RZ/T2M Group

EnDat Safety Sample Program

Summary

This application note explains a sample program for acquiring and indicating information including safety data from an EnDat 2.2 compliant encoder by using the Encoder I/F Configuration Library (EC-Lib) of the RZ/T2M.

The major features of the program are listed below.
・Supports the mode command and the MRS codes used in EnDat 2.2.
・Obtain angle information, etc. from a functional safety-relevant encoder (ECN1123 FS from HEIDENHAIN) compliant with EnDat 2.2 specifications

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 and Figure 1.1 shows the operating environment when the sample code is being executed.

Table 1.1 Peripheral Modules and Applications

Peripheral Module	Application
EnDat I/F	Communication with the EnDat 2.2 compliant encoder (Incremental signal is not supported)
Interrupt controller (ICU)	EnDat 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 Encoder I/F
Serial Communication Interface (SCI) UART	Asynchronous communications of the SCI are used for COM port communications by using USB interface.

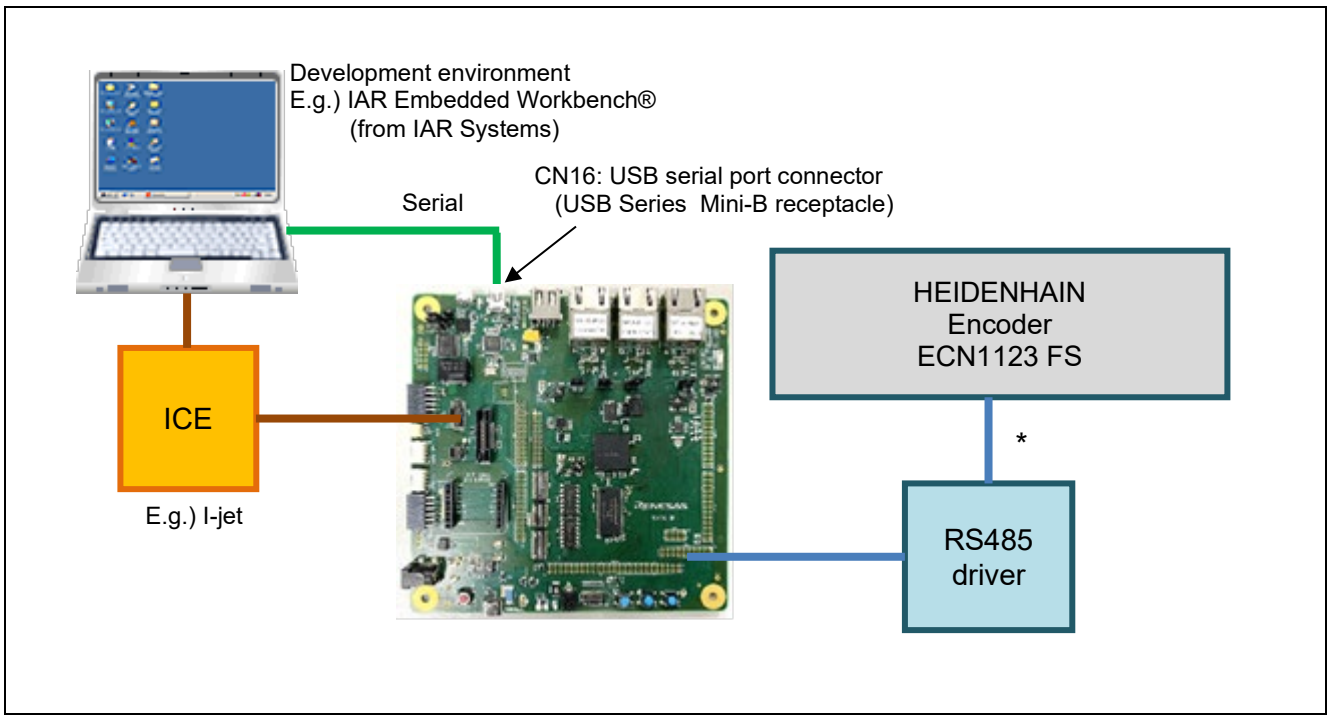

Figure 1.1 Operating Environment

Note: For cable lengths that can be sent and received, please refer to the "EnDat Specification", which can be obtained by contacting HEIDENHAIN.

