDst, VOID* pRspduDst, BOOL blEnable)		
Arguments	Type Name	Variable Name	Description	I/O
	VOID*	pRyDst	RY area*1	Output
	VOID*	pRWwDst	RWw area*1	Output
	VOID*	pRspduDst	RSPDU area*1, *2	Output
	BOOL	blEnable	Enables/disables copying R_IN_TRUE: Enable R_IN_FALSE: Disable	Input
Return Value	R_IN_OK: Normal completion (received data present) R_IN_ERR: Abnormal completion (no received data)			
Description	This function stores cyclic received data from the master station in the address ranges indicated by pRyDst, pRWwDst, and pRspduDst. Note that when blEnable is set to R_IN_FALSE, the received cyclic received data are discarded. (The return value changes to R_IN_ERR.) To use this function for CC-Link IE TSN Class A, set blEnable to R_IN_TRUE. When this function is in use, do not use any of the following functions: • 6.4.4(4) gerR_IN_UpdateReceivedCyclicData • 6.4.4(5) gerR_IN_FinishReceivedCyclicDataAcquisition • 6.4.4(6) gerR_IN_GetReceivedCyclicRY • 6.4.4(7) gerR_IN_GetReceivedCyclicRWw • 6.4.4(14) gerR_IN_GetReceivedCyclicBuffer • 6.4.4(15) gerR_IN_FinishReceivedCyclicBuffer Note 1. The start address must be in units of 4 bytes. Note 2. The processing related to RSPDU is enabled when the compiler switch "SAFETY_PDU_ENABLE" is enabled.			

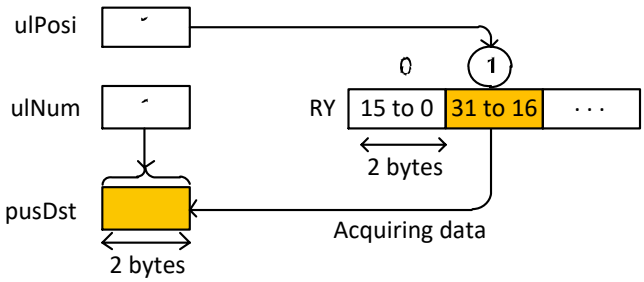
(4) gerR_IN_UpdateReceivedCyclicData

Function	Updating cyclic received data			
Call Format	ERRCODE gerR_IN_UpdateReceivedCyclicData (VOID)			
Arguments	Type Name	Variable Name	Description	I/O
	—	—	—	—
Return Value	R_IN_OK: Normal completion R_IN_ERR: Abnormal completion			
Description	By executing this function, the latest RY and RWw data can be acquired by using the following functions. • 6.4.4(6) gerR_IN_GetReceivedCyclicRY • 6.4.4(7) gerR_IN_GetReceivedCyclicRWw When this function is in use, do not use any of the following functions: • 6.4.4(3) gerR_IN_GetReceivedCyclicData • 6.4.4(14) gerR_IN_GetReceivedCyclicBuffer • 6.4.4(15) gerR_IN_FinishReceivedCyclicBuffer			

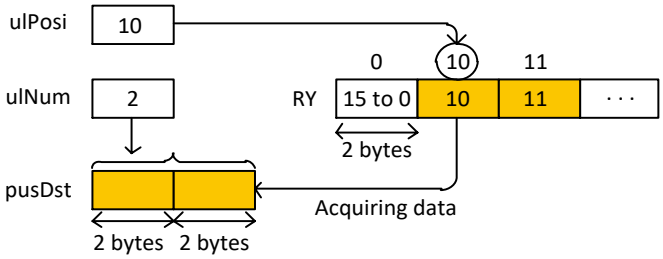
(5) gerR_IN_FinishReceivedCyclicDataAcquisition

Function	Completion of cyclic received data acquisition			
Call Format	ERRCODE gerR_IN_FinishReceivedCyclicDataAcquisition (VOID)			
Arguments	Type Name	Variable Name	Description	I/O
	—	—	—	—
Return Value	R_IN_OK: Normal completion R_IN_ERR: Abnormal completion			
Description	This function notifies R-IN32M4-CL3 of the completion of acquisition of cyclic received data. When this function is in use, do not use any of the following functions: • 6.4.4(3) gerR_IN_GetReceivedCyclicData • 6.4.4(14) gerR_IN_GetReceivedCyclicBuffer • 6.4.4(15) gerR_IN_FinishReceivedCyclicBuffer			

(6) gerR_IN_GetReceivedCyclicRY

Function	Acquiring cyclic received data (RY)			
Call Format	ERRCODE gerR_IN_GetReceivedCyclicRY (ULONG ulPosi, ULONG ulNum, USHORT *pusDst)			
Arguments	Type Name	Variable Name	Description	I/O
	ULONG	ulPosi	RY acquisition start position	Input
	ULONG	ulNum	RY acquisition number of data (word)	Input
	USHORT*	pusDst	RY storage destination area*1	Output
Return Value	R_IN_OK: Normal completion R_IN_ERR: Abnormal completion R_IN_ERR_NODATA: Abnormal completion (no received data) R_IN_ERR_OUTOFRANGE: Abnormal completion (out of range) R_IN_ERR_BOUNDARY: Abnormal completion (boundary error)			
Description	The latest RY data are stored in the address range indicated by pusDst by executing the gerR_IN_UpdateReceivedCyclicData function (6.4.4(4)).  When this function is in use, do not use any of the following functions: • 6.4.4(3) gerR_IN_GetReceivedCyclicData • 6.4.4(14) gerR_IN_GetReceivedCyclicBuffer • 6.4.4(15) gerR_IN_FinishReceivedCyclicBuffer Note 1. The start address must be in units of 2 bytes.			

(7) gerR_IN_GetReceivedCyclicRWw

Function	Acquiring cyclic received data (RWw)			
Call Format	ERRCODE gerR_IN_GetReceivedCyclicRWw (ULONG ulPosi, ULONG ulNum, USHORT *pusDst)			
Arguments	Type Name	Variable Name	Description	I/O
	ULONG	ulPosi	RWw acquisition start position	Input
	ULONG	ulNum	RWw acquisition number of data (word)	Input
	USHORT*	pusDst	RWw storage destination area*1	Output
Return Value	R_IN_OK: Normal completion R_IN_ERR: Abnormal completion R_IN_ERR_NODATA: Abnormal completion (no received data) R_IN_ERR_OUTOFRANGE: Abnormal completion (out of range) R_IN_ERR_BOUNDARY: Abnormal completion (boundary error)			
Description	The latest RWw data are stored in the address range indicated by pusDst by executing the gerR_IN_UpdateReceivedCyclicData function (6.4.4(4)).  When this function is in use, do not use any of the following functions: • 6.4.4(3) gerR_IN_GetReceivedCyclicData • 6.4.4(14) gerR_IN_GetReceivedCyclicBuffer • 6.4.4(15) gerR_IN_FinishReceivedCyclicBuffer Note 1. The start address must be in units of 2 bytes.			

(8) gerR_IN_GetReceivedCyclicRspdu

Function	Cyclic receive data acquisition (RSPDU)			
Call Format	ERRCODE gerR_IN_GetReceivedCyclicRspdu(USHORT* pusDst)			
Arguments	Type Name	Variable Name	Description	I/O
	USHORT*	pusDst	RSPDU storage destination area*1	Output
Return Value	R_IN_OK: Normal end R_IN_ERR: Abnormal end R_IN_ERR_NODATA: Abnormal end (no received data) R_IN_ERR_BOUNDARY: Abnormal end (boundary error)			
Description	This function stores cyclic data (values after S-Header of RSPDU) received from the safety master station in the addresses indicated by pusDst. Executing the gerR_IN_UpdateReceivedCyclicData function (6.4.4(4)) stores the latest RSPDU data in the area specified by pusDst. When using this function, do not use the following: • 6.4.4(3) gerR_IN_GetReceivedCyclicData • 6.4.4(14) gerR_IN_GetReceivedCyclicBuffer • 6.4.4(15) gerR_IN_FinishReceivedCyclicBuffer Note 1. The start address must be in units of 2 bytes.			

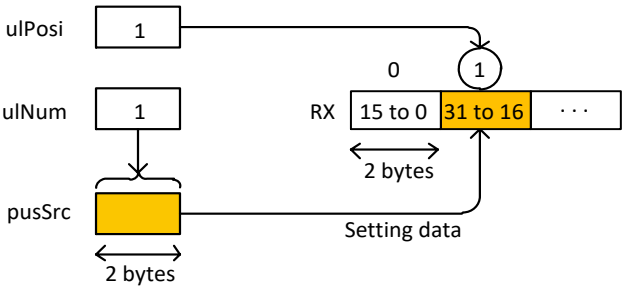
(9) gerR_IN_SetSendCyclicData

Function	Setting data for cyclic transmission (batch)			
Call Format	Standard communications	ERRCODE gerR_IN_SetSendCyclicData(const VOID* pRxSrc, const VOID* pRWrSrc, BOOL blEnable)		
	Safety communications	ERRCODE gerR_IN_SetSendCyclicData(const VOID* pRxSrc, const VOID* pRWrSrc, const VOID* pSspduSrc, BOOL blEnable)		
Arguments	Type Name	Variable Name	Description	I/O
	const VOID*	pRxSrc	RX area*1	Input
	const VOID*	pRWrSrc	RWr area*1	Input
	const VOID*	pSspduSrc,	SSPDU area*1, *2	Input
	BOOL	blEnable	Updating enabled or disabled R_IN_TRUE: Enabled R_IN_FALSE: Disabled	Input
Return Value	R_IN_OK: Normal completion R_IN_ERR: Abnormal completion			
Description	This function sets the data for cyclic transmission stored in the address ranges indicated by pRxSrc, pRWrSrc, and pSspduSrc in R-IN32M4-CL3. Note that the data for cyclic transmission are not set when blEnable is set to R_IN_FALSE. To use this function for CC-Link IE TSN Class A, set blEnable to R_IN_TRUE. When this function is in use, do not use any of the following functions: • 6.4.4(10) gerR_IN_FinishSendCyclicDataSetting • 6.4.4(11) gerR_IN_SetSendCyclicRX • 6.4.4(12) gerR_IN_SetSendCyclicRWr • 6.4.4(16) gerR_IN_GetSendCyclicBuffer • 6.4.4(17) gerR_IN_GetSplitSendCyclicBuffer • 6.4.4(18) gerR_IN_FinishSendCyclicBuffer Note 1. The start address must be in units of 4 bytes. Note 2. The processing related to SSPDU is enabled when the compiler switch "SAFETY_PDU_ENABLE" is enabled.			

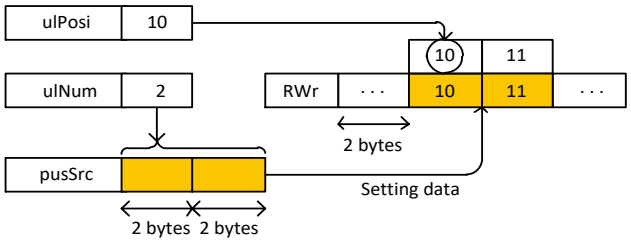
(10) gerR_IN_FinishSendCyclicDataSetting

Function	Completion of setting data for cyclic transmission			
Call Format	ERRCODE gerR_IN_FinishSendCyclicDataSetting (VOID)			
Arguments	Type Name	Variable Name	Description	I/O
	—	—	—	—
Return Value	R_IN_OK: Normal completion R_IN_ERR: Abnormal completion R_IN_ERR_NODATA: Abnormal end (Send data setting is not required when the product that is ranked as CC-Link IE TSN Class A operates.)			
Description	Execute this function when data for cyclic transmission are set by the gerR_IN_SetSendCyclicRX function (6.4.4(11)) and the gerR_IN_SetSendCyclicRWr function (6.4.4(12)). This function sends the RX and RWr data set in R-IN32M4-CL3 and the latest home station state information in the next cycle of communications. When the product that is ranked as CC-Link IE TSN Class A operates, if cyclic data is not received, "R_IN_ERR_NODATA" is returned as the return value. When this function is in use, do not use any of the following functions: • 6.4.4(9) gerR_IN_SetSendCyclicData • 6.4.4(16) gerR_IN_GetSendCyclicBuffer • 6.4.4(17) gerR_IN_GetSplitSendCyclicBuffer • 6.4.4(18) gerR_IN_FinishSendCyclicBuffer			

(11) gerR_IN_SetSendCyclicRX

Function	Setting data for cyclic transmission (RX)			
Call Format	ERRCODE gerR_IN_SetSendCyclicRX (const USHORT *pusSrc, ULONG ulPosi, ULONG ulNum)			
Arguments	Type Name	Variable Name	Description	I/O
	const USHORT*	pusSrc	RX setting source area*1	Input
	ULONG	ulPosi	RX setting start position	Input
	ULONG	ulNum	Number of RX data to be set (word)	Input
Return Value	R_IN_OK: Normal completion R_IN_ERR: Abnormal completion R_IN_ERR_OUTOFRANGE: Abnormal completion (out of range) R_IN_ERR_BOUNDARY: Abnormal completion (boundary error) R_IN_ERR_NODATA: Abnormal end (Send data setting is not required when the product that is ranked as CC-Link IE TSN Class A operates.)			
Description	This function sets the data for cyclic transmission stored in the address range indicated by pusSrc in R-IN32M4-CL3. The set RX data are sent in the next cycle of communications by executing the gerR_IN_FinishSendCyclicDataSetting function (6.4.4(10)).  When the product that is ranked as CC-Link IE TSN Class A operates, if cyclic data is not received, "R_IN_ERR_NODATA" is returned as the return value. When this function is in use, do not use any of the following functions: • 6.4.4(9) gerR_IN_SetSendCyclicData • 6.4.4(16) gerR_IN_GetSendCyclicBuffer • 6.4.4(17) gerR_IN_GetSplitSendCyclicBuffer • 6.4.4(18) gerR_IN_FinishSendCyclicBuffer" Note 1. The start address must be in units of 2 bytes.			

(12) gerR_IN_SetSendCyclicRWr

Function	Setting data for cyclic transmission (RWr)			
Call Format	ERRCODE gerR_IN_SetSendCyclicRWr (const USHORT *pusSrc, ULONG ulPosi, ULONG ulNum)			
Arguments	Type Name	Variable Name	Description	I/O
	const USHORT*	pusSrc	RWr setting source area*1	Input
	ULONG	ulPosi	RWr setting start position	Input
	ULONG	ulNum	Number of RWr data to be set (word)	Input
Return Value	R_IN_OK: Normal completion R_IN_ERR: Abnormal completion R_IN_ERR_OUTOFRANGE: Abnormal completion (out of range) R_IN_ERR_BOUNDARY: Abnormal completion (boundary error) R_IN_ERR_NODATA: Abnormal end (Send data setting is not required when the product that is ranked as CC-Link IE TSN Class A operates.)			
Description	This function sets the data for cyclic transmission stored in the area indicated by pusSrc in R-IN32M4-CL3. The set RWr data are sent in the next cycle of communications by executing the gerR_IN_FinishSendCyclicDataSetting function (6.4.4(10)).  When the product that is ranked as CC-Link IE TSN Class A operates, if cyclic data is not received, "R_IN_ERR_NODATA" is returned as the return value. When this function is in use, do not use any of the following functions: • 6.4.4(9) gerR_IN_SetSendCyclicData • 6.4.4(16) gerR_IN_GetSendCyclicBuffer • 6.4.4(17) gerR_IN_GetSplitSendCyclicBuffer • 6.4.4(18) gerR_IN_FinishSendCyclicBuffer" Note 1. The start address must be in units of 2 bytes.			

(13) gerR_IN_SetSendCyclicSspdu

Function	Cyclic send data setting (SSPDU)			
Call Format	ERRCODE gerR_IN_SetSendCyclicSspdu(const USHORT* pusSrc)			
Arguments	Type Name	Variable Name	Description	I/O
	const USHORT*	pusSrc	SSPDU setting source area*1	Input
Return Value	R_IN_OK: Normal end R_IN_ERR: Abnormal end R_IN_ERR_BOUNDARY: Abnormal end (boundary error) R_IN_ERR_NODATA: Abnormal end (Send data setting is not required when the product that is ranked as CC-Link IE TSN Class A operates.)			
Description	This function sets the cyclic send data (S-Header start address of SSPDU) stored in the area specified by pusSrc to R-IN32M4-CL3. Executing the gerR_IN_FinishSendCyclicDataSetting function (6.4.4(10)) sends the set SSPDU data at the next communication cycle. When the product that is ranked as CC-Link IE TSN Class A operates, if cyclic data is not received, "R_IN_ERR_NODATA" is returned as the return value. When using this function, do not use the following: • 6.4.4(9) gerR_IN_SetSendCyclicData • 6.4.4(16) gerR_IN_GetSendCyclicBuffer • 6.4.4(17) gerR_IN_GetSplitSendCyclicBuffer • 6.4.4(18) gerR_IN_FinishSendCyclicBuffer Note 1. Set the start address in increments of 2 bytes.			

(14) gerR_IN_GetReceivedCyclicBuffer

Function	Acquiring the cyclic reception buffer			
Call Format	Standard communications	ERRCODE gerR_IN_GetReceivedCyclicBuffer (ULONG* pulRyBuffer, ULONG* pulRWwBuffer)		
	Safety communications	ERRCODE gerR_IN_GetReceivedCyclicBuffer(ULONG* pulRyBuffer, ULONG* pulRWwBuffer, ULONG* pulRspduBuffer)		
Arguments	Type Name	Variable Name	Description	I/O
	ULONG*	pulRyBuffer	RY storage destination address	Output
	ULONG*	pulRWwBuffer	RWw storage destination address	Output
	ULONG*	pulRspduBuffer	RSPDU storage destination address*1	Output
Return Value	R_IN_OK: Normal completion (received data present) R_IN_ERR: Abnormal completion R_IN_ERR_NODATA: Abnormal completion (no received data)			
Description	This function acquires the destination addresses for storage of the cyclic data received from the master station. After cyclic received data acquisition is completed, execute the gerR_IN_FinishReceivedCyclicBuffer function (0). When this function is in use, do not use any of the following functions: • 6.4.4(3) gerR_IN_GetReceivedCyclicData • 6.4.4(4) gerR_IN_UpdateReceivedCyclicData • 6.4.4(5) gerR_IN_FinishReceivedCyclicDataAcquisition • 6.4.4(6) gerR_IN_GetReceivedCyclicRY • 6.4.4(7) gerR_IN_GetReceivedCyclicRWw • 6.4.4(8) gerR_IN_GetReceivedCyclicRspdu Note 1. The processing related to RSPDU is enabled when the compiler switch "SAFETY_PDU_ENABLE" is enabled.			

(15) gerR_IN_FinishReceivedCyclicBuffer

Function	Completion of cyclic reception buffer acquisition			
Call Format	ERRCODE gerR_IN_FinishReceivedCyclicBuffer (VOID)			
Arguments	Type Name	Variable Name	Description	I/O
	—	—	—	—
Return Value	R_IN_OK: Normal completion (received data present) R_IN_ERR: Abnormal completion R_IN_ERR_NODATA: Abnormal completion (no received data)			
Description	This function notifies R-IN32M4-CL3 of the completion of acquisition of cyclic data received from the master station. When this function is in use, do not use any of the following functions: • 6.4.4(3) gerR_IN_GetReceivedCyclicData • 6.4.4(4) gerR_IN_UpdateReceivedCyclicData • 6.4.4(5) gerR_IN_FinishReceivedCyclicDataAcquisition • 6.4.4(6) gerR_IN_GetReceivedCyclicRY • 6.4.4(7) gerR_IN_GetReceivedCyclicRWw • 6.4.4(8) gerR_IN_GetReceivedCyclicRspdu			

(16) gerR_IN_GetSendCyclicBuffer

Function	Acquiring the cyclic transmission buffer			
Call Format	Standard communications	ERRCODE gerR_IN_GetSendCyclicBuffer (ULONG* pulRxBuffer, ULONG* pulRWrBuffer)		
	Safety communications	ERRCODE gerR_IN_GetSendCyclicBuffer (ULONG* pulRxBuffer, ULONG* pulRWrBuffer, ULONG* pulSspduBuffer)		
Arguments	Type Name	Variable Name	Description	I/O
	ULONG*	pulRxBuffer	RX storage destination address	Output
	ULONG*	pulRWrBuffer	RWr storage destination address	Output
	ULONG*	pulSspduBuffer	SSPDU storage destination address*1	Output
Return Value	R_IN_OK: Normal completion R_IN_ERR: Abnormal completion R_IN_ERR_NODATA: Abnormal end (Send data setting is not required when the product that is ranked as CC-Link IE TSN Class A operates.)			
Description	This function acquires the destination addresses for storage of data for cyclic transmission to be sent to the master station. After the completion of setting data for cyclic transmission, execute the gerR_IN_FinishSendCyclicBuffer function (6.4.4(18)). The destination addresses for storage of RX and RWr are only set as arguments when the return value is R_IN_OK. When the product that is ranked as CC-Link IE TSN Class A operates, if cyclic data is not received, "R_IN_ERR_NODATA" is returned as the return value. When this function is in use, do not use any of the following functions: • 6.4.4(9) gerR_IN_SetSendCyclicData • 6.4.4(10) gerR_IN_FinishSendCyclicDataSetting • 6.4.4(11) gerR_IN_SetSendCyclicRX • 6.4.4(12) gerR_IN_SetSendCyclicRWr • 6.4.4(13) gerR_IN_SetSendCyclicSspdu Note 1. The processing related to SSPDU is enabled when the compiler switch "SAFETY_PDU_ENABLE" is enabled. If data for cyclic transmission are too large for a single frame so that two frames for cyclic transmission are required, use the gerR_IN_GetSplitSendCyclicBuffer function (6.4.4(17)).			

