RZ/T2M Group

EnDat 3 Sample Program

Summary

This application note explains a sample program for acquiring and indicating information from an EnDat 3 compliant encoder by using the Encoder I/F Configuration Library of the RZ/T2M.

The major features of the program are listed below.

- Supports the foreground communication and background communication of the EnDat 3.
- Obtain angle information, etc. from an encoder (ECI 1319 E30-R2 from HEIDENHAIN) compliant with EnDat 3 specification or bus connection supported encoders (EQN 1337 E30-RB from HEIDENHAIN) of the EnDat 3 specification.
- Supports memory access of the encoder compliant with EnDat 3 specification.

Target Device

RZ/T2M

When applying the sample program covered in this application note to another microcomputer, modify the program according to the specifications for the target microcomputer and conduct an extensive evaluation of the modified program.

Table of Contents

1. Specifications

Table 1.1 lists the peripheral modules to be used and their applications. Figure 1.1 shows the operating environment when the sample code is being executed, and Figure 1.2 shows example of bus connection.

Table 1.1 Peripheral Modules and Applications

Peripheral Module	Application
EnDat 3 I/F	Communication with the EnDat 3 compliant encoder
Interrupt Controller (ICU)	EnDat 3 I/F interrupt control
General PWM Timer (GPT) channel 0	Generation of event cycles to be input to ELC
Event Link Controller (ELC)	Link events output by GPT channel 0 to EnDat 3 I/F
Serial Communication Interface (SCI) UART	Asynchronous communications of the SCI are used for COM port communications by using USB interface. It is used for console interface of the sample program.

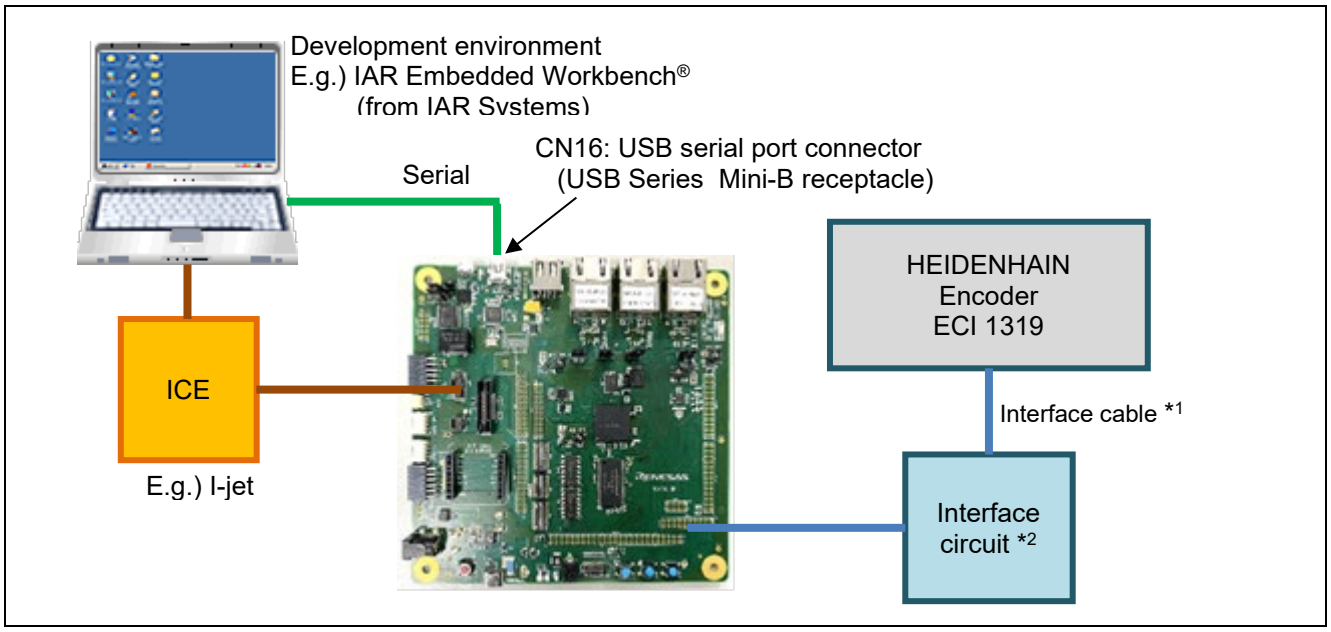

Figure 1.1 Operating Environment

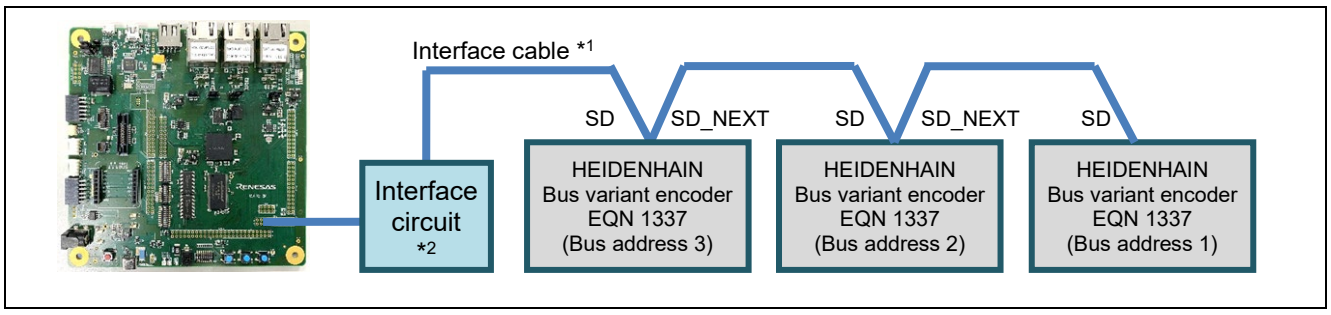

Figure 1.2 Example of Bus Connection

Note: 1. For allowable cable lengths, please refer to the encoder manuals.
2. For the interface circuit, please refer to the "EnDat 3 Hardware Specification", which can be obtained by contacting HEIDENHAIN.

2. Operating Environment

The sample code covered in this application note is for the environment below.

Table 2.1 Operating Environment

Item	Description
MCU	RZ/T2M Group
Operating frequency	CPUCLK = 800MHz
Operating voltage	1.1 V (Core) / 1.8 V (PLL, etc.) / 3.3 V (I/O)
Integrated development environment *	IAR Systems: IAR Embedded Workbench® for Arm®
	RENESAS: e² studio
Board	RSK+RZT2M（RTK9RZT2M0C00000BE）
Devices（function to be used on the board）	None

Note Refer to the RZ/T2M Group Encoder I/F EnDat 3 sample program Release Note to check the version number of the integrated development environment.

3. Peripheral Modules

The basics of the peripheral modules, operating modes, and registers are described in the “RZ/T2M Group User’s Manual: Hardware”.

3.1 Pins

Table 3.1 lists the pins used and their functions.

Table 3.1 Pins Used and Their Functions

Channel	**Port Name** (Function Pin Name)	**I/O Port**	**Input/ Output**	**Description**
ENDAT3_CH0	soen0 (ENCIF4)	P02_3	Output	Data output enable pin
	sdout0 (ENCIF3)	P02_2	Output	Data output pin
	sdin00 (ENCIF0)	P01_6	Input	Data input pin 0 *
	sdin10 (ENCIF1)	P01_7	Input	Data input pin 1 *
ENDAT3_CH1	soen1 (ENCIF9)	P03_3	Output	Data output enable pin
	sdout1 (ENCIF8)	P03_0	Output	Data output pin
	sdin01 (ENCIF5)	P17_3	Input	Data input pin 0 *
	sdin11 (ENCIF6)	P17_4	Input	Data input pin 1 *

Note. Please input the same receiving signal to the data input pin 0 and the data input pin 1.

4. Software

4.1 EnDat 3 Driver Function

The functions of the EnDat 3 driver are listed below.

1) Initial settings
2) Acquiring positional data
3) Accessing to memory area of the encoder
4) Receiving additional information *

Note: In this document, “additional information” represents the low priority frame (LPF) contents designated by the various frame identifiers (FIDs). For details, see the “EnDat 3 Interface Specification” which is available from HEIDENHAIN GmbH on request.

4.2 File Structures

For the file structure, refer to the release note for the RZ/T2M Group Encoder I/F EnDat 3 sample program.

4.3 Functions

Table 4.1 lists the functions to be used.

Table 4.1 List of Functions

Category	Function Name	Page Number
EnDat 3 driver API functions	R_ENDAT3_Open	8
	R_ENDAT3_Close	8
	R_ENDAT3_GetVersion	9
	R_ENDAT3_Control	9
User-defined functions	endat3_init_reset_wait_callback	13
	endat3_callback	13
	endat3_pos_callback	14
	endat3_ready_callback	14
Interrupt-handlers	enc_ch0_fg_isr	15
	enc_ch1_fg_isr	15
	enc_ch0_bg_isr	15
	enc_ch1_bg_isr	15
	enc_ch0_fifo_isr	15
	enc_ch1_fifo_isr	16

4.4 Specifications of API Functions

4.4.1 R_ENDAT3_Open

R_ENDAT3_Open	
Synopsis	Starting control of the EnDat 3 encoder
Header	r_endat3_rzt2_if.h
Declaration	r_endat3_err_t R_ENDAT3_Open(const int32_t id, r_endat3_info_t* p_info);
Description	EnDat 3 Driver initializes following settings. 1. Initialization of the encoder 2. Sending HELLO request Execute this function 1.3 seconds after the encoder is turned on. After sending this HELLO request, insert wait time more than 300 milliseconds. Response data of the HELLO request is not received.
Arguments	id : Specifies the ID to be used. (It is defined in r_endat3_rzt2_dat.h.) R_ENDAT0_ID : Specifies channel 0. R_ENDAT1_ID : Specifies channel 1. Others : Setting is not allowed. p_info : Specifies encoder information. Specify the address of the structure r_endat3_info_t that contains the encoder information.
Return value	ENDAT3_SUCCESS : Normal termination ENDAT3_ERR_INVALID_ARG : Abnormal termination (The id or p_info member of the r_endat3_info_t structure is not specified.) ENDAT3_ERR_ACCESS : Abnormal termination (The driver is already opened.)
Note	Before executing this function, be sure to configure and start Multi-Protocol Encoder IF using EC-Lib.

4.4.2 R_ENDAT3_Close

R_ENDAT3_Close	
Synopsis	Ending control of the EnDat 3 encoder
Header	r_endat3_rz2_if.h
Declaration	r_endat3_err_t R_ENDAT3_Close(const int32_t id);
Description	This function handles ending of the encoder interface driver.
Arguments	id : Specifies the ID to be used. (It is defined in r_endat3_rzt2_dat.h.) R_ENDAT0_ID : Specifies channel 0. R_ENDAT1_ID : Specifies channel 1. Others : Setting is not allowed.
Return value	ENDAT3_SUCCESS : Normal termination ENDAT3_ERR_INVALID_ARG : Abnormal termination (The id is not specified.) ENDAT3_ERR_ACCESS : Abnormal termination (A request is being transmitted.)

4.4.3 R_ENDAT3_GetVersion

R_ENDAT3_GetVersion	
Synopsis	Acquiring the version number of the encoder interface driver
Header	r_endat3_rz2_if.h
Declaration	uint32_t R_ENDAT3_GetVersion(void);
Description	This function acquires the version number of the EnDat 3 driver.
Arguments	None
Return value	Version information : The major and the minor parts of the version number are stored in the sixteen MSBs and the sixteen LSBs, respectively. Ex.) For ver.1.2, the value returned is 0x00010002.

4.4.4 R_ENDAT3_Control

R_ENDAT3_Control	
Synopsis	Controlling the EnDat 3 encoder
Header	r_endat3_rz2_if.h
Declaration	r_endat3_err_t R_ENDAT3_Control(const int32_t id, const r_endat3_cmd_t cmd, r_endat3_req_t *p_req);
Description	This function controls the EnDat 3 encoder by using argument cmd. See "4.4.5 Control Commands" for the operation of the control commands.
Arguments	id : Specifies the ID to be used. (It is defined in r_endat3_rzt2_dat.h.) R_ENDAT0_ID : Specifies channel 0. R_ENDAT1_ID : Specifies channel 1. Others : Setting is not allowed. cmd : Command For details, see section “4.10.3(2), r_endat3_cmd_t”. p_req : Arguments for each cmd
Return value	ENDAT3_SUCCESS : Normal termination ENDAT3_ERR_INVALID_ARG : Abnormal termination (id or cmd is not specified.) For other return values, see "4.4.5 Control Commands".
Note	Be sure to execute R_ENDAT3_Open before executing this function.

4.4.5 Control Commands

(1) ENDAT3_CMD_REQ

ENDAT3_CMD_REQ		
Synopsis	Sends request to the EnDat 3 encoder	
Header	r_endat3_rz2_if.h	
Declaration	r_endat3_err_t R_ENDAT3_Control(const int32_t id, const r_endat3_cmd_t cmd, r_endat3_req_t *p_req);	
Description	Sends foreground request to the EnDat 3 encoder. The endat3_callback and endat3_ready_callback functions are called once in response to each request. If the ELC mode setting is enabled, the requests are repeatedly sent in synchronize with ELC events until ENDAT3_CMD_STOP is executed. In response to each request, the endat3_pos_callback and the endat3_ready_callback functions are called.	
Arguments	id	: Specifies the ID to be used. (It is defined in r_endat3_rzt2_dat.h.)
		R_ENDAT0_ID : Specifies channel 0.
		R_ENDAT1_ID : Specifies channel 1.
		Others : Setting is not allowed.
	cmd	: ENDAT3_CMD_REQ
	p_req	: Request information Specifies the pointer to the r_endat3_req_t structure which holds the request information. For details of the r_endat3_req_t structure, see section “4.10.1(2), r_endat3_req_t”.
Return value	ENDAT3_SUCCESS	: Normal termination
	ENDAT3_ERR_INVALID_ARG	: Abnormal termination (id or cmd is not specified, p_req is NULL or the structure r_endat3_req_t member is not specified.)
	ENDAT3_ERR_BUSY	: Abnormal termination (Transfer is in progress or FG_STATUS register MASTER_READY bit is 0.)
	ENDAT3_ERR_ACCESS	: Abnormal termination (The channel is not started.)

(2) ENDAT3_CMD_BGREQ

ENDAT3_CMD_BGREQ		
Synopsis	Sends background request to the EnDat 3 encoder	
Header	r_endat3_rz2_if.h	
Declaration	r_endat3_err_t R_ENDAT3_Control(const int32_t id, const r_endat3_cmd_t cmd, r_endat3_req_t *p_req);	
Description	Sends background request to the EnDat 3 encoder. The backbround request is separated into three foreground requests. After sending these foreground requests, foreground requests are continuously sent until updating background response. In response to each foreground request, endat3_callback is called. After receiving background response, endat3_callback and endat3_ready callback functions are called.	
Arguments	id	: Specifies the ID to be used. (It is defined in r_endat3_rzt2_dat.h.)
		R_ENDAT0_ID : Specifies channel 0.
		R_ENDAT1_ID : Specifies channel 1.
		Others : Setting is not allowed.
	cmd	: ENDAT3_CMD_BGREQ
	p_req	: Request information Specifies the pointer to the r_endat3_req_t structure which holds the request information. For details of the r_endat3_req_t structure, see section “4.10.1(2),r_endat3_req_t”.
Return value	ENDAT3_SUCCESS	: Normal termination
	ENDAT3_ERR_INVALID_ARG	: Abnormal termination (id or cmd is not specified, p_req is NULL or the structure r_endat3_req_t member is not specified.)
	ENDAT3_ERR_BUSY	: Abnormal termination (Transfer is in progress or FG_STATUS register MASTER_READY bit is 0.)
	ENDAT3_ERR_ACCESS	: Abnormal termination (The channel is not started.)

(3) R_ENDAT3_CMD_RATE

R_ENDAT3_CMD_RATE		
Synopsis	Change data rate of the encoder interface	
Header	r_endat3_rz2_if.h	
Declaration	r_endat3_err_t R_ENDAT3_Control(const int32_t id, const r_endat3_cmd_t cmd, r_endat3_req_t *p_req);	
Description	Change data rate of the EnDat 3 interface to designated value by the request information. The EnDat 3 device data rate should be switched in advance by using the foreground request.	
Arguments	id	: Specifies the ID to be used. (It is defined in r_endat3_rzt2_dat.h.)
		R_ENDAT0_ID : Specifies channel 0.
		R_ENDAT1_ID : Specifies channel 1.
		Others : Setting is not allowed.
	cmd	: R_ENDAT3_CMD_RATE
	p_req	: Request information Specifies the pointer to the r_endat3_req_t structure which holds the request information. Member req_data of the structure is referred in this command. In case req_data is 0, data rate is switched to 12.5 Mbps. In case req_data is 1, data rate is switched to 25 Mbps.
Return value	ENDAT3_SUCCESS	: Normal termination
	ENDAT3_ERR_INVALID_ARG	: Abnormal termination (id or cmd is not specified.)
	ENDAT3_ERR_BUSY	: Abnormal termination (Transfer is in progress.)
	ENDAT3_ERR_ACCESS	: Abnormal termination (The channel is not started.)