IAR Embedded Workbench is a registered trademark of IAR Systems.

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
Operating frequency	CPUCLK=800MHz
Operating voltage	1.1 V (Core) / 1.8 V (PLL, etc.) / 3.3 V (I/O)
Integrated development environment *	IAR Embedded Workbench® for ARM from IAR Systems
	e² studio from Renesas
Board	RSK+RZT2M（RTK9RZT2M0C00000BE）
Devices （function to be used on the board）	None

Note Refer to the release note for the RZ/T2M Group Encoder I/F EnDat Safety sample program 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
ENDAT_CH0	TCLK0 (ENCIF4)	P02_3	Output	Clock output pin
	DE0 (ENCIF3)	P02_2	Output	Data output enable pin
	DATA_DV0 (ENCIF2)	P02_0	Output	Data output pin
	DATA_RC0 (ENCIF0)	P01_6	Input	Data input pin
ENDAT_CH1	TCLK1 (ENCIF9)	P03_3	Output	Clock output pin
	DE1 (ENCIF8)	P03_0	Output	Data output enable pin
	DATA_DV1 (ENCIF7)	P17_5	Output	Data output pin
	DATA_RC1 (ENCIF5)	P17_3	Input	Data input pin

4. Software

4.1 EnDat Driver Function

The functions of the EnDat driver are listed below.

1) Initial settings
   A) Settings of the noise filter
   B) Initialization of the encoder (Encoder with battery unit is not supported)
   C) Settings of propagation delay compensation
2) Transmission of the following request information
   A) Mode command
   B) MRS code
   C) Parameters
3) Reception of the encoder data
   A) Positional value
   B) Parameters
   C) Additional information *1

Note 1. In this document, “additional information” represents the additional data 1 and 2. For details, see the “EnDat 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 Safety 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 driver API functions	R_ENDAT_Open	8
	R_ENDAT_Close	8
	R_ENDAT_GetVersion	9
	R_ENDAT_Control	9
User-defined functions	enc_init_reset_wait_callback	12
	enc_init_mem_wait_callback	12
	enc_init_pram_wait_callback	12
	enc_init_cable_wait_callback	13
	endat_callback	13
	endat_poscon_callback	14
	endat_fifodt_callback	14
	endat_rdst_callback	14
Interrupt-handlers	endat0_rx_int_isr	15
	endat1_rx_int_isr	15
	endat0_fifo_int_isr	15
	endat1_fifo_int_isr	15

4.4 Specifications of API Functions

4.4.1 R_ENDAT_Open

R_ENDAT_Open			
Synopsis	Starting control of the encoder		
Header	r_endat_rzt2_if.h, r_endat_rzt2_dat.h		
Declaration	r_endat_err_t R_ENDAT_Open(const int32_t id, r_endat_info_t* pinfo);		
Description	EnDat Driver initializes following initial settings. 1. Initial settings of the noise filter 2. Initialization of the encoder (Encoder with a battery unit is not supported) 3. Settings of propagation delay compensation Execute this function 1.3 seconds after the encoder is turned on. The propagation delay of the cable is automatically measured, but if any of the R_ENDAT_CABLE_DELAY measurements fail, the number of measurements is reduced by that amount. If all measurements fail, the value ENDAT_ERR_DRV is returned.		
Arguments	id	Specifies the ID to be used.	
		R_ENDAT0_ID	: Specifies channel 0
		R_ENDAT1_ID	: Specifies channel 1
		Others	: Setting is not allowed
	pinfo	: Specifies encoder information Specify the address of the structure r_endat_info_t that contains the encoder information	
Return value	ENDAT_SUCCESS		: Normal termination
	ENDAT_ERR_INVALID_ARG		: Abnormal termination (the id or pinfo member variable of the r_endat_info_t structure is not specified value).
	ENDAT_ERR_ACCESS		: Abnormal termination (the driver is already open).
	ENDAT_ERR_DRV		: Abnormal termination (encoder initialization failed).
Note	Before executing this function, be sure to configure and start Multi-Protocol Encoder IF using EC-Lib. The encoder initialization process includes sending the mode command "Encoder receive reset", reading Word 13: "Number of clocks", clearing Word 0 "Error messages", and clearing Word 1: "Warning messages". When initializing the encoder with a battery unit, add the required procedures to the encoder after executing this function by referring to the "Power-on procedure" in the HEIDENHAIN application note (v03).		