(17) gerR_IN_GetSplitSendCyclicBuffer

Function	Acquiring the cyclic split transmission buffer			
Call Format	ERRCODE gerR_IN_GetSplitSendCyclicBuffer (R_IN_SEND_CYCLIC_BUFFER_T* pstBuffer)			
Arguments	Type Name	Variable Name	Description	I/O
	R_IN_SEND_CYCLIC_BUFFER_T	*pstBuffer	Cyclic transmission buffer For details, refer to Table 6.21.	Output
Return Value	R_IN_OK: Normal completion R_IN_ERR: Abnormal completion R_IN_ERR_NODATA: Abnormal end (Send data setting is not required when the product that is ranked as CC-Link IE TSN Class A operates.)			
Description	This function acquires the destination addresses for storage of data for cyclic transmission to be sent to the master station. After the completion of setting data for cyclic transmission, execute the gerR_IN_GetSplitSendCyclicBuffer function (6.4.4(18)). The areas of RX or RWr are split when a frame for cyclic transmission is configured into two frames due to RX or RWr being too large for a single frame. This function allows acquisition of the start addresses and sizes of the areas into which RX or RWr have been split. The destination addresses for storage of RX and RWr are only set as arguments when the return value is R_IN_OK. When the product that is ranked as CC-Link IE TSN Class A operates, if cyclic data is not received, "R_IN_ERR_NODATA" is returned as the return value. When this function is in use, do not use any of the following functions: • 6.4.4(9) gerR_IN_SetSendCyclicData • 6.4.4(10) gerR_IN_FinishSendCyclicDataSetting • 6.4.4(11) gerR_IN_SetSendCyclicRX • 6.4.4(12) gerR_IN_SetSendCyclicRWr • 6.4.4(13) gerR_IN_SetSendCyclicSspdu			

Table 6.21 R_IN_SEND_CYCLIC_BUFFER_T List

No.	Member			Description
1	stRx	—		RX buffer information
2	[R_IN_SEND_CYCLIC_BUFFER_RX_NUM]	ULONG	ulAddr	Buffer start address
3		USHORT	usSize	Buffer size (byte)
4	stRWr	—		RWr buffer information
5	[R_IN_SEND_CYCLIC_BUFFER_RWR_NUM]	ULONG	ulAddr	Buffer start address
6		USHORT	usSize	Buffer size (byte)
7	stSspdu	-		SSPDU buffer information*1
8	[R_IN_SEND_CYCLIC_BUFFER_SSPDU_NUM]	ULONG	ulAddr	Buffer start address
9		USHORT	usSize	Buffer size (byte)
10	USHORT		usRxNum	Number of RX buffers
11	USHORT		usRWrNum	Number of RWr buffers
12	USHORT		usSspduNum	Number of SSPDU buffers *1

Note 1. This is used when the safety PDU send/receive is performed (the compiler switch "SAFETY_PDU_ENABLE" is enabled).

Table 6.22 Macro Definitions for Number of Cyclic Send Buffer Areas

Constant	Value (for other than CANopen communications)	Value (for CANopen communications)
R_IN_SEND_CYCLIC_BUFFER_RX_NUM	2	2
R_IN_SEND_CYCLIC_BUFFER_RWR_NUM	2	R_IN_CAN_MAX_ODTABLE_NUM*1 + 1
R_IN_SEND_CYCLIC_BUFFER_SSPDU_NUM	2	Not defined

Note 1. Maximum number of object dictionaries

(18) gerR_IN_FinishSendCyclicBuffer

Function	Completion of setting the cyclic transmission buffer			
Call Format	ERRCODE gerR_IN_FinishSendCyclicBuffer (VOID)			
Arguments	Type Name	Variable Name	Description	I/O
	—	—	—	—
Return Value	R_IN_OK: Normal completion R_IN_ERR: Abnormal completion R_IN_ERR_NODATA: Abnormal end (Send data setting is not required when the product that is ranked as CC-Link IE TSN Class A operates.)			
Description	This function notifies R-IN32M4-CL3 of the completion of setting the cyclic transmission buffer and sends the data to the master station. When the product that is ranked as CC-Link IE TSN Class A operates, if cyclic data is not received, "R_IN_ERR_NODATA" is returned as the return value. When this function is in use, do not use any of the following functions: • 6.4.4(9) gerR_IN_SetSendCyclicData • 6.4.4(10) gerR_IN_FinishSendCyclicDataSetting • 6.4.4(11) gerR_IN_SetSendCyclicRX • 6.4.4(12) gerR_IN_SetSendCyclicRWr • 6.4.4(13) gerR_IN_SetSendCyclicSspdu			

(19) gvR_IN_SendCyclicFrameClassA

Function	Cyclic frame send (CC-Link IE TSN Class A)			
Call Format	VOID gvR_IN_SendCyclicFrameClassA(VOID)			
Arguments	Type Name	Variable Name	Description	I/O
	—	—	—	—
Return Value	—			
Description	This function sends the set cyclic send data to the master station during CC-Link IE TSN Class A. When this function is executed, cyclic frames are actually sent. When the product that is ranked as CC-Link IE TSN Class B operates, execution of this function is not required.			

6.4.5 Home Station State Setting

(1) gerR_IN_SetNodeStatus

Function	Setting the home station state			
Call Format	ERRCODE gerR_IN_SetNodeStatus (ULONG ulErrSts)			
Arguments	Type Name	Variable Name	Description	I/O
	ULONG	ulErrSts	Application error state R_IN_ERRSTS_NONE (0): No error R_IN_ERRSTS_WARNING (1): Minor abnormality R_IN_ERRSTS_ERROR (2): Moderate abnormality R_IN_ERRSTS_FATALERROR (3): Major abnormality	Input
Return Value	R_IN_OK: Normal completion R_IN_ERR_OUTOFRANGE: Abnormal completion (out of range)			
Description	This function sets the state of the home station as information to be sent in cyclic transfer. When a WDT error occurs, a major error is set by the R-IN32M4-CL3 driver.			

6.4.6 Home Station State Acquisition

(1) gerR_IN_GetIPAddress

Function	Acquiring the IP address			
Call Format	ERRCODE gerR_IN_GetIPAddress (ULONG *pulIPAddress, ULONG *pulSubnetmask, ULONG *pulDefaultGateway)			
Arguments	Type Name	Variable Name	Description	I/O
	ULONG*	pulIPAddress	IP address	Output
	ULONG*	pulSubnetmask	Subnet mask	Output
	ULONG*	pulDefaultGateway	Default gateway	Output
Return Value	R_IN_OK: Normal completion R_IN_ERR: Abnormal completion			
Description	This function acquires the setting values of the IP address, subnet mask, and default gateway. If the address is "192.168.3.10", the stored value of *pulIPAddress and *pulDefaultGateway is "0xC0A8030AH". If the subnet mask is "255.255.255.0", the value to be stored in *pulSubnetmask is "0xFFFFFF00H". The value stored in *pulDefaultGateway of the default gateway is always "0x00000000H". Use this function after the gerR_IN_Initialize function (6.4.1(2)) has completed.			

(2) gerR_IN_GetCurrentCyclicSize

Function	Acquiring the cyclic transfer size specified by the master station			
Call Format	ERRCODE gerR_IN_GetCurrentCyclicSize (R_IN_CYCLIC_SIZE_T *pstCyclicSize)			
Arguments	Type Name	Variable Name	Description	I/O
	R_IN_CYCLIC_SIZE_T*	pstCyclicSize	Cyclic transfer size For detail, refer to Table 6.23.	Output
Return Value	R_IN_OK: Normal completion R_IN_ERR: Abnormal completion			
Description	This function acquires the cyclic transfer size specified by the master station. Functions gerR_IN_GetReceivedCyclicData (6.4.4(3)) and gerR_IN_SetSendCyclicData (6.4.4(9)) input and output cyclic transmitted/received data in the size acquired by this function.			

Table 6.23 R_IN_ CYCLIC_SIZE_T List

No.	Member		Description
1	ULONG	ulRySize	RY size (byte (octet))
2	ULONG	ulRWwSize	RWw size (byte (octet))
3	ULONG	ulRxSize	RX size (byte (octet))
4	ULONG	ulRWrSize	RWr size (byte (octet))
5	ULONG	ulRspduSize	RSPDU size (byte)*1
6	ULONG	ulSspduSize	SSPDU size (byte)*1

Note 1. This is used when the safety PDU send/receive is performed (the compiler switch "SAFETY_PDU_ENABLE" is enabled).

(3) gerR_IN_GetCyclicStatus

Function	Acquiring the cyclic transfer state			
Call Format	ERRCODE gerR_IN_GetCyclicStatus (UCHAR* puchCyclicStatus)			
Arguments	Type Name	Variable Name	Description	I/O
	UCHAR*	puchCyclicStatus	Cyclic transfer state 00H: Normal communications or power-on 02H: Monitoring time timeout 12H: Reserved station setting of the home station 13H: Home station number overlap	Output
Return Value	R_IN_OK: Normal completion R_IN_ERR: Abnormal completion			
Description	This function acquires the state of cyclic transfer.			

(4) gerR_IN_GetCommumicationStatus

Function	Acquiring the data link state			
Call Format	ERRCODE gerR_IN_GetCommumicationStatus (ULONG *pulCommSts)			
Arguments	Type Name	Variable Name	Description	I/O
	ULONG*	pulCommSts	Data link state R_IN_COMMSTS_CYC_DLINK (2): Data link in progress (cyclic transfer in progress) R_IN_COMMSTS_CYC_STOP (1): Data link in progress (cyclic transfer stopped) R_IN_COMMSTS_DISCONNECT (0): No data link (disconnected)	Output
Return Value	R_IN_OK: Normal completion R_IN_ERR: Abnormal completion			
Description	This function acquires the data link state. Turn D LINK LED on or off in accord with the data link state. R_IN_COMMSTS_CYC_DLINK: LED on R_IN_COMMSTS_CYC_STOP: LED blinking R_IN_COMMSTS_DISCONNECT: LED off			

(5) gerR_IN_GetPortStatus

Function	Acquiring the PHY link state			
Call Format	ERRCODE gerR_IN_GetPortStatus (ULONG ulPort, ULONG *pulLinkStatus, ULONG *pulSpeed, ULONG *pulDuplex)			
Arguments	Type Name	Variable Name	Description	I/O
	ULONG	ulPort	Port selection R_IN_PORT1 (0): Port 1 R_IN_PORT2 (1): Port 2	Input
	ULONG*	pulLinkStatus	Link state R_IN_LINKUP (1): LinkUp R_IN_LINKDOWN (0): LinkDown	Output
	ULONG*	pulSpeed	Speed R_IN_SPEED_1G (0): 1 Gbps R_IN_SPEED_100M (1): 100 Mbps R_IN_SPEED_10M (2): 10 Mbps (When *pulLinkState is LinkDown: Don't care.)	Output
	ULONG*	pulDuplex	Full duplex/Half duplex R_IN_DUPLEX_FULL (0): Full duplex R_IN_DUPLEX_HALF (1): Half duplex (When *pulLinkState is LinkDown: Don't care.)	Output
Return Value	R_IN_OK: Normal completion R_IN_ERR: Abnormal completion R_IN_ERR_OUTOFRANGE: Abnormal completion (out of range) R_IN_ERR_SEMAPHORE: Abnormal completion (semaphore acquisition failure) R_IN_ERR_FATAL: Abnormal completion (fatal error occurred)			
Description	This function acquires the PHY link state. Execute the function after the completion of the gerR_IN_Start function (6.4.1(4)).			

(6) gerR_IN_GetStatisticalInformation

Function	Acquiring statistical information			
Call Format	ERRCODE gerR_IN_GetStatisticalInformation (R_IN_STATISTICS_T *pstStatisticalInformation)			
Arguments	Type Name	Variable Name	Description	I/O
	R_IN_STATISTICS_T*	pstStatisticalInformation	Statistical information acquisition For details, refer to Table 6.24.	Output
Return Value	R_IN_OK: Normal completion R_IN_ERR: Abnormal completion (parameter error)			
Description	This function acquires statistical information in the MIB held by the R-IN32M4-CL3 driver. If the argument is a null pointer, this function returns R_IN_ERR. The accumulated number of statistical information sets can be held by defining the compiler switch "ACCUMULATE_STATISTICS_INFORMATION", and the upper limit number of information sets is held when the number reaches the limit. The statistical information update processing is performed by the updating task. If the function is executed in the lower priority task than the updating task, data inconsistency occurs. To prevent data inconsistency, execute the function in the task having the same priority as the updating task.			

Table 6.24 R_IN_STATISTICS_T List

No.	Member		Description
1	USHORT	usCyclicReceiveCounter	Cyclic reception counter
2	USHORT	usCyclicReceiveDiscardCounter	Cyclic reception discard counter
3	USHORT	usCyclicFrameReceiveCounter	Cyclic frame reception counter
4	USHORT	usNonCyclicReceiveCounter	Non-cyclic reception counter
5	USHORT	usNonCyclicReceiveDiscardCounter	Non-cyclic reception discard counter
6	USHORT	usNumberOfHecErrorFrame	Number of HEC error frames
7	USHORT	usNumberOfDcsErrorFrame	Number of DCS error frames
8	USHORT	usNumberOfFcsErrorFrame	Number of FCS error frames
9	USHORT	usNumberOfSdcrcErrorFrame	Number of SDCRC error frames
10	USHORT	usNumberOfShortPacketFrame	Number of short packet frames detected
11	USHORT	usNumberOfJumboFrame	Number of jumbo frames detected
12	USHORT	usNumberOfLongPacketFrame	Number of long packets detected
13	USHORT	usNumberOfFailedCcLinkIePduSize	Number of CC-Link IE TSN PDU length errors
14	USHORT	usNumberOfFlagmentErrorFrame	Number of fragment error frames
15	USHORT	usNumberOfPriorityControlFrame	Number of priority control frames
16	USHORT	usNumberOfIpFrame	Number of IP frames
17	USHORT	usNumberOfIeee802or1588Frame	Number of IEEE802.1AS/IEEE1588 frames
18	USHORT	usNumberOfLldpFrame	Number of LLDP frames
19	USHORT	usNumberOfSyncFrame	Number of Sync frames

(7) gerR_IN_ClearStatisticalInformation

Function	Clearing statistical information			
Call Format	ERRCODE gerR_IN_ClearStatisticalInformation (VOID)			
Arguments	Type Name	Variable Name	Description	I/O
	—	—	—	—
Return Value	R_IN_OK: Normal completion R_IN_ERR_BUSY: The request for clearing has already been received or is currently being processed			
Description	This function clears the statistical information in the MIB held by the R-IN32M4-CL3 driver. The statistical information is cleared at the next update of statistical information.			

(8) gulR_IN_GetNetworkTopology

Function	Acquiring network topology information			
Call Format	ULONG gulR_IN_GetNetworkTopology (VOID)			
Arguments	Type Name	Variable Name	Description	I/O
	—	—	—	—
Return Value	R_IN_NETWORK_TOPOLOGY_UNKNOWN (0): Network topology is unknown. R_IN_NETWORK_TOPOLOGY_OTHER (1): Other network topology R_IN_NETWORK_TOPOLOGY_RING (2): Ring connection			
Description	This function acquires information on the detected topology (single-ring) which has been set from the master station.			

(9) gerR_IN_GetNodeOperationMode

Function	Acquiring the home station's operating mode			
Call Format	ERRCODE gerR_IN_GetNodeOperationMode (R_IN_NODE_OPERATION_MODE_T* pstMode)			
Arguments	Type Name	Variable Name	Description	I/O
	R_IN_NODE_OPERATION_MODE_T*	pstMode	Home station operating mode R_IN_ASYNCHRONOUS (0): Asynchronous R_IN_SYNCHRONOUS (1): Synchronous	Output
Return Value	R_IN_OK: Normal completion R_IN_ERR: Abnormal completion			
Description	This function acquires the operating mode of the home station (R-IN32M4-CL3).			

(10) gerR_IN_GetLinkSpeed

Function	Communication speed acquisition			
Call Format	ERRCODE gerR_IN_GetLinkSpeed(ULONG* pulLinkSpeed)			
Arguments	Type Name	Variable Name	Description	I/O
	ULONG*	pulLinkSpeed	Communication speed	Output
Return Value	R_IN_OK: Normal completion R_IN_ERR: Abnormal completion			
Description	This function acquires the communication speed setting value. 0: 1 Gbps 1: 100 Mbps Execute the function after the gerR_IN_Initialize function has completed.			

6.4.7 LED Control

(1) gerR_IN_SetUSER1LED

Function	LED lighting control (USER LED1)			
Call Format	ERRCODE gerR_IN_SetUSER1LED (ULONG ulCtrl)			
Arguments	Type Name	Variable Name	Description	I/O
	ULONG	ulCtrl	LED lighting control parameter For details, refer to Table 6.25.	Input
Return Value	R_IN_OK: Normal completion R_IN_ERR_OUTOFRANGE: Abnormal completion (out of range)			
Description	This function controls lighting of USER LED1. Since processing to disable and enable dispatching of service calls is called from within this function, processing by the function is mutually exclusive of processing for other tasks.			

(2) gerR_IN_SetUSER2LED

Function	LED lighting control (USER LED2)			
Call Format	ERRCODE gerR_IN_SetUSER2LED (ULONG ulCtrl)			
Arguments	Type Name	Variable Name	Description	I/O
	ULONG	ulCtrl	LED lighting control parameter For details, refer to Table 6.25.	Input
Return Value	R_IN_OK: Normal completion R_IN_ERR_OUTOFRANGE: Abnormal completion (out of range)			
Description	This function controls lighting of USER LED2. Since processing to disable and enable dispatching of service calls is called from within this function, processing by the function is mutually exclusive of processing for other tasks.			

(3) gerR_IN_SetRUNLED

Function	LED lighting control (RUN)			
Call Format	ERRCODE gerR_IN_SetRUNLED (ULONG ulCtrl)			
Arguments	Type Name	Variable Name	Description	I/O
	ULONG	ulCtrl	LED lighting control parameter For details, refer to Table 6.25.	Input
Return Value	R_IN_OK: Normal completion R_IN_ERR_OUTOFRANGE: Abnormal completion (out of range)			
Description	This function controls lighting of RUN LED. Since processing to disable and enable dispatching of service calls is called from within this function, processing by the function is mutually exclusive of processing for other tasks.			

(4) gerR_IN_SetERRLED

Function	LED lighting control (ERR)			
Call Format	ERRCODE gerR_IN_SetERRLED (ULONG ulCtrl)			
Arguments	Type Name	Variable Name	Description	I/O
	ULONG	ulCtrl	LED lighting control parameter For details, refer to Table 6.25.	Input
Return Value	R_IN_OK: Normal completion R_IN_ERR_OUTOFRANGE: Abnormal completion (out of range)			
Description	This function controls lighting of ERR LED. Since processing to disable and enable dispatching of service calls is called from within this function, processing by the function is mutually exclusive of processing for other tasks.			

Table 6.25 LED Lighting Control Parameter List

Name	Value	Description
R_IN_LED_OFF	0	LED on
R_IN_LED_ON	1	LED off
R_IN_LED_BLINK_1000	2	LED blinking (cycle: 1 s)
R_IN_LED_BLINK_500	3	LED blinking (cycle: 500 ms)
R_IN_LED_BLINK_200	4	LED blinking (cycle: 200 ms)

(5) gerR_IN_SetLERR1LED

Function	LED lighting control (L ER1 LED)			
Call Format	ERRCODE gerR_IN_SetLERR1LED (ULONG ulCtrl)			
Arguments	Type Name	Variable Name	Description	I/O
	ULONG	ulCtrl	R_IN_LED_OFF (0): LED off R_IN_LED_ON (1): LED on	Input
Return Value	R_IN_OK: Normal completion R_IN_ERR_OUTOFRANGE: Abnormal completion (out of range)			
Description	This function controls lighting of L ER1 LED. Since processing to disable and enable dispatching of service calls is called from within this function, processing by the function is mutually exclusive of processing for other tasks.			

(6) gerR_IN_SetLERR2LE

Function	LED lighting control (L ER2 LED)			
Call Format	ERRCODE gerR_IN_SetLERR2LED (ULONG ulCtrl)			
Arguments	Type Name	Variable Name	Description	I/O
	ULONG	ulCtrl	R_IN_LED_OFF (0): LED off R_IN_LED_ON (1): LED on	Input
Return Value	R_IN_OK: Normal completion R_IN_ERR_OUTOFRANGE: Abnormal completion (out of range)			
Description	This function controls lighting of L ER2 LED. Since processing to disable and enable dispatching of service calls is called from within this function, processing by the function is mutually exclusive of processing for other tasks.			

(7) gerR_IN_DisableLED

Function	Disabling the LED lighting function			
Call Format	ERRCODE gerR_IN_DisableLED (USHORT usDisable)			
Arguments	Type Name	Variable Name	Description	I/O
	USHORT	usDisable	LED lighting function disable (ON: Disable, OFF: Hold the previous value) Bit 0: Disable RUN LED Bit 1: Disable USER LED2 Bit 3: Disable ERR LED Bit 5: Disable USER LED1 Bit 6: Disable DLINK LED Bit 9: Disable L ER 1 LED Bit 10: Disable L ER 2 LED (Other than the above: Unused)	Input
Return Value	R_IN_OK: Normal completion R_IN_ERR_OUTOFRANGE: Abnormal completion (out of range)			
Description	This function disables the LED lighting function. If "ON: Disable" is set to one or more unused bits in the argument, this function returns R_IN_ERR_OUTOFRANGE: Abnormal completion (out of range). Since processing to disable and enable dispatching of service calls is called from within this function, processing by the function is mutually exclusive of processing for other tasks.			

(8) gerR_IN_EnableLED

Function	Enabling the LED lighting function			
Call Format	ERRCODE gerR_IN_EnableLED (USHORT usEnable)			
Arguments	Type Name	Variable Name	Description	I/O
	USHORT	usEnable	LED lighting function enable (ON: Enable, OFF: Hold the previous value) Bit 0: Enable RUN LED Bit 1: Enable USER LED2 Bit 3: Enable ERR LED Bit 5: Enable USER LED1 Bit 6: Enable DLINK LED Bit 9: Enable L ER 1 LED Bit 10: Enable L ER 2 LED (Other than the above: Unused)	Input
Return Value	R_IN_OK: Normal completion R_IN_ERR_OUTOFRANGE: Abnormal completion (out of range)			
Description	This function enables the LED lighting function. If "ON: Enable" is set to one or more unused bits in the argument, this function returns R_IN_ERR_OUTOFRANGE: Abnormal completion (out of range). Since processing to disable and enable dispatching of service calls is called from within this function, processing by the function is mutually exclusive of processing for other tasks.			

(9) gerR_IN_UpdateLedStatus

Function	Updating the communications state display LED			
Call Format	ERRCODE gerR_IN_UpdateLedStatus (VOID)			
Arguments	Type Name	Variable Name	Description	I/O
	—	—	—	—
Return Value	R_IN_OK: Normal completion			
Description	This function turns RUN, ERR, and D LINK LED on or off in accord with the data link state. For details, refer to Table 6.26, D LINK LED Control. Since processing to disable and enable dispatching of service calls is called from within this function, processing by the function is mutually exclusive of processing for other tasks.			

Table 6.26 D LINK LED Control

Data Link State	D LINK LED
Data link in progress (cyclic transfer in progress)	On
Data link in progress (cyclic transfer stopped)	Blinking (Cycle: 500 ms)
No data link (disconnected)	Off

(10) gerR_IN_SetSDRDLEDMode

Function	Setting SD/RD lighting mode			
Call Format	ERRCODE gerR_IN_SetSDRDLEDMode (ULONG ulMode)			
Arguments	Type Name	Variable Name	Description	I/O
	ULONG	ulMode	SD/RD lighting mode specification R_IN_LEDMODE_SDRD1_SDRD2 lighting mode: SDRD1/SDRD2 R_IN_LEDMODE_SD_RD lighting mode: SD/RD	Input
Return Value	R_IN_OK: Normal completion R_IN_ERR_OUTOFRANGE: Abnormal completion (out of range)			
Description	This function sets the lighting mode of SD/RD. Since processing to disable and enable dispatching of service calls is called from within this function, processing by the function is mutually exclusive of processing for other tasks.			