(4) R_ENDAT3_CMD_STOP

R_ENDAT3_CMD_STOP		
Synopsis	Stop continuous acquisition of position values	
Header	r_endat3_rz2_if.h	
Declaration	r_endat3_err_t R_ENDAT3_Control(const int32_t id, const r_endat3_cmd_t cmd, r_endat3_req_t * p_req);	
Description	Disables the ELC mode setting during event-synchronized transmission in ELC mode, thus continuous reception of position values from the EnDat 3 encoder is stopped. An error is returned if there is no continuous reception of position values.	
Arguments	id	: Specifies the ID to be used. (It is defined in r_endat3_rzt2_dat.h.)
		R_ENDAT0_ID : Specifies channel 0.
		R_ENDAT1_ID : Specifies channel 1.
		Others : Setting is not allowed.
	cmd	: R_ENDAT3_CMD_STOP
	p_req	: Not used (Specify NULL.)
Return value	ENDAT3_SUCCESS	: Normal termination
	ENDAT3_ERR_INVALID_ARG	: Abnormal termination (id or cmd is not specified.)
	ENDAT3_ERR_ACCESS	: Abnormal termination (ELC mode request has not been sent.)

4.5 Specifications of User-defined Functions

4.5.1 endat3_init_reset_wait_callback

endat3_init_reset_wait_callback	
Synopsis	Function to generate wait time after the first HELLO request
Header	-
Declaration	void endat3_init_reset_wait_callback(void);
Description	Callback function to be registered with the R_ENDAT3_Open function. In the initialization process of the connected encoder, this function generates the time to wait after the HELLO request. Set 300 ms waiting time or longer. The function name is an example and can be set freely.
Arguments	None
Return value	None

4.5.2 endat3_callback

endat3_callback		
Synopsis	Data reception notification for sending requests	
Header	-	
Declaration	void endat3_callback(uint16_t num_result, r_endat3_result_t * p_result, r_endat3_protocol_err_t *per, uint16_t runtime);	
Description	This callback function is registered in the R_ENDAT3_Control() function with command argument ENDAT3_CMD_REQ or ENDAT3_CMD_BGREQ. This function notifies response of the request. This is called when the foreground interrupt is occurred. While background request is executing by the ENDAT3_CMD_BGREQ, callbacks by this function are generated multiple times. Background response is reported as the result of the last callback. This function is the context of the interrupt handler. To ensure interrupt responsiveness, return immediately. The function name is an example and can be set freely.	
Argument	num_result	: Number of transmission results
	p_result	: Result of transmission Pointer to the structure r_endat3_result_t that contains the result of transmission. The reception data is valid until the next request is sent.
	per	: Error information Pointer to the structure r_endat3_protocol_err_t that contains information of the transmission error. The information is valid until the next request is sent.
	runtime	: Propagation time indication Propagation time from request transmission to response reception is reported.
Return value	None	

4.5.3 endat3_pos_callback

endat3_pos_callback	
Synopsis	Data reception notification for ELC mode foreground requests
Header	-
Declaration	void endat3_pos_callback(uint16_t num_result, r_endat3_result_t * p_result, endat3_protocol_err_t *p_err, uint16_t runtime);
Description	This callback function is registered in the R_ENDAT3_Control (ENDAT3_CMD_REQ) function in ELC mode. This function notifies response of the request. This is called each time, when the foreground interrupt is occurred. This function is the context of the interrupt handler. To ensure interrupt responsiveness, return immediately. The function name is an example and can be set freely.
Arguments	num_result : Number of transmission results p_result : Result of transmission Pointer to the structure r_endat3_result_t that contains the result of transmission. The reception data is valid until the next request is sent. p_err : Error information Pointer to the structure r_endat3_protocol_err_t that contains information of the transmission error. The information is valid until the next request is sent. runtime : Propagation time indication Propagation time from request transmission to response reception is reported.
Return value	None

4.5.4 endat3_ready_callback

endat3_ready_callback	
Synopsis	Callback function to notify next communication readiness
Header	-
Declaration	void endat3_ready_callback(void);
Description	This callback function is registered in the R_ENDAT3_Control() function with command argument ENDAT3_CMD_REQ or ENDAT3_CMD_BGREQ. This function notifies that the data reception of the request is completed, and next communication is ready to start. If the ENDAT3_CMD_REQ command is executed, this function is called after the endat3_callback function when a foreground interrupt occurs. While operating in ELC mode, this function is called after the endat3_pos_callback function each time, when foreground interrupt occurs. If the ENDAT3_CMD_BGREQ command is executed, this function is called once in the foreground interrupt after receiving background response. This function is the context of the interrupt handler. To ensure interrupt responsiveness, return immediately. The function name is an example and can be set freely.
Arguments	None
Return value	None

4.6 Interrupt Handlers

4.6.1 enc_ch0_fg_isr

enc_ch0_fg_isr	
Synopsis	Channel 0 foreground interrupt handler
Header	-
Declaration	void enc_ch0_fg_isr(void);
Description	Foreground interrupt handler on EnDat 3 channel 0.
Arguments	None
Return value	None

4.6.2 enc_ch1_fg_isr

enc_ch1_fg_isr	
Synopsis	Channel 1 foreground interrupt handler
Header	-
Declaration	void enc_ch1_fg_isr(void);
Description	Foreground interrupt handler on EnDat 3 channel 1.
Arguments	None
Return value	None

4.6.3 enc_ch0_bg_isr

enc_ch0_bg_isr	
Synopsis	Channel 0 background interrupt handler
Header	-
Declaration	void enc_ch0_bg_isr(void);
Description	Background interrupt handler on EnDat 3 channel 0.
Arguments	None
Return value	None

4.6.4 enc_ch1_bg_isr

enc_ch1_bg_isr	
Synopsis	Channel 1 background interrupt handler
Header	-
Declaration	void enc_ch1_bg_isr(void);
Description	Background interrupt handler on EnDat 3 channel 1.
Arguments	None
Return value	None

4.6.5 enc_ch0_fifo_isr

enc_ch0_fifo_isr	
Synopsis	Channel 0 data reception interrupt handler
Header	-
Declaration	void enc_ch0_fifo_isr(void);
Description	Data reception interrupt handler on EnDat 3 channel 0.
Arguments	None
Return value	None

4.6.6 enc_ch1_fifo_isr

enc_ch1_fifo_isr	
Synopsis	Channel 1 data reception interrupt handler
Header	-
Declaration	void enc_ch1_fifo_isr(void);
Description	Data reception interrupt handler on EnDat 3 channel 1.
Arguments	None
Return value	None

4.7 Interrupts

Table 4.2 lists the interrupt for the EnDat 3 driver.

Table 4.2 Interrupt for the EnDat 3 Driver

Interrupt	ID	Outline
ENCIF_INT0	372	Interrupts are generated when the channel 0 is changed to standby state by the ending of foreground communication or the error.
ENCIF_INT1	373	When the channel 0 state is changed to standby, interrupts are generated if the background communication is in any of following situations. 1. Background response is received. 2. Background request is received by the encoder. 3. An error is occurred in the background communication.
ENCIF_INT2	374	Interrupts are generated when the channel 0 data are received.
ENCIF_INT4	376	Interrupts are generated when the channel 1 is changed to standby state by the ending of foreground communication or the error.
ENCIF_INT5	377	When the channel 1 state is changed to standby, interrupts are generated if the background communication is in any of following situations. 1. Background response is received. 2. Background request is received by the encoder. 3. An error is occurred in the background communication.
ENCIF_INT6	378	Interrupts are generated when the channel 1 data are received.

4.8 Constants and Error Codes

The major constants and error codes are listed below. Table 4.3 lists EnDat 3 request codes and data, Table 4.4 lists background request codes, Table 4.5 lists EnDat 3 frame identifiers, and Table 4.6 lists setting values for the EnDat 3 interface registers. The error code is given in Section 4.10.3(1), r_endat3_err_t.

Table 4.3 EnDat 3 Request Codes / Data

Constant	Value	Content *
R_ENDAT3_REQC_DATA0	0x00	DATA0 request, Activate LPF send list 0
R_ENDAT3_REQC_DATA1	0x01	DATA1 request, Activate LPF send list 1
R_ENDAT3_REQC_DATA2	0x02	DATA2 request, Activate LPF send list 2
R_ENDAT3_REQC_DATA3	0x03	DATA3 request, Activate LPF send list 3
R_ENDAT3_REQC_DATA4	0x04	DATA4 request, Activate LPF send list 4
R_ENDAT3_REQC_DATA5	0x05	DATA5 request, Activate LPF send list 5
R_ENDAT3_REQC_DATA6	0x06	DATA6 request, Activate LPF send list 6
R_ENDAT3_REQC_DATA7	0x07	DATA7 request, Activate LPF send list 7
R_ENDAT3_REQC_DATA	0x08	DATA request
R_ENDAT3_REQC_DATANOP	0x09	DATANOP request
R_ENDAT3_REQC_RESET	0x0B	RESET request, Encoder reset
R_ENDAT3_REQC_CLEAR	0x0C	CLEAR request, Resetting of states
R_ENDAT3_REQC_ECHO	0x0E	ECHO request, Echo for measuring the propagation time
R_ENDAT3_REQC_RATE	0x10	RATE request, Set- data transfer rate
R_ENDAT3_REQC_HELLO	0x22	HELLO request, Test an encoder’s readiness
R_ENDAT3_REQC_BUSBC	0x80	BUSBC request, Bus command for broadcast
R_ENDAT3_REQC_BUSP2P	0x81	BUSP2P request, Bus command for P2P communication
R_ENDAT3_REQC_BUSINIT	0x82	BUSINIT request, Initialization for a bus setup
R_ENDAT3_REQC_FORCE	0x90	FORCE request, Forced dynamic sampling
R_ENDAT3_REQD_DATANOP	0x0000	Data value for DATANOP request
R_ENDAT3_REQD_RESET	0xBBBB	Data value for RESET request
R_ENDAT3_REQD_CLEAR_MAX	0x0007	Maximum data value for CLEAR request
R_ENDAT3_REQD_RATE_MAX	0x0001	Maximum rate value for RATE request
R_ENDAT3_REQD_HELLO	0x2222	Data value for HELLO request
R_ENDAT3_REQD_BUSINIT	0x8282	Data value for BUSINIT request
R_ENDAT3_REQD_FORCE	0x0000	Data value for FORCE request

Note: For details, refer to the “EnDat 3 Interface Specification" which is available from HEIDENHAIN on request.

Table 4.4 Background Request Codes

Constant	Value	Content *
R_ENDAT3_BGREQ_NOP	0x01	NOP request
R_ENDAT3_BGREQ_READ	0x02	READ request, Read from memory area
R_ENDAT3_BGREQ_WRITE	0x03	WRITE request, Write to memory area
R_ENDAT3_BGREQ_RECONFIGURE	0x04	RECONFIGURE request
R_ENDAT3_BGREQ_AUTH	0x80	AUTH request
R_ENDAT3_BGREQ_PROTECT	0x81	PROTECT request
R_ENDAT3_BGREQ_SETPASS	0x83	SETPASS request
R_ENDAT3_BGREQ_LOCATE	0x84	LOCATE request

Note: For details, refer to the “EnDat 3 Interface Specification" which is available from HEIDENHAIN on request.

Table 4.5 EnDat 3 Frame IDs

Constant	Value	Content *
R_ENDAT3_FID_NOP	0x00	Empty content
R_ENDAT3_FID_POS1	0x01	Position 1
R_ENDAT3_FID_POS2	0x02	Position 2
R_ENDAT3_FID_TOUCHPROBE	0x03	Probe signal and timestamp for touch probes
R_ENDAT3_FID_POS_ABS	0x04	Absolute position for incremental encoders
R_ENDAT3_FID_ZERODATA	0x05	Data values 0
R_ENDAT3_FID_ERRMSG	0x0A	Error messages and warning messages
R_ENDAT3_FID_EVALNUM	0x11	Valuation numbers
R_ENDAT3_FID_MOUNT0	0x12	Mounting parameters 0 to 2
R_ENDAT3_FID_MOUNT1	0x13	Mounting parameters 3 to 5
R_ENDAT3_FID_COMMU	0x1A	Commutation and limit position
R_ENDAT3_FID_SENSOR_TEMP_MAX	0x20	Maximum value from motor winding temperatures
R_ENDAT3_FID_SENSOR_TEMP_INT	0x21	Internal temperature of the encoder
R_ENDAT3_FID_SENSOR_TEMP_M1	0x22	Motor winding temperature external 1
R_ENDAT3_FID_SENSOR_TEMP_M2	0x23	Motor winding temperature external 2
R_ENDAT3_FID_SENSOR_TEMP_M3	0x24	Motor winding temperature external 3
R_ENDAT3_FID_SF_POS1	0x50	Safety frame for validation of position 1
R_ENDAT3_FID_BGRSP	0x60	Most recently generated background response
R_ENDAT3_FID_BGREQ	0x68	Most recently processed background response
R_ENDAT3_FID_BUS_ADDR	0x70	Bus address

Note: For details, refer to the “EnDat 3 Interface Specification" which is available from HEIDENHAIN on request.

Table 4.6 Setting Values for the EnDat 3 Interface Registers

Constant	Value	Content
R_ENDAT3_BRATE_25M	0x0001u	Register setting value for data rate 25 Mbps
R_ENDAT3_BRATE_12_5M	0x0003u	Register setting value for data rate 12.5 Mbps
R_ENDAT3_P2P_MODE	false	Topology setting value for P2P mode
R_ENDAT3_BUS_MODE	true	Topology setting value for bus mode
R_ENDAT3_NUM_BUS_SLAVES	1 (P2P mode) 3 (Bus mode)	Number of connecting slaves
R_ENDAT3_WD_11US	0x016Fu	Register setting value for watchdog timer waiting time 11 us

4.9 Fixed-Width Integer Types

Table 4.7 lists the fixed-width integers for the sample program. These fixed-width integers are defined in the standard libraries.

Table 4.7 Fixed-Width Integers for the Sample Program

Symbol	Description
int8_t	8-bit signed integer (defined in the standard libraries)
int16_t	16-bit signed integer (defined in the standard libraries)
int32_t	32-bit signed integer (defined in the standard libraries)
int64_t	64-bit signed integer (defined in the standard libraries)
uint8_t	8-bit unsigned integer (defined in the standard libraries)
uint16_t	16-bit unsigned integer (defined in the standard libraries)
uint32_t	32-bit unsigned integer (defined in the standard libraries)
uint64_t	64-bit unsigned integer (defined in the standard libraries)

4.10 Structures, Unions, and Enumerated Types

4.10.1 Structures

(1) r_endat3_info_t

Initialization information of the EnDat 3 control unit

```
typedef struct
{
    bool                topology;               P2P mode or bus mode setting
                                                Set R_ENDAT3_P2P_MODE in P2P mode. Set
                                                R_ENDAT3_BUS_MODE in bus mode.
    uint16_t            num_bus_slaves;         Number of bus mode slaves setting
                                                See "Table 4.6 Setting Values for the EnDat 3 Interface
                                                Registers". This setting is reflected in the
                                                NUM_BUS_SLAVES bit of the CONFIG_0 register.
    uint16_t            bitrate;                Data rate setting
                                                See "Table 4.6 Setting Values for the EnDat 3 Interface
                                                Registers". This setting is reflected in the CLKSEL bit
                                                and CLKDIV bit of the BRATE register.
    endat3_wait_cb_t    p_enc_init_reset_wait;  A pointer to a callback function that generates the wait
                                                time after an initial HELLO request.
                                                See "4.5.1 endat3_init_reset_wait_callback" for details.
                                                Do not set NULL.
    uint16_t *          p_rd_buf;               A pointer to a buffer for transferring received data
                                                Set a pointer to a buffer allocated for storing received
                                                data. The buffer length should be enough for storing
                                                received data. Receiving data length varies by using
                                                number of LPF frames. Required buffer length of
                                                received data with n LPF frames is 9 + 4n per one
                                                connected encoder. In bus mode, received data from all
                                                encoders are stored together.
    uint16_t            rd_buf_length;          Buffer length for transferring received data
                                                Set length of the buffer pointed by p_rd_buf.
} r_endat3_info_t
```