4.4.2 R_ENDAT_Close

R_ENDAT_Close			
Synopsis	Ending control of the EnDat encoder		
Header	r_endat_rz2_if.h, r_endat_rzt2_dat.h		
Declaration	r_endat_err_t R_ENDAT_Close(const int32_t id);		
Description	This function handles ending of the encoder IF driver.		
Arguments	id	: Specifies the ID to be used.	
		R_ENDAT0_ID	: Specifies channel 0
		R_ENDAT1_ID	: Specifies channel 1
		Others	: Setting is not allowed
Return value	ENDAT_SUCCESS		: Normal termination
	ENDAT_ERR_INVALID_ARG		: Abnormal termination (a value for argument id has not been specified)
	ENDAT_ERR_ACCESS		: Abnormal termination (Request is being sent.)

4.4.3 R_ENDAT_GetVersion

R_ENDAT_GetVersion		
Synopsis	Acquiring the version number of the encoder interface driver	
Header	r_endat_rz2_if.h	
Declaration	uint32_t R_ENDAT_GetVersion(void);	
Description	This function acquires the version number of the EnDat driver.	
Arguments	None	
Return value	Version information	: The major and minor parts of the version number are stored in the 16 higher and lower-order bits, respectively. Ex.) For ver.1.2, the value returned is 0x00010002

4.4.4 R_ENDAT_Control

R_ENDAT_Control			
Synopsis	Controlling the EnDat encoder		
Header	r_endat_rz2_if.h, r_endat_rzt2_dat.h		
Declaration	r_endat_err_t R_ENDAT_Control(const int32_t id, const r_endat_cmd_t cmd, void *const pbuf);		
Description	Use the argument cmd to control the EnDat encoder. See "4.4.5 List of Control Commands" for the operation of the control commands.		
Arguments	id	: Specifies the ID to be used.	
		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_endat_cmd_t.
	pbuf		: Arguments for each cmd
Return value	ENDAT_SUCCESS		: Normal termination
	ENDAT_ERR_INVALID_ARG		: Abnormal termination (id or cmd is not specified)
	For other return values, see "4.4.5 List of Control Commands".		
Note	Be sure to execute R_ENDAT_Open before executing this function.		

4.4.5 List of Control Commands

(1) ENDAT_CMD_REQ

ENDAT_CMD_REQ			
Synopsis	Sends requests to the EnDat encoder		
Header	r_endat_rz2_if.h, r_endat_rzt2_dat.h		
Declaration	r_endat_err_t R_ENDAT_Control(const int32_t id, const r_endat_cmd_t cmd, void *const pbuf);		
Description	Sends requests to the EnDat encoder. The endat_callback function is called once for each request sent. If the Continuous mode setting is enabled, the endat_poscon_callback function is repeatedly called until ENDAT_CMD_POS_STOP is executed and the endat_rdst_callback function is called. If the ELC mode setting is enabled, the request is repeatedly sent in synchronize with ELC events until ENDAT_CMD_POS_STOP is executed and the endat_rdst_callback function is called. Each time a request is sent, the endat_poscon_callback function is called.		
Arguments	id	: Specifies the ID to be used.	
		R_ENDAT0_ID	: Specifies channel 0
		R_ENDAT1_ID	: Specifies channel 1
		Others	: Setting is not allowed
	cmd		: ENDAT_CMD_REQ
	pbuf		: Request information Specifies the pointer to the r_endat_req_t structure which holds the request information. For details of the r_endat_req_t structure, see Section 4.10.1(1) r_endat_info_t.
Return value	R_ENDAT_SUCCESS		: Normal termination
	R_ENDAT_ERR_INVALID_ARG		: Abnormal termination (id or cmd is not specified, pbuf is NULL or the structure r_endat_req_t member is not specified)
	R_ENDAT_ERR_BUSY		: Abnormal termination (the operation is not possible since transfer is in progress)
	R_ENDAT_ERR_ACCESS		: Abnormal termination (the given channel has not been started)