(11) gerR_IN_StartTestLED

Function	Starting the LED test			
Call Format	ERRCODE gerR_IN_StartTestLED (USHORT usTestMode)			
Arguments	Type Name	Variable Name	Description	I/O
	const USHORT	usTestMode	Test function enable/disable specification (ON: Enable, OFF: Disable) Bit 0: RUN LED Bit 1: USER LED 2 Bit 2: USER LED 1 Bit 3: D LINK LED Bit 4: ERR LED Bit 5: SD/SDRD1 LED Bit 6: RD/SDRD2 LED Bit 7: L ER 1 LED Bit 8: L ER 2 LED Bit 14: LINK1 LED Bit 15: LINK2 LED (Other than the above: Unused)	Input
Return Value	R_IN_OK: Normal completion R_IN_ERR_OUTOFRANGE: Abnormal completion (out of range) R_IN_ERR_SEMAPHORE: Abnormal completion (semaphore acquisition failure) R_IN_ERR_FATAL: Abnormal completion (fatal error occurred)			
Description	This function specifies enabling or disabling of the LED test and sets the enabled LEDs to the initial state (off). Execute the function to start the LED test. In the period from the start to execution to the end of the LED test, only call this function once at the start of the test. If the return value is R_IN_ERR_FATAL, there is a fatal error in R-IN32M4-CL3 and gR_IN_CallbackFatalError callback function (6.6(1)) created by the user is called. Acquire the R-IN32M4-CL3 fatal error. Since processing to disable and enable dispatching of service calls is called from within this function, processing by the function is mutually exclusive of processing for other tasks.			

(12) gerR_IN_ExecuteTestLED

Function	Executing the LED test			
Call Format	ERRCODE gerR_IN_ExecuteTestLED (USHORT usTestLed)			
Arguments	Type Name	Variable Name	Description	I/O
	USHORT	usTestLed	Forced on/off specification (ON: Forced on, OFF: Forced off) Bit 0: RUN LED Bit 1: USER LED 2 Bit 2: USER LED 1 Bit 3: D LINK LED Bit 4: ERR LED Bit 5: SD/SDRD1 LED Bit 6: RD/SDRD2 LED Bit 7: L ER 1 LED Bit 8: L ER 2 LED Bit 14: LINK1 LED Bit 15: LINK2 LED (Other than the above: Unused)	Input
Return Value	R_IN_OK: Normal completion R_IN_ERR_OUTOFRANGE: Abnormal completion (out of range) R_IN_ERR_SEMAPHORE: Abnormal completion (semaphore acquisition failed) R_IN_ERR_FATAL: Abnormal completion (fatal error occurred)			
Description	This function forcibly turns LEDs on or off. Use this function in the period from the stat to the end of the LED test. If the return value is R_IN_ERR_FATAL, there is a fatal error in R-IN32M4-CL3 and gR_IN_CallbackFatalError callback function (6.6(1)) created by the user is called. Acquire the R-IN32M4-CL3 fatal error. Since processing to disable and enable dispatching of service calls is called from within this function, processing by the function is mutually exclusive of processing for other tasks.			

(13) gerR_IN_StopTestLED

Function	Ending the LED test			
Call Format	ERRCODE gerR_IN_StopTestLED (VOID)			
Arguments	Type Name	Variable Name	Description	I/O
	—	—	—	—
Return Value	R_IN_OK: Normal completion R_IN_ERR_SEMAPHORE: Abnormal completion (semaphore acquisition failed) R_IN_ERR_FATAL: Abnormal completion (fatal error occurred)			
Description	Execute this function to end the LED test. In the period from the start to execution to the end of the LED test, only call this function once at the end of the test. If the return value is R_IN_ERR_FATAL, there is a fatal error in R-IN32M4-CL3 and gR_IN_CallbackFatalError callback function (6.6(1)) created by the user is called. Acquire the R-IN32M4-CL3 fatal error. Since processing to disable and enable dispatching of service calls is called from within this function, processing by the function is mutually exclusive of processing for other tasks.			

6.4.8 Network Time

(1) gerR_IN_GetNetworkTime

Function	Acquiring the network time (serial value)			
Call Format	ERRCODE gerR_IN_GetNetworkTime (USHORT* pusSerial)			
Arguments	Type Name	Variable Name	Description	I/O
	USHORT*	pusSerial	Network time (serial value) pusSerial [0]: Network time (bits 15 to 0) pusSerial [1]: Network time (bits 31 to 16) pusSerial [2]: Network time (bits 47 to 32)	Output
Return Value	R_IN_OK: Normal completion R_IN_ERR: Abnormal completion R_IN_ERR_OUTOFRANGE: Abnormal completion (out of range)			
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). When the network time is before the year of 2000, the function returns R_IN_ERR_OUTOFRANGE (Network time (serial value) out of range). Note that the acquired time is the UTC time.			

(2) gerR_IN_NetworkTimeToDate

Function	Network time (serial value) to clock information conversion			
Call Format	ERRCODE gerR_IN_NetworkTimeToDate (R_IN_TIMEINFO_T* pstTimeInfo, const USHORT* pusSerial)			
Arguments	Type Name	Variable Name	Description	I/O
	R_IN_TIMEINFO_T*	pstTimeInfo	Clock information For details, refer to Table 6.27.	Output
	const USHORT*	pusSerial	Network time (serial value) pusSerial [0]: Network time (bits 15 to 0) pusSerial [1]: Network time (bits 31 to 16) pusSerial [2]: Network time (bits 47 to 32)	Input
Return Value	R_IN_OK: Normal completion R_IN_ERR: Abnormal completion R_IN_ERR_OUTOFRANGE: Abnormal completion (out of range)			
Description	This function converts the network time (serial value in increments of 15.2587890625 µs given a starting point of January 1, 2000, 00:00:00) to the clock information (year, month, day, hour, minute, second, millisecond, and day of the week). For the millisecond of the clock information, the value obtained by multiplying the network time (bits 15 to 0) by 0.0152587890625 and rounding down to the nearest integer is set. When the year value of the network time (serial value) is 2106 or later, the function returns R_IN_ERR_OUTOFRANGE (Network time (serial value) out of range).			

Table 6.27 R_IN_TIMEINFO_T List

No.	Member		Description
1	USHORT	usYear	Year (1970 to 2105)
2	USHORT	usMonth	Month (1 to 12)
3	USHORT	usDay	Day (1 to 31)
4	USHORT	usHour	Hour (0 to 23)
5	USHORT	usMin	Minute (0 to 59)
6	USHORT	usSec	Second (0 to 59)
7	USHORT	usMsec	Millisecond (0 to 999)
8	USHORT	usWday	Day of week (0 (Sunday) to 6 (Saturday))

(3) gerR_IN_DateToNetworkTime

Function	Clock information to network time (serial value) conversion			
Call Format	ERRCODE gerR_IN_DateToNetworkTime (const R_IN_TIMEINFO_T* pstTimeInfo, USHORT* pusSerial)			
Arguments	Type Name	Variable Name	Description	I/O
	const R_IN_TIMEINFO_T*	pstTimeInfo	Clock information For details, refer to Table 6.27.	Input
	USHORT*	pusSerial	Network time (serial value)	Output
Return Value	R_IN_OK: Normal completion R_IN_ERR: Abnormal completion R_IN_ERR_OUTOFRANGE: Abnormal completion (out of range)			
Description	This function converts the clock information (year, month, day, hour, minute, second, and millisecond) to the network time (serial value in increments of 15.2587890625 µs given a starting point of January 1, 2000, 00:00:00). For the network time (bits 15–0), the value obtained by dividing the millisecond value of the clock information by 0.0152587890625 and rounding down to the nearest integer is set. When the year value of the clock information is not between 2000 and 2105, the function returns R_IN_ERR_OUTOFRANGE (clock information out of range). The R-IN32M4-CL3 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) gerR_IN_GetUnixTime

Function	Acquiring the network time (UNIX TIME)			
Call Format	ERRCODE gerR_IN_GetUnixTime (R_IN_UNIX_TIME_T* pstUnixTime)			
Arguments	Type Name	Variable Name	Description	I/O
	R_IN_UNIX_TIME_T*	pstUnixTime	Network time (UNIX time) For details, refer to Table 6.28.	Output
Return Value	R_IN_OK: Normal completion R_IN_ERR: Abnormal completion			
Description	This function acquires the Network time (UNIX time (seconds and nanoseconds)). Note that the acquired time is the UTC time.			

Table 6.28 R_IN_UNIX_TIME_T List

No.	Member		Description
1	ULONG	ulNanoSecond	UNIX time (nanoseconds)
2	ULONG	ulSecond	UNIX time (seconds)

(5) gerR_IN_UnixTimeToDate

Function	Network time (UNIX time) to clock information conversion			
Call Format	ERRCODE gerR_IN_UnixTimeToDate (R_IN_TIMEINFO_T* pstTimeInfo, const R_IN_UNIX_TIME_T* pstUnixTime)			
Arguments	Type Name	Variable Name	Description	I/O
	R_IN_TIMEINFO_T*	pstTimeInfo	Clock information For details, refer to Table 6.27.	Output
	const R_IN_UNIX_TIME_T*	pstUnixTime	Network time (UNIX time) For details, refer to Table 6.28.	Input
Return Value	R_IN_OK: Normal completion R_IN_ERR: Abnormal completion R_IN_ERR_OUTOFRANGE: Abnormal completion (out of range)			
Description	This function converts the network time (UNIX time (seconds and nanoseconds)) to the clock information (year, month, day, hour, minute, second, millisecond, and day of the week). When the value is after the year 2106 in UNIX time (second) or the value is greater than 999999992 ns in UNIX time (nanosecond), the function returns R_IN_ERR_OUTOFRANGE (UNIX time out of range). UNIX time can be converted to clock information only when a value of January 1, 1970, 00:00:01:00 or later is set. For the millisecond of the clock information, the value obtained by dividing the nanosecond value of the network time (UNIX time) by 1000000 and rounding down to the nearest integer is set.			

(6) gerR_IN_DateToUnixTime

Function	Clock information to network time (UNIX time) conversion			
Call Format	ERRCODE gerR_IN_DateToUnixTime (const R_IN_TIMEINFO_T* pstTimeInfo, R_IN_UNIX_TIME_T* pstUnixTime)			
Arguments	Type Name	Variable Name	Description	I/O
	const R_IN_TIMEINFO_T*	pstTimeInfo	Clock information For details, refer to Table 6.27.	Input
	R_IN_UNIX_TIME_T*	pstUnixTime	Network time (UNIX time) For details, refer to Table 6.28.	Output
Return Value	R_IN_OK: Normal completion R_IN_ERR: Abnormal completion R_IN_ERR_OUTOFRANGE: Abnormal completion (out of range)			
Description	This function converts the clock information (year, month, day, hour, minute, second, and millisecond) to the network time (UNIX time (seconds and nanoseconds)). When the year value of the clock information is not between 1970 and 2105, the function returns R_IN_ERR_OUTOFRANGE (clock information out of range). The R-IN32M4-CL3 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.			

(7) gerR_IN_SetUnixTimeClassA

Function	Network time (UNIX time) setting (CC-Link IE TSN Class A)			
Call Format	ERRCODE gerR_IN_SetUnixTimeClassA(R_IN_UNIX_TIME_T* pstUnixTime)			
Arguments	Type Name	Variable Name	Description	I/O
	R_IN_UNIX_TIME_T*	pstUnixTime	Network time (UNIX time) For details, refer to Table 6.28.	Input
Return Value	R_IN_OK: Normal completion R_IN_ERR: Abnormal completion R_IN_ERR_OUTOFRANGE: Abnormal completion (out of range)			
Description	This function sets the network time as the UNIX time for CC-Link IE TSN Class A. The function cannot be used for CC-Link IE TSN Class B.			

(8) gvR_IN_SetUnixOffsetTime

Function	Setting the network time offset			
Call Format	VOID gvR_IN_SetUnixOffsetTime (LONGLONG llOffsetSec, LONG lOffsetNsec, SHORT sUtcOffsetMin, SHORT sSummerTimeOffsetMin)			
Arguments	Type Name	Variable Name	Description	I/O
	LONGLONG	llOffsetSec	Offset (seconds)	Input
	LONG	lOffsetNsec	Offset (nanoseconds)	Input
	SHORT	sUtcOffsetMin	UTC offset (minutes)	Input
	SHORT	sSummerTimeOffsetMin	Summertime offset (minutes)	Input
Return Value	—			
Description	This function sets the correction value of the network time (UNIX time).			

(9) gerR_IN_GetUnixOffsetTime

Function	Acquiring the network time offset			
Call Format	ERRCODE gerR_IN_GetUnixOffsetTime (LONGLONG* pllOffsetSec, LONG* plOffsetNsec, SHORT* psUtcOffsetMin, SHORT* psSummerTimeOffsetMin)			
Arguments	Type Name	Variable Name	Description	I/O
	LONGLONG*	pllOffsetSec	Offset (seconds)	Input
	LONG*	plOffsetNsec	Offset (nanoseconds)	Input
	SHORT*	psUtcOffsetMin	UTC offset (minutes)	Input
	SHORT*	psSummerTimeOffsetMin	Summertime offset (minutes)	Input
Return Value	R_IN_OK: Normal completion R_IN_ERR: Abnormal completion R_IN_ERR_NODATA: Abnormal completion (no data)			
Description	This function acquires the correction value of the network time (UNIX time).			

The following shows an example of using the offsets to correct the network time (UNIX time).

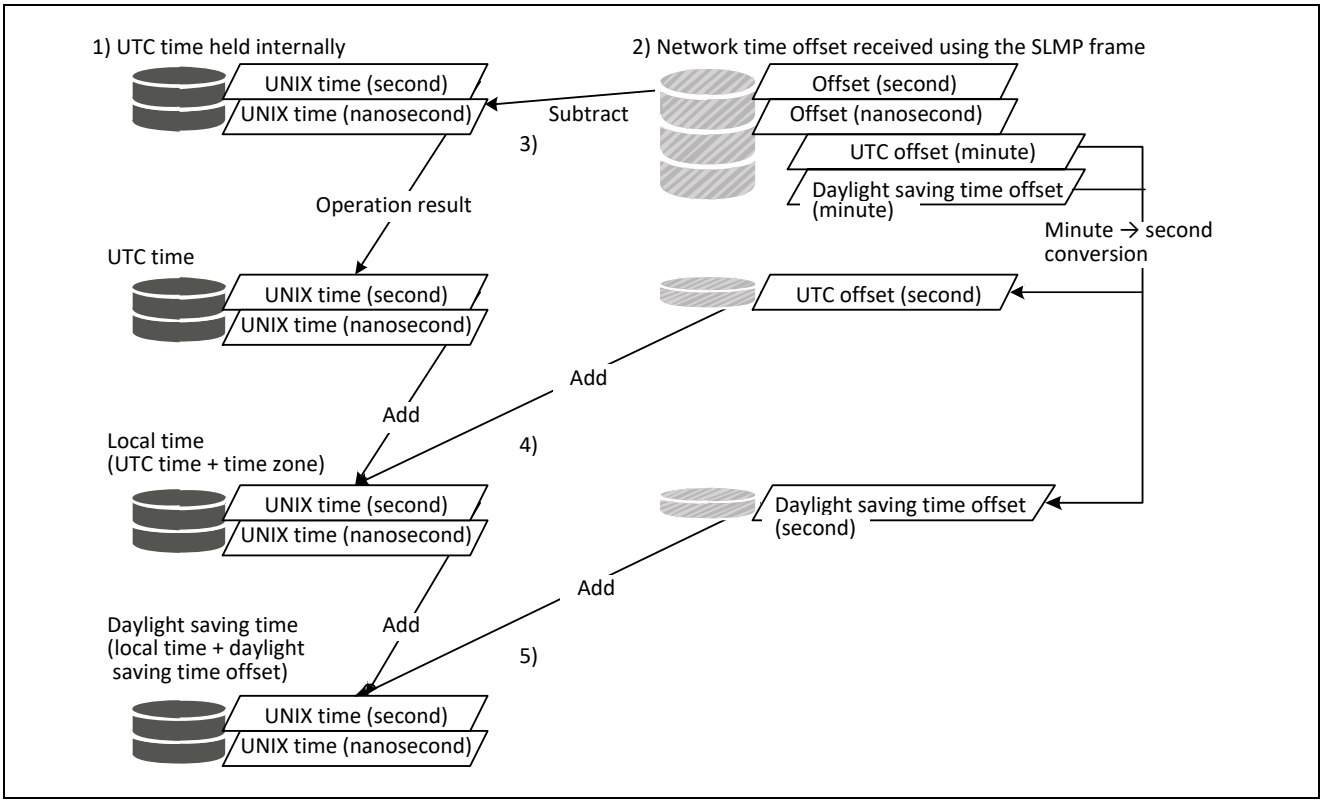


Figure 6.2 Example of Time Correction with Offset Values

1) Execute the gerR_IN_GetUnixTime function (Network time (UNIX TIME) acquisition), and acquire the network time (UNIX time).
2) Execute the gerR_IN_GetUnixOffsetTime function (Network time offset acquisition), and acquire the offset values.
3) Subtract the offset value in nanoseconds from the UTC time (UNIX time) in nanoseconds held internally in 1). (Operation result A)
   Then, perform the following (a), (b), or (c) depending on the operation result.

(a) When the operation result A is a negative value	
1.	Subtract 1 from the network time (UNIX time) in seconds. If the time in seconds before subtraction is 0 (before January 1, 1970, 00:00:01), the processing is stopped because the time cannot be corrected.
2.	Add 1000000000, which is the maximum value in nanoseconds, to the operation result A, and substitute it for the network time (UNIX time) in nanoseconds.
(b) When the operation result A is the maximum value (1000000000) in nanoseconds or greater	
1.	Add 1 to the network time (UNIX time) in seconds.
2.	Subtract 1000000000, which is the maximum value in nanoseconds, from the operation result A, and substitute it for the network time (UNIX time) in nanoseconds.
(c) When the operation result A is other than the above	
1.	No processing

2). Subtract the offset value in seconds from the UTC time (UNIX time) in seconds held internally in 2). (Operation result B)
   Note that if the operation result B is a negative value, the processing is stopped because the time cannot be corrected.

4) Add the UTC offset value in seconds to the UTC time (UNIX time) synchronized with the master station in seconds. At that time, check whether the UTC offset is other than 8000H or not.
   When the result is 8000H, the offset value does not need to be added. (8000H: No offset specification)
   When the result is other than 8000H,
   (1) multiply the UTC offset by 60 seconds.
   (2) add the operation result of (1) above to the network time (UNIX time) in seconds.

5) Add the daylight saving time offset value in seconds and the local time (UNIX time) in seconds. At that time, check whether the daylight saving time offset is other than 8000H or not.
   When the result is 8000H, the offset value does not need to be added. (8000H: No offset specification)
   When the result is other than 8000H,
   (1) multiply the UTC offset by 60 seconds.
   (2) add the operation result of (1) above to the network time (UNIX time) in seconds.

6.4.9 MDIO Access

(1) gerR_IN_EnableMACIPAccess

Function	Enabling MAC_IP access			
Call Format	ERRCODE gerR_IN_EnableMACIPAccess (VOID)			
Arguments	Type Name	Variable Name	Description	I/O
	—	—	—	—
Return Value	R_IN_OK: Normal completion R_IN_ERR_SEMAPHORE: Semaphore acquisition failed			
Description	This function enables access to MAC_IP. The function executes the service call and acquires semaphores. Therefore, shorten the period from "enabling MAC_IP access" to "disabling MAC_IP access" to the extent possible (If the user uses interrupts, use the function with the interrupts disabled during the period from "enabling MAC IP access" to "disabling MAC IP access").			

(2) gerR_IN_DisableMACIPAccess

Function	Disabling MAC_IP access			
Call Format	ERRCODE gerR_IN_DisableMACIPAccess (VOID)			
Arguments	Type Name	Variable Name	Description	I/O
	—	—	—	—
Return Value	R_IN_OK: Normal completion R_IN_ERR_FATAL: Abnormal completion (fatal error occurred)			
Description	This function disables access to MAC_IP. Execute this function after the gerR_IN_Initialize function (6.4.1(2)). If the return value is R_IN_ERR_FATAL, there is a fatal error in R-IN32M4-CL3. The gR_IN_CallbackFatalError callback function (6.6(1)) created by the user is called. Acquire the R-IN32M4-CL3 fatal error.			

(3) gerR_IN_WritePhy

Function	PHY internal register writing			
Call Format	ERRCODE gerR_IN_WritePhy (ULONG ulPort, ULONG ulAddr, ULONG ulData)			
Arguments	Type Name	Variable Name	Description	I/O
	ULONG	ulPort	Port subject to register writing R_IN_PORT1 (0): Port 1 R_IN_PORT2 (1): Port 2	Input
	ULONG	ulAddr	PHY register address	Input
	ULONG	ulData	Data to be written to PHY	Input
Return Value	R_IN_OK: Normal completion R_IN_ERR_OUTOFRANGE: Abnormal completion (out of range) R_IN_ERR_FATAL: Abnormal completion (fatal error occurred)			
Description	This function writes to PHY internal registers through the MDIO. Execute this function after the gerR_IN_Initialize function (6.4.1(2)). Use this function during the period from gerR_IN_EnableMACIPAccess (6.4.9(1)) to gerR_IN_DisableMACIPAccess (6.4.9(2)). If the return value is R_IN_ERR_FATAL, there is a fatal error in R-IN32M4-CL3. The gR_IN_CallbackFatalError callback function (6.6(1)) created by the user is called. Acquire the R-IN32M4-CL3 fatal error. For reference: Reading from or writing to PHY internal registers takes approximately 12 µs (minimum) to 24 µs (maximum).			

(4) gerR_IN_ReadPhy

Function	PHY internal register reading			
Call Format	ERRCODE gerR_IN_ReadPhy (ULONG ulPort, ULONG ulAddr, ULONG* pulData)			
Arguments	Type Name	Variable Name	Description	I/O
	ULONG	ulPort	Port subject to register reading R_IN_PORT1 (0): Port 1 R_IN_PORT2 (1): Port 2	Input
	ULONG	ulAddr	PHY register address	Input
	ULONG*	pulData	Data read from PHY	Input
Return Value	R_IN_OK: Normal completion R_IN_ERR: Abnormal completion (MDIO command end wait error) R_IN_ERR_OUTOFRANGE: Abnormal completion (out of range) R_IN_ERR_FATAL: Abnormal completion (fatal error occurred)			
Description	This function reads PHY internal registers through the MDIO. Execute this function after the gerR_IN_Initialize function (6.4.1(2)). Use this function during the period from gerR_IN_EnableMACIPAccess (6.4.9(1)) to gerR_IN_DisableMACIPAccess (6.4.9(2)). If the return value is R_IN_ERR_FATAL, there is a fatal error in R-IN32M4-CL3. The gR_IN_CallbackFatalError callback function (6.6(1)) created by the user is called. Acquire the R-IN32M4-CL3 fatal error. For reference: Reading from or writing to PHY internal registers takes approximately 12 µs (minimum) to 24 µs (maximum).			