(2) r_endat3_req_t

Request information to send to the EnDat 3 compliant encoder

By the foreground request transmission command, combination of the request code and the request data is sent to the encoder. By the background request transmission command, background request code and background request data are sent to the encoder by separating into three foreground requests and combining with request codes for background data transmission.

```
typedef struct
{
```

bool	en_bus_req;	Bus mode setting (true: enabled, false: disabled)
uint8_t	bus_code;	Bus code Set request code of the bus request. See “Table 4.3 EnDat 3 Request Codes / Data”. It is valid only if bus mode is enabled. Set BUSBC or BUSP2P request.
uint8_t	bus_addr;	Bus address Set bus address of the bus request. It is used as foreground address in BUSBC request, or bus address in BUSP2P request. Setting value is ignored in other case.
uint8_t	bus_bgaddr;	Bus background address It is used as background address in BUSBC bus request. Setting value is ignored in other case.
uint8_t	req_code;	Request code See "Table 4.3 EnDat 3 Request Codes / Data". If the background request control command (“4.4.5(2) ENDAT3_CMD_BGREQ”) is used, set one of DATA0 to DATA7 or DATA requests.
uint16_t	req_data;	Request data See “Table 4.3 EnDat 3 Request Codes / Data”. If the data value or range is specified in the table depending on the request code, value in the range is available. The setting value is ignored in the background request control command.
uint16_t	watchdog;	Setting value of the watchdog timer See “Table 4.6 Setting Values for the EnDat 3 Interface Registers”. The timer watches transmission time from end of the request transmission to the end of the high priority frame (HPF) reception. Watching time is setting value × 30 us.
uint8_t	cmd;	Background request code See “Table 4.4 Background Request Codes”. This setting is available only with the background request control command (“4.4.5(2) ENDAT3_CMD_BGREQ“).
uint8_t	args[5];	Background request data This setting is available only with the background request control command.
bool	elc;	ELC mode setting (true: enabled, false: disabled) This setting is available only with the foreground request control command (“4.4.5(1) ENDAT3_CMD_REQ”).
r_endat3_isr_result_cb_t	p_isr_result;	Pointer to a callback function to be called when the response of the request is notified. See "4.5.2 endat3_callback" and “4.5.3 endat3_pos_callback” for details. Do not set NULL.
r_endat3_isr_ready_cb_t	p_isr_ready;	Pointer to a callback function to notify next transmission readiness.. See “4.5.4 endat3_ready_callback” for details. Do not set NULL.

```
} r_endat3_req_t
```

(3) r_endat3_result_t

Results of transmission

```
typedef struct
{
    r_endat3_req_err_t      result;     Results of transmitting requests
                                        See “4.10.3(3) r_endat3_req_err_t”.
    r_endat3_data_t         data;       Received data
                                        See “4.10.1(4) r_endat3_data_t”.
    r_endat3_status_t       status;     Encoder Status
                                        See “4.10.1(5) r_endat3_status_t”.
} r_endat3_result_t
```

(4) r_endat3_data_t

Received data

```
typedef struct
{
    uint8_t         timestamp;      Timestamp indication
    uint64_t        data;           Position value (Position 1)
                                    The fifth to the zeroth bytes of the high priority frame (HPF) are
                                    stored.
    uint8_t         status;         Status byte of the HPF is stored.
    uint32_t        lph;            The third to the zeroth bytes of the low priority header (LPH) are
                                    stored.
    uint8_t         lph_bg;         BG field (BG.ERR_EXEC and BG.STATUS) of the LPH is stored.
    uint8_t         lph_zact;       ZACT field of the LPH is stored.
    uint8_t         lph_yact;       YACT field of the LPH is stored.
    uint8_t         lph_xdim;       XDIM field of the LPH is stored.
    uint16_t        lpf[60];        Low priority frames (LPFs) are stored in the array by every 2
                                    bytes. One frame consists of 8 bytes, thus four elements in the
                                    array corresponds to a frame. In response to the single
                                    foreground request, maximum number of received LPFs is up to
                                    fifteen. Data for lph_xdim frames from top of the array are valid.
    uint32_t        crcerr;         CRC error flags of the HPF, LPFs, and HPH frames are stored.
    uint64_t        bgrsp;          The fifth to the zeroth bytes of the background response are
                                    stored. It is valid in callback function in response to the
                                    background request control command ("4.4.5(2)
                                    ENDAT3_CMD_BGREQ”).
} r_endat3_data_t
```

(5) r_endat3_status_t

Encoder Status and validity of the High Priority Frame (HPF) (HPF.STATUS)

```
typedef struct
{
    bool        err_req;    Request code support error
                            (true: request code is not supported, false: supported)
    bool        rm;         Availability of absolute value
                            (true: absolute value is available, false: not available)
    bool        hpfv        Validity of the HPF data
                            (true: HPF data is valid, false: not valid)
    bool        w;          Warning status inside the encoder
                            (true: there is a warning, false: no warning)
    bool        f;          Fault status inside the encoder
                            (true: a fault was found, false: no fault)
} r_endat3_status_t
```

(6) r_endat3_protocol_err_t

EnDat 3 I/F and encoder foreground communication error information

```
typedef struct
{
    bool        comm_error; Communication error (logical sum of cs_error, phy_error, wd_error,
                            strobe_error and buffer_ovr) (true: an error was found, false: no error)
    bool        cs_error;   CRC check error (true: an error was found, false: no error)
    bool        phy_error;  Manchester code error (true: an error was found, false: no error)
    bool        wd_error;   Watchdog error (true: an error was found, false: no error)
    bool        strobe_error;; Strobe error (strobe is detected during an ongoing transmission)
                            (true: an error was found, false: no error)
    bool        buffer_ovr; Reception buffer overrun (true: an error was found, false; no error)
} r_endat3_protocol_err_t
```

4.10.2 Unions

No unions are used.

4.10.3 Enumerated Types

(1) r_endat3_err_t

Error codes of the encoder I/F

```
typedef enum
{
    ENDAT3_SUCCESS          =0,   Normal termination
    ENDAT3_ERR_INVALID_ARG  ,     Argument error
    ENDAT3_ERR_BUSY         ,     API is not executable
    ENDAT3_ERR_ACCESS       ,     Error in the execution order of APIs
    ENDAT3_ERR_DRV                Internal error in driver
} r_endat3_err_t
```

(2) r_endat3_cmd_t

Command settings when the R_ENDAT3_Control function is used

```
typedef enum
{
    ENDAT3_CMD_REQ      ,     Send foreground request to the encoder
    ENDAT3_CMD_BGREQ    ,     Send background request to the encoder
    ENDAT3_CMD_RATE     ,     Change data rate of the encoder interface
    ENDAT3_CMD_STOP           Stop ELC mode request transmission
} r_endat3_cmd_t
```

(3) r_endat3_req_err_t

Result of the request

```
typedef enum
{
    ENDAT3_REQ_SUCCESS  =0,   Normal termination of the transmission
    ENDAT3_REQ_ERR            Data transmission error occurs
} r_endat3_req_err_t
```

4.11 Description of the Sample Program

4.11.1 Operation Outline

This sample program supports the EnDat 3 compliant encoder "ECI 1319 E30-R2" and bus connection encoders “EQN 1337 E30-RB”. This sample program performs the following process.

1) Send requests to the EnDat 3 encoder by commands entered from the console.
2) Read/write to EnDat 3 encoder registers by commands entered from the console.
3) Display the data received from the EnDat 3 encoder on the console.
4) Send requests using the ELC event input trigger function of the EnDat 3 I/F. (GPT events are linked as input events.)

(1) System Block Diagram

Figure 4.1 shows a block diagram of the system.

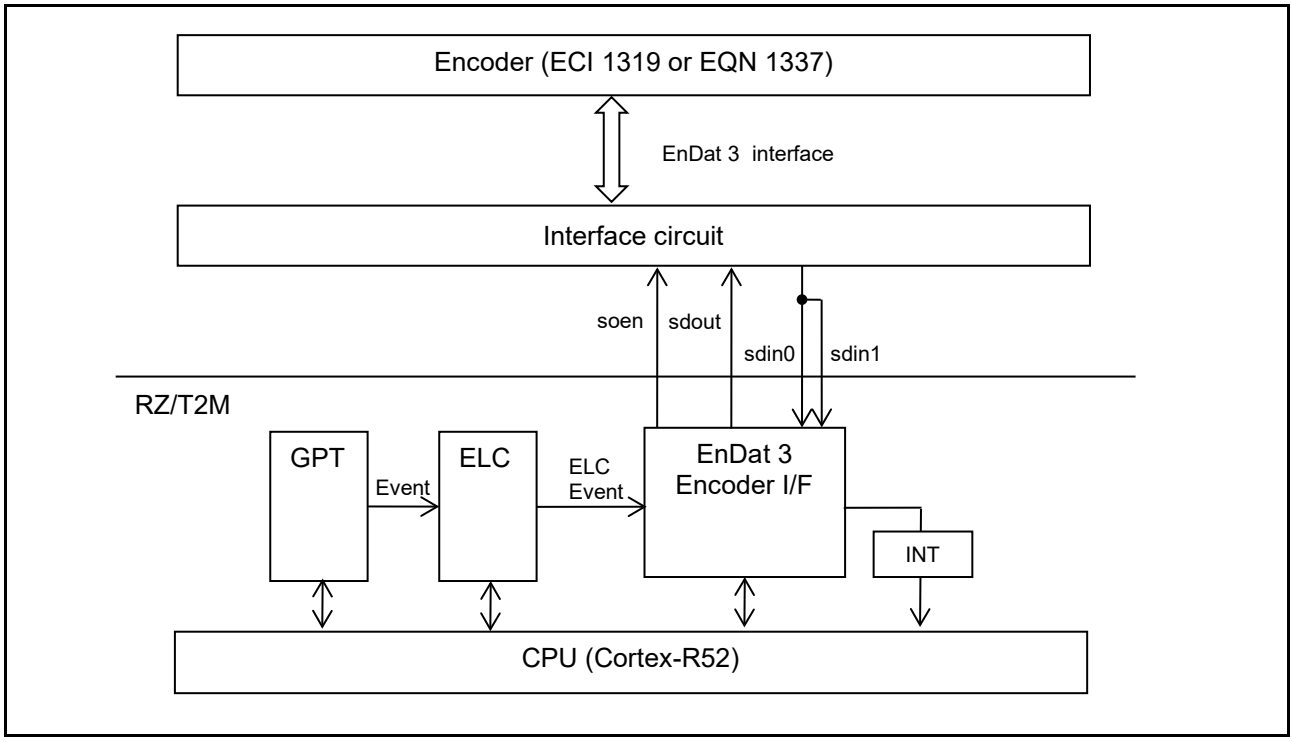

Figure 4.1 System Block Diagram

(2) Software Structure

Figure 4.2 shows the structure of the software.

The EnDat 3 driver has an opening process part composed of the R_ENDAT3_Open function, a closing process part composed of the R_ENDAT3_Close function, a sending requests part composed of the R_ENDAT3_Control function, and a data reception part (interrupt handler) composed of callback functions.

The sample program has an EnDat 3 driver controller section that controls the EnDat 3 driver and sends requests, and a results indication section (callback) that displays the results of data reception.

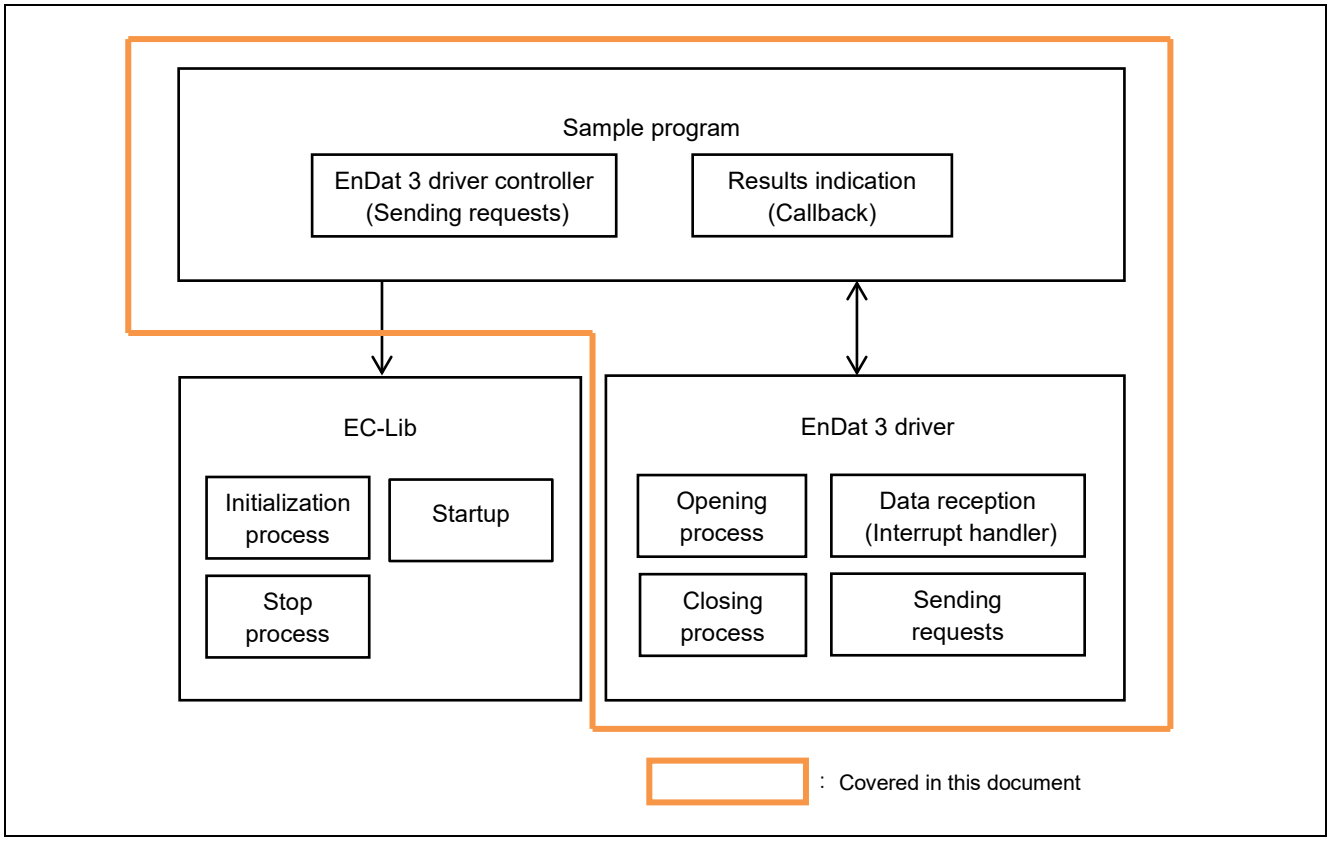

Figure 4.2 Software Structure

4.11.2 Sample Program Functions

Table 4.8 lists the major functions of the sample program.