(2) R_ENDAT_CMD_POS_STOP

R_ENDAT_CMD_POS_STOP			
Synopsis	Stop continuous acquisiton of position values		
Header	r_endat_rz2_if.h, r_endat_rzt2_dat.h		
Declaration	r_endat_err_t R_ENDAT_Control(const int32_t id, const r_endat_cmd_t cmd, void *const pbuf);		
Description	Disables the Continuous mode setting during reception processing in Continuous mode, and also disables the ELC mode setting during event-synchronized send/receive processing in ELC mode, thus continuous reception of position values from the EnDat encoder is stopped. An error is returned if there is no continuous reception processing of position values.		
Arguments	id	: Specifies the ID to be used.	
		R_ENDAT0_ID	: Specifies channel 0
		R_ENDAT1_ID	: Specifies channel 1
		Others	: Setting is not allowed
	cmd		: R_ENDAT_CMD_POS_STOP
	pbuf		: Not used (specify NULL).
Return value	R_ENDAT_SUCCESS		: Normal termination
	R_ENDAT_ERR_INVALID_ARG		: Abnormal termination (id or cmd is not specified)
	R_ENDAT_ERR_ACCESS		: Abnormal termination (continuous mode or ELC mode request has not been sent.)

4.5 Specifications of User-Defined Functions

4.5.1 enc_init_reset_wait_callback

enc_init_reset_wait_callback	
Synopsis	Function to generate wait time after encoder reset
Header	r_endat_rzt2_if.h
Declaration	void enc_init_reset_wait_callback(void);
Description	Callback function to be registered with the R_ENDAT_Open function. Initialization process of the connected encoder generates the time to wait after the encoder reset process. Set 60 ms waiting time or longer. The function name is an example and can be freely set.
Arguments	None
Return value	None

4.5.2 enc_init_mem_wait_callback

enc_init_mem_wait_callback	
Synopsis	Function to generate wait time for detecting memory area selection timeout
Header	r_endat_rzt2_if.h
Declaration	void enc_init_mem_wait_callback(void);
Description	Callback function to be registered with the R_ENDAT_Open function. Generates a wait time used for detecting a timeout error in the process of selecting a memory area in the initialization process of the connected encoder. Set 743 us* waiting time or longer. The function name is an example and can be freely set. Note: This value is based on the assumption of (2 clock cycles + mode command(6 clock cycles) + MRS code(8 clock cycles) + 16 clock cycles + 2T(2 clock cycles) + maximum 7 clock cycles +Start(1 clock cycle) + MRS code(8 clock cycles) + 16 clock cycles + CRC(5 clock cycles))×(1/100 kHz) + t_m(30 us) + t_R(0.5 us) + t_D(1.7 us) = 742.2 us. The transmission clock frequency is set to 100kHz in the driver during the encoder initialization process. The delay time t_D assumes a cable length of 150 m. Users are required to adjust the waiting time according to the encoder and the cable length.
Arguments	None
Return value	None

4.5.3 enc_init_pram_wait_callback

enc_init_pram_wait_callback	
Synopsis	Function to generate wait time for detecting parameter transmission timeout
Header	r_endat_rzt2_if.h
Declaration	void enc_init_pram_wait_callback(void);
Description	Callback function to be registered with the R_ENDAT_Open function. Generates a wait time for the initialization process of the connected encoder to detect timeout errors in the process of sending and receiving parameters by the encoder. Set 13 ms* waiting time or longer. The function name is an example and can be freely set. Note: Assumes memory access time (12 ms) + (Start(1 clock cycle) + Address(8 clock cycles) + Parameters(16 clock cycles) + CRC(5 clock cycles))×(1/100 kHz) + t_m(30 us) + t_R(0.5 us) + t_D(1.7 us) = 12.33 ms. During the encoder initialization process, the transmission clock frequency is set to 100 kHz in the driver. The delay time t_D assumes a cable length of 150 m. Users are required to adjust the waiting time according to the encoder and the cable length.
Arguments	None
Return value	None