6.4.10 SLMP Transmission/Reception

(1) gerR_IN_ReceivedSlmpMain

Function	SLMP reception main processing			
Call Format	ERRCODE gerR_IN_ReceivedSlmpMain (VOID)			
Arguments	Type Name	Variable Name	Description	I/O
	—	—	—	—
Return Value	R_IN_OK: Normal completion R_IN_ERR: Abnormal completion			
Description	This function handles processing for reception of SLMP frames.			

(2) gvR_IN_ReceivedSlmpExecution

Function	SLMP reception command execution			
Call Format	VOID gvR_IN_ReceivedSlmpExecution (VOID)			
Arguments	Type Name	Variable Name	Description	I/O
	—	—	—	—
Return Value	—			
Description	This function handles command processing corresponding to the received frame.			

(3) gblR_IN_GetReceiveSlmpStatus

Function	Acquiring SLMP reception enabled state for user reasons			
Call Format	BOOL gblR_IN_GetReceiveSlmpStatus (VOID)			
Arguments	Type Name	Variable Name	Description	I/O
	—	—	—	—
Return Value	R_IN_TRUE: Receive enabled R_IN_FALSE: Receive disabled			
Description	This function acquires the SLMP reception enabled state for user reasons.			

(4) gerR_IN_SetReceiveSlmpStatus

Function	Setting to enable SLMP reception for user reasons			
Call Format	ERRCODE gerR_IN_SetReceiveSlmpStatus (BOOL blEnable)			
Arguments	Type Name	Variable Name	Description	I/O
	BOOL	blEnable	Reception enable setting	Input
Return Value	R_IN_OK: Normal completion R_IN_ERR_OUTOFRANGE: Abnormal completion (out of range)			
Description	This function enables or disables the state in terms of SLMP reception for reasons of the user.			

(5) gerR_IN_SetSlmpCommand

Function	SLMP command determination registration			
Call Format	ERRCODE gerR_IN_SetSlmpCommand (const R_IN_SLMP_EXECUTION_RECEIVE_TBL_T* pstUserSlmpReceiveFunctionTable)			
Arguments	Type Name	Variable Name	Description	I/O
	const R_IN_SLMP_EXECUTION_RECEIVE_TBL_T*	pstUserSlmpReceiveFunctionTable	SLMP reception execution function table pointer For details, refer to Table 6.29.	Input
Return Value	R_IN_OK: Normal completion R_IN_ERR: Abnormal completion			
Description	This function notifies the R-IN32M4-CL3 driver of the SLMP reception execution function table pointer which is acquired as an argument.			

Table 6.29 R_IN_SLMP_EXECUTION_RECEIVE_TBL_T List

No.	Member		Description
1	USHORT	usRequestNumber	Number of request command registrations
2	USHORT	usResponseNumber	Number of response command registrations
3	R_IN_SLMP_FUNCTION_REQUEST_TBL_T*	pstRequest	Request command execution function table pointer (Table 6.30)
4	R_IN_SLMP_FUNCTION_RESPONSE_TBL_T*	pstResponse	Response command execution function table pointer (Table 6.33)
5	R_IN_SLMP_RESPONSE_FUNCTION	fpv3Efunction	ST response frame reception function pointer

Table 6.30 R_IN_SLMP_FUNCTION_REQUEST_TBL_T

No.	Member		Description
1	USHORT	usCommand	SLMP command
2	USHORT	usSubCommand	SLMP sub-command
3	R_IN_SLMP_REQUEST_FUNCTION	fperFunction	Command function pointer A pointer to the function that processes the received SLMP command. For the SLMP command list, refer to Table 4.3. typedefERRCODE (*R_IN _SLMP_REQUEST_FUNCTION) (VOID* pvRequestData, R_IN _SLMP_RECEIVE_INFORMATION_T* pstRequestInformation, VOID* pvReceiveData, R_IN_SLMP_SEND_INFORMATION_T*pstReceiveInformation). For R_IN_SLMP_RECEIVE_INFORMATION_T, refer to Table 6.31. For R_IN_SLMP_SEND_INFORMATION_T, refer to Table 6.32.
4	BOOL	blBroadCastSend	Broadcast transmission state
5	ULONG	ulFrameType	Frame type specification Bit 0: SLMP ST frame reception (0b: Not supported / 1b: Supported) Bit 1: SLMP MT frame reception (0b: Not supported / 1b: Supported) Bit 3: SLMP LMT frame reception (0b: Not supported / 1b: Supported) Other than the above: Fixed to 0
6	BOOL	blFrameTypeErrorSuppression	Error response suppression at the time of a mismatch in the specification of the frame type 0b: No suppression (error response) 1b: Suppression (no error response)
7	USHORT	usDataLengthErrorFincode	End code for request data length error[*1] R_IN_SLMP_FIN_CODE_NORMAL (0000H): Executes the corresponding command function registered at the command function pointer. Other than the above: Sets the setting value to the end code and sends an error response inside the R-IN32M4-CL3 driver.

Note 1. Set whether the R-IN32M4-CL3 driver sends an error response or error processing is performed in the corresponding command function when an error in the data length after the subcommand section of the received LMT frame has been detected.

To execute the corresponding command function, set an error code to the SLMP receive result in request data (receive) included information (R_IN_SLMP_RECEIVE_INFORMATION_T). For error codes, refer to "Table 6.35".

SLMP frames detected as a data length error include frames without frame numbers and frames where the data after the subcommand section is lost, as shown in the following figure.

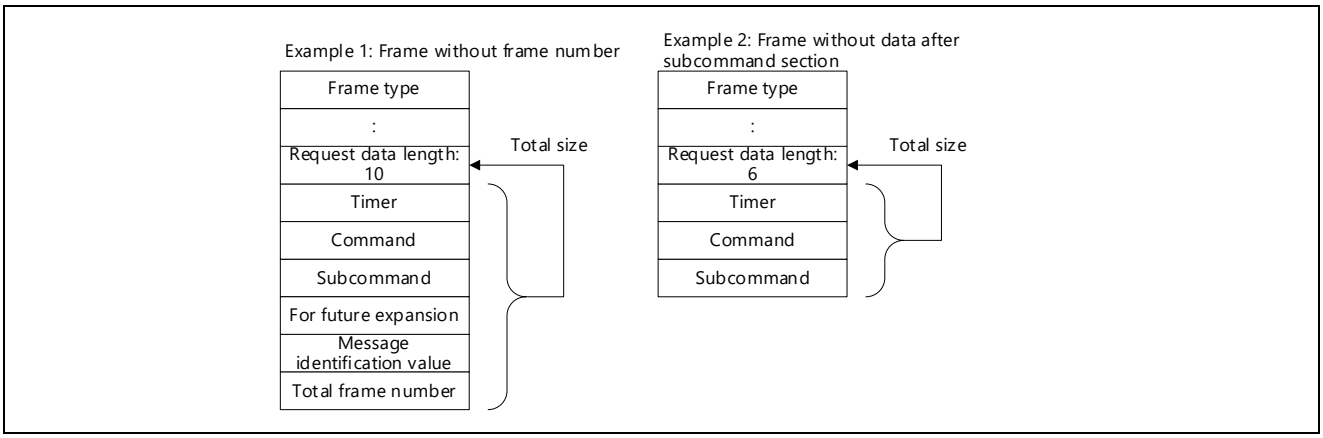


Figure 6.3 Image of Data Length Error SLMP Frames

Table 6.31 R_IN_SLMP_RECEIVE_INFORMATION_T

No.	Member		Description
1	USHORT	usRequestDataSize	Request data size
2	USHORT	usFType	Frame type
3	USHORT	usSerialNo	Serial number
4	USHORT	usReserved2	For future expansion
5	USHORT	usDstProcNo	Request destination station processor number
6	UCHAR	uchMultiDropNo	Request destination station processor subnumber
7	UCHAR	uchReserved3	For future expansion
8	USHORT	usLargeNodeNo	Request destination station extended station number
9	USHORT	usTimer	Timer
10	UCHAR	uchReserved4	For future expansion
11	UCHAR	uchReqDataId	Message identification value
12	USHORT	usDataDevideNum	Total number of divisions
13	USHORT	usDataNumber	Division number
14	ULONG	ulIPAddress	IP address
15	USHORT	usPhysicalPort	Physical port number
16	USHORT	usReceivedResult	SLMP receive result*1

Note 1. For details, refer to "Table 6.35".

Table 6.32 R_IN_SLMP_SEND_INFORMATION_T

No.	Member		Description
1	USHORT	usResponseDataSize	Response data size
2	USHORT	usFinishCode	End code

Table 6.33 R_IN_SLMP_FUNCTION_RESPONSE_TBL_T

No.	Member		Description
1	USHORT	usCommand	SLMP command
2	USHORT	usSubCommand	SLMP subcommand
3	R_IN_SLMP_REQUEST_FUNCTION	fperFunction	Command function pointer A pointer to the function that processes the received SLMP command. For the SLMP command list, refer to Table 4.3. typedefVOID (*R_IN_SLMP_RESPONSE_FUNCTION) (const VOID* pvResponseDataOffset, R_IN_SLMP_RECEIVE_RESPONSE_INFORMATION_T* pstReceiveResponseInformation). For R_IN_SLMP_RECEIVE_RESPONSE_INFORMATION_T, refer to Table 6.34.
4	ULONG	ulFrameType	Frame type specification Bit 1: SLMP MT frame reception (0b: Not supported / 1b: Supported) Bit 3: SLMP LMT frame reception (0b: Not supported / 1b: Supported) Other than the above: Fixed to 0
5	BOOL	blDataLengthErrorDetection	Setting for response data length error[*1] R_IN_TRUE: Executes the function registered at the command function pointer. R_IN_FALSE: Discards the received frame.

Note 1. Set whether the R-IN32M4-CL3 driver sends an error response or error processing is performed in the corresponding command function when an error in the data length after the subcommand section of the received LMT frame has been detected.
To execute the corresponding command function, set an error code to the SLMP receive result in response data (receive) included information (R_IN_SLMP_RECEIVE_RESPONSE_INFORMATION_T). For error codes, refer to "Table 6.35".

Frames detected as a data length error include frames without frame ID numbers and frames where the data after the subcommand section is lost, as shown in "Figure 6.3".

Table 6.34 R_IN_SLMP_RECEIVE_RESPONSE_INFORMATION_T

No.	Member		Description
1	USHORT	usResponseDataSize	Response data size
2	USHORT	usFType	Frame type
3	USHORT	usSerialNo	Serial number
4	USHORT	usReserved2	For future expansion
5	USHORT	usDstProcNo	Request destination station processor number
6	UCHAR	uchMultiDropNo	Request destination station processor subnumber
7	UCHAR	uchReserved3	For future expansion
8	USHORT	usLargeNodeNo	Request destination station extended station number
9	USHORT	usFinishCode	End code
10	UCHAR	uchReserved4	For future expansion
11	UCHAR	uchResDataId	Message identification value
12	USHORT	usDataDevideNum	Total number of divisions
13	USHORT	usDataNumber	Division number
14	ULONG	ulIPAddress	IP address
15	USHORT	usPhysicalPort	Physical port number
16	USHORT	usReceivedResult	SLMP receive result*1

Note 1. For details, refer to "Table 6.35".

Table 6.35 List of SLMP Receive Result Error Codes

No.	Error code	Value	Description
1	R_IN_SLMP_RECEIVED_RESULT_NORMAL	0000H	Received normally
2	R_IN_SLMP_RECEIVED_RESULT_ERR_DATALEN	0100H	Detected a data length error

(6) gerR_IN_SendSlmpFrame

Function	SLMP frame transmission			
Call Format	ERRCODE gerR_IN_SendSlmpFrame (VOID)			
Arguments	Type Name	Variable Name	Description	I/O
	—	—	—	—
Return Value	R_IN_OK: Normal completion R_IN_ERR: Abnormal completion			
Description	This function handles processing for transmission of SLMP frames.			

(7) gerR_IN_WriteSlmpSendBuffer

Function	SLMP transmission buffer writing			
Call Format	ERRCODE gerR_IN_WriteSlmpSendBuffer (const VOID* pvReceiveBuffer, USHORT usSize, ULONG ulIpAddress, USHORT usPort, USHORT usPhysicalPort, VOID* pvSendBuffer)			
Arguments	Type Name	Variable Name	Description	I/O
	const VOID*	pvReceiveBuffer	Reception buffer	Input
	USHORT	usSize	Transmission size	Input
	ULONG	ulIpAddress	Destination IP address	Input
	USHORT	usPort	Destination port number	Input
	USHORT	usPhysicalPort	Physical port number	Input
	VOID*	pvSendBuffer	Transmission buffer	Input
Return Value	R_IN_OK: Normal completion R_IN_ERR: Abnormal completion R_IN_ERR_OUTOFRANGE: Abnormal completion (out of range)			
Description	This function writes SLMP frames to the transmission buffer of R-IN32M4-CL3. Set the reception buffer as follows in accord with the frame to be sent. • When sending the SLMP request frame, set NULL. • When sending the SLMP response frame, set the start address of the received request frame. Do not set 45238 for the destination port number. Specify the start address of the SLMP frame for transmission in the transmission buffer.			

(8) gblR_IN_GetSlmpSendBufferUsed

Function	SLMP transmission buffer usage acquisition			
Call Format	BOOL gblR_IN_GetSlmpSendBufferUsed (VOID)			
Arguments	Type Name	Variable Name	Description	I/O
	—	—	—	—
Return Value	R_IN_TRUE: The SLMP transmission buffer contains data. R_IN_FALSE: The SLMP transmission buffer contains no data.			
Description	This function acquires the usage state of the SLMP transmission buffer which is held internally.			

(9) gerR_IN_SetSlmpResponseFrameSendRequest

Function	SLMP response frame transmission request			
Call Format	ERRCODE gerR_IN_SetSlmpResponseFrameSendRequest (VOID* pvResponseDataTopAddress, R_IN_SLMP_SEND_INFORMATION_T* pstReceiveInformation)			
Arguments	Type Name	Variable Name	Description	I/O
	VOID*	pvResponseDataTopAddress	Response data start address	Input
	R_IN_SLMP_SEND_INFORMATION_T*	pstReceiveInformation	Response data size For details, refer to Table 6.32.	Input
Return Value	R_IN_OK: Normal completion R_IN_ERR: Abnormal completion			
Description	This function creates a response frame based on the response data specified as an argument and received data (SLMP request frame) which is held internally. Use this function when sending a response to the frame acquired from "gvR_IN_CallbackNotifyReceivedSlmp"(6.6(3)).			

(10) gvR_IN_ReleaseSlmpReceiveFrame

Function	SLMP reception buffer release request			
Call Format	ERRCODE gvR_IN_ReleaseSlmpReceiveFrame (VOID)			
Arguments	Type Name	Variable Name	Description	I/O
	—	—	—	—
Return Value	—			
Description	This function releases the area secured by the received data (SLMP frame).			

6.4.11 SLMP Command Execution

(1) gvR_IN_ExecuteReset

Function	Home station reset			
Call Format	VOID gvR_IN_ExecuteReset (VOID)			
Arguments	Type Name	Variable Name	Description	I/O
	—	—	—	—
Return Value	—			
Description	This function resets the home station (R-IN32M4-CL3).			

(2) gerR_IN_StartSlmpRequestTimer

Function	Starting the SLMP request waiting timer			
Call Format	ERRCODE gerR_IN_StartSlmpRequestTimer (const R_IN_TIMEOUT_FUNCTION fpvTimeoutFunction , ULONG ulLimitTime , USHORT usTimerId)			
Arguments	Type Name	Variable Name	Description	I/O
	const R_IN_TIMEOUT_FUNCTION	fpvTimeoutFunction	Timeout execution function typedef VOID (*R_IN_TIMEOUT_FUNCTION) (VOID);	Input
	ULONG	ulLimitTime	Timer time (ms)	Input
	USHORT	usTimerId	Timer ID	Input
Return Value	R_IN_OK: Normal completion R_IN_ERR: Abnormal completion R_IN_ERR_OUTOFRANGE: Abnormal completion (out of range)			
Description	This function starts a timer for monitoring the time from transmission of a response message for the received request message to reception of a next request message. Use this function for commands that use timers, such as data backup/restoration.			

(3) gerR_IN_StopSlmpRequestTimer

Function	Stopping the SLMP request waiting timer			
Call Format	ERRCODE gerR_IN_StopSlmpRequestTimer (USHORT usTimerId)			
Arguments	Type Name	Variable Name	Description	I/O
	USHORT	usTimerId	Timer ID	Input
Return Value	R_IN_OK: Normal completion R_IN_ERR_OUTOFRANGE: Abnormal completion (out of range)			
Description	This function stops a timer for monitoring the time from transmission of a response message for the received request message to reception of a next request message. Use this function for commands that use timers, such as data backup/restoration.			

(4) gerR_IN_GetMasterIPAddress

Function	Acquiring the management master station's IP address			
Call Format	ERRCODE gerR_IN_GetMasterIPAddress (ULONG* pulIPAddress, USHORT* pusPhysicalPort)			
Arguments	Type Name	Variable Name	Description	I/O
	ULONG*	pulIPAddress	IP address	Output
	USHORT*	pusPhysicalPort	Physical port number	Output
Return Value	R_IN_OK: Normal completion R_IN_ERR: Abnormal completion			
Description	This function acquires the IP address and physical port number of the connected management master station.			

(5) gerR_IN_CheckIpAddressSlmp

Function	Checking the IP address			
Call Format	ERRCODE gerR_IN_CheckIpAddressSlmp (ULONG ulIPAddress, ULONG ulSubnetMask)			
Arguments	Type Name	Variable Name	Description	I/O
	ULONG	ulIPAddress	IP address	Input
	ULONG	ulSubnetMask	Subnet mask	Input
Return Value				
Description	This function checks the validity of the IP address.			

6.4.12 Error History

(1) gerR_IN_SetErrorHistory

Function	Registering the error history			
Call Format	ERRCODE gerR_IN_SetErrorHistory (R_IN_ERROR_INFORMATION_T* pstErrorInformation)			
Arguments	Type Name	Variable Name	Description	I/O
	R_IN_ERROR_INFORMATION_T*	pstErrorInformation	Error information For details, refer to Table 6.36.	Input
Return Value	R_IN_OK: Normal completion R_IN_ERR: Abnormal completion R_IN_ERR_OUTOFRANGE: Abnormal completion (out of range)			
Description	This function registers error information as history. The function registers error information of the main module (controller information) when extension modules/slice remote I/O modules are used and the compiler switch "CURERR_OPTIONINFO_ENABLE" is defined. Do not perform this processing while the gvR_IN_ClearErrorHistory function is being executed.			

Table 6.36 R_IN_ERROR_INFORMATION_T List

No.	Member		Description
1	USHORT	usErrorCode	Error code
2	UCHAR	uchErrorDetailSize	Error detail size
3	USHORT	ausErrorDetail [10]	Error detail information

(2) gerR_IN_ClearErrorHistory

Function	Clearing the error history			
Call Format	VOID gvR_IN_ClearErrorHistory (VOID)			
Arguments	Type Name	Variable Name	Description	I/O
	—	—	—	—
Return Value	—			
Description	This function clears the error history information. When the extension modules/slice remote I/O modules are used and the compiler switch "CURERR_OPTIONINFO_ENABLE" is defined, do not use this function because it clears all the error history logs including the number of extension modules/slice remote I/O modules. When the compiler switch "CURERR_OPTIONINFO_ENABLE" is defined, use the following to clear the error history logs of the main module (controller information) and extension modules/slice remote I/O modules (option information). ・ gvR_IN_ClearErrorHistoryController (Section 6.4.12(3)) ・ gerR_IN_ClearErrorHistoryOption (Section 6.4.12(5)) Do not perform this processing while the gerR_IN_SetErrorHistory function is being executed.			

(3) gvR_IN_ClearErrorHistoryController

Function	Error history clear (controller information)			
Call Format	VOID gvR_IN_ClearErrorHistoryController(VOID)			
Arguments	Type Name	Variable Name	Description	I/O
	—	—	—	—
Return Value	—			
Description	This function clears error history logs of controller information. Do not perform this processing while the error history is being registered.			

(4) gerR_IN_SetErrorHistoryOption

Function	Error history registration (option information)			
Call Format	ERRCODE gerR_IN_SetErrorHistoryOption (R_IN_ERROR_INFORMATION_T* pstErrorInformation, USHORT usOptNum)			
Arguments	Type Name	Variable Name	Description	I/O
	R_IN_ERROR_INFORMATION_T*	pstErrorInformation	Error information For details, refer to "Table 6.36".	Input
	USHORT	usOptNum	Option information number	Input
Return Value	R_IN_OK: Normal completion R_IN_ERR: Abnormal completion R_IN_ERR_OUTOFRANGE: Abnormal completion (out of range)			
Description	This function registers the option information number in the argument and error information (error code, error detail size, error detail information) as error history logs. Do not perform this processing while the gerR_IN_ClearErrorHistoryOption function is being executed.			

(5) gerR_IN_ClearErrorHistoryOption

Function	Error history clear (option information)			
Call Format	ERRCODE gerR_IN_ClearErrorHistoryOption(USHORT usOptNum)			
Arguments	Type Name	Variable Name	Description	I/O
	USHORT	usOptNum	Option information number	Input
Return Value	R_IN_OK: Normal completion R_IN_ERR: Abnormal completion R_IN_ERR_OUTOFRANGE: Abnormal completion (out of range)			
Description	This function clears error history logs corresponding to the option information number in the argument. Do not perform this processing while the gerR_IN_SetErrorHistoryOption function is being executed.			

6.4.13 Hardware Test

(1) gerR_IN_IEEETest

Function	IEEE802.3ab compliance test			
Call Format	VOID gerR_IN_IEEETest (USHORT usMode)			
Arguments	Type Name	Variable Name	Description	I/O
	USHORT	usMode	Test Mode R_IN_IEEE_MODE1 (0): Mode 1 R_IN_IEEE_MODE2 (1): Mode 2 R_IN_IEEE_MODE3 (2): Mode 3 R_IN_IEEE_MODE4 (3): Mode 4 R_IN_IEEE_END (4): Test end	Input
Return Value	R_IN_OK: Normal completion R_IN_ERR: Abnormal completion R_IN_ERR_FATAL: Abnormal completion (fatal error)			
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 is R_IN _ERR_FATAL, there is a fatal error in R-IN32M4-CL3. Acquire the R-IN32M4-CL3 fatal error.			

(2) gerR_IN_InitializeLoopBackTest

Function	Loopback communications test initialization			
Call Format	ERRCODE gerR_IN_InitializeLoopBackTest(VOID)			
Arguments	Type Name	Variable Name	Description	I/O
	–	–	–	–
Return Value	R_IN_OK: Normal completion R_IN_ERR: Abnormal completion R_IN_ERR_OUTOFRANGE: Abnormal completion (out of range) R_IN_ERR_FATAL: Abnormal completion (fatal err)			
Description	This function handles initialization to run the loopback communications test.			