Table 4.8 Major Functions of the Sample Program

Function Name	Page Number	
	Specification	Flowchart
hal_entry	28	-
enc_main	28	37
endat3_cmd_control	28	38
endat3_power_on_wait	28	-
endat3_init_reset_wait_callback	29	-
endat3_pos	29	39
endat3_read	29	40
endat3_write	29	41
endat3_bus_addr	30	42
endat3_rate	30	43
endat3_elctimer	30	44
endat3_stop	31	45
endat3_req	31	46
endat3_bus_req	31	47
endat3_callback	32	48
endat3_pos_callback	32	49
endat3_ready_callback	32	49
get_cmd	32	-
cmd_exit	33	-
result_display	33	-
result_access_display	33	
result_fg_display	34	
timer_start	34	-
timer_stop	34	-

4.11.3 Specifications of Sample Program Functions

(1) hal_entry

hal_entry	
Synopsis	Entry function of the EnDat 3 sample program
Header	-
Declaration	void hal_entry(void);
Description	This is the entry function of the EnDat 3 sample program. From here, the function enc_main() is called.
Arguments	None
Return value	None

(2) enc_main

enc_main		
Synopsis	Main function of the EnDat 3 sample program	
Header	-	
Declaration	int32_t enc_main(uint8_t ch);	
Description	This is the main function of the EnDat 3 sample program. For details, see section “4.11.6(1), Flowchart of enc_main”.	
Arguments	ch	Encoder channel number 0: specify channel 0, 1: specify channel 1
Return value	0	: Normal termination
	Others	: Abnormal termination (error code of the encoder IF driver)

(3) endat3_cmd_control

endat3_cmd_control	
Synopsis	EnDat 3 driver control function
Header	-
Declaration	static void endat3_cmd_control(void);
Description	This function performs the following operations: • Start of EnDat 3 encoder control • Console command input processing • Termination of EnDat 3 encoder control
Arguments	None
Return value	None

(4) endat3_power_on_wait

endat3_power_on_wait	
Synopsis	Waiting time generation function after encoder power-on
Header	-
Declaration	static void endat3_power_on_wait(void);
Description	This callback function generates the required 1.3 seconds standby time after the encoder is turned on.
Arguments	None
Return value	None

(5) endat3_init_reset_wait_callback

endat3_init_reset_wait_callback	
Synopsis	Waiting time generation function after encoder reset
Header	-
Declaration	static void endat3_init_reset_wait_callback(void);
Description	This callback function generates a waiting time of 300 milliseconds after the encoder reset process in the initialization process of the connected encoder. See “4.5.1 endat3_init_reset_wait_callback”.
Arguments	None
Return value	None

(6) endat3_pos

endat3_pos		
Synopsis	Function to get a positional value from the encoder	
Header	-	
Declaration	static void endat3_pos(uint32_t arg_num, char_t *p_arg[]);	
Description	This function is executed when the console command pos is entered. It acquires a positional value from the encoder. A foreground request with the request code R_ENDAT3_REQC_DATANOP is transmitted to get positional value. See “4.11.6(3) Flowchart of endat3_pos“ for details.	
Arguments	arg_num	: Number of strings entered from the console (Not used.)
	*p_arg[]	: First address of string entered from console (Not used.)
Return value	None	

(7) endat3_read

endat3_read	
Synopsis	Function to read designated amount of data from memory of the encoder
Header	-
Declaration	static void endat3_read(uint32_t arg_num, char_t *p_arg[]);
Description	This function is executed when the console command read is entered. A background request R_ENDAT3_BGREQ_READ with the request code R_ENDAT3_REQC_DATA0 is transmitted to read data. See “4.11.6(4) Flowchart of endat3_read” for details.
Argumetns	arg_num : Number of strings entered from the console
	*p_arg[] : First address of string entered from console
Return value	None

(8) endat3_write

endat3_write	
Synopsis	Function to write data to memory of the encoder
Header	-
Declaration	static void endat3_write(uint32_t arg_num, char_t *p_arg[]);
Description	This function is executed when the console command write is entered. A background request R_ENDAT3_BGREQ_WRITE with the request code R_ENDAT3_REQC_DATA0 is transmitted to write data. See “4.11.6(5) Flowchart of endat3_write” for details.
Argumetns	arg_num : Number of strings entered from the console
	*p_arg[] : First address of string entered from console
Return value	None

(9) endat3_bus_addr

endat3_bus_addr	
Synopsis	Function to change bus address of the encoder interface
Header	-
Declaration	static void endat3_bus_addr(uint32_t arg_num, char_t *p_arg[]);
Description	This function is executed when the console command bus_addr is entered. It changes target encoder of the read and write commands when encoders are connected as a bus mode. See “4.11.6(6) Flowchart of endat3_bus_addr” for details.
Argumetns	arg_num : Number of strings entered from the console *p_arg[] : First address of string entered from console
Return value	None

(10) endat3_rate

endat3_rate	
Synopsis	Function to change data rate of the encoder interface
Header	-
Declaration	static void endat3_rate(uint32_t arg_num, char_t *p_arg[]);
Description	This function is executed when the console command rate is entered. It changes data rate of the encoder interface. See “4.11.6(7) Flowchart of endat3_rate” for details. (Data rate of the encoder device should be changed by foreground request in advance. If there is data rate mismatch between the encoder interface and the encoder device, communication will fail.)
Argumetns	arg_num : Number of strings entered from the console *p_arg[] : First address of string entered from console
Return value	None

(11) endat3_elctimer

endat3_elctimer	
Synopsis	Function to get position values from the encoder synchronously with ELC events
Header	-
Declaration	static void endat3_elctimer(uint32_t arg_num, char_t *p_arg[]);
Description	This function is executed when the console command elctimer is entered. It acquires position values continuously from the encoder synchronously with the ELC events. Foreground requests with the request code R_ENDAT3_REQC_DATANOP are used. See “4.11.6(8) Flowchart of endat3_elctimer“ for details.
Arguments	arg_num : Number of strings entered from the console *p_arg[] : First address of string entered from console
Return value	None

(12) endat3_stop

endat3_stop	
Synopsis	Function to stop position value acquisition synchronized with ELC events
Header	-
Declaration	static void endat3_stop(uint32_t arg_num, char_t *p_arg[]);
Description	This function is executed when the console command stop is entered. This function cancels ELC mode operation and stops sending position value acquisition request. After the continuous positional value transmission from the encoder is stopped, the latest 10 position values are displayed. See “4.11.6(9) Flowchart of endat3_stop” for details.
Arguments	arg_num : Number of strings entered from the console (Not used.) *p_arg[] : First address of string entered from console (Not used.)
Return value	None

(13) endat3_req

endat3_req	
Synopsis	Function to send foreground request to the encoder
Header	-
Declaration	static void endat3_req(uint32_t arg_num, char_t *p_arg[]);
Description	This function is executed when the console command req is entered. This function sends foreground request the encoder and desplays received foreground response. See “4.11.6(10) Flowchart of endat3_req” for details.
Arguments	arg_num : Number of strings entered from the console *p_arg[] : First address of string entered from console
Return value	None

(14) endat3_bus_req

endat3_bus_req	
Synopsis	Function to send foreground request in bus mode to the encoders
Header	-
Declaration	static void endat3_bus_req(uint32_t arg_num, char_t *p_arg[]);
Description	This function is executed when the console command bus_req is entered. This function sends foreground request in bus mode to the encoders and desplays received foreground response. See “4.11.6(11) Flowchart of endat3_bus_req” for details.
Arguments	arg_num : Number of strings entered from the console *p_arg[] : First address of string entered from console
Return value	None

(15) endat3_callback

endat3_callback		
Synopsis	Callback function that notifies response of the request	
Header	-	
Declaration	static void endat3_callback(uint16_t num_result, r_endat3_result_t *p_result, r_endat3_protocol_err_t *p_err, uint16_t runtime);	
Description	This function stores the result in memory.	
Arguments	num_result	: Number of transmission results
	p_result	: Result of the request transmission
	p_err	: EnDat 3 I/F and encoder error information
	runtime	: Propagation time indication
Return value	None	

(16) endat3_pos_callback

endat3_pos_callback		
Synopsis	Callback function that notifies response of the request	
Header	-	
Declaration	static void endat3_pos_callback (uint16_t num_result, r_endat3_result_t *p_result, r_endat3_protocol_err_t *p_err, uint16_t runtime);	
Description	This function stores the continuously acquired results in memory.	
Arguments	num_result	: Number of transmission results
	p_result	: Result of the request transmission
	p_err	: EnDat 3 I/F and encoder error information
	runtime	: Propagation time indication
Return value	None	

(17) endat3_ready_callback

endat3_ready_callback	
Synopsis	Callback function that notifies the next communication readiness
Header	-
Declaration	static void endat3_ready_callback(void);
Description	This function notifies that the data reception of the request is completed, and next communication is ready to start. It is called each time data reception is completed while operating in ELC mode. This function sets the transmission completion flag.
Arguments	None
Return value	None

(18) get_cmd

get_cmd		
Synopsis	Function to get commands from the console	
Header	-	
Declaration	static uint32_t get_cmd(char_t *p_arg[], const uint32_t arg_max);	
Description	This function gets the command from the console	
Arguments	p_arg	: Pointer to an array that stores commands acquired from the console
	arg_max	: Maximum number of strings to acquire
Return value	Number of strings acquired from the console	

(19) cmd_exit

cmd_exit		
Synopsis	Function to indicate end of the EnDat 3 sample program	
Header	-	
Declaration	static void cmd_exit(uint32_t arg_num, char_t *p_arg[]);	
Description	This function is executed when the console command exit is entered. It indicates an end of the EnDat 3 sample program in the console.	
Arguments	arg_num	: Number of strings entered from console (Not used.)
	*p_arg[]	: First address of string entered from console (Not used.)
Return value	None	

(20) result_display

result_display		
Synopsis	Function to display the result of data reception	
Header	-	
Declaration	static void result_display(uint16_t num_result, r_endat3_result_t *p_result, r_endat3_protocol_err_t *p_err);	
Description	This function indicates the result of data reception in response to a request sent to the encoder.	
Arguments	num_result	: Number of transmission results
	p_result	: Result of the request transmission
	p_err	: EnDat 3 I/F and encoder error information
Return value	None	

(21) result_access_display

result_access_display		
Synopsis	Function to display the result of memory access	
Header	-	
Declaration	static void result_access_display(uint32_t addr, uint32_t num_words, r_endat3_result_t *p_result, r_endat3_protocol_err_t *p_err);	
Description	This function indicates the result in response to a memory access request.	
Arguments	addr	: Memory address
	num_words	: Number of access words
	p_result	: Result of the request transmission
	p_err	: EnDat 3 I/F and encoder error information
Return value	None	

(22) result_fg_display

result_fg_display		
Synopsis	Function to display the result of foreground request to the encoder	
Header	-	
Declaration	static void result_fg_display(uint16_t num_result, r_endat3_result_t *p_result, r_endat3_protocol_err_t *p_err, uint16_t runtime);	
Description	This function indicates the result in response to a foreground request sent to the encoder.	
Arguments	num_result	: Number of transmission results
	p_result	: Result of the request transmission
	p_err	: EnDat 3 I/F and encoder error information
	runtime	: Propagation time indication
Return value	None	

(23) timer_start

timer_start	
Synopsis	GPT channel 0 cycle setting/startup function
Header	bsp_api.h hal_data.h
Declaration	static void timer_start(uint32_t us);
Description	This function sets the timer interval on GPT channel 0, and starts the timer.
Arguments	us : Timer interval [us]
Return value	None

(24) timer_stop

timer_stop	
Synopsis	GPT channel 0 timer stop
Header	bsp_api.h hal_data.h
Declaration	static void timer_stop(void);
Description	This function stops GPT channel 0 timer.
Arguments	None
Return value	None

4.11.4 Variables of Sample Program

Table 4.9 lists the major static type variables.

Table 4.9 Major Static Variables

Type	Variable Name	Description	Function to be used
bool	endat3_flag	Transmission completion flag (true: transmission completed, false: transmission is in progress)	endat3_pos endat3_read endat3_write endat3_stop endat3_req endat3_bus_req endat3_callback endat3_ready_callback
uint16_t	endat3_num_result	Number of transmission results	endat3_pos endat3_req endat3_bus_req endat3_callback
r_endat3_result_t	*p_endat3_result	Pointer to the data acquisition results	endat3_pos endat3_read endat3_write endat3_req endat3_bus_req endat3_callback
r_endat3_protocol_err_t	*p_endat3_err	Pointer to the error information	endat3_pos endat3_read endat3_write endat3_reqx endat3_bus_req endat3_callback
uint16_t	endat3_runtime	Propagation time indication	endat3_req endat3_bus_req endat3_callback
r_endat3_req_err_t	result_ti[ENDAT3_POS_NUM][ENDAT3_NUM_BUS_SLAVES]	Errors of continuously acquired position values An array with 10 elements is used as a ring buffer to store the results of the latest 10 acquisitions.	endat3_elctimer endat3_stop endat3_pos_callback
uint64_t	pos_ti[ENDAT3_POS_NUM][ENDAT3_NUM_BUS_SLAVES]	Continuously acquired position values An array with 10 elements is used as a ring buffer to store the results of the latest 10 acquisitions.	endat3_elctimer endat3_stop endat3_pos_callback
r_endat3_staus_t	status_ti[ENDAT3_POS_NUM][ENDAT3_NUM_BUS_SLAVES]	Status of continuous acquisition An array with 10 elements is used as a ring buffer to store the results of the latest 10 acquisitions.	endat3_elctimer endat3_stop endat3_pos_callback
uint8_t	pos_ti_valid	Number of valid elements in result_ti, pos_ti and status_ti arrays. Indicates the number of valid elements of position values stored in the array.	endat3_elctimer endat3_stop endat3_pos_callback

Type	Variable Name	Description	Function to be used
uint8_t	pos_ti_num	Update position indices for result_ti, pos_ti and status_ti arrays. The index to be updated by the next acquired position values.	endat3_elctimer endat3_stop endat3_pos_callback
bool	pos_ti_empty	Space information in result_ti, pos_ti and status_ti arrays. (true: has space, false: has no space)	endat3_elctimer endat3_pos_callback
int32_t	cur_id	Used EnDat 3 I/F driver ID	enc_main endat3_cmd_control endat3_pos endat3_read endat3_write endat3_rate endat3_elctimer endat3_stop endat3_req endat3_bus_req

4.11.5 Constants of Sample Program

Table 4.10 lists the major constants used in the sample program.

Table 4.10 Major constants

Constants	Value	Content
ENDAT3_ENC_TSAT_WAIT	1300u	Standby time after power-on (1.3 s)
ENDAT3_ENC_100US_WAIT	100u	Waiting time after EC-Lib startup (100 us)
ENDAT3_ENC_INIT_RESET_WAIT	300u	Time to wait after encoder reset process (300 ms)
ENDAT3_POS_NUM	10u	Number of elements of the array for storing continuously received position values
ENDAT3_ADDR_MAX	0xFFFFFF	Maximum address of the read, write commands
ENDAT3_DATA_MAX	0xFFFF	Maximum data value of the write command
ENDAT3_NUM_WORDS_MIN	1	Minimum number of words for read command
ENDAT3_NUM_WORDS_MAX	3	Maximum number of words for read command