4.5.4 enc_init_cable_wait_callback

enc_init_cable_wait_callback	
Synopsis	Function to generate wait time for detecting propagation delay measurement timeout
Header	r_endat_rzt2_if.h
Declaration	void enc_init_cable_wait_callback(void);
Description	Callback function to be registered with the R_ENDAT_Open function. Generates a wait time used for detecting a timeout error in the process of measuring cable propagation delay in the initialization process of the connected encoder. Set 588 us* waiting time or longer. The function name is an example and can be freely set. Note: Assumes t_{cal}(5 us) + (Start(1 clock cycle) + Error(1 clock cycle) + maximum number of bits of main received data (48 bits) + number of CRC bits (5 bits))×(1/100 kHz) + t_m(30 us) + t_R(0.5 us) + t_D(1.7 us) = 587.2 us. During the encoder initialization process, the transmission clock frequency is set to 100 kHz in the driver. The delay time t_D assumes a cable length of 150 m. Users are required to adjust the waiting time according to the encoder and the cable length.
Arguments	None
Return value	None

4.5.5 endat_callback

endat_callback		
Synopsis	Data reception result notification function for sending requests	
Header	r_endat_rzt2_if.h	
Declaration	void endat_callback(r_endat_result_t * presult, r_endat_protocol_err_t *perr);	
Description	This callback function is registered with the R_ENDAT_Control(ENDAT_CMD_REQ) function. This function is called when the MBERR interrupt, the WDG interrupt or the RXEND 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 freely set.	
Argument	presult	: Result of sending/receiving Pointer to structure r_endat_result_t that stores the result of sending/receiving data. The data reception result is valid until the next request is sent.
	per	: Error information Pointer to the structure r_endat_protocol_err_t that contains the results of sending and receiving. The data reception result is valid until the next request is sent.
Return value	None	

4.5.6 endat_poscon_callback

endat_poscon_callback	
Synopsis	Data reception result notification function for sending requests (Continuous mode, ELC mode)
Header	r_endat_rzt2_if.h
Declaration	void endat_poscon_callback(r_endat_result_t * presult, endat_protocol_err_t *perr);
Description	This callback function is registered with the R_ENDAT_Control (ENDAT_CMD_REQ) function when data transmission is performed in Continuous mode or ELC mode. This function notifies the result of data reception in response to a request, and is called when the MBERR interrupt, the WDG interrupt or the RXEND 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 freely set.
Arguments	presult : Result of sending/receiving Pointer to the structure r_endat_result_t that stores the result of sending/receiving data. The data reception result is valid until the next request is sent/received. perr : Error information Pointer to the structure r_endat_protocol_err_t that stores the results of sending and receiving. The data reception result is valid until the next request is sent/received.
Return value	None

4.5.7 endat_fifodt_callback

endat_fifodt_callback	
Synopsis	FIFO data reception result notification function
Header	r_endat_rzt2_if.h
Declaration	void endat_fifodt_callback(r_endat_fifodt_t * pfdat);
Description	This callback function is registered with the R_ENDAT_Control (ENDAT_CMD_REQ) function. This function notifies the result of FIFO data reception. This function is the context of the interrupt handler. To ensure interrupt responsiveness, return immediately. The function name is an example and can be freely set.
Arguments	pfdat : Result of receiving Pointer to the structure r_endat_fifodt_t that stores the result of receiving data by FIFO. The data reception result is valid until the next request is sent/received.

4.5.8 endat_rdst_callback

endat_rdst_callback	
Synopsis	Callback function to notify that the next data communication is ready to start
Header	r_endat_rzt2_if.h
Declaration	void endat_rdst_callback(void);
Description	This callback function is registered with the R_ENDAT_Control(ENDAT_CMD_REQ) function. It is called after the endat_callback function when an RDSTC interrupt occurs. While operating in Continuous mode or ELC mode, this function is called after the endat_poscon_callback function each time data reception is completed. This function is the context for the interrupt handler. To ensure interrupt responsiveness, return promptly. The function name is an example and can be set freely.
Arguments	None
Return value	None

4.6 Interrupt Handler

4.6.1 endat0_rx_int_isr

endat0_rx_int_isr	
Synopsis	Channel 0 data reception completion interrupt handler
Header	-
Declaration	void endat0_rx_int_isr(void);
Description	Interrupt handler for the following interrupt factors on EnDat channel 0. 1. MBERR interrupt 2. WDG interrupt 3. RXEND interrupt 4. RDSTC interrupt
Arguments	None
Return value	None