(3) gerR_IN_SendLoopBackTest

Function	Loopback communications test data transmission			
Call Format	ERRCODE gerR_IN_SendLoopBackTest (ULONG ulPort)			
Arguments	Type Name	Variable Name	Description	I/O
	ULONG	ulPort	Test target port	Input
Return Value	R_IN_OK: Normal completion R_IN_ERR: Abnormal completion R_IN_ERR_OUTOFRANGE: Abnormal completion (out of range)			
Description	This function sends communications test data from the test target port specified by the argument. To run the test, connect Ethernet port 1 and Ethernet port 2 with an Ethernet cable.			

(4) gerR_IN_ReceiveLoopBackTest

Function	Loopback communications test data reception			
Call Format	ERRCODE gerR_IN_ReceiveLoopBackTest (ULONG ulPort)			
Arguments	Type Name	Variable Name	Description	I/O
	ULONG	ulPort	Test target port	Input
Return Value	R_IN_OK: Normal completion R_IN_ERR: Abnormal completion R_IN_ERR_OUTOFRANGE: Abnormal completion (out of range) R_IN_ERR_NODATA: Abnormal completion (no data) R_IN_ERR_FATAL: Abnormal completion (fatal error) R_IN_ERR_TIMEOUT: Abnormal completion (timeout)			
Description	This function receives non-cyclic frames and compares the received data with the transmitted data.			

6.4.14 General Common Functions

(1) gverR_IN_CopyMemory

Function	Memory copy			
Call Format	ERRCODE gerR_IN_CopyMemory (VOID* pvDestination, const VOID* pvSource, ULONG ulSize)			
Arguments	Type Name	Variable Name	Description	I/O
	VOID*	pvDestination	Copy destination data	Output
	const VOID*	pvSource	Copy source data	Input
	ULONG	ulSize	Copy data size (in units of byte)	Input
Return Value	R_IN_OK: Normal completion R_IN_ERR: Abnormal completion			
Description	This function copies memory data in 32-, 16-, or 8-bit units.			

(2) gerR_IN_FillMemory

Function	Memory fill (8-bit)			
Call Format	ERRCODE gerR_IN_FillMemory (VOID* pvDestination, UCHAR uchFillData, ULONG ulSize)			
Arguments	Type Name	Variable Name	Description	I/O
	VOID*	pvDestination	Fill destination data	Output
	UCHAR	uchFillData	Fill data (00H–FFH)	Input
	ULONG	ulSize	Fill destination data size (in units of byte)	Input
Return Value	R_IN_OK: Normal completion R_IN_ERR: Abnormal completion			
Description	This function fills memory areas with the specified data in 8-bit units.			

(3) gerR_IN_FillMemory16

Function	Memory fill (16-bit)			
Call Format	ERRCODE gerR_IN_FillMemory16 (VOID* pvDestination, USHORT usFillData, ULONG ulSize)			
Arguments	Type Name	Variable Name	Description	I/O
	VOID*	pvDestination	Fill destination data	Output
	USHORT	usFillData	Fill data (0000H–FFFFH)	Input
	ULONG	ulSize	Fill destination data size (in units of byte)	Input
Return Value	R_IN_OK: Normal completion R_IN_ERR: Abnormal completion			
Description	This function fills memory areas with the specified data in 16-bit units.			

(4) gerR_IN_FillMemory32

Function	Memory fill (32-bit)			
Call Format	ERRCODE gerR_IN_FillMemory32 (VOID* pvDestination, ULONG ulFillData, ULONG ulSize)			
Arguments	Type Name	Variable Name	Description	I/O
	VOID*	pvDestination	Fill destination data	Output
	ULONG	ulFillData	Fill data (00000000H–FFFFFFFFH))	Input
	ULONG	ulSize	Fill destination data size (in units of byte)	Input
Return Value	R_IN_OK: Normal completion R_IN_ERR: Abnormal completion			
Description	This function fills memory areas with the specified data in 32-bit units.			

(5) gerR_IN_EndianShort

Function	Endian conversion (USHORT)			
Call Format	ERRCODE gerR_IN_EndianShort (USHORT* pusVal)			
Arguments	Type Name	Variable Name	Description	I/O
	USHORT*	pusVal	Conversion target	Output
Return Value	R_IN_OK: Normal completion R_IN_ERR: Abnormal completion			
Description	This function performs endian conversion on 2-byte arguments.			

(6) gerR_IN_EndianLong

Function	Endian conversion (ULONG)			
Call Format	ERRCODE gerR_IN_EndianLong (ULONG* pulVal)			
Arguments	Type Name	Variable Name	Description	I/O
	ULONG*	pulVal	Conversion target	Output
Return Value	R_IN_OK: Normal completion R_IN_ERR: Abnormal completion			
Description	This function performs endian conversion on 4-byte arguments.			

(7) gerR_IN_EndianLongLong

Function	Endian conversion (ULONGULONG)			
Call Format	ERRCODE gerR_IN_EndianLongLong (ULONGLONG* pullVal)			
Arguments	Type Name	Variable Name	Description	I/O
	ULONGLONG*	pullVal	Conversion target	Output
Return Value	R_IN_OK: Normal completion R_IN_ERR: Abnormal completion			
Description	This function performs endian conversion on 8-byte arguments.			

(8) gvR_IN_DisableInt

Function	Disabling interrupts			
Call Format	VOID gvR_IN_DisableInt (VOID)			
Arguments	Type Name	Variable Name	Description	I/O
	—	—	—	—
Return Value	—			
Description	This function disables execution of the interrupt program. When the interrupt disable or interrupt enable function is to be executed by a task to which exclusive control applies, execute the dispatch disable or dispatch enable function before and after the interrupt disable or interrupt enable function so that the current task does not switch to another higher priority task before execution of the interrupt disable or interrupt enable function. Example: 1. Disabling dispatching 2. Disabling interrupts 3. Processing to which exclusive control applies 4. Enabling interrupts 5. Enabling dispatching			

(9) gvR_IN_EnableInt

Function	Enabling interrupts			
Call Format	VOID gvR_IN_EnableInt (VOID)			
Arguments	Type Name	Variable Name	Description	I/O
	—	—	—	—
Return Value	—			
Description	This function enables the execution of the interrupt program. When the interrupt disable or interrupt enable function is to be executed by a task to which exclusive control applies, execute the dispatch disable or dispatch enable function before and after the interrupt disable or interrupt enable function so that the current task does not switch to another higher priority task before execution of the interrupt disable or interrupt enable function. Example: 1. Disabling dispatching 2. Disabling interrupts 3. Processing to which exclusive control applies 4. Enabling interrupts 5. Enabling dispatching			

(10) gvR_IN_DisableDispatch

Function	Disabling dispatching			
Call Format	VOID gvR_IN_DisableDispatch (VOID)			
Arguments	Type Name	Variable Name	Description	I/O
	—	—	—	—
Return Value	—			
Description	This function disables the scheduling (switching) of tasks. Do not execute the dispatch disable or enable function while interrupts are disabled. If tasks are switched at the time of executing the dispatch enable function, an SVC interrupt cannot be executed, resulting in a hard fault exception.			

(11) gvR_IN_EnableDispatch

Function	Enabling dispatching			
Call Format	VOID gvR_IN_EnableDispatch (VOID)			
Arguments	Type Name	Variable Name	Description	I/O
	—	—	—	—
Return Value	—			
Description	This function enables the scheduling (switching) of tasks. Do not execute the dispatch disable or enable function while interrupts are disabled. If tasks are switched at the time of executing the dispatch enable function, an SVC interrupt cannot be executed, resulting in a hard fault exception.			

6.4.15 Network-Synchronized Communications

(1) gerR_IN_GetSyncDeviationFlag

Function	Time synchronization deviation detection			
Call Format	ERRCODE gerR_IN_GetSyncDeviationFlag (BOOL* pblSyncDeviationFlag)			
Arguments	Type Name	Variable Name	Description	I/O
	BOOL*	pblSyncDeviationFlag	Time synchronization deviation detection R_IN_FALSE: Time synchronization deviation not detected R_IN_TRUE: Time synchronization deviation detected	Output
Return Value	R_IN_OK: Normal completion R_IN_ERR: Abnormal completion			
Description	This function acquires the state in terms of deviation of time synchronization which has arisen due to starting to participate in or being reconnected to the network.			

(2) gerR_IN_StopAppSyncSignal

Function	Stopping the output of the synchronization signal			
Call Format	ERRCODE gerR_IN_StopAppSyncSignal (USHORT* pusResult)			
Arguments	Type Name	Variable Name	Description	I/O
	USHORT*	pusResult	The result of execution R_IN_SUCCEED: Success R_IN_FAIL: Failure	Output
Return Value	R_IN_OK: Normal completion R_IN_ERR: Abnormal completion			
Description	This function stops the output of the synchronization signal. Once this function has been called, the output of the synchronization signal stopping will no longer awaken the given synchronization processing task. The return value being R_IN_FAIL indicates that the output of the synchronization signal has already been stopped.			

(3) gerR_IN_SetWdcThreshold

Function	Setting the counter threshold for consecutive watchdog counter checking errors			
Call Format	ERRCODE gerR_IN_SetWdcThreshold (USHORT usWdcThreshold)			
Arguments	Type Name	Variable Name	Description	I/O
	USHORT	usWdcThreshold	Threshold for the number of consecutive WDC errors For details, refer to "Table 6.37".	Input
Return Value	R_IN_OK: Normal completion R_IN_ERR_OUTOFRANGE: Abnormal completion (out of range)			
Description	This function sets the counter threshold for consecutive watchdog counter checking errors specified by the request message of the SLMP watchdog counter setting command.			

Table 6.37 Continuous WDC Error Count Threshold Values

Bit	Name	Setting value
15	Watchdog counter check error continuous counter threshold value setting status	0: Not set 1: Set
14 to 0	Watchdog counter check error continuous counter threshold value	0000H to 0FFFH (0 to 4095)

(4) gerR_IN_MakeResponseWdcInfomation

Function	Creating watchdog counter information setting response data			
Call Format	ERRCODE gerR_IN_MakeResponseWdcInfomation (VOID* pvResponseData, USHORT* pusDataSize)			
Arguments	Type Name	Variable Name	Description	I/O
	VOID*	pvResponseData	Response data*1	Output
	USHORT*	pusDataSize	Response data size	Output
Return Value	R_IN_OK: Normal completion R_IN_ERR: Abnormal completion R_IN_ERR_OUTOFRANGE: Abnormal completion (out of range)			
Description	This function creates response data of the SLMP watchdog counter setting command (request) based on the information managed by the R-IN32M4-CL3 driver. If this function is executed while sub-payload information is not fixed, an R_IN_ERR_NODATA error occurs. Call this function after sub-payload information has been fixed upon reception of a CyclicConfigTrnSubPayload frame or CyclicConfigRcvSubPayload frame. Since the SLMP watchdog counter information setting request command is designed to suit the protocol whereby commands are sent by the master station after the reception of a CyclicConfigTrnSubPayload frame or CyclicConfigRcvSubPayload frame, this function must be executed upon reception of the SLMP watchdog counter information setting request command.			

Note 1.The following is the response data structure. For details, refer to the "SLMP Specification" published by the CC-Link Partner Association.

No.	Item in the response data (ResSetWatchdogCounterInfo)		Size
1	transmitSubPayloadNum		2
2	transmitSubPayloadInfo1	watchdogCounterExistence	1
3		Reserved	1
4		receiveMemoryAddr	4
5		watchdogCounterUlIndex	2
6		watchdogCounterUlSubIndex	1
7		Reserved	1
8		watchdogCounterUlOffset	2
9	transmitSubPayloadInfo2 to m (Same as transmitSubPayloadInfo1)		12 × (m - 1)
10	receiveSubPayloadNum		2
11	receiveSubPayloadInfo1	watchdogCounterExistence	1
12		Reserved	1
13		watchdogCounterDlIndex	2
14		watchdogCounterDlSubIndex	1
15		Reserved	1
16		watchdogCounterDlOffset	2
17	receiveSubPayloadInfo2 to n (Same as receiveSubPayloadInfo1)		8 × (n - 1)

m = transmitSubPayloadNum

n = receiveSubPayloadNum

(5) gvR_IN_IncrementWdcUL

Function	Incrementing the watchdog counter (for transmission)			
Call Format	VOID gvR_IN_IncrementWdcUL (VOID)			
Arguments	Type Name	Variable Name	Description	I/O
	—	—	—	—
Return Value	—			
Description	This function increments the normal value of the transmission watchdog counter. The value to be added to the normal value of the transmission watchdog counter in response to a single execution of this function is set by a member of the R-IN32M4-CL3 unit information (pstUnitInfo) structure, which is an argument of gerR_IN_Initialize (R-IN32M4-CL3 initialization).			

(6) gvR_IN_LatchWdcUL

Function	Updating the watchdog counter latched value (for transmission)			
Call Format	VOID gvR_IN_LatchWdcUL (VOID)			
Arguments	Type Name	Variable Name	Description	I/O
	—	—	—	—
Return Value	—			
Description	This function sets the normal value for the transmission watchdog counter as the value of the watchdog counter to be sent to and latched by the master station.			

(7) gerR_IN_GetWdcUL

Function	Acquiring the watchdog counter UL			
Call Format	ERRCODE gerR_IN_GetWdcUL (BOOL blWdcULValidFlag, USHORT* pusWdcUL)			
Arguments	Type Name	Variable Name	Description	I/O
	BOOL	blWdcULValidFlag	Watchdog counter enable/disable flag transmission control flag R_IN_FALSE: Watchdog counter disabled R_IN_TRUE: Watchdog counter enabled	Input
	USHORT*	pusWdcUL	Watchdog counter (for transmission)	Output
Return Value	R_IN_OK: Normal completion R_IN_ERR: Abnormal completion			
Description	This function acquires the value of the watchdog counter UL to be sent to the master station as cyclic data based on the watchdog counter value latched by "Watchdog counter latched value update (for transmission)". To disable the watchdog counter UL, set the watchdog counter enable/disable flag transmission control flag to "R_IN_FALSE".			

(8) gerR_IN_CheckWdcDL

<table>
<tr><td>Function</td><td colspan="4">Checking the watchdog counter (for reception)</td></tr>
<tr><td>Call Format</td><td colspan="4">ERRCODE gerR_IN_CheckWdcDL (USHORT usWdc, USHORT* pusCheckResult)</td></tr>
<tr><td rowspan="3">Arguments</td><td>Type Name</td><td>Variable Name</td><td>Description</td><td>I/O</td></tr>
<tr><td>USHORT</td><td>usWdc</td><td>Received watchdog counter</td><td>Input</td></tr>
<tr><td>USHORT*</td><td>pusCheckResult</td><td>Received watchdog counter check result
R_IN_WDC_OK: Normal
R_IN_WDC_NOT_COMP: Completed without comparison
R_IN_WDC_NG_LESS: Abnormal (within the threshold range)
R_IN_WDC_NG_OVER: Abnormal (out of the threshold range)</td><td>Output</td></tr>
<tr><td>Return Value</td><td colspan="4">R_IN_OK: Normal completion
R_IN_ERR: Abnormal completion</td></tr>
<tr><td>Description</td><td colspan="4">This function checks the received watchdog counter based on the held threshold and returns the result.
<table>
<tr><td>Check Result</td><td>State</td></tr>
<tr><td>R_IN_WDC_OK</td><td>None of the following (the received watchdog counter is normal)</td></tr>
<tr><td>R_IN_WDC_NOT_COMP</td><td>If any of the following conditions are satisfied
• The receive watchdog counter specified in the argument is invalid.
• The receive watchdog counter specified in the argument is valid, but the receive watchdog counter at the last execution is invalid.
• The data link status is not Data link (cyclic communications being performed).</td></tr>
<tr><td>R_IN_WDC_NG_LESS</td><td>Either of the following states continues a number of times less than the counter threshold for consecutive watchdog counter checking errors.
• The value received from the watchdog counter specified as the argument is less than or equal to the value counted by the same watchdog counter in the previous execution.
• The difference in the counter values exceeds twice the watchdog counter increment value.</td></tr>
<tr><td>R_IN_WDC_NG_OVER</td><td>Either of the following states continues a number of times greater than or equal to the counter threshold for consecutive watchdog counter checking errors.
• The value received from the watchdog counter specified as the argument is less than or equal to the value counted by the same watchdog counter in the previous execution.
• The difference in the counter values exceeds twice the watchdog counter increment value.</td></tr>
</table></td></tr>
</table>

6.4.16 CANopen Communications

(1) gerR_IN_CanInit

Function	CANopen communications function initialization			
Call Format	ERRCODE gerR_IN_CanInit (const R_IN_CAN_OD_TABLE_T* pstObjectDictionary, USHORT usObjectNumber R_IN_CAN_ERROR_DETAIL_T* pstErrorDetail)			
Arguments	Type Name	Variable Name	Description	I/O
	const R_IN_CAN_OD_TABLE_T*	pstObjectDictionary	Start address of the object dictionary Refer to Table 6.38.	Input
	USHORT	usObjectNumber	Number of elements of the object dictionary	Input
	R_IN_CAN_ERROR_DETAIL_T*	pstErrorDetail	Error detail information Refer to Table 6.39.	Output
Return Value	R_IN_OK: Normal completion R_IN_ERR: Abnormal completion			
Description	This function initializes the CANopen communications function. The number of object dictionaries which can be registered is fixed to 1 to 8.			

Table 6.38 R_IN_CAN_OD_TABLE_T List

No.	Member		Description
1	const R_IN_CAN_OD_T	*pstObjectDictionary	Start address of the object dictionary*3
2	USHORT	usObjectDictionaryNum	Number of elements of the object dictionary

Note 3. For details of the object dictionary structure, refer to Table 6.40 R_IN_CAN_OD_T List.

Table 6.39 R_IN_CAN_ERROR_DETAIL_T List

No.	Member		Description
1	ULONG	ulAbortCode	Abort code
2	USHORT	usObjectDictionaryNo	Object dictionary number
3	USHORT	usIndex	Index
4	UCHAR	uchSubIndex	Sub-index

(2) gerR_IN_CanGetValidObjectDictionaryNumber

Function	Object dictionary handling acquisition			
Call Format	ERRCODE gerR_IN_CanGetValidObjectDictionaryNumber (USHORT* pusObjectDictionaryNumber)			
Arguments	Type Name	Variable Name	Description	I/O
	USHORT*	pusObjectDictionaryNumber	Number of valid object dictionaries	Output
Return Value	R_IN_OK: Normal completion R_IN_ERR: Abnormal completion R_IN_ERR_NODATA: Abnormal end (Number of valid object dictionaries not determined)			
Description	This function acquires the number of valid object dictionaries. The number of valid object dictionaries is determined while the NMT state is INIT. If the function is executed before the number has been determined, the function returns R_IN_ERR_NODATA.			

Table 6.40 R_IN_CAN_OD_T List

No.	Member		Description
1	USHORT	usIndex	Index
2	UCHAR	uchMaxSubIndex	Maximum sub-index This member is set to indicate a sub-index with a number no less than 1. Sub-index 0 is not an available setting because it is the number of entries. If the object code is "R_IN_CAN_VAR", the setting is 0.
3	UCHAR	uchObjectCode	The following object codes are set. • R_IN_CAN_VAR (07H): Single value • R_IN_CAN_ARRAY (08H): Multiple sub-indices of the same type are included. • R_IN_CAN_RECORD (09H): Multiple sub-indices of different types are included. For details, refer to the CC-Link IE TSN Specification (Overview) published by the CC-Link Partner Association.
4	R_IN_CAN_OBJECT_DATA_T	*pstObjectData	This parameter is setting the start address of the table which defines data of each sub-index. The setting is required even if the number of uchMaxSubIndex is 0. Refer to Table 6.41.
5	const CHAR	*pcName	This parameter is for setting the start address of the area which holds the object name. Set the terminating character "00H" at the end of the area which holds the name. The length of the string must be within 500H (1280 characters).

<table>
<tr><th>No.</th><th colspan="2">Member</th><th>Description</th></tr>
<tr><td>6</td><td>ULONG</td><td>(*pfulRead) ()</td><td>This parameter is for creating and registering a desired function if there is any processing you want to run whenever gerR_IN_CanReadObject (6.4.16(4)) or gerR_IN_CanReadBlockObject (6.4.16(6)) has proceeded. In the case of gerR_IN_CanReadBlockObject, this function is executed the same number of times as the number of sub-indices in the target for reading. If the setting is null, only the default reading proceeds.

<Function format to be registered in *pfulRead>
<table>
<tr><td>Call Format</td><td colspan="4">ULONG gulReadFunc
(USHORT usIndex, UCHAR uchSubindex,
ULONG ulSize, UCHAR *puchData)</td></tr>
<tr><td rowspan="5">Arguments</td><td>Type Name</td><td>Variable Name</td><td>Description</td><td>I/O</td></tr>
<tr><td>USHORT</td><td>usIndex</td><td>Index</td><td>Input</td></tr>
<tr><td>UCHAR</td><td>uchSubindex</td><td>Sub-index</td><td>Input</td></tr>
<tr><td>ULONG</td><td>ulSize</td><td>Read size (byte)</td><td>Input</td></tr>
<tr><td>UCHAR</td><td>*puchData</td><td>Read data*4</td><td>Output</td></tr>
<tr><td>Return Value</td><td colspan="4">Abort code</td></tr>
</table>
Note 4. For data reading, set the value corresponding to the amount to be read. If a value is not set, the previous value will be sent.</td></tr>
<tr><td>7</td><td>ULONG</td><td>(*pfulWrite) ()</td><td>This parameter is for creating and registering a desired function if there is any processing you want to run whenever gerR_IN_CanWriteObject (6.4.16(5)) or gerR_IN_CanWriteBlockObject (6.4.16(7)) has proceeded. In the case of gerR_IN_CanWriteBlockObject, this function is executed the same number of times as the number of sub-indices in the target for writing. If the setting is null, only the default writing proceeds.