4.11.6 Flowchart of Main Process

(1) Flowchart of enc_main

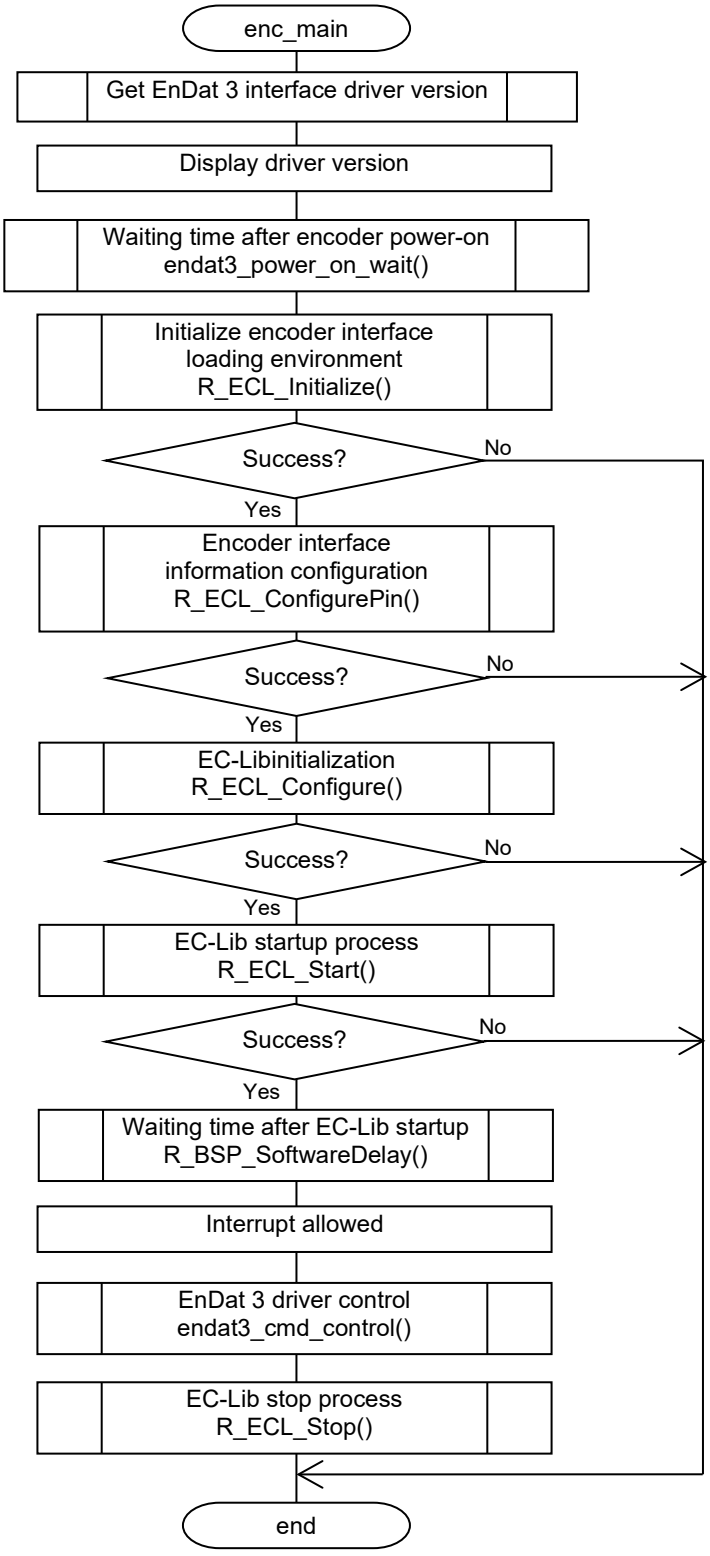

Figure 4.3 Flowchart of enc_main Function

(2) Flowchart of endat3_cmd_control

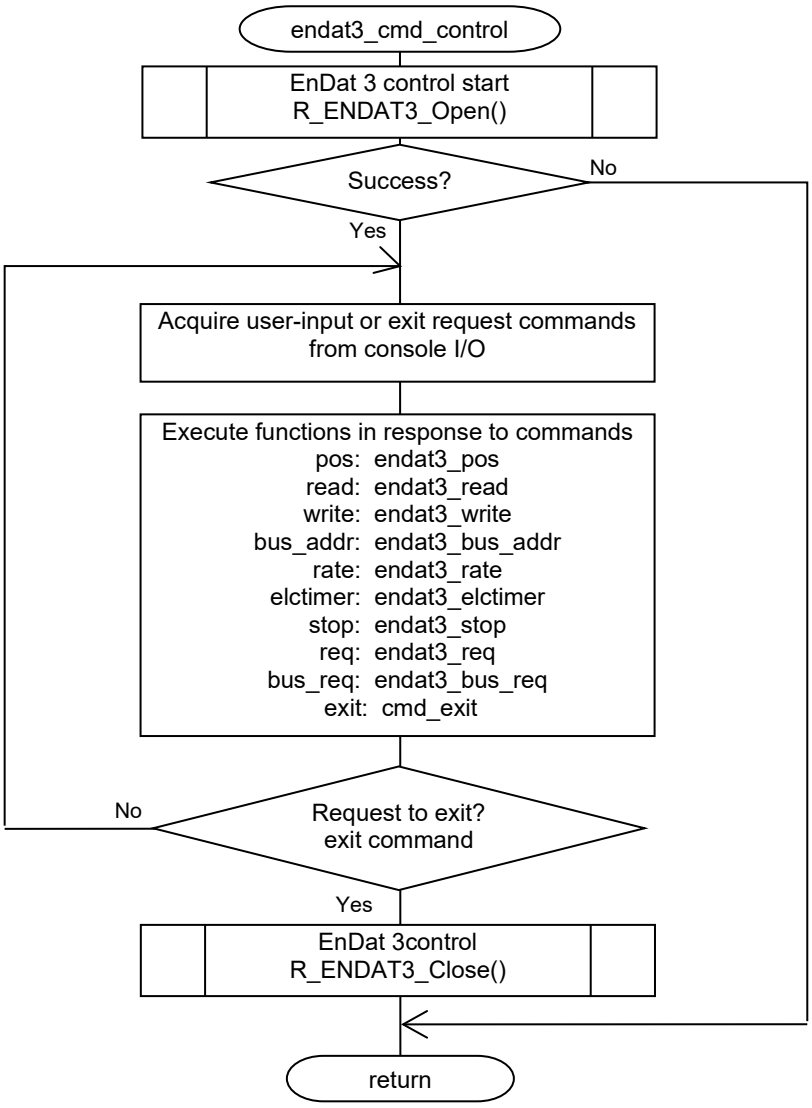

Figure 4.4 Flowchart of endat3_cmd_control Function

(3) Flowchart of endat3_pos

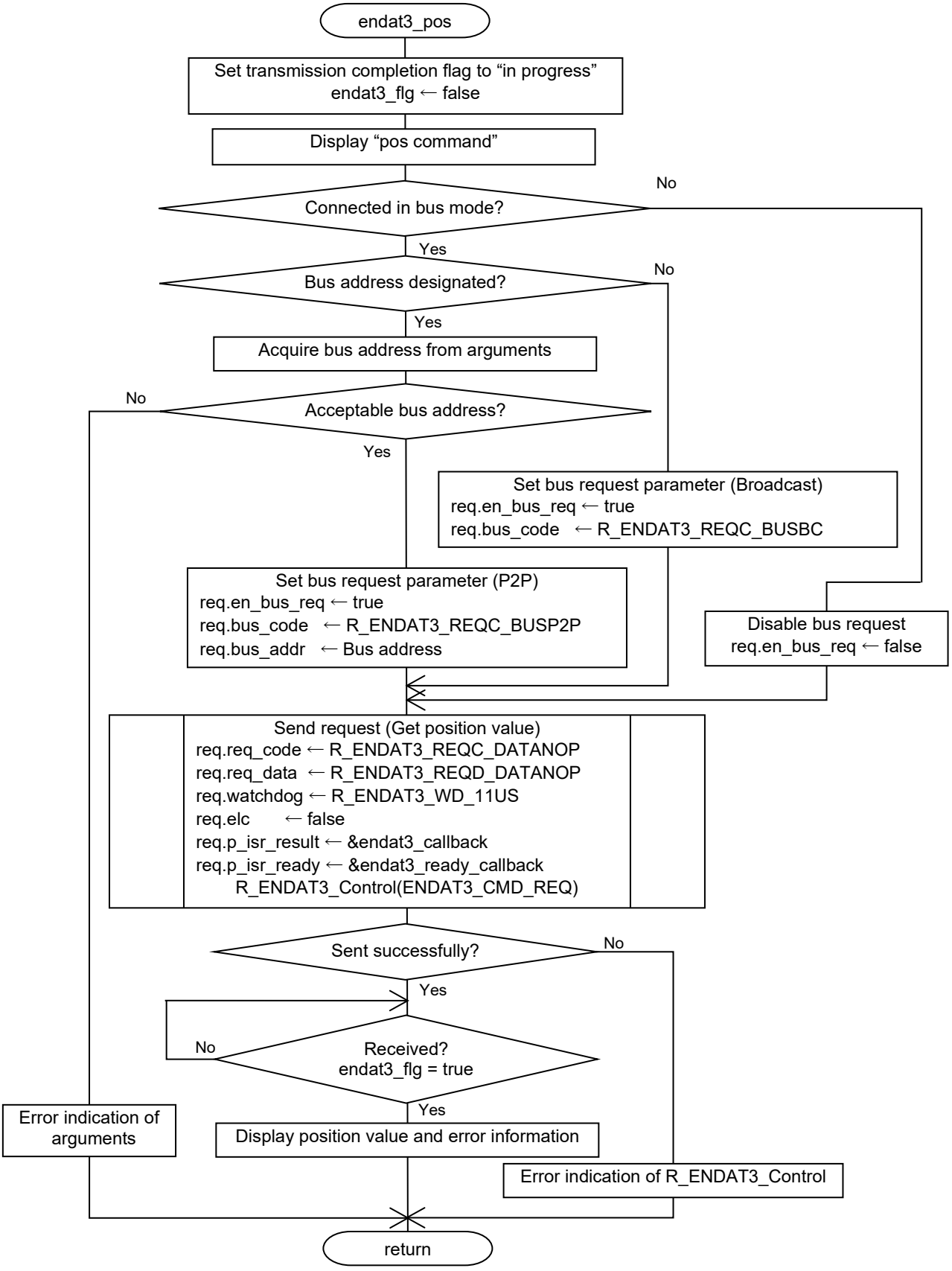

Figure 4.5 Flowchart of endat3_pos Function

(4) Flowchart of endat3_read

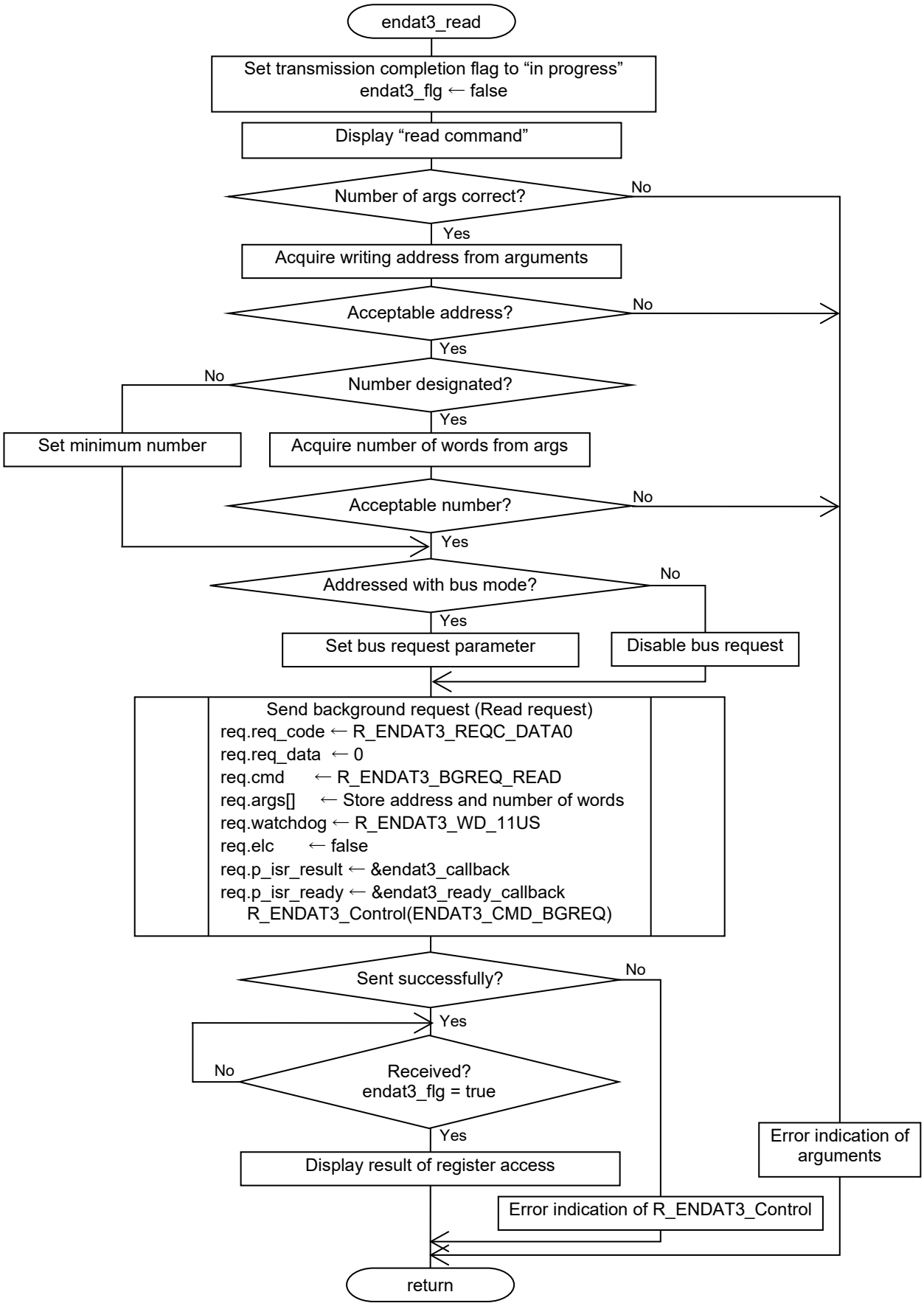

Figure 4.6 Flowchart of endat3_read Function

(5) Flowchart of endat3_write

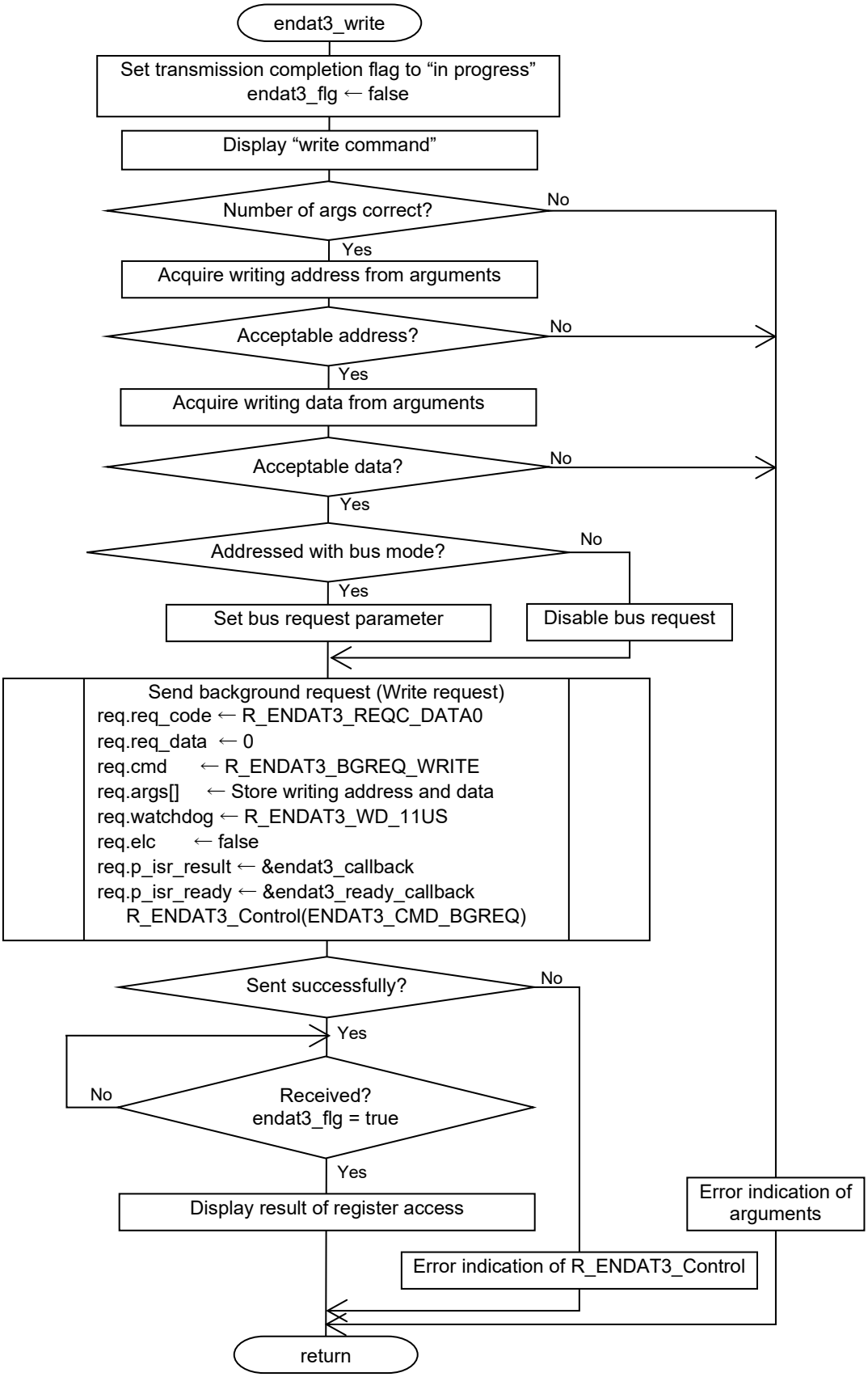

Figure 4.7 Flowchart of endat3_write Function

(6) Flowchart of endat3_bus_addr

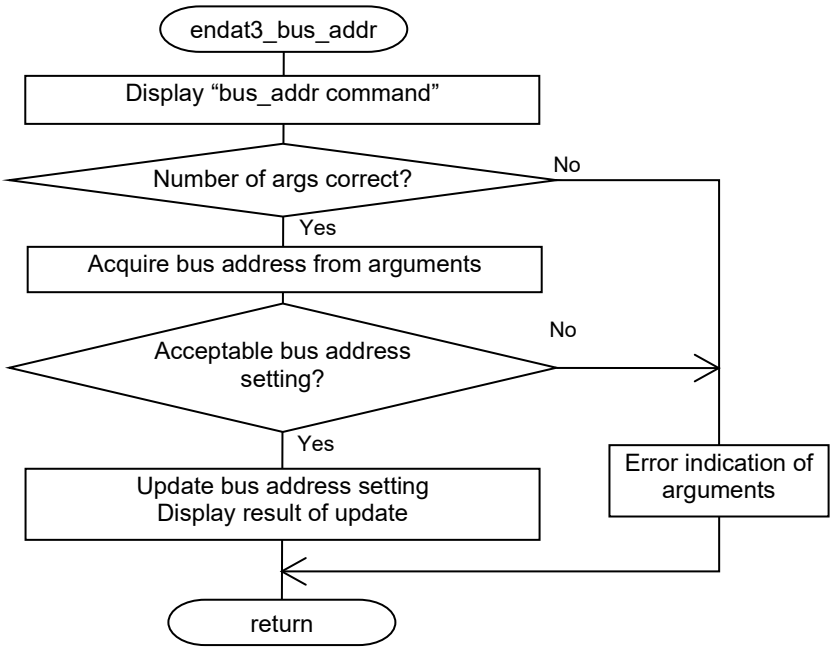

Figure 4.8 Flowchart of endat3_bus_addr Function

(7) Flowchart of endat3_rate

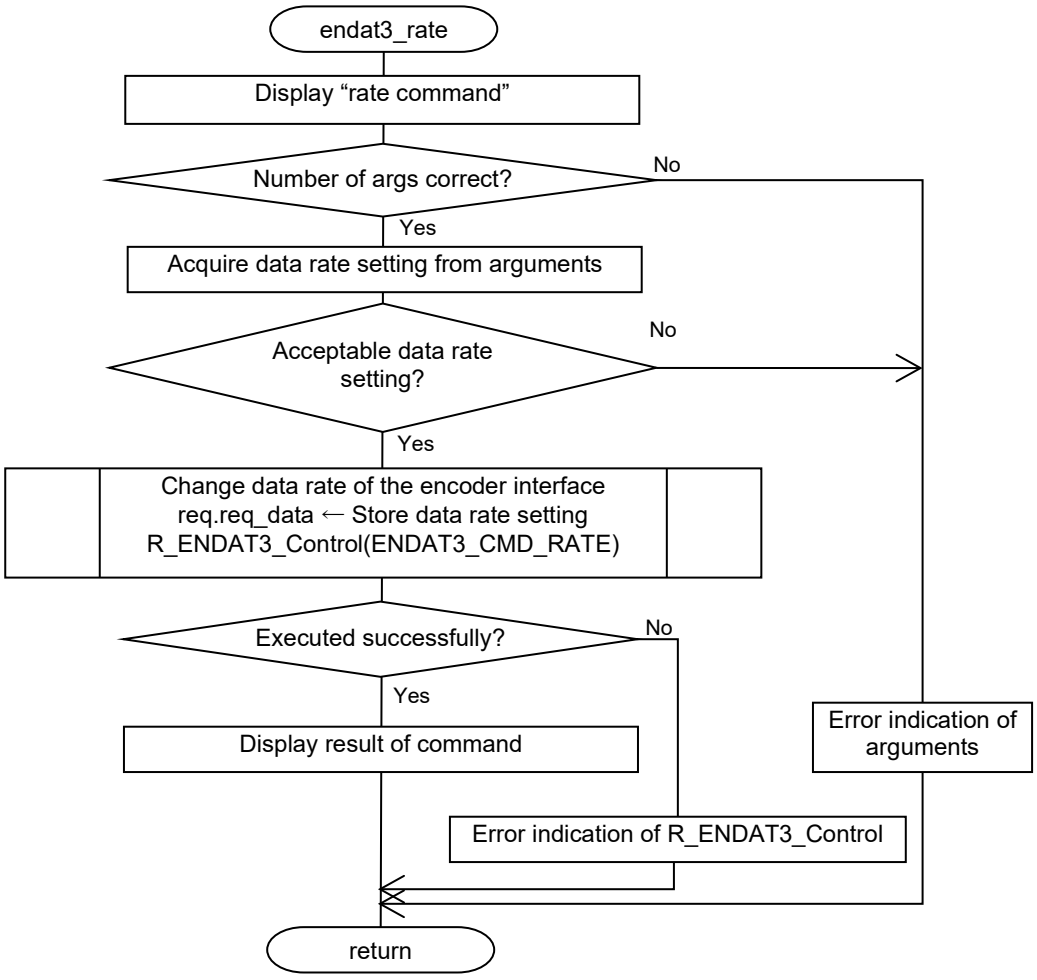

Figure 4.9 Flowchart of endat3_rate Function

(8) Flowchart of endat3_elctimer

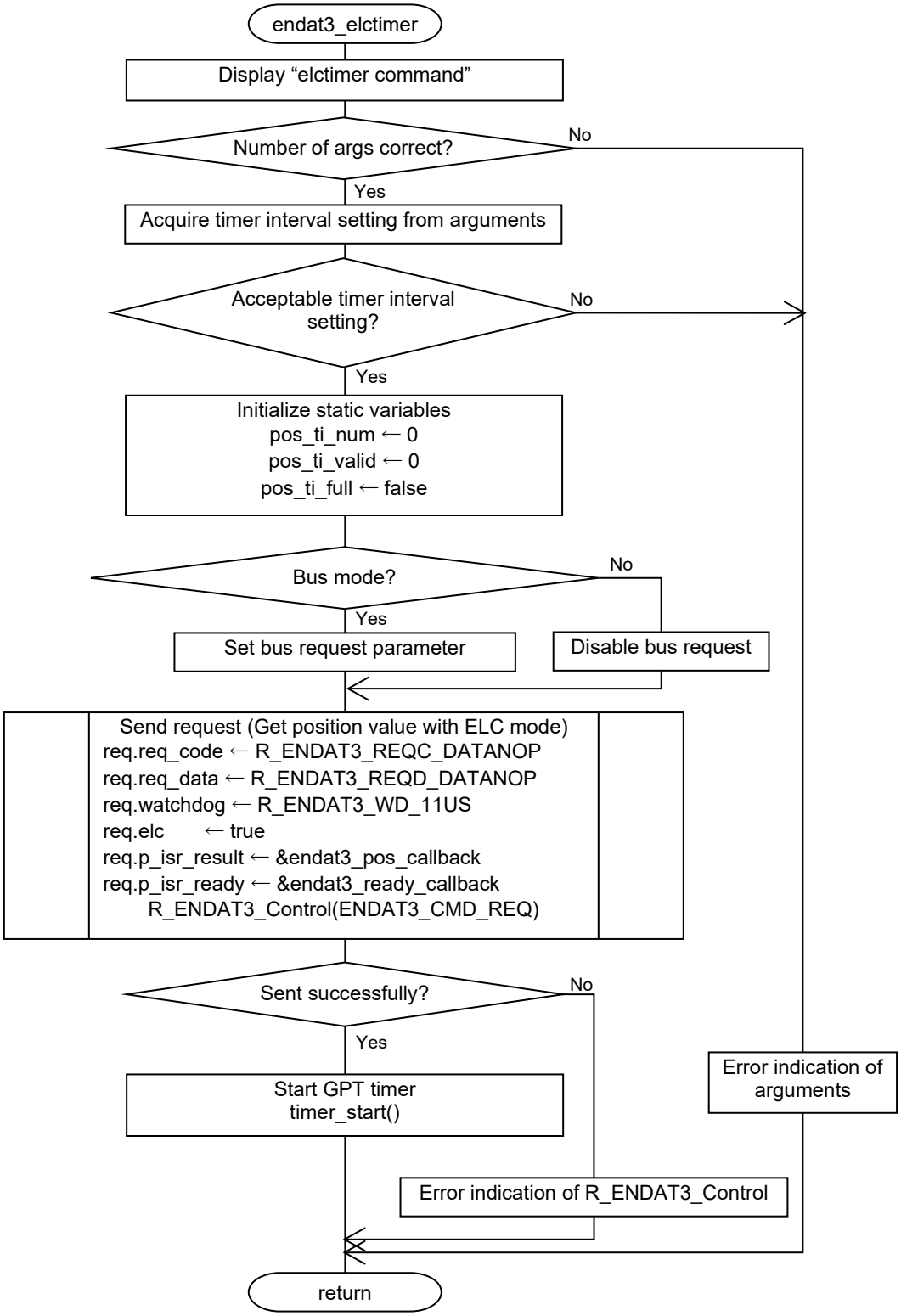

Figure 4.10 Flowchart of endat3_elctimer Function

(9) Flowchart of endat3_stop

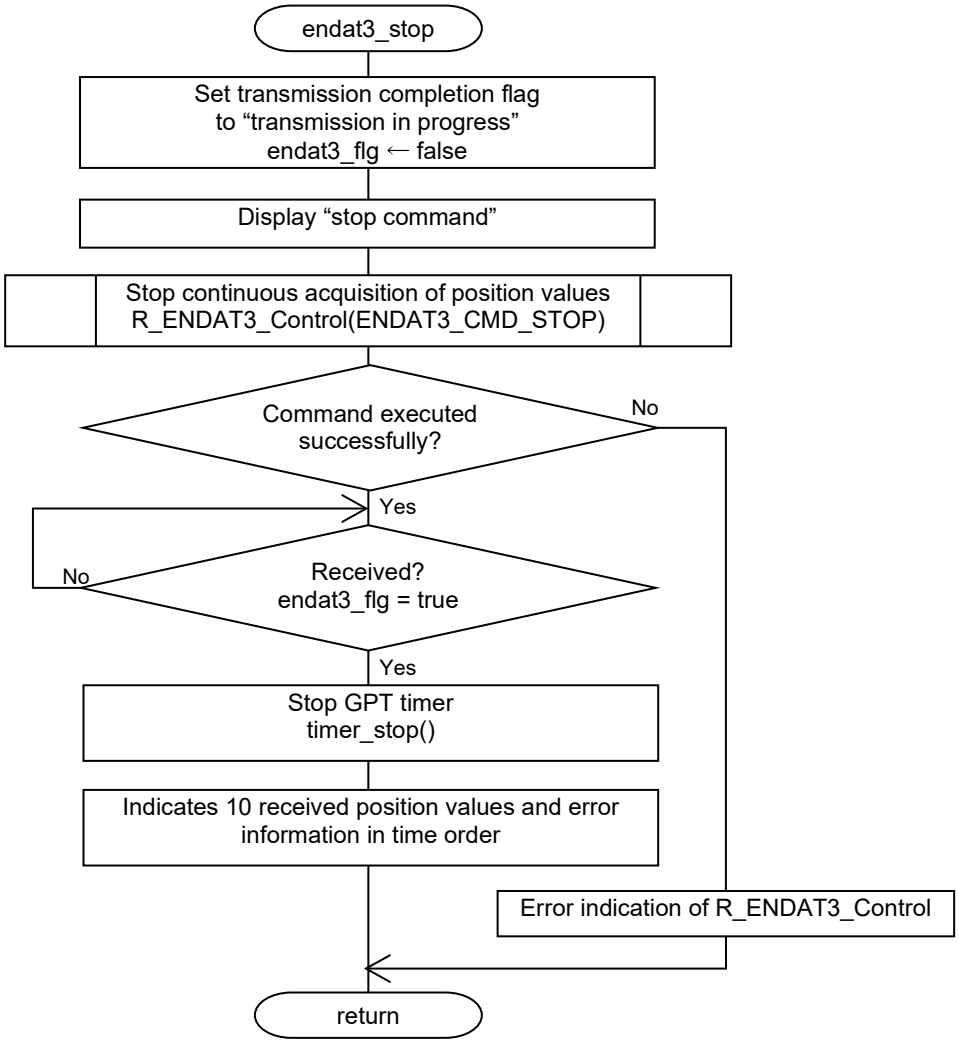

Figure 4.11 Flowchart of endat3_stop Function

(10) Flowchart of endat3_req

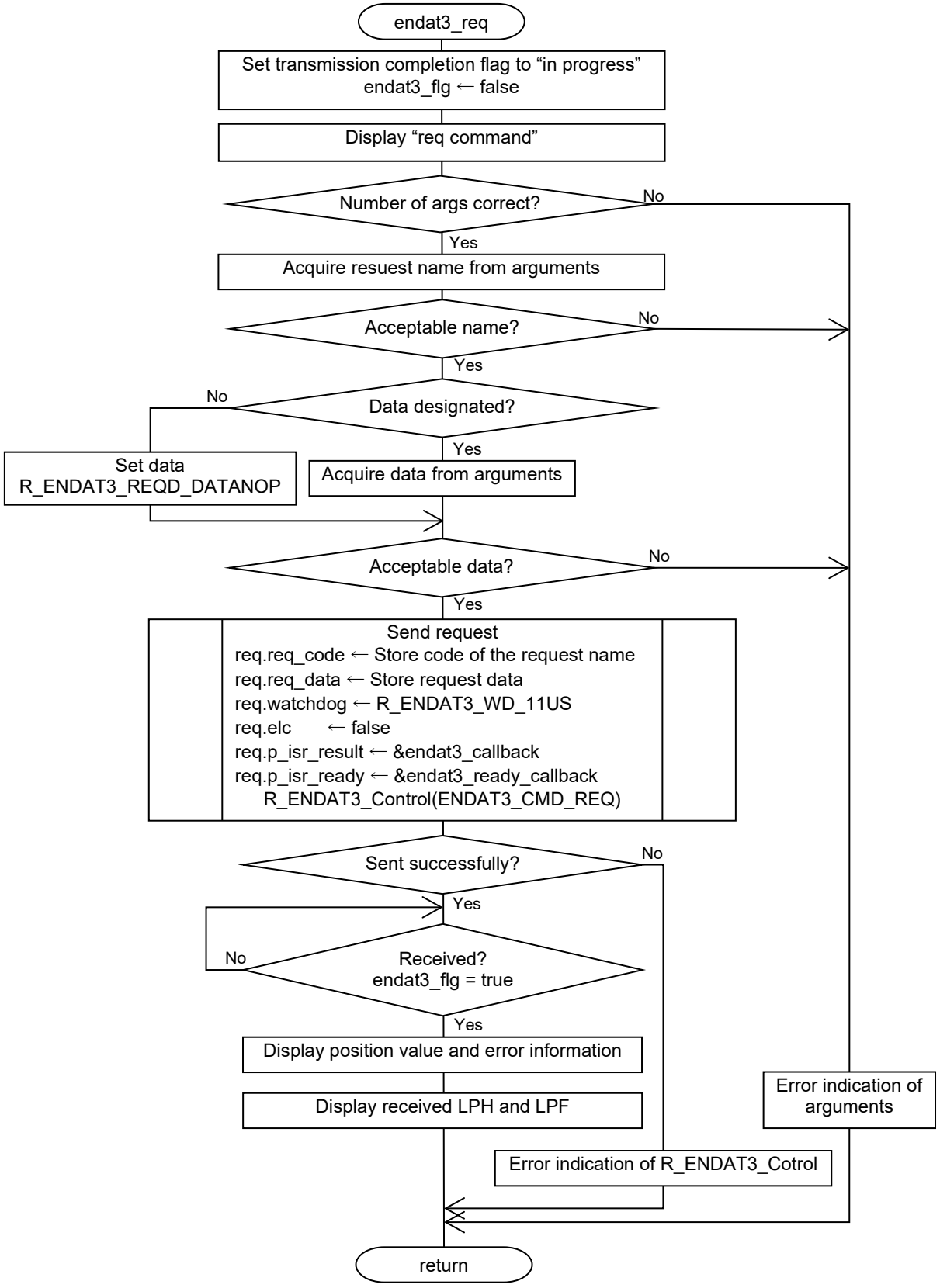

Figure 4.12 Flowchart of endat3_req Function

(11) Flowchart of endat3_bus_req

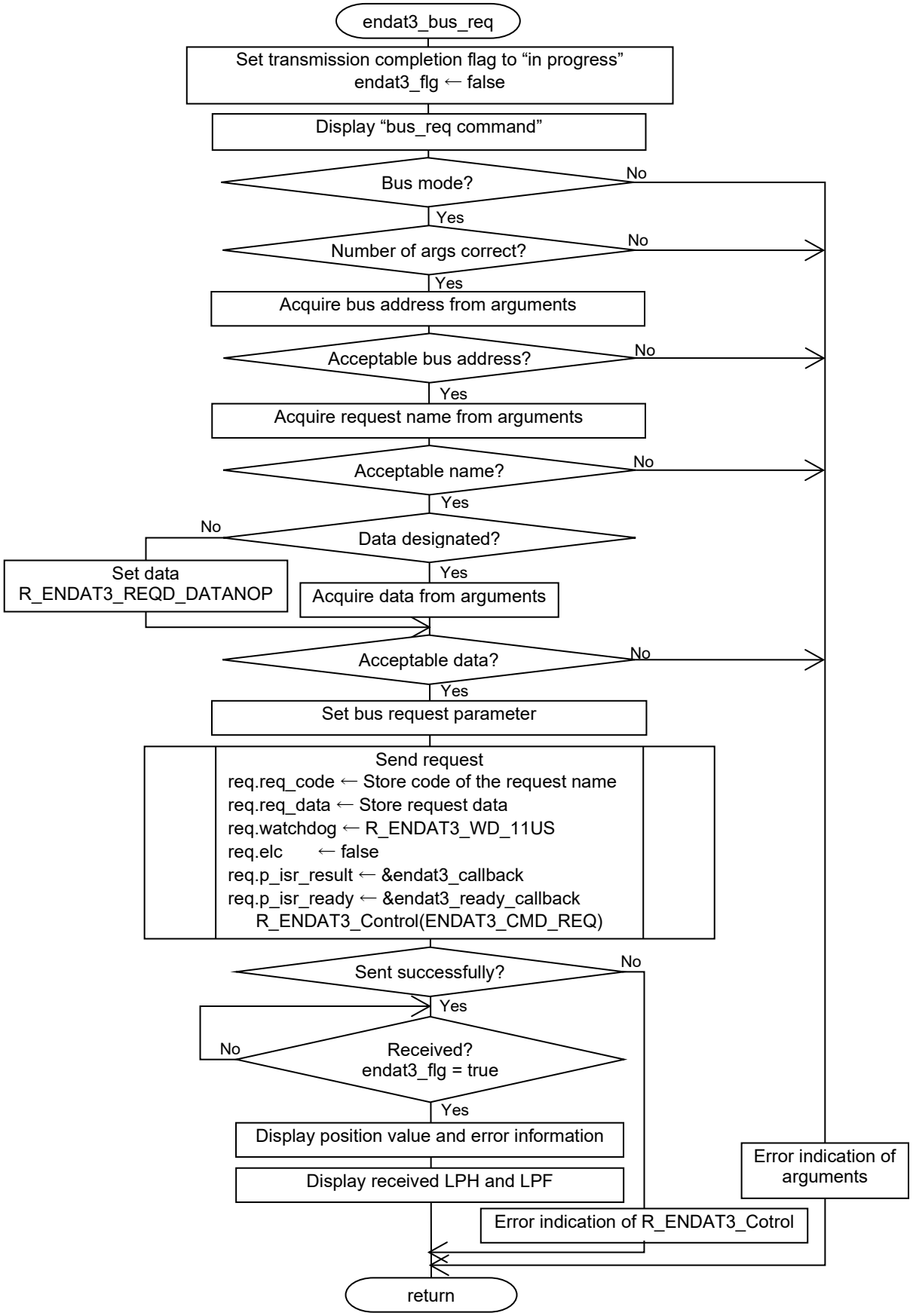

Figure 4.13 Flowchart of endat3_bus_req Function

(12) Flowchart of endat3_callback

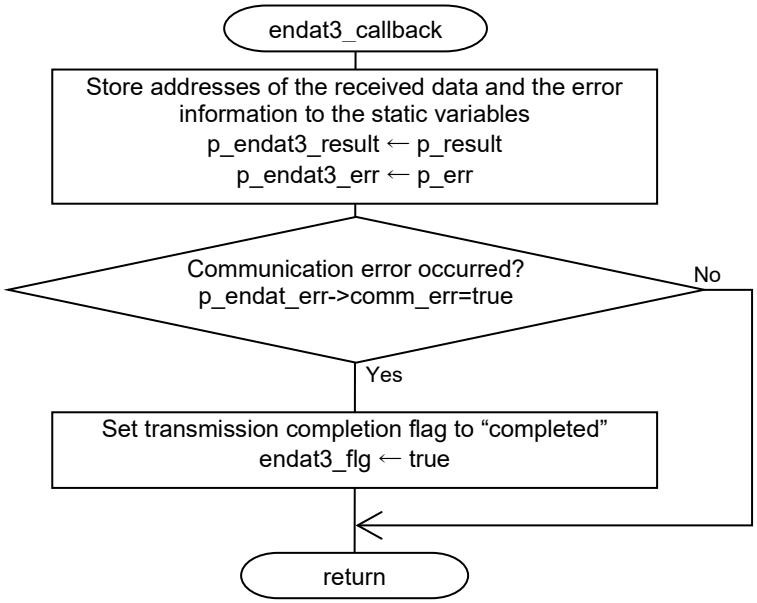

Figure 4.14 Flowchart endat3_callback Function

(13) Flowchart of endat3_pos_callback

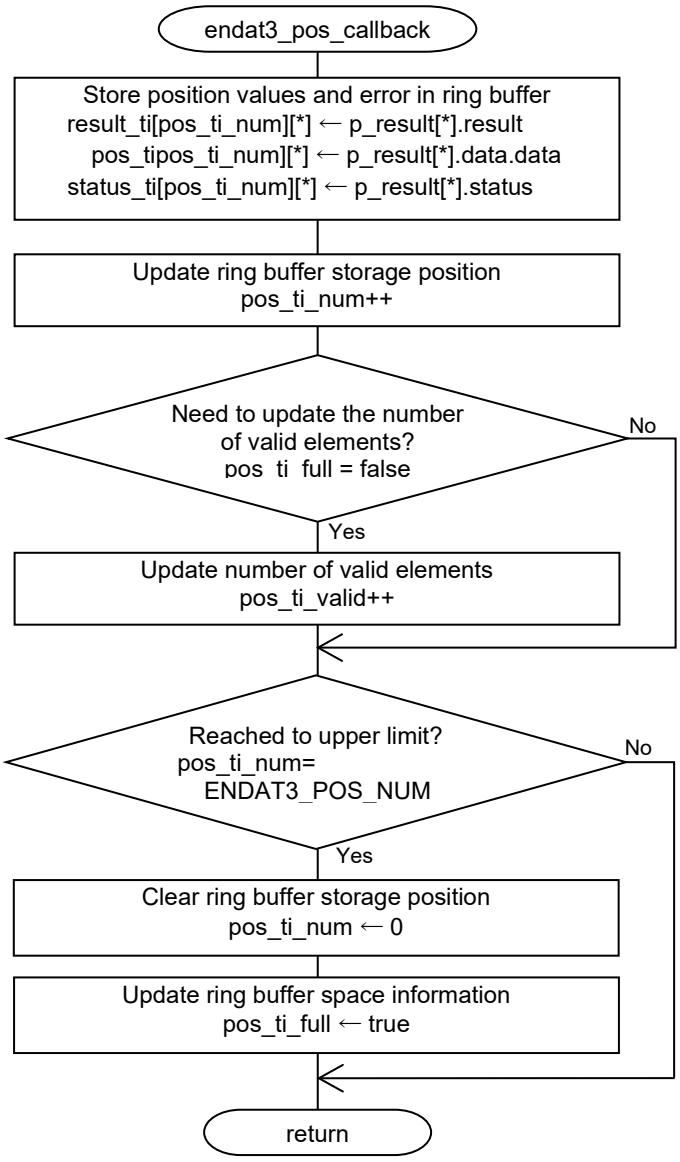

Figure 4.15 Flowchart endat3_pos_callback Function

(14) Flowchart of endat3_ready_callback

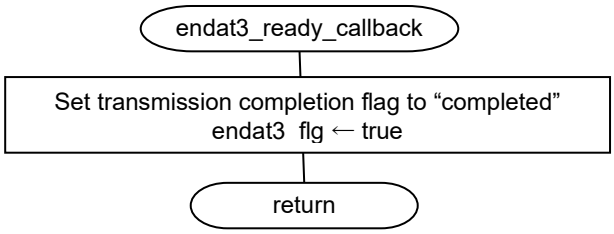

Figure 4.16 Flowchart of endat3_ready_callback Function

4.11.7 Operation Sequence

(1) Startup Sequence

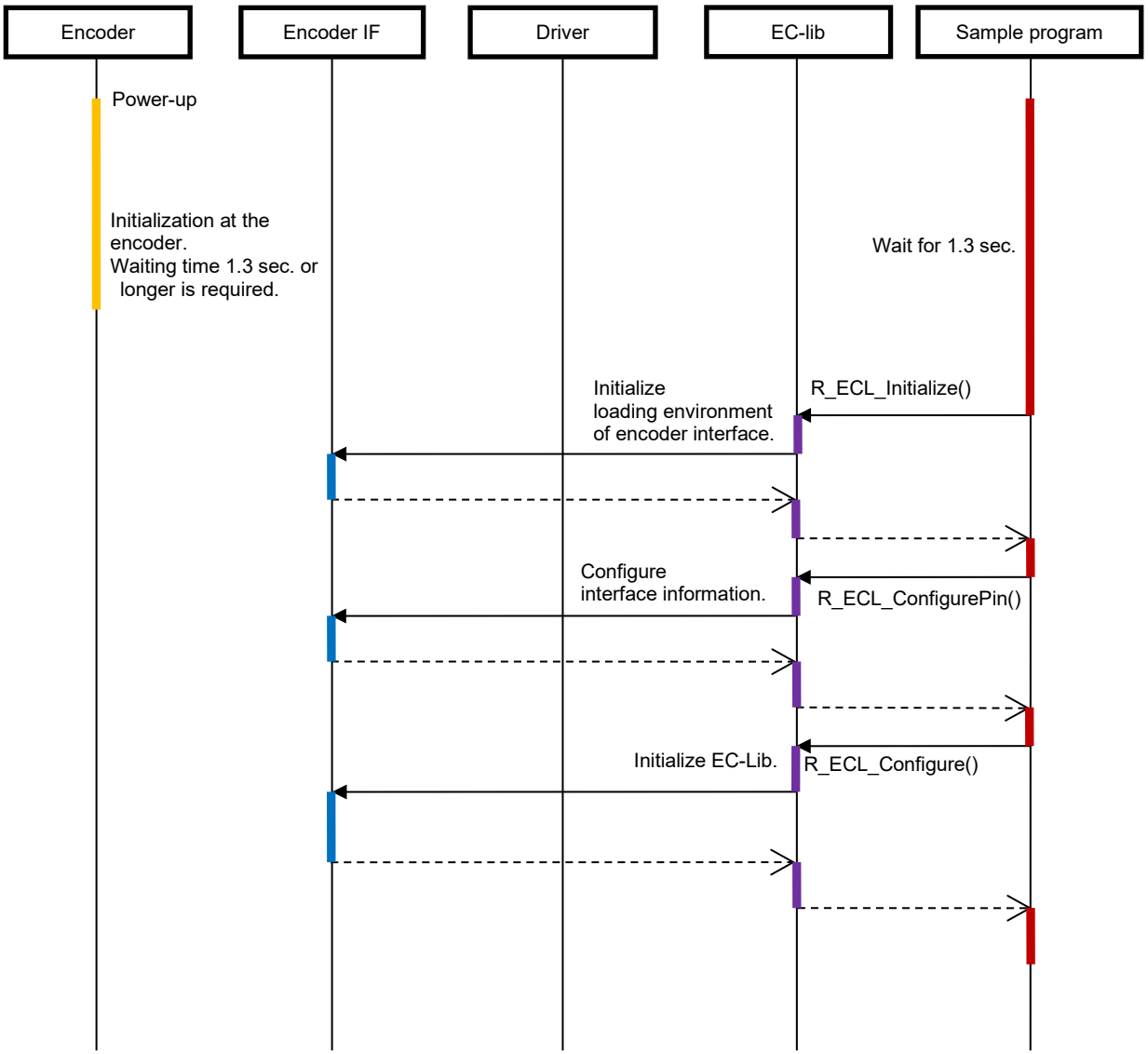

Figure 4.17 Startup Sequence Diagram

(2) Start Sequence

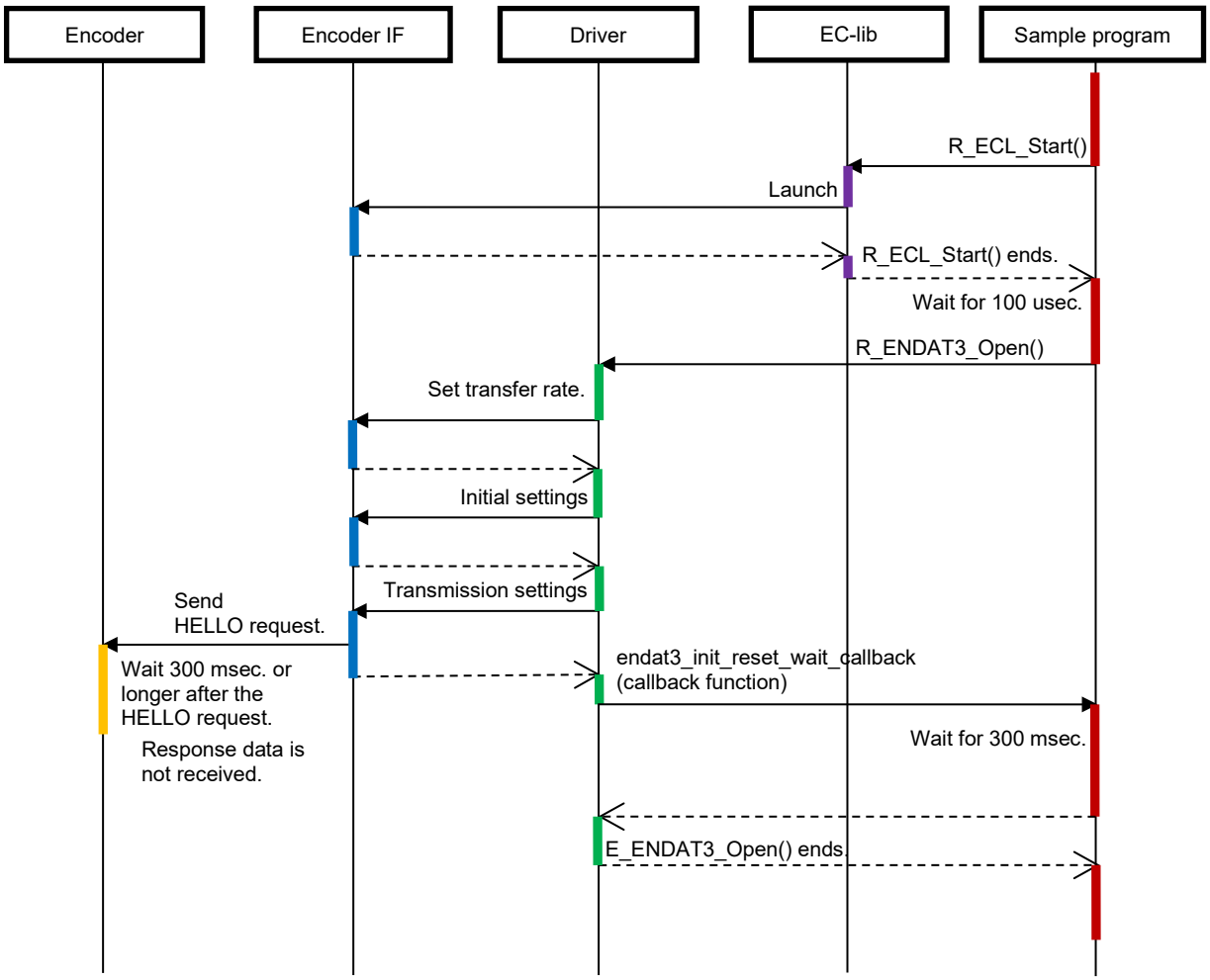

Figure 4.18 Start Sequence Diagram

(3) Sequences of Request Transmission and Data Reception

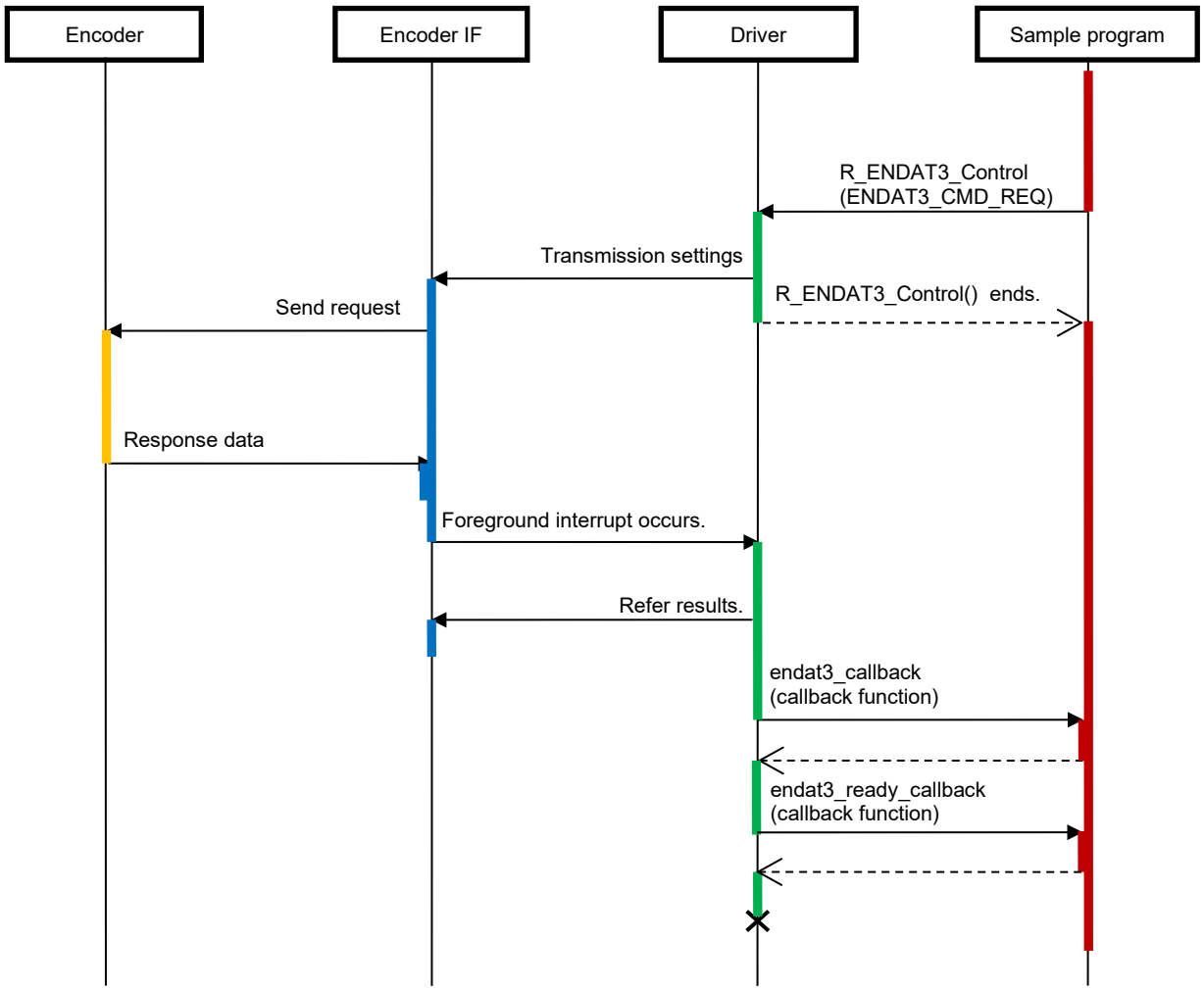

Figure 4.19 Sequences of Request Transmission and Data Reception Diagram

(4) Sequence of Request Transmission (ELC Mode) and Continuous Data Reception

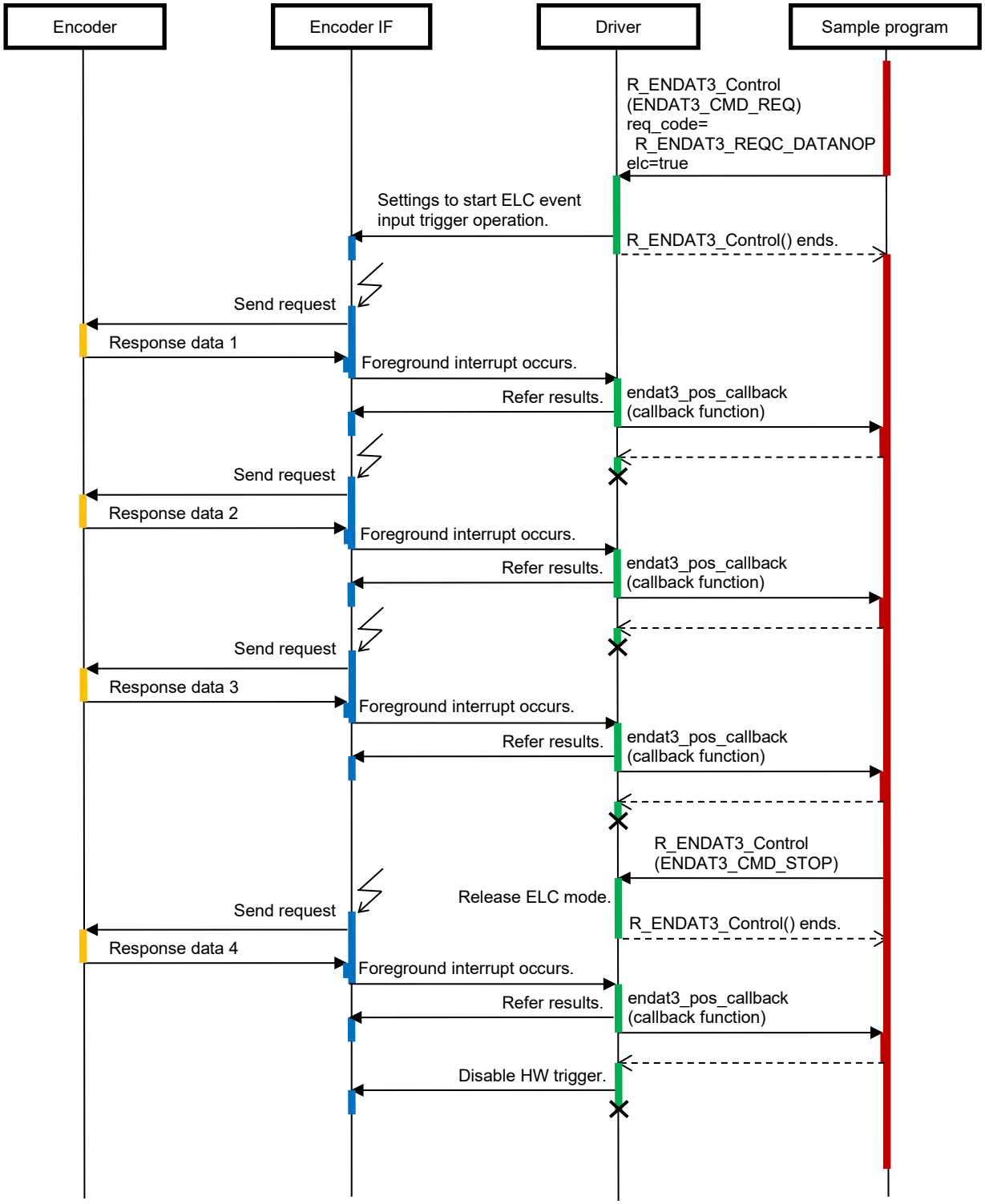