4.6.2 endat1_rx_int_isr

endat1_rx_int_isr	
Synopsis	Channel 1 data reception completion interrupt handler
Header	-
Declaration	void endat1_rx_int_isr(void);
Description	Interrupt handler for the following interrupt factors on EnDat channel 1. 1. MBERR interrupt 2. WDG interrupt 3. RXEND interrupt 4. RDSTC interrupt
Arguments	None
Return value	None

4.6.3 endat0_fifo_int_isr

endat0_fifo_int_isr	
Synopsis	Channel 0 FIFO data reception completion interrupt handler
Header	-
Declaration	void endat0_fifo_int_isr(void);
Description	Interrupt handler for the FIFO data reception interrupt on EnDat channel 0.
Arguments	None
Return value	None

4.6.4 endat1_fifo_int_isr

endat1_fifo_int_isr	
Synopsis	Channel 1 FIFO data reception completion interrupt handler
Header	-
Declaration	void endat1_fifo_int_isr(void);
Description	Interrupt handler for the FIFO data reception interrupt on EnDat channel 1.
Arguments	None
Return value	None

4.7 Interrupts

Table 4.2 lists the interrupt for the EnDat driver.

Table 4.2 Interrupt for the EnDat Driver

Interrupt	ID	Outline
Channel 0 data reception	372	Interrupts are generated by the following interrupt factors. 1. MBERR interrupt 2. WDG interrupt 3. RXEND interrupt 4. RDSTC interrupt
Channel 1 data reception	376	Interrupts are generated by the following interrupt factors. 1. MBERR interrupt 2. WDG interrupt 3. RXEND interrupt 4. RDSTC interrupt
Channel 0 FIFO data reception	374	Interrupt is generated by storing reception data to FIFO.
Channel 1 FIFO data reception	378	Interrupt is generated by storing reception data to FIFO.

4.8 Constants and Error Codes

The major constants and error codes are listed below. Table 4.3 lists user-defined constants for the EnDat driver (r_endat_rzt2_config.h), Table 4.4 lists EnDat 2.2 mode commands, Table 4.5 lists transmission clock frequencies, Table 4.6 lists watchdog timer time units, Table 4.7 lists low-level period at the start of data transmission, Table 4.8 lists the MRS codes, and Table 4.9 lists parameters for FIFO data reception settings. The error code is given in Section 4.10.3(1), r_endat_err_t.

Table 4.3 User-defined Constants for the EnDat Driver (r_endat_rzt2_config.h)

Constant	Set value	Content
R_ENDAT_CABLE_DELAY	5	The number of times the propagation delay is automatically measured. Set it to 5 to 255 times.
R_ENDAT_ADD_NUM	0u	Number of additional information to receive

Table 4.4 EnDat 2.2 Mode Commands

Constant	Value	Content
R_ENDAT_POS	0x07u	“Encoder send position values” command
R_ENDAT_MEM	0x0Eu	“Selection of memory area” command
R_ENDAT_RX_PARAM	0x1Cu	“Encoder receive parameter” command
R_ENDAT_PARAM	0x23u	“Encoder send parameter” command
R_ENDAT_RESET	0x2Au	“Encoder receive reset” command
R_ENDAT_POS_ADD_DATA	0x38u	“Encoder send position values with additional data” command
R_ENDAT_POS_MEM	0x09u	“Encoder send position values and selection of the memory area” command
R_ENDAT_POS_RX_PARAM	0x1Bu	“Encoder send position values and receive parameter” command
R_ENDAT_POS_PARAM	0x24u	“Encoder send position values and send parameter” command
R_ENDAT_POS_RX_ERR_RESET	0x2Du	“Encoder send position values and receiver error reset” command

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

Table 4.5 Transmission Clock Frequencies

Constant	Value	Content
R_ENDAT_FTCLK_16670	0x3u	16.67 MHz *
R_ENDAT_FTCLK_8330	0x6u	8.33 MHz *
R_ENDAT_FTCLK_4160	0xBu	4.16 MHz *
R_ENDAT_FTCLK_4000	0x8u	4 MHz *
R_ENDAT_FTCLK_2000	0xCu	2 MHz
R_ENDAT_FTCLK_1000	0xDu	1 MHz
R_ENDAT_FTCLK_200	0xEu	0.2 MHz
R_ENDAT_FTCLK_100	0xFu	0.1 MHz

Note: Propagation delay compensation should be enabled (delay_comp=true) to use.