<Function format to be registered in *pfulWrite>
<table>
<tr><td>Call Format</td><td colspan="4">ULONG gulWriteFunc
(USHORT usIndex, UCHAR uchSubindex, ULONG ulSize, UCHAR *puchData)</td></tr>
<tr><td rowspan="5">Arguments</td><td>Type Name</td><td>Variable Name</td><td>Description</td><td>I/O</td></tr>
<tr><td>USHORT</td><td>usIndex</td><td>Index</td><td>Input</td></tr>
<tr><td>UCHAR</td><td>uchSubindex</td><td>Sub-index</td><td>Input</td></tr>
<tr><td>ULONG</td><td>ulSize</td><td>Write size (byte)</td><td>Input</td></tr>
<tr><td>UCHAR</td><td>*puchData</td><td>Write data</td><td>Input</td></tr>
<tr><td>Return Value</td><td colspan="4">Abort code</td></tr>
</table></td></tr>
</table>

Table 6.41 R_IN_CAN_OBJECT_DATA_T List

<table>
<tr><th>No.</th><th colspan="2">Member</th><th>Description</th></tr>
<tr><td>1</td><td>USHORT</td><td>usDataType</td><td>The data type of the area indicated by "*pvValue" is set.
<table>
<tr><th>Definition</th><th>Setting</th><th>Description</th></tr>
<tr><td>R_IN_CAN_INTEGER8</td><td>0002H</td><td>Signed 8-bit integer</td></tr>
<tr><td>R_IN_CAN_INTEGER16</td><td>0003H</td><td>Signed 16-bit integer</td></tr>
<tr><td>R_IN_CAN_INTEGER32</td><td>0004H</td><td>Signed 32-bit integer</td></tr>
<tr><td>R_IN_CAN_UNSIGNED8</td><td>0005H</td><td>Unsigned 8-bit integer</td></tr>
<tr><td>R_IN_CAN_UNSIGNED16</td><td>0006H</td><td>Unsigned 16-bit integer</td></tr>
<tr><td>R_IN_CAN_UNSIGNED32</td><td>0007H</td><td>Unsigned 32-bit integer</td></tr>
<tr><td>R_IN_CAN_VISIBLESTRING</td><td>0009H</td><td>String</td></tr>
</table></td></tr>
<tr><td>2</td><td>USHORT</td><td>usBitLength</td><td>The valid bit length of the area indicated by "*pvValue" is set.
<table>
<tr><th>Definition</th><th>Setting</th></tr>
<tr><td>R_IN_CAN_INTEGER8</td><td>8</td></tr>
<tr><td>R_IN_CAN_INTEGER16</td><td>16</td></tr>
<tr><td>R_IN_CAN_INTEGER32</td><td>32</td></tr>
<tr><td>R_IN_CAN_UNSIGNED8</td><td>8</td></tr>
<tr><td>R_IN_CAN_UNSIGNED16</td><td>16</td></tr>
<tr><td>R_IN_CAN_UNSIGNED32</td><td>32</td></tr>
<tr><td>R_IN_CAN_VISIBLESTRING</td><td>8 to 1280 (a multiple of 8)</td></tr>
</table></td></tr>
<tr><td>3</td><td>UCHAR</td><td>uchAccess</td><td>The access type of SDOs is set.
<table>
<tr><th>Definition</th><th>Setting</th><th>Description</th></tr>
<tr><td>R_IN_CAN_READWRITE</td><td>0077H</td><td>Always readable/writable</td></tr>
<tr><td>R_IN_CAN_READ</td><td>0007H</td><td>Always readable</td></tr>
<tr><td>R_IN_CAN_READ_PREOP</td><td>0001H</td><td>Only readable during PREOP</td></tr>
<tr><td>R_IN_CAN_READ_SAFEOP</td><td>0002H</td><td>Only readable during SAFEOP</td></tr>
<tr><td>R_IN_CAN_READ_OP</td><td>0004H</td><td>Only readable during OP</td></tr>
<tr><td>R_IN_CAN_WRITE</td><td>0070H</td><td>Always writable</td></tr>
<tr><td>R_IN_CAN_WRITE_PREOP</td><td>0010H</td><td>Only writable during PREOP</td></tr>
<tr><td>R_IN_CAN_WRITE_SAFEOP</td><td>0020H</td><td>Only writable during SAFEOP</td></tr>
<tr><td>R_IN_CAN_WRITE_OP</td><td>0040H</td><td>Only writable during OP</td></tr>
</table></td></tr>
<tr><td>4</td><td>UCHAR</td><td>uchMapping</td><td>This parameter is for setting whether PDOs can be mapped.
<table>
<tr><th>Definition</th><th>Setting</th><th>Description</th></tr>
<tr><td>R_IN_CAN_NOPDOMAPPING</td><td>0000H</td><td>PDOs cannot be mapped.</td></tr>
<tr><td>R_IN_CAN_RPDOMAPPING</td><td>0040H</td><td>RPDOs can be mapped. "R_IN_CAN_VISIBLESTRING" cannot be set for usDataType.</td></tr>
<tr><td>R_IN_CAN_TPDOMAPPING</td><td>0080H</td><td>TPDOs can be mapped. "R_IN_CAN_VISIBLESTRING" cannot be set for usDataType.</td></tr>
</table></td></tr>
</table>

<table>
<tr><th>No.</th><th colspan="2">Member</th><th>Description</th></tr>
<tr><td>5</td><td>UCHAR</td><td>uchValueInfo</td><td>This parameter is for setting whether “unit type (ulUnitType)”, “initial value (ulDefaultValue)”, “minimum value (ulMinValue)”, and “maximum value (ulMaxValue)” are effective.
<table>
<tr><th>Definition</th><th>Setting</th><th>Description</th></tr>
<tr><td>R_IN_CAN_VALUEINFO_UNITTYPE</td><td>08H</td><td>Unit type data become effective.</td></tr>
<tr><td>R_IN_CAN_VALUEINFO_VALUE</td><td>70H</td><td>The initial value, minimum value, and maximum value settings become effective.</td></tr>
<tr><td>R_IN_CAN_VALUEINFO_DEFAULTVALUE</td><td>10H</td><td>The initial value setting becomes effective.</td></tr>
<tr><td>R_IN_CAN_VALUEINFO_MINMAXVALUE</td><td>60H</td><td>The minimum and maximum value settings become effective.</td></tr>
<tr><td>R_IN_CAN_VALUEINFO_MINVALUE</td><td>20H</td><td>The minimum setting becomes effective.</td></tr>
<tr><td>R_IN_CAN_VALUEINFO_MAXVALUE</td><td>40H</td><td>The maximum setting becomes effective.</td></tr>
</table></td></tr>
<tr><td>6</td><td>ULONG</td><td>ulUnitType</td><td>The unit of entry is set as a 4-byte numerical value.
(Example: In the case of km/h, the setting is 0301 4800H (k: 03, m: 01, h: 48).)</td></tr>
<tr><td>7</td><td>ULONG</td><td>ulDefaultValue</td><td>Sets the initial value.*5</td></tr>
<tr><td>8</td><td>ULONG</td><td>ulMinValue</td><td>Sets the minimum value.*5</td></tr>
<tr><td>9</td><td>ULONG</td><td>ulMaxValue</td><td>Sets the maximum value.*5</td></tr>
<tr><td>10</td><td>VOID</td><td>*pvValue</td><td>Sets the start address of the variables used by the application.
If “R_IN_CAN_VISIBLESTRING” is set as the data type, the following condition must be satisfied.
• Set the terminating character “00H” at the end of the area which holds the string.</td></tr>
</table>

Note 5. If this setting is not required, it can be skipped by setting the corresponding bit of “uchValueInfo” to “OFF”. If “R_IN_CAN_VISIBLESTRING” is set as the data type, setting this member is not allowed.

(3) gerR_IN_CanGetObjectHandle

Function	Object dictionary handling acquisition			
Call Format	ERRCODE gerR_IN_CanGetObjectHandle (USHORT usObjectDictionaryNo, USHORT usIndex, R_IN_CAN_OD_T** pstObject, R_IN_CAN_ERROR_DETAIL_T* pstErrorDetail)			
Arguments	Type Name	Variable Name	Description	I/O
	USHORT	usObjectDictionaryNo	Object dictionary number	Input
	USHORT	usIndex	Index	Input
	R_IN_CAN_OD_T**	pstObject	Start address corresponding to the index of the object dictionary Refer to Table 6.40.	Output
	R_IN_CAN_ERROR_DETAIL_T*	pstErrorDetail	Error detail information Refer to Table 6.39.	Output
Return Value	R_IN_OK: Normal completion R_IN_ERR: Abnormal completion			
Description	This function acquires the start address corresponding to the applicable object of the object dictionary based on the specified index.			

(4) gerR_IN_CanReadObject

Function	Object dictionary reading			
Call Format	ERRCODE gerR_IN_CanReadObject (USHORT usObjectDictionaryNo, USHORT usIndex,UCHAR uchSubIndex, USHORT usRequestDataSize, USHORT* pusDataSize,UCHAR* puchReadData,R_IN_CAN_ERROR_DETAIL_T* pstErrorDetail)			
Arguments	Type Name	Variable Name	Description	I/O
	USHORT	usObjectDictionaryNo	Object Dictionary number	Input
	USHORT	usIndex	Index	Input
	UCHAR	uchSubIndex	Sub-index	Input
	USHORT	usRequestDataSize	Data read instruction size	Input
	USHORT*	pusDataSize	Size of data to be read	Output
	UCHAR*	puchReadData	Read data	Output
	R_IN_CAN_ERROR_DETAIL_T*	pstErrorDetail	Error detail information Refer to Table 6.39.	Output
Return Value	R_IN_OK: Normal completion R_IN_ERR: Abnormal completion			
Description	This function reads data stored in the corresponding object of the object dictionary based on the specified index and sub-index. When the data type of the target object for reading is "R_IN_CAN_VISIBLESTRING": • For the amount of data to be read, specify 0 or the byte size indicated by the valid bit length (usBitLength) of the corresponding object in the object dictionary. • Data are read from detection of the start character to the end of the data for reading, so pad the set data with the terminating character to indicate the end of the data to be read. When the data type of the target object for reading is "Other than R_IN_CAN_VISIBLESTRING": • For the amount of data to be read, specify 0 or the byte size indicated by the data type (usDataType) of the corresponding object in the object dictionary.			

(5) gerR_IN_CanWriteObject

Function	Object dictionary writing			
Call Format	ERRCODE gerR_IN_CanWriteObject (USHORT usObjectDictionaryNo,USHORT usIndex,UCHAR uchSubIndex,USHORT usDataSize, UCHAR* puchWriteData,R_IN_CAN_ERROR_DETAIL_T* pstErrorDetail)			
Arguments	Type Name	Variable Name	Description	I/O
	USHORT	usObjectDictionaryNo	Object Dictionary number	Input
	USHORT	usIndex	Index	Input
	UCHAR	uchSubIndex	Sub-index	Input
	USHORT	usDataSize	Size of the data to be written	Input
	UCHAR*	puchWriteData	Write data	Input
	R_IN_CAN_ERROR_DETAIL_T*	pstErrorDetail	Error detail information Refer to Table 6.39.	Output
Return Value	R_IN_OK: Normal completion R_IN_ERR: Abnormal completion			
Description	This function writes specified data to the corresponding object of the object dictionary based on the specified index and sub-index. When the data type of the target object for writing is "R_IN_CAN_VISIBLESTRING": • For the amount of SLMP data for writing, specify the byte size indicated by the valid bit length (usBitLength) of the corresponding object in the object dictionary. • Set the terminating character after the end of the string. When the data type of the target object for writing is "Other than R_IN_CAN_VISIBLESTRING": • For the amount of data for writing, specify the byte size indicated by the data type (usDataType) of the corresponding object in the object dictionary.			

(6) gerR_IN_CanReadBlockObject

Function	Object dictionary block reading			
Call Format	gerR_IN_CanReadBlockObject (USHORT usObjectDictionaryNo, USHORT usIndex, UCHAR uchSubIndex, USHORT usDataSize, UCHAR* puchReadData, R_IN_CAN_ERROR_DETAIL_T* pstErrorDetail);			
Arguments	Type Name	Variable Name	Description	I/O
	USHORT	usObjectDictionaryNo	Object Dictionary number	Input
	USHORT	usIndex	Index	Input
	UCHAR	uchSubIndex	Sub-index	Input
	USHORT	usDataSize	Size of data to be read	Input
	UCHAR*	puchReadData	Read data	Output
	R_IN_CAN_ERROR_DETAIL_T*	pstErrorDetail	Error detail information Refer to Table 6.39.	Output
Return Value	R_IN_OK: Normal completion R_IN_ERR: Abnormal completion			
Description	The specified amount of data is read from the corresponding object in the object dictionary based on the specified index and sub-index. As an SLMP frame, the readable size is 500H (1280 bytes). When the data type of the target object for reading is "R_IN_CAN_VISIBLESTRING": • The amount of SLMP data to be read should be calculated as the byte size indicated by the valid bit length (usBitLength) of the corresponding object in the object dictionary. • Data are read from detection of the start character to the end of the data for reading, so pad the set data with the terminating character to indicate the end of the data to be read. When the data type of the target object for reading is "Other than R_IN_CAN_VISIBLESTRING": • The amount of data to be read should be calculated as the byte size indicated by the data type (usDataType) of the corresponding object in the object dictionary.			

(7) gerR_IN_CanWriteBlockObject

Function	Object dictionary block writing			
Call Format	ERRCODE gerR_IN_CanWriteBlockObject (USHORT usObjectDictionaryNo,USHORT usIndex, UCHAR uchSubIndex,USHORT usDataSize,UCHAR* puchWriteData,R_IN_CAN_ERROR_DETAIL_T* pstErrorDetail)			
Arguments	Type Name	Variable Name	Description	I/O
	USHORT	usObjectDictionaryNo	Object Dictionary number	Input
	USHORT	usIndex	Index	Input
	UCHAR	uchSubIndex	Sub-index	Input
	USHORT	usDataSize	Size of the data to be written	Input
	UCHAR*	puchWriteData	Write data	Input
	R_IN_CAN_ERROR_DETAIL_T*	pstErrorDetail	Error detail information Refer to Table 6.39.	Output
Return Value	R_IN_OK: Normal completion R_IN_ERR: Abnormal completion			
Description	The specified amount of data is written to the corresponding object in the object dictionary based on the specified index and sub-index. As an SLMP frame, the writable size is 500H (1280 bytes). When the data type of the target object for writing is "R_IN_CAN_VISIBLESTRING": • The amount of SLMP data for writing should be calculated as the byte size indicated by the valid bit length (usBitLength) of the corresponding object in the object dictionary. • Set the terminating character after the end of the string. When the data type of the target object for writing is "Other than R_IN_CAN_VISIBLESTRING": • For the amount of data for writing, specify the byte size indicated by the data type (usDataType) of the corresponding object in the object dictionary.			

(8) gerR_IN_CanGetNmtState

Function	Acquiring the NMT state			
Call Format	ERRCODE gerR_IN_CanGetNmtState(USHORT usObjectDictionaryNo, USHORT* pusCurrentNmtState, USHORT* pusMasterRequestedNmtState, R_IN_CAN_ERROR_DETAIL_T* pstErrorDetail)			
Arguments	Type Name	Variable Name	Description	I/O
	USHORT	usObjectDictionaryNo	Object dictionary number	Input
	USHORT*	pusCurrentNmtState	Current NMT state	Output
	USHORT*	pusMasterRequestedNmtState	NMT state received from the master station	Output
	R_IN_CAN_ERROR_DETAIL_T*	pstErrorDetail	Error detail information Refer to Table 6.39.	Output
Return Value	R_IN_OK: Normal completion R_IN_ERR: Abnormal completion			
Description	This function acquires the current NMT state and the NMT state specified by the master station. If the master has not specified an NMT state, Init is acquired.			

(9) gerR_IN_CanSetNmtState

Function	Setting the NMT state			
Call Format	ERRCODE gerR_IN_CanSetNmtState(USHORT usObjectDictionaryNo, USHORT usMasterRequestNmtState, R_IN_CAN_ERROR_DETAIL_T* pstErrorDetail)			
Arguments	Type Name	Variable Name	Description	I/O
	USHORT	usObjectDictionaryNo	Object dictionary number	Input
	USHORT	usMasterRequestedNmtState	The NMT state received from the master station	Input
	R_IN_CAN_ERROR_DETAIL_T*	pstErrorDetail	Error detail information Refer to Table 6.39.	Output
Return Value	R_IN_OK: Normal completion R_IN_ERR: Abnormal completion			
Description	This function sets the specified NMT state in the R-IN32M4-CL3 driver.			

(10) gerR_IN_CanUpdateRPDO

Function	Updating RPDOs			
Call Format	ERRCODE gerR_IN_CanUpdateRPDO (R_IN_CAN_ERROR_DETAIL_T* pstErrorDetail)			
Arguments	Type Name	Variable Name	Description	I/O
	R_IN_CAN_ERROR_DETAIL_T*	pstErrorDetail	Error detail information Refer to Table 6.39.	Output
Return Value	R_IN_OK: Normal completion R_IN_ERR_NODATA: Abnormal completion (no received data) R_IN_ERR: Abnormal completion			
Description	This function reflects the received RWw data in the object dictionary in accord with the PDO mapping setting.			

(11) gerR_IN_CanUpdateTPDO

Function	Updating TPDOs			
Call Format	ERRCODE gerR_IN_CanUpdateTPDO (R_IN_CAN_ERROR_DETAIL_T* pstErrorDetail)			
Arguments	Type Name	Variable Name	Description	I/O
	R_IN_CAN_ERROR_DETAIL_T*	pstErrorDetail	Error detail information Refer to Table 6.39.	Output
Return Value	R_IN_OK: Normal completion R_IN_ERR: Abnormal completion R_IN_ERR_NODATA: Abnormal end (Send data setting is not required when the product that is ranked as CC-Link IE TSN Class A operates.)			
Description	This function reflects the data of the object dictionary in RWr in accord with the PDO mapping setting. When the product that is ranked as CC-Link IE TSN Class A operates, if cyclic data is not received, "R_IN_ERR_NODATA" is returned as the return value.			

6.4.17 MCU-MCU interface

(1) gvR_IN_SchedulerMain

Function	Scheduler task main			
Call Format	VOID gvR_IN_SchedulerMain(VOID)			
Arguments	Type Name	Variable Name	Description	I/O
	-	-	-	-
Return Value	-			
Description	This function performs the scheduler task providing timer processing of the R-IN32M4-CL3 driver.			

(2) gvR_IN_McuIfGpioSendMain

Function	GPIO communication send task main			
Call Format	VOID gvR_IN_McuIfGpioSendMain(VOID)			
Arguments	Type Name	Variable Name	Description	I/O
	-	-	-	-
Return Value	-			
Description	This function performs the event notification send main processing of the R-IN32M4-CL3 driver.			

(3) gvR_IN_McuIfGpioResponseMain

Function	GPIO communication response receive task main			
Call Format	VOID gvR_IN_McuIfGpioResponseMain(VOID)			
Arguments	Type Name	Variable Name	Description	I/O
	-	-	-	-
Return Value	-			
Description	This function performs the event notification response receive processing of the R-IN32M4-CL3 driver.			

(4) gvR_IN_McuIfGpioReceivedMain

Function	GPIO communication receive task main			
Call Format	VOID gvR_IN_McuIfGpioReceivedMain(VOID)			
Arguments	Type Name	Variable Name	Description	I/O
	-	-	-	-
Return Value	-			
Description	This function performs the event notification receive main processing of the R-IN32M4-CL3 driver.			

(5) gvR_IN_McuIfRtDmaCompleteMain

Function	RT DMA transfer completion task main			
Call Format	VOID gvR_IN_McuIfRtDmaCompleteMain(VOID)			
Arguments	Type Name	Variable Name	Description	I/O
	-	-	-	-
Return Value	-			
Description	This function performs the RT DMA transfer completion task main processing of the R-IN32M4-CL3 driver.			

(6) gvR_IN_McuIfInitial

Function	MCU-MCU interface initial			
Call Format	VOID gvR_IN_McuIfInitial(VOID)			
Arguments	Type Name	Variable Name	Description	I/O
	-	-	-	-
Return Value	-			
Description	This function performs the following processing of the R-IN32M4-CL3 driver. ・ Memory block management function initialization ・ MCU-MCU interface function initialization ・ Scheduler task provided timer function initial processing			

(7) gerR_IN_McuIfSetCommand

Function	MCU-MCU interface registration processing			
Call Format	ERRCODE gerR_IN_McuIfSetCommand (R_IN_MCU_IF_GPIO_MANAGEMENT_TBL_T* pstMcuIfMng , R_IN_MCU_IF_RTDMA_FUNCTION_TBL_T* pstRtDmaFunction)			
Arguments	Type Name	Variable Name	Description	I/O
	R_IN_MCU_IF_GPIO_MANAGEMENT_TBL_T*	pstMcuIfMng	MCU-MCU interface management table For details, refer to "Table 6.42".	Input
	R_IN_MCU_IF_RTDMA_FUNCTION_TBL_T*	pstRtDmaFunction	Execution function table at completion of serial communication For details, refer to "Table 6.45".	Input
Return Value	R_IN_OK: Normal completion R_IN_ERR: Abnormal completion			
Description	This function registers a function address specified in the argument to a lower layer.			

No.	Function to register interface functions of the R-IN32M4-CL3 driver	Argument to be set at execution
1	Callback function for MCU-MCU communications	MCU-MCU interface management pointer
2	RT DMA transfer completion callback function	Call function when the RT DMA receive is completed

Table 6.42 R_IN_MCU_IF_GPIO_MANAGEMENT_TBL_T List

No.	Member		Description
1	USHORT	usRequestNumber	Number of request command registrations
2	USHORT	usResponseNumber	Number of response command registrations
3	CONST R_IN_MCU_IF_GPIO_FUNCTION_REQUEST_TBL_T*	pstRequest	Execution function table when the GPIO communication request is received This defines the command function executed when a notification from the external MCU is detected. For details, refer to "Table 6.43".
4	CONST R_IN_MCU_IF_GPIO_FUNCTION_RESPONSE_TBL_T*	pstResponse	Execution function table when the GPIO communication response is received This defines the function that receives the execution result when a notification is set to the external MCU. For details, refer to "Table 6.44".
5	R_IN_MCU_IF_ACK_GPIO_SET_FUNCTION	fpvAckGpioSet	Function pointer of the GPIO communication acceptance processing
6	R_IN_MCU_IF_ACK_GPIO_CLEAR_FUNCTION	fpvAckGpioClear	Function pointer of the GPIO communication acceptance clear processing
7	R_IN_MCU_IF_REQUEST_GPIO_SET_FUNCTION	ferRequestGpioSet	Function pointer of the GPIO communication event type/code setting processing

Table 6.43 R_IN_MCU_IF_GPIO_FUNCTION_REQUEST_TBL_T List

No.	Member		Description
1	USHORT	usKind	Event type
2	USHORT	usCode	Event code
3	R_IN_MCU_IF_REQUEST_FUNCTION	fpvFunction	Command function pointer*1

Note 1. In the default setting, "UserMcuIfSafetyPduTransfer" and "UserMcuIfSafetyPduReceived" are defined. For details on the functions, refer to Section 5.10.5 and Section 5.10.6.

Table 6.44 R_IN_MCU_IF_GPIO_FUNCTION_RESPONSE_TBL_T List

No.	Member		Description
1	USHORT	usKind	Event type
2	USHORT	usCode	Event code
3	R_IN_MCU_IF_REQUEST_FUNCTION	fpvFunction	Command function pointer*1

Note 1. In the default setting, "UserMcuIfSafetyPduTransferResponse" and "UserMcuIfSafetyPduReceivedResponse" are defined. For details on the functions, refer to Section 5.10.7 and Section 5.10.8.

Table 6.45 R_IN_MCU_IF_RTDMA_FUNCTION_TBL_T List

No.	Member		Description
1	R_IN_MCU_IF_RTDMA_RESERVED_FUNCTION	fpvReceivedFunction	Command function pointer*1
2	R_IN_MCU_IF_RTDMA_SEND_FUNCTION	fpvSendFunction	Command function pointer*2

Note 1. In the default setting, "UserMcuIfSafetyPduReceivedComplete" is defined. For details on the function, refer to Section 5.10.9.