Figure 4.20 Sequence of Request Transmission (ELC Mode) and Continuous Data Reception

(5) Sequence of Background Request Transmission and Data Reception

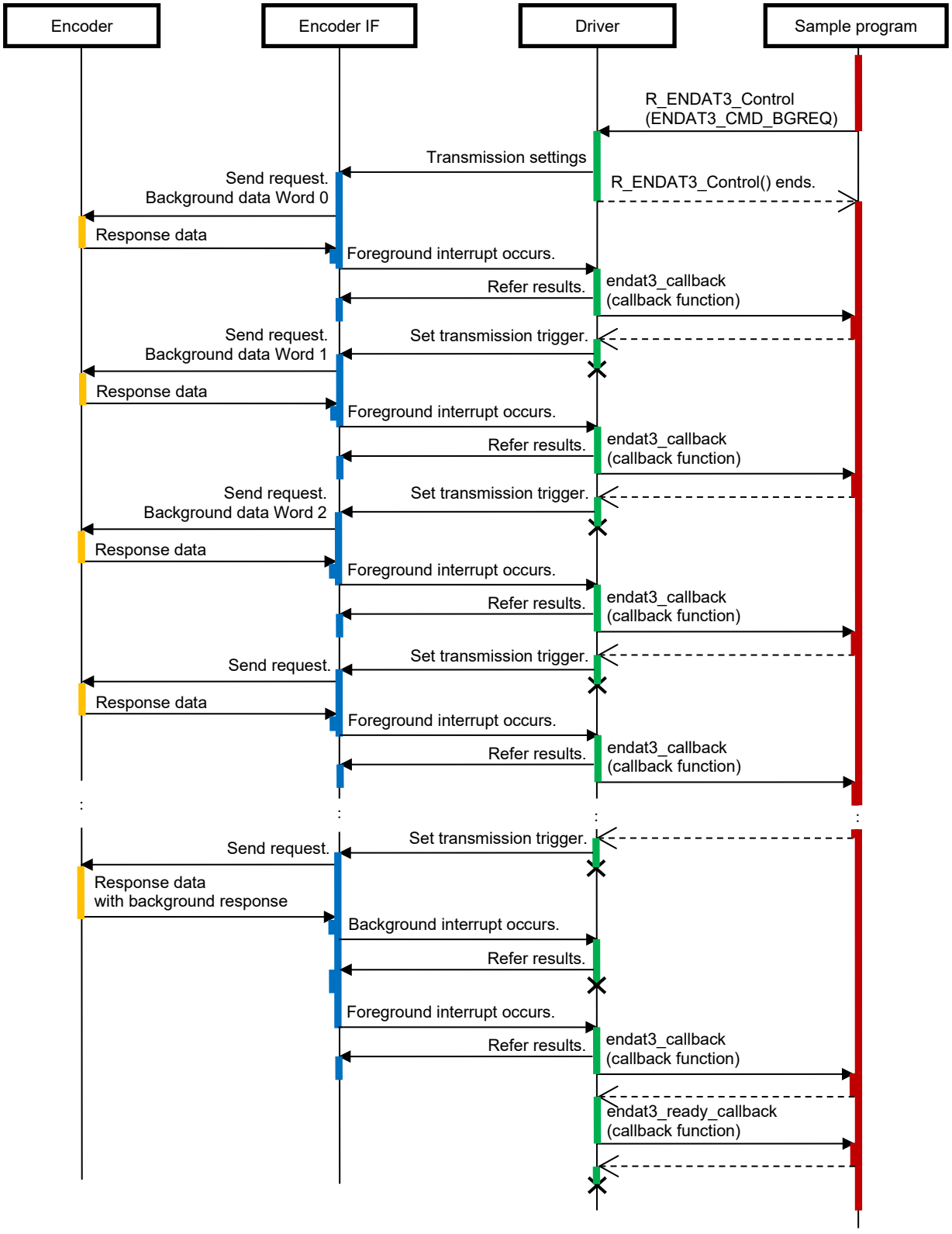

Figure 4.21 Sequence of Background Request Transmission and Data Reception Diagram

(6) Stop Sequence

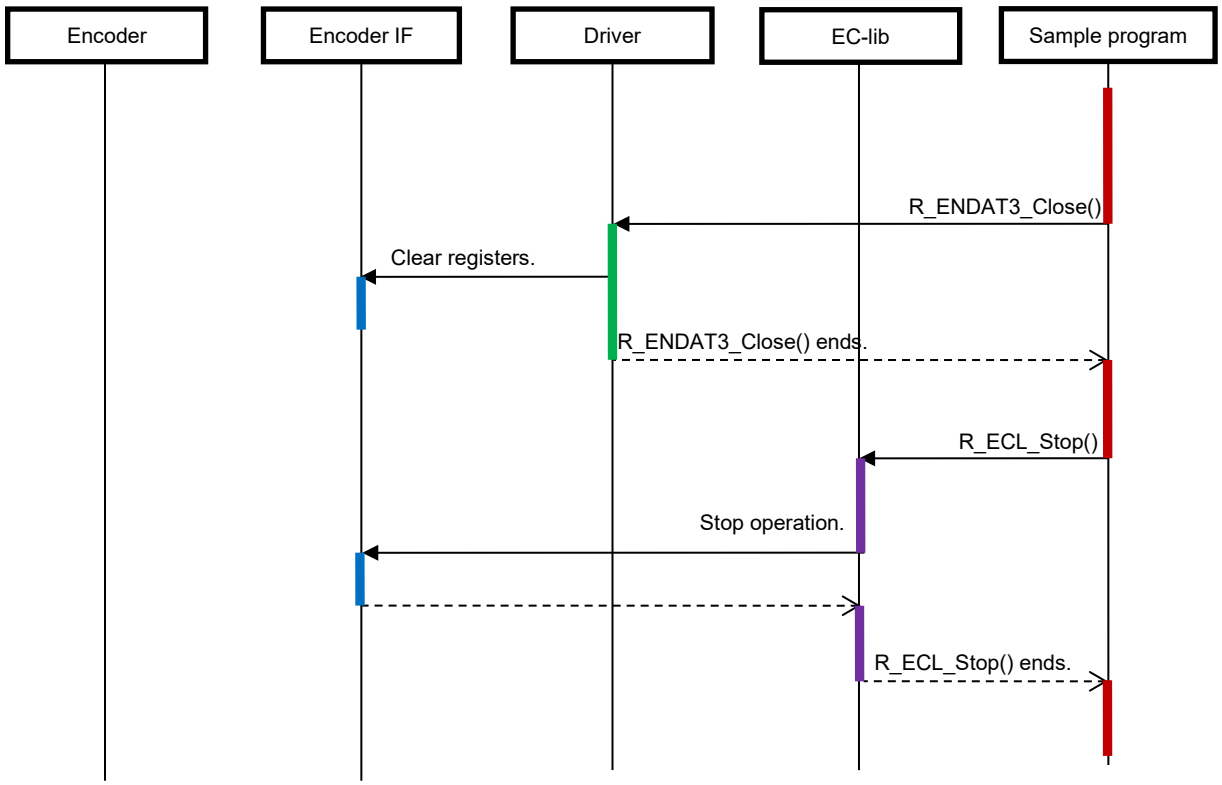

Figure 4.22 Stop Sequence Diagram

4.11.8 Console Commands

This sample program corresponds to the EnDat 3 compliant encoder “ECI 1319 E30-R2” and bus connection encoders “EQN 1337 E30-RB”. The command available for input from the console are listed below.

Table 4.11 Console Commands

Command	Content
pos *ba*	In P2P mode, gets position value once. Bus address [*ba*] is ignored in P2P mode. In bus mode, gets position value once from the encoder designated by bus address [*ba*]. If the bas address designation is omitted, gets position values once from all the encoders by broadcast.
read *addr num*	Reads data from memory of the encoder. Readout address [*addr*] is specified in hexadecimal within the range of 000000 to FFFFFF. Number of words to readout [*num*] is specified in the range of 1 to 3. If the number of words is omitted, 1 is used.
write *addr data*	Writes data to memory of the encoder. Writing address [*addr*] is specified in hexadecimal within the range of 000000 to FFFFFF. Writing data [*data*] is specified in hexadecimal within the range of 0000 to FFFF.
bus_addr *ba*	Sets target encoder of the read / write command of bus mode. Bus address [*ba*] is specified in hexadecimal within the range of 0 to the number of encoders. If 0 is specified as bas address, access without bus request is tried. Initial value of the bus address is 0.
rate *data*	Sets data rate of the encoder interface. Data rate indication [*data*] is selected from 0 or 1. Indication 0 is for 12.5 Mbps. 1 is for 25 Mbps. Data rate of the encoder device should be switched by the foreground request in advance.
elctimer *val*	The position value is continuously acquired in a timer cycle as an ELC event input trigger operation. The timer cycle [*val*] is specified in decimal in units of usec. To stop continuous acquisition, enter the "stop" command.
stop	Stops continuous acquisition of positional values.
req *name data*	Sends foreground request. For the request name [*name*] and request data [*data*], see “Table 4.12 List of Foreground Request Name / Request Data”. Request name is specified in text, request data is specified in hexadecimal. If the request data is omitted, 0000 is used.
bus_req *ba name data*	Sends foreground request in bus mode. Bus address [*ba*] is specified in hexadecimal within the range of 0 to the number of encoders, or FF. If FF is designated as the bas address, broadcast is used. For the request name [*name*] and request data [*data*], see “Table 4.12 List of Foreground Request Name / Request Data”. Request name is specified in text, request data is specified in hexadecimal. If the request data is omitted, 0000 is used.
exit	Exit the program.

Table 4.12 List of Foreground Request Name / Request Data

Request Name	Request Data	Content
DATA0	BGD	Activate low priority frame (LPF) send list 0
DATA1	BGD	Activate low priority frame (LPF) send list 1
DATA2	BGD	Activate low priority frame (LPF) send list 2
DATA3	BGD	Activate low priority frame (LPF) send list 3
DATA4	BGD	Activate low priority frame (LPF) send list 4
DATA5	BGD	Activate low priority frame (LPF) send list 5
DATA6	BGD	Activate low priority frame (LPF) send list 6
DATA7	BGD	Activate low priority frame (LPF) send list 7
DATA	BGD	Keep send list
DATANOP	0x0000	Keep send list and do not transfer BGD
RESET	0xBBBB	Encoder reset
CLEAR	Type	Resetting of status Type bit 0 : Reset error, Type bit 1 : Reset the warning, Type bit 2 : Clear the absolute value, Type bit 15 to 3 : Reserved (The value 0 is to be assigned.)
ECHO	Data	Echo for measuring the propagation time
RATE	Type	Set data transfer rate Type = 0x0000 : Switch to 12.5 Mbps, Type = 0x0001 : Switch to 25 Mbps (All other settings are reserved. Do not use.)
HELLO	0x2222	Test an encoder’s communication readiness

Note. The abbreviation BGF stands for background data. For details, refer to the “EnDat 3 Interface Specification" which is available from HEIDENHAIN on request.

(1) Result of running

After running, it will display the command prompt. Enter the command after “endat3 >”.

```
EnDat3 sample program start
R_ENDAT3_GetVersion = 4.0

endat3 >
```

(2) Example of command execution