Note 2. Not defined in the default setting (NULL)

(8) gerR_IN_McuIfCheckTransferRtDma

Function	RT DMA transfer request check			
Call Format	ERRCODE gerR_IN_McuIfCheckTransferRtDma(VOID)			
Arguments	Type Name	Variable Name	Description	I/O
	-	-	-	-
Return Value	R_IN_OK: Normal completion R_IN_ERR: Abnormal completion R_IN_BUSY：During processing			
Description	This function check the RT DMA transfer request.			

(9) gerR_IN_McuIfStartSendCsih

Function	DMA transfer ready (for send)			
Call Format	ERRCODE gerR_IN_McuIfStartSendCsih(CONST VOID* pSrc ,ULONG ulLen)			
Arguments	Type Name	Variable Name	Description	I/O
	CONST VOID*	pSrc	Send source address	Input
	ULONG	ulLen	Send data length	Input
Return Value	R_IN_OK: Normal completion R_IN_ERR: Abnormal completion			
Description	This function sets RT DMA and CSIH to send data in the area specified in the argument in serial communication.			

(10) gerR_IN_McuIfStartReceiveCsih

Function	DMA transfer ready (for receive)			
Call Format	ERRCODE gerR_IN_McuIfStartReceiveCsih(VOID* pDst ,ULONG ulLen)			
Arguments	Type Name	Variable Name	Description	I/O
	VOID*	pDst	Receive data storage destination address	Input
	ULONG	ulLen	Receive data length	Input
Return Value	R_IN_OK: Normal completion R_IN_ERR: Abnormal completion			
Description	This function sets RT DMA and CSIH to store the received data to the area specified in the argument in serial communication.			

(11) gerR_IN_McuIfStopReceiveCsih

Function	CSIH receive stop			
Call Format	ERRCODE gerR_IN_McuIfStopReceiveCsih(VOID)			
Arguments	Type Name	Variable Name	Description	I/O
	-	-	-	-
Return Value	R_IN_OK: Normal completion R_IN_ERR: Abnormal completion			
Description	This function stops the CSIH receive operation.			

(12) gerR_IN_McuIfStopSendCsih

Function	CSIH send stop			
Call Format	ERRCODE gerR_IN_McuIfStopSendCsih(VOID)			
Arguments	Type Name	Variable Name	Description	I/O
	-	-	-	-
Return Value	R_IN_OK: Normal completion R_IN_ERR: Abnormal completion			
Description	This function stops the CSIH send operation.			

(13) gerR_IN_McuIfCheckGpioState

Function	GPIO communication status check			
Call Format	ERRCODE gerR_IN_McuIfCheckGpioState(VOID)			
Arguments	Type Name	Variable Name	Description	I/O
	-	-	-	-
Return Value	R_IN_OK: Normal completion (Reserved) R_IN_BUSY：During processing (Used)			
Description	This function returns the GPIO communication status to the call source.			

(14) gerR_IN_McuIfGetGpioState

Function	GPIO communication status acquisition			
Call Format	ERRCODE gerR_IN_McuIfGetGpioState(VOID)			
Arguments	Type Name	Variable Name	Description	I/O
	-	-	-	-
Return Value	R_IN_OK: Normal completion R_IN_BUSY：During processing (Used)			
Description	This function changes the GPIO communication status from "Reserved" to "Used".			

(15) gvR_IN_McuIfClearGpioState

Function	GPIO communication status clear			
Call Format	VOID gvR_IN_McuIfClearGpioState(VOID)			
Arguments	Type Name	Variable Name	Description	I/O
	-	-	-	-
Return Value	-			
Description	This function sets the GPIO communication status to "Reserved".			

(16) gerR_IN_McuIfGpioSend

Function	GPIO communication data create request			
Call Format	ERRCODE gerR_IN_McuIfGpioSend(UCHAR uchKind ,UCHAR uchEvtCode)			
Arguments	Type Name	Variable Name	Description	I/O
	-	-	-	-
Return Value	-			
Description	This function performs the mailbox send processing to the GPIO communication send task.			

6.5 R-IN32M4-CL3 Driver Callback Function List

The table below lists the R-IN32M4-CL3 driver callback functions.

Table 6.46 R-IN32M4-CL3 Driver Callback Function List

No.	Function Category	Function Name	Function Type	Overview
1	Fatal error management	gR_IN_CallbackFatalError	VOID	R-IN32M4-CL3 fatal error acquisition
2	SLMP reception	gerR_IN_CallbackReceivedSlmp	ERRCODE	SLMP frame acquisition
3		gvR_IN_CallbackNotifyReceivedSlmp	VOID	SLMP frame reception notification
4	Cyclic transmission data initialization	gulR_IN_CallbackInitSendCyclicData	ULONG	Initializing data for cyclic transmission
5	Data link acceleration	gulR_IN_CallbackTimeSyncComplete	ULONG	Data link acceleration (at the completion of time synchronization)
6		gulR_IN_CallbackCyclicStart	ULONG	Data link acceleration (at the start of cyclic transfer)
7		gulR_IN_CallbackDisconnectPartwayThrough	ULONG	Data link acceleration (in the case of disconnection partway through)
8	Network-synchronized communications	gulR_IN_CallbackCheckComCycle	ULONG	Communications cycle check
9		gulR_IN_CallbackSyncCom	ULONG	Synchronous cyclic communications processing

6.6 R-IN32M4-CL3 Driver Callback Function Details

The internal processing of R-IN32M4-CL3 driver callback functions needs to be customized as required. The following describes the details of the R-IN32M4-CL3 driver callback functions.

(1) gR_IN_CallbackFatalError

Function	R-IN32M4-CL3 fatal error acquisition			
Call Format	VOID gR_IN_CallbackFatalError (ULONG ulErrorCode, ULONG ulErrorInfo)			
Arguments	Type Name	Variable Name	Description	I/O
	ULONG	ulErrorCode	Fatal error code	Input
	ULONG	ulErrorInfo	Fatal error information (address of the function when an error occurred.)	Input
Return Value	—			
Description	A callback function to acquire the fatal error information from the R-IN32M4-CL3 driver.			

(2) gerR_IN_CallbackReceivedSlmp

Function	SLMP frame acquisition			
Call Format	ERRCODE gerR_IN_CallbackReceivedSlmp (const VOID* pvFrameAddress, USHORT usFrameSize, ULONG ulIpAddress, USHORT usPort)			
Arguments	Type Name	Variable Name	Description	I/O
	const VOID*	pvFrameAddress	Frame start address	Input
	USHORT	usFrameSize	Frame size	Input
	ULONG	ulIpAddress	Originating IP address	Input
	USHORT	usPort	Originating port number	Input
Return Value	R_IN_OK: Normal completion R_IN_ERR: Abnormal completion			
Description	This function determines whether to acquire SLMP frames based on the reception buffer, received size, originating IP address, and originating port number, and sets the result as the return value as the argument. When an SLMP frame is received, this function is called before processing of the command functions registered in the SLMP execution function table is handled in response. Customization is not required unless there is a specific reason (e.g., SLMP frames are not to be acquired).			

(3) gvR_IN_CallbackNotifyReceivedSlmp

Function	SLMP frame reception notification			
Call Format	VOID gvR_IN_CallbackNotifyReceivedSlmp (const VOID* pvFrameAddress, USHORT usFrameSize , ULONG ulIpAddress, USHORT usPort)			
Arguments	Type Name	Variable Name	Description	I/O
	const VOID*	pvFrameAddress	Frame start address	Input
	USHORT	usFrameSize	Frame size	Input
	ULONG	ulIpAddress	Send source IP address	Input
	USHORT	usPort	Send source port number	Input
Return Value	—			
Description	This function notifies the start address of the received SLMP frame and the received size. When an SLMP frame is received, this function is called if the corresponding command function in the SLMP execution function table is null. If the SLMP frame of a non-supported command is received, a response does not always need to be sent. When sending a response, use the R-IN32M4-CL3 driver interface function "gerR_IN_SetSlmpResponseFrameSendRequest" (6.4.10(9)) to send the response. Also, execute "gvR_IN_ReleaseSlmpReceiveFrame" (6.4.10(10)) after this processing to receive a next SLMP frame.			

(4) gulR_IN_CallbackInitSendCyclicData

Function	Initializing data for transmission at the start of cyclic transfer			
Call Format	ULONG gulR_IN_CallbackInitSendCyclicData (ULONG ulParam1, ULONG ulParam2)			
Arguments	Type Name	Variable Name	Description	I/O
	ULONG	ulParam1	Parameter 1 (fixed to 0)	Input
	ULONG	ulParam2	Parameter 2 (fixed to 0)	Input
Return Value	Fixed to 0			
Description	For cyclic data to be sent immediately after return from disconnection, data that were to have been sent immediately before the disconnection may also be sent. If you want to send only the desired data, set data for cyclic transmission as part of this callback processing. When setting data for cyclic transmission, use the same function as the R-IN32M4-CL3 driver interface function (batch, for individual sections of data, or high-speed) for use in "UserSendCyclic" (5.4.3, Cyclic Transmission Processing). This callback processing is executed while dispatching is disabled. Using the gerR_IN_RegistCallback function (6.4.1(7)) to register this callback processing leads to this processing being called back at the time of reconnection of a disconnected data link. (Callback processing type: R_IN_FUNCTIONTYPE_CYCLICDATA_INITIALIZE (1))			

(5) gulR_IN_CallbackTimeSyncComplete

Function	Data link acceleration (at the completion of time synchronization)			
Call Format	ULONG gulR_IN_CallbackTimeSyncComplete (ULONG ulParam1, ULONG ulParam2)			
Arguments	Type Name	Variable Name	Description	I/O
	ulParam1	ulParam1	Parameter 1 (fixed to 0)	Input
	ULONG	ulParam2	Parameter 2 (fixed to 0)	Input
Return Value	Fixed to 0			
Description	Setting the priority of the user task (TSKID_PERIODIC) higher than that of the low priority interrupt task (TSKID_NX_LOW_INT) may affect*1 the start of data link. Use this callback processing as a workaround*1 to this issue. With the default priority settings, the start of data link is not affected. Default priority of the user task: 3 Default priority of the low priority interrupt task: 2 *1: For details, refer to [Supplementary information for data link speed-up] in this section. This callback function can be executed in dispatch disabled state. By registering this callback processing using the gerR_IN_RegistCallback function (Callback processing registration) (Section 6.4.1(7)), this function is called back when the time synchronization is completed. (Callback processing type: R_IN_FUNCTIONTYPE_TIMESYNC_COMPLETE (0))			

(6) gulR_IN_CallbackCyclicStart

Function	Data link acceleration (at the start of cyclic transfer)			
Call Format	ULONG gulR_IN_CallbackCyclicStart (ULONG ulParam1, ULONG ulParam2)			
Arguments	Type Name	Variable Name	Description	I/O
	ULONG	ulParam1	Parameter 1 (fixed to 0)	Input
	ULONG	ulParam2	Parameter 2 (fixed to 0)	Input
Return Value	Fixed to 0			
Description	Setting the priority of the user task (TSKID_PERIODIC) higher than that of the low priority interrupt task (TSKID_NX_LOW_INT) may affect*1 the start of data link. Use this callback processing as a workaround*1 to this issue. With the default priority settings, the start of data link is not affected. Default priority of the user task: 3 Default priority of the low priority interrupt task: 2 *1: For details, refer to [Supplementary information for data link speed-up] in this section. This callback function can be executed in dispatch disabled state. By registering this callback processing using the gerR_IN_RegistCallback function (Callback processing registration) (Section 6.4.1(7)), this function is called back when cyclic transmission starts. (Callback processing type: R_IN_FUNCTIONTYPE_CYCLIC_START (2))			

(7) gulR_IN_CallbackDisconnectPartwayThrough

Function	Data link acceleration (in the case of disconnection partway through)			
Call Format	ULONG gulR_IN_CallbackDisconnectPartwayThrough (ULONG ulParam1, ULONG ulParam2)			
Arguments	Type Name	Variable Name	Description	I/O
	ULONG	ulParam1	Parameter 1 (fixed to 0)	Input
	ULONG	ulParam2	Parameter 2 (fixed to 0)	Input
Return Value	Fixed to 0			
Description	Setting the priority of the user task (TSKID_PERIODIC) higher than that of the low priority interrupt task (TSKID_NX_LOW_INT) may affect*1 the start of data link. Use this callback processing as a workaround*1 to this issue. With the default priority settings, the start of data link is not affected. Default priority of the user task: 3 Default priority of the low priority interrupt task: 2 *1: For details, refer to [Supplementary information for data link speed-up] in this section. This callback function can be executed in dispatch disabled state. By registering this callback processing using the gerR_IN_RegistCallback function (Callback processing registration) (Section 6.4.1(7)), this function is called back when the station is disconnected in the middle of the initial processing. (Callback processing type: R_IN_FUNCTIONTYPE_DISCONNECT_PARTWAY_THROUGH (3))			

[Supplementary information for data link speed-up]

The R-IN32M4-CL3 driver performs processing required to start data link, such as processing that sets data received from the master station and time synchronization processing, in the low priority interrupt task (TSKID_NX_LOW_INT). Therefore, setting the priority of the user task (TSKID_PERIODIC) higher than that of the low priority interrupt task may affect the start of data link of the own station.
The details of this issue and the workarounds to the issue are described below.

(a) Issue

- When disconnection does not occur partway through

When the processing of a user task with a higher priority starts, the low-priority interrupt task processing is placed in the waiting state and its processing only continues after the end of processing for the user task. This extends the actual processing time from the start to end of processing of the low-priority interrupt task.

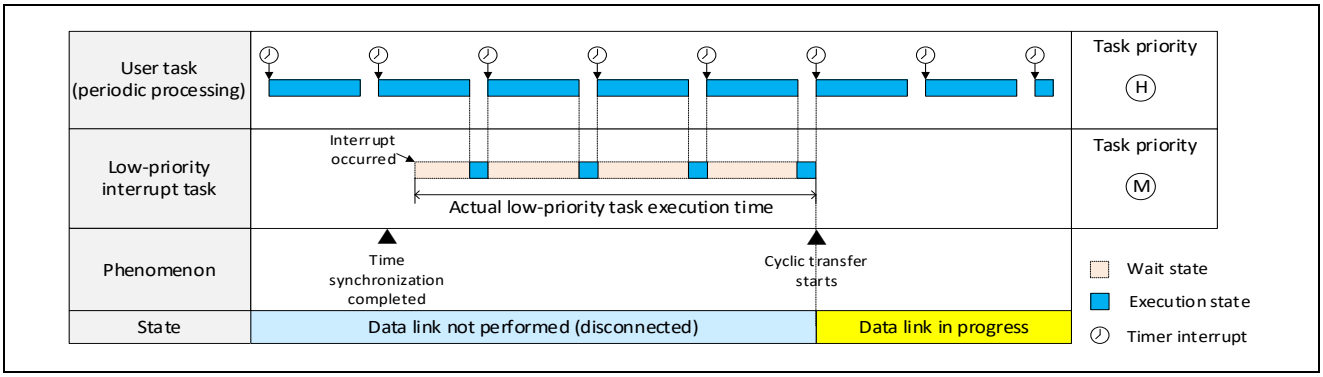


Figure 6.4 Problem that Arises when the Priority of the User Task is High (when Disconnection does not Occurs Partway Through)

- When disconnection occurs partway through

After having detected a slave station, the master station monitors processing through to the reception of cyclic data from this slave station. When the time this monitoring takes reaches a timeout, the master station judges that the slave station has been disconnected partway through and restarts the initialization sequence from the beginning. Disconnecting the station partway through initialization for cyclic transfer further extends the time until the start of cyclic transfer because time synchronization processing and processing to receive the various items of setting information required for the data link from the master station must be repeated.

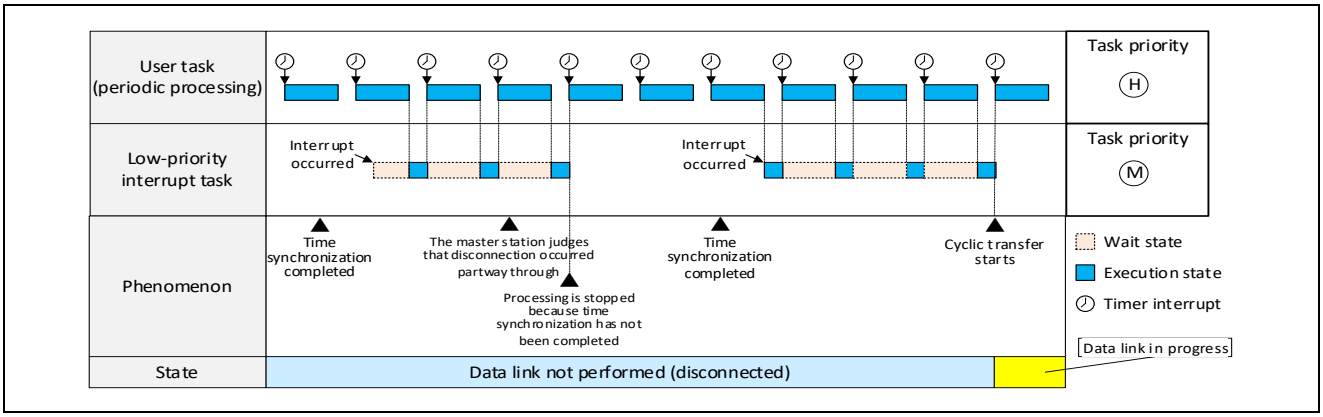


Figure 6.5 Problem that Arises when the Priority of the User Task is High (when Disconnection Occurs Partway Through)

(b) Workarounds

- When disconnection does not occur partway through

Temporarily lowering the priority of the user task leads to user task processing entering the waiting state during execution of the low-priority interrupt task, avoiding the generation of a delay until processing of the low-priority interrupt task.

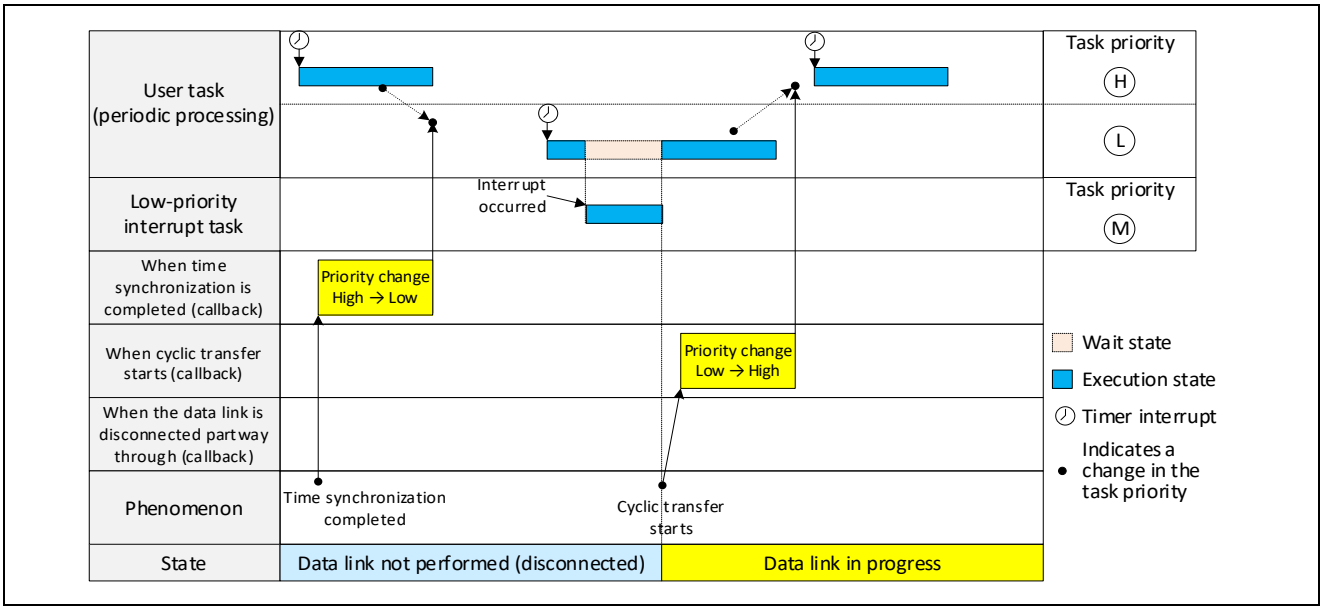


Figure 6.6 Effects of Temporarily Lowering the Priority of the User Task (when Disconnection does not Occur Partway Through)

[gulR_IN_CallbackTimeSyncComplete]
Using the RTOS service call (chg_pri), this function temporarily sets the priority of the priority change target lower than that of the low priority interrupt task.
[gulR_IN_CallbackCyclicStart]
Using the RTOS service call (chg_pri), this function sets the priority of the priority change target task back to the original.

- When disconnection occurs partway through

If disconnection occurs partway through the processing before the start of a data link because the priority of the user task has temporarily been lowered after the completion of time synchronization, the task will wait for completion of the next time synchronization at the lowered priority, so restore the task to its original priority with the data link disconnected partway through. After that, wait for the data link to start again following completion of the next time synchronization.

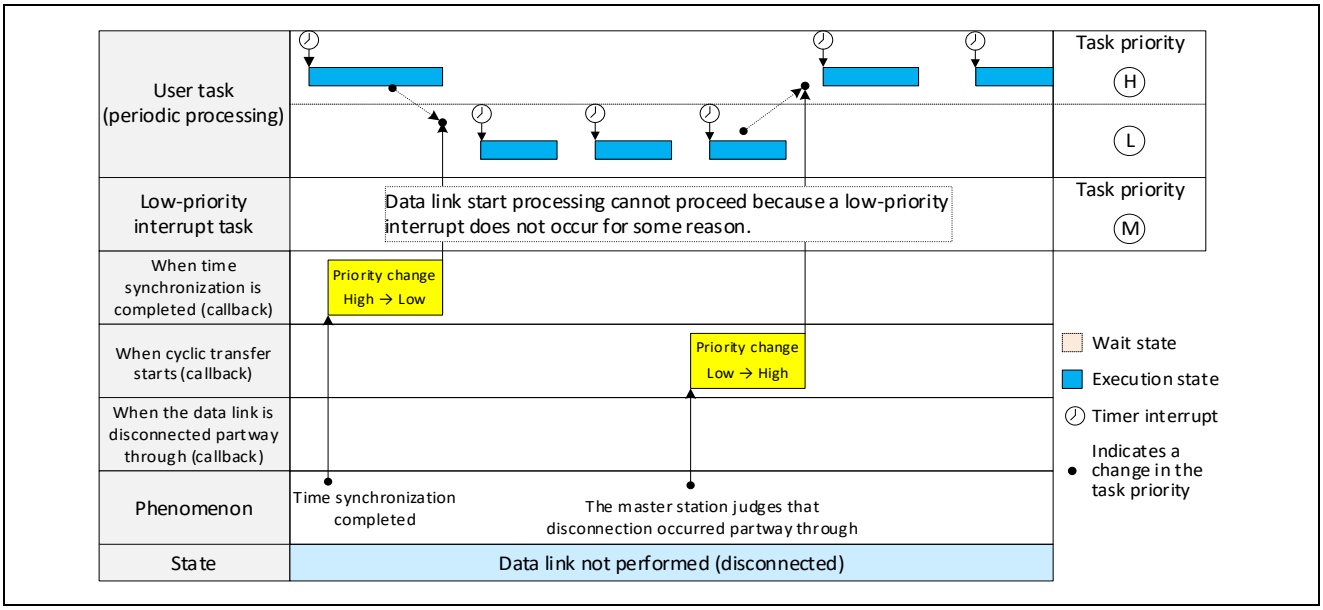


Figure 6.7 Effects of Temporarily Lowering the Priority of the User Task (when Disconnection Occurs Partway Through)

[gulR_IN_CallbackTimeSyncComplete]
Using the RTOS service call (chg_pri), this function temporarily sets the priority of the priority change target lower than that of the low priority interrupt task.
[gulR_IN_CallbackDisconnectPartwayThrough]
Using the RTOS service call (chg_pri), this function sets the priority of the priority change target task back to the original.