It is an example of executing pos command. Results of sending and receiveing requests, received positonal value and status information are reported as the response from the encoder.

```
endat3 >pos
pos command
  result      : ENDAT3_REQ_SUCCESS
  pos mt      : 0x0D13
  pos st      : 0xDA50DE80
  rm          : true
  hpfv        : true
endat3 >
```

4.11.9 Initializing procedure of bus connected encoders

EnDat 3 bus connection supported encoders are accessible in bus mode by daisy-chain connection.

To access bus connected three encoders (assign bus address in ascending order from furthest to RZ/T2M), execute following commands at first.

```
req HELLO 2222
bus_req 3 HELLO 2222
req HELLO 2222
bus_req 2 HELLO 2222
req HELLO 2222
bus_req 1 HELLO 2222
```

Here after, each encoder accepts bus mode commands. For details of the communication with bus connected encoders, refer to the “EnDat 3 Interface Specification” and “EnDat 3 Application Notes” which are available from HEIDENHAIN on request.

5. Sample Code

The sample code is available from the Renesas Electronics website.

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Nov 15.24		First Edition issued
3.00	Oct 24.25	1, 4, 5 26	Change description for trademarks. Revise description of software structure.
4.00	Mar 13.26	8 - 35 13, 14	Change prefix of pointer variables to “p_”. Revise header column of the specifications of user-defined functions.

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.

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 be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.
5. 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.
6. 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.
7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION (“Vulnerability Issues”). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.
8. 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.
9. 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.
10. 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.
11. 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.
12. 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.
13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
14. 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.5.0-1 October 2020)

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/.