(8) gulR_IN_CallbackCheckComCycle

Function	Communications cycle check			
Call Format	ULONG gulR_IN_CallbackCheckComCycle (ULONG ulParam1, ULONG ulParam2)			
Arguments	Type Name	Variable Name	Description	I/O
	ULONG	ulParam1	Start address of communications cycle information For details, refer to "Table 6.47.	Input
	ULONG	ulParam2	Parameter 2 (fixed to 0)	Input
Return Value	Result of judgment 0: Normal 1: Abnormal			
Description	In the initialization phase of the CC-Link IE TSN protocol, the slave receives information on the communications cycle from the master station and processing proceeds at the time specified for network-synchronized communications. This function judges whether the device side is capable of the given communications cycle and returns the result of judgment. If the communications cycle information raises a problem for the user application, the return value of the function is non-zero. This callback processing is executed while dispatching is disabled. Using the gerR_IN_RegistCallback function (6.4.1(7)) to register this callback processing leads to this processing being called back. This callback processing does not proceed if network communications are not to proceed. (Callback processing type: R_IN_FUNCTIONTYPE_COMCYLCE_DEFINITION (4))			

Table 6.47 R_IN_COMCYCLE_INFO_T List

No.	Member		Description
1	UCHAR	uchTimeslotNum	Number of time slots
2	UCHAR	uchReserve	Reserved
3	USHORT	usRepetitionCount	Communications cycle repetition count
4	ULONGLONG	aullTsStartTime [R_IN_TIMESLOT_MAX]	Start time of TS0 to TS7 (ns)
5	ULONGLONG	aullTsEndTime [R_IN_TIMESLOT_MAX]	End time of TS0 to TS7 (ns)
6	ULONGLONG	ullComCycle	Communications cycle (ns)

(9) gulR_IN_CallbackSyncCom

Function	Synchronous cyclic communications processing			
Call Format	ULONG gulR_IN_CallbackSyncCom (ULONG ulParam1, ULONG ulParam2)			
Arguments	Type Name	Variable Name	Description	I/O
	ULONG	ulParam1	Parameter 1 (fixed to 0)	Input
	ULONG	ulParam2	Parameter 2 (fixed to 0)	Input
Return Value	Callback execution result (fixed to 0)			
Description	When running network-synchronized communications, this processing proceeds in response to the completion of cyclic transfer (the time at which timeslot 2 starts). This function handles processing for received cyclic data and preparations to receive the cyclic data sent in the next round of transmission. This callback processing is executed while dispatching is disabled. If processing requires time, consider the measures such as execution by another task, Using the gerR_IN_RegistCallback function (6.4.1(7)) to register this callback processing leads to this processing being called back. This processing is not called back if network-synchronized communications are not to proceed. (Callback processing type: R_IN_FUNCTIONTYPE_SYNC_CYCLIC_COM (5))			

The table below gives an overview of synchronous cyclic communications processing.

Table 6.48 Synchronous Cyclic Communications Processing Overview

No.	Item	Description
i	Latching the normal value of the transmission WDC	"gvR_IN_LatchWdcUL" (6.4.15(6)) is executed to latch the normal value of the WDC for transmission which proceeds in synchronous timing processing.
ii	Cyclic data receive	"UserReceiveCyclic" (5.4.1) is used to acquire cyclic received data.
iii	Checking the received WDC	This processing acquires the WDC value from the received cyclic data and judges whether the received cyclic data are normal by calling "gerR_IN_CheckWdcDL" (6.4.15(8)). If the result of checking is not normal, the data are discarded.
iv	Disconnection and synchronization error checking processing	This processing checks whether time synchronization has deviated by using the function for detecting deviations in time synchronization (6.4.15(1)). It also checks whether the home station is disconnected by using the function for acquiring the state of the data link (6.4.6(4)). In case of deviation from time synchronization or disconnection of the home station, this function executes application-dependent processing such as stopping synchronous timing processing by executing the function for stopping output of the synchronization signal (6.4.15(2)). Furthermore, to judge whether the application information in the StsW (the home station state register) indicates "asynchronous", when a deviation in time synchronization is detected (indicated in R_IN_TRUE) even once following the system startup, the function holds an indication of this in the global variable "time synchronization deviation detected flag". *1
v	Creating the WDC UL	"gerR_IN_GetWdcUL" (6.4.15(7)) is executed to acquire the value of the watchdog counter UL to be sent to the master station as cyclic data.
vi	Setting cyclic data for transmission	This processing sets cyclic data for transmission and embeds the created watchdog counter UL in the cyclic data.
vii	Home station state transmission processing	The home station state is sent as cyclic data.
viii	Cyclic transmission processing	"UserSendCyclic" (5.4.3) is used to send the data created for cyclic transmission.
ix	Own station status and cyclic transmission status update	This processing updates the communications state of the home station and the state of cyclic transfer.

Note 1. The application information in the StsW (the home station state register) created within the R-IN32M4-CL3 driver indicates "asynchronous" if a deviation in time synchronization is detected even once following the system startup. On the other hand, if the disconnected data link is reconnected after the detection of a deviation in time synchronization, this is acquired as "Time synchronization deviation not detected" (R_IN_FALSE) by the function for detecting deviations in time synchronization (6.4.15(1)).

The following describes operations assumed in the sample code and its timing chart for synchronous cyclic communications processing (in this section) and synchronous timing processing (in section 5.2.5).

(a) Operation Timing Chart 1 for Synchronous Cyclic Communications Processing and Synchronous Timing Processing (a case where synchronous timing processing is completed within the period of time slot 1)

1) Operation for synchronous cyclic communications processing

A series of processing for cyclic transmission/reception proceeds (refer to Table 6.48, Synchronous Cyclic Communications Processing OverviewTable 6.48 Synchronous Cyclic Communications Processing Overview). At this time, the value of the transmission WDC being incremented in synchronous timing processing is latched, and WDC_UL is created from the latched value and embedded in the cyclic data.

2) Operation for synchronous timing processing

User application processing to align the timing with other slave stations proceeds. Internal processing based on cyclic received data and data for cyclic transmission are created. Also, the normal value of the transmission WDC is incremented in every synchronization cycle.

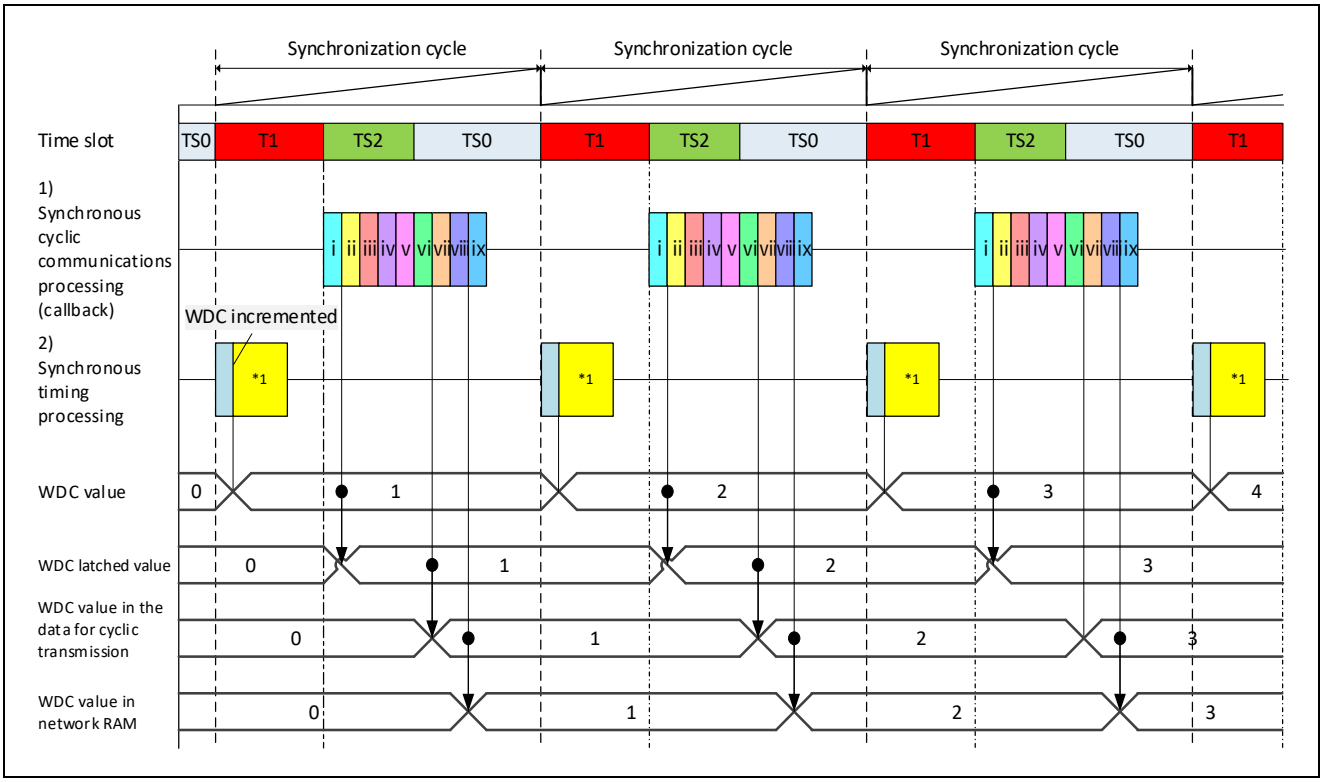


Figure 6.8 Operation Timing Chart 1 for Synchronous Cyclic Communications Processing and Synchronous Timing Processing

Note 1. User application processing to align the timing with other slave stations

[Implementation of error processing]

The time obtained by subtracting the time of time slot 1 from the communications cycle notified by "gulR_IN_CallbackCheckComCycle" (6.6(8)) is the time over which synchronous cyclic communications processing is executable. Check this time and if a time that does not allow execution is specified, handle the result of execution of "Communications cyclic check" as an error.

(b) Operation Timing Chart 2 for Synchronous Cyclic Communications Processing and Synchronous Timing Processing (a case where synchronous timing processing is not completed within the period of time slot 1)

Depending on the priority order of task startup, synchronous cyclic communications processing which should be called back at the start of time slot 2 does not proceed, and synchronous cyclic communications processing only starts after a wait for the completion of processing for timing synchronization.

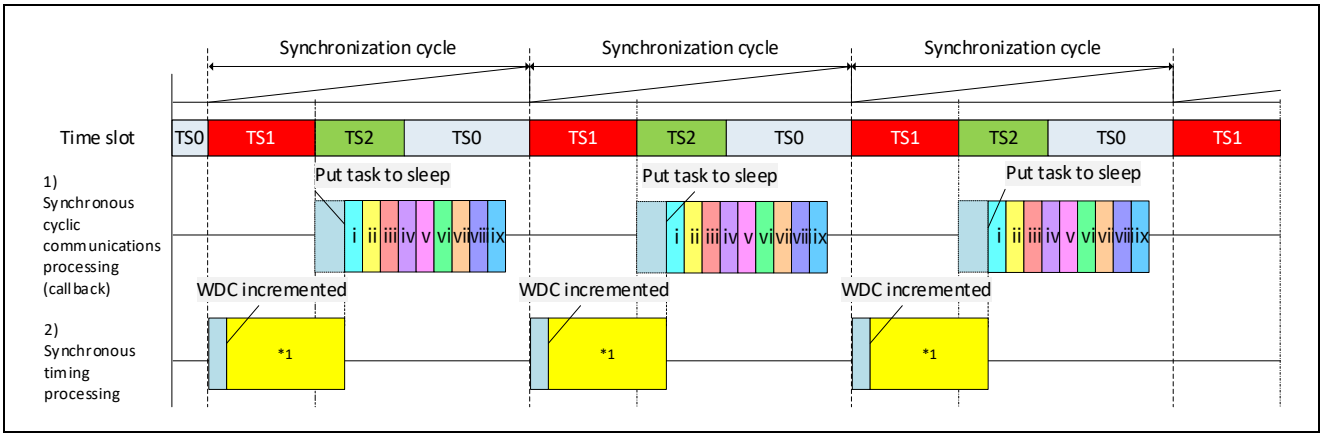


Figure 6.9 Operation Timing Chart 2 for Synchronous Cyclic Communications Processing and Synchronous Timing Processing

Note 1. User application processing to align the timing with other slave stations

The details of synchronous cyclic communications processing and synchronous timing processing shown in the above figure are the same as those shown in "(a) Operation Timing Chart 1 for Synchronous Cyclic Communications Processing and Synchronous Timing Processing".

[Implementation of error processing]
Synchronous timing processing not being completed within the period of time slot 1 raises a problem in the time of time slot 1.
Check whether the time of time slot 1 notified by "gulR_IN_CallbackCheckComCycle" (6.6(8)) is at least the execution time of synchronous timing processing. If a time that does not allow execution is specified, handle the result of execution of "Communications cycle check" as an error. This can prevent setting up the data link.

(c) Operation Timing Chart 3 for Synchronous Cyclic Communications Processing and Synchronous Timing Processing (a case where synchronous timing processing and synchronous cyclic communications processing is not completed within the synchronization cycle)

If the receive watchdog counter continuously repeats an error and the error count exceeds the threshold value, "gerR_IN_GetSyncDeviationFlag " (6.4.15(1)) in the synchronization cyclic communication processing detects synchronization deviation.

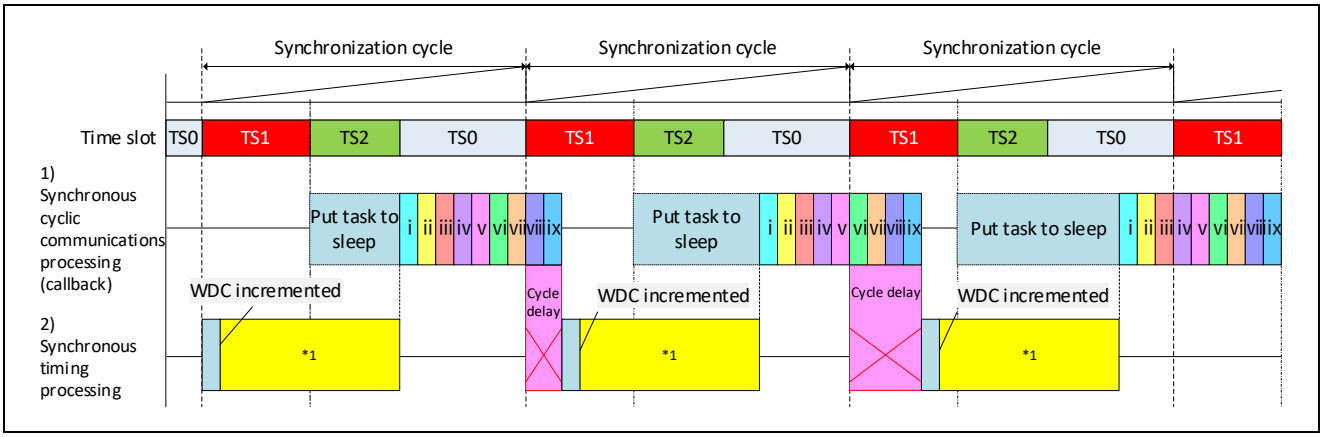


Figure 6.10 Operation Timing Chart 3 for Synchronous Cyclic Communications Processing and Synchronous Timing Processing

Note 1. User application processing to align the timing with other slave stations

The details of synchronous cyclic communications processing (callback) and synchronous timing processing shown in the above figure are the same as those shown in "(a) Operation Timing Chart 1 for Synchronous Cyclic Communications Processing and Synchronous Timing Processing".

[Implementation of error processing]
The total time of synchronous cyclic communications processing and synchronous timing processing not being completed within the synchronization cycle raises a problem in the communications cycle specified by the master station.
Check whether the communications cycle notified by "gulR_IN_CallbackCheckComCycle" (6.6(8)) is a communications cycle that allows execution. If a time that does not allow execution is specified, handle the result of execution of "Communications cycle check" as an error. This can prevent setting up the data link.

REVISION HISTORY	R-IN32M4-CL3 User's Manual: CC-Link IE TSN edition

Rev.	Date	Description	
		Page	Summary
1.00	Nov 21, 2019	—	First edition issued
1.01	Apr 30, 2020	6	2.3 R-IN32M4-CL3 Application Product Communications Specifications Table 2.2, modified
		18	4. Functions of the R-IN32M4-CL3 Application Product Table 4.2, added
		21 to 23	4.2.2 SLMP The description, added and updated
		27 to 29	5.1 List of User Programs Items, added. Overview, updated
		30, 31	5.2 User Program Tasks The description, added
		38	5.3.1 Initialization Processing Figure 5.7, updated
		40, 41	5.3.3 SLMP Receive Initialization Processing 5.3.4 SLMP Request Frame Creation Initialization Processing 5.3.5 Parameter Operation Initialization Processing Newly added
		42 to 48	5.4 User Program Details (Cyclic Transmission Related) The description, added and updated
		54	5.5.5 SLMP Receive Processing The description, updated
		56	5.5.8 SLMP ST (3E) Response Frame Receive Processing Newly added
		57, 58	5.6.1 SLMP Memory Read Request Command Receive Processing 5.6.2 SLMP Memory Write Request Command Receive Processing The figures and descriptions, updated
		61 to 71	5.6.4 SLMP Memory Read Response Command Receive Processing to 5.6.14 SLMP Network Time Offset Distribution Command Receive Processing The figures and descriptions, updated
		72	5.7.1 Hardware Test Processing (IEEE802.3ab Compliance Test) Figure 5.43, updated
		81 to 83	6.2 R-IN32M4-CL3 Driver Interface Function List The tables, updated
		84, 85	6.3 R-IN32M4-CL3 Driver Tasks The tables, newly added
		87 to 89	6.4.1 Initial Setup Tables 6.11 to 6.13, updated
		100 to 109	6.4.4 Cyclic Transmission The description, added and updated
		129 to 134	6.4.10 SLMP Send and Receive The description, updated

Rev.	Date	Description	
		Page	Summary
1.01	Apr 30, 2020	141 to 142	6.4.14 General Purpose Common The description, updated
		143	6.5 R-IN32M4-CL3 Driver Callback Function List Table 6.35, updated
		145	6.6 R-IN32M4-CL3 Driver Callback Function Details The description, added
1.02	Oct 31, 2020	16 to 18	3.4.2 Development Environment to 3.4.4 Compilation Switches Newly added
		28 to 31	4.2.2 About SLMP Newly added
		35 to 37	4.5 Network-Synchronized Communications Newly added
		37, 38	4.6 CANopen Communications Newly added
		39 to 42	5.1 List of User Programs The tables, added and updated
		74 to 75	5.5.9 Processing for Checking a Fatal Error Detected in the R-IN32M4-CL3 Driver Newly added
		87	5.6.11 SLMP Parameter Distribution Necessity Check Processing Newly added
		91	5.6.15 SLMP Parameter Data Write Request Processing Newly added
		93	5.6.17 SLMP Watchdog Counter Information Setting Request Command Reception Processing Newly added
		94 to 109	5.7 User Program Details (CANopen Communications Related) Newly added
		120 to 122	6.2 R-IN32M4-CL3 Driver Interface Function List The tables, updated
		158	6.4.6 Home Station State Acquisition (8) and (9), newly added
		188 to 191	6.4.15 Network-Synchronized Communications Newly added
		192 to 202	6.4.16 CANopen Communications Newly added
		206 to 212	6.6 R-IN32M4-CL3 Driver Callback Function Details (5) to (9), newly added
1.05	Dec 31, 2021	10	Added Sample CSP+ file
		19	Added Compilation Switches
		25	4. Functions of the R-IN32M4-CL3 Application Product Table 4.1 updated
		29-30	Added SLMP commands sent and received by user programs
		39-40	4.6.1 Expansion unit for CANopen communication Newly added

Rev.	Date	Description	
		Page	Summary
1.05	Dec 31, 2021	40-41	4.7 Communication speed setting via SLMP Newly added
		43	SLMP command execution related user program function added
		107-108	5.7 User Program Details (CANopen Communications Related) (4) newly added
		135-138	Added R-IN32M4-CL3 driver interface function
1.06	Jul 29, 2022	4	1.4 CC-Link Partner Association (CLPA) (1), (2) updated
		7	2.3 R-IN32M4-CL3 Application Product Communications Specifications Table 2.2 updated
		8	2.3.1 Precautions when the product ranked as CC-Link IE TSN Class A operates Newly added
		11	2.6 Sample CSP+ file overview Table 2.5 updated
		18	3.4.2 Development Environment Table 3.3 updated
		19	3.4.3.2 Flash ROM Table Settings Table 3.4 updated
		20	3.4.4 Compilation Switches Table 3.5 updated
		26	4. Functions of the R-IN32M4-CL3 Application Product Table 4.1 updated
		35-37	4.5 Network-Synchronized Communications The description, added
		40-41	4.7 Safety PDU Send/Receive In Safety Communications Newly added
		42	4.8 Communication Speed and CC-Link IE TSN Class Setting via SLMP Newly added
		44-45	5.1 List of User Programs Table 5.4 – 5.7 updated
		48-65	5.2 User Program Tasks updated The description, added and updated
		66-77	5.4 User Program Details (Cyclic Transfer Related) The description, added and updated
		102	5.6.10 SLMP network time distribution command receive processing Newly added
		118-119	5.8.5 SLMP function setting read (CC-Link IE TSN Class) request command receive Processing 5.8.6 SLMP function setting write (CC-Link IE TSN Class) request command receive processing Newly added
		140-151	5.10 Details on Processing of User Programs (MCU-MCU Interface Related) Newly added
		162-165	6.2 R-IN32M4-CL3 Driver Interface Function List Table 6.5 updated

Rev.	Date	Description	
		Page	Summary
1.06	Jul 29, 2022	169-177	6.4.1 Initial Setup The description added, modified
		188-200	6.4.4 Cyclic Transfer The description added, modified
		254-259	6.4.17 MCU-MCU interface Newly added

R-IN32M4-CL3 User's Manual: CC-Link IE TSN edition

Publication Date: Rev.1.06 Jul 29, 2022

Published by: Renesas Electronics Corporation

R-IN32M4-CL3 User's Manual: CC-Link IE TSN edition

Renesas Electronics Corporation