Table 4.6 Watchdog Timer Time Units

Constant	Value	Content
R_ENDAT_WD_RANGE_US	0x00u	Watchdog Timer time unit is microseconds
R_ENDAT_WD_RANGE_MS	0x80u	Watchdog Timer time unit is milliseconds

Table 4.7 Low-level Period at the Start of Data Transmission

Constant	Value	Content
R_ENDAT_TST_HALF_TCLK	0x00u	1/2 TCLK
R_ENDAT_TST_500NS	0x01u	0.5 us *
R_ENDAT_TST_1US	0x02u	1 us *
R_ENDAT_TST_1500NS	0x03u	1.5 us *
R_ENDAT_TST_2US	0x04u	2 us *
R_ENDAT_TST_4US	0x05u	4 us *
R_ENDAT_TST_8US	0x06u	8 us *
R_ENDAT_TST_10US	0x07u	10 us *

Note The low-level period has a margin of error. See the hardware manual for details.

Table 4.8 MRS Codes

Constant	Value	Content
R_ENDAT_MRS_INFO1_NOP	0x40u	Send additional info 1 without data contents (NOP)
R_ENDAT_MRS_DIA	0x41u	Send diagnostic values
R_ENDAT_MRS_POS2_LSB	0x42u	Send position value 2, word 1 LSB
R_ENDAT_MRS_POS2_CENTER	0x43u	Send position value 2, word 2
R_ENDAT_MRS_POS2_MSB	0x44u	Send position value 2, word 3 MSB
R_ENDAT_MRS_MEM_LSB	0x45u	Acknowledge memory content LSB
R_ENDAT_MRS_MEM_MSB	0x46u	Acknowledge memory content MSB
R_ENDAT_MRS_MRS_CODE	0x47u	Acknowledge MRS code
R_ENDAT_MRS_TEST_SMD	0x48u	Acknowledge test command
R_ENDAT_MRS_TEST_LSB	0x49u	Send test values, word 1 LSB
R_ENDAT_MRS_TEST_CENTER	0x4Au	Send test values, word 2
R_ENDAT_MRS_TEST_MSB	0x4Bu	Send test values, word 3 MSB
R_ENDAT_MRS_TEMP1	0x4Cu	Send temperature 1
R_ENDAT_MRS_TEMP2	0x4Du	Send temperature 2
R_ENDAT_MRS_ADD_SEN	0x4Eu	Additional sensors
R_ENDAT_MRS_NOT_INFO1	0x4Fu	Stop sending additional datum 1
R_ENDAT_MRS_INFO2_NOP	0x50u	Send additional datum 2 without data contents
R_ENDAT_MRS_COM	0x51u	Send commutation
R_ENDAT_MRS_ACC	0x52u	Send acceleration
R_ENDAT_MRS_COM_ACC	0x53u	Send commutation & acceleration
R_ENDAT_MRS_LIM_POS	0x54u	Send limit position signals
R_ENDAT_MRS_LIM_POS_ACC	0x55u	Send limit position signals & acceleration
R_ENDAT_MRS_ASY_POS_LSB	0x56u	Asynchronous position value, word 1 LSB
R_ENDAT_MRS_ASY_POS_CENTER	0x57u	Asynchronous position value, word 2
R_ENDAT_MRS_ASY_POS_MSB	0x58u	Asynchronous position value, word 3 MSB
R_ENDAT_MRS_OPE_STA_ERR	0x59u	Operating status error sources
R_ENDAT_MRS_TIM_STA	0x5Bu	Timestamp
R_ENDAT_MRS_NOT_INFO2	0x5Fu	Stop sending additional datum 2
R_ENDAT_MRS_OPE_STAT	0xB9u	Operating status
R_ENDAT_MRS_ENC_MANU1	0xA1u	Parameters of the encoder manufacturer 1
R_ENDAT_MRS_ENC_MANU2	0xA3u	Parameters of the encoder manufacturer 2
R_ENDAT_MRS_ENC_MANU3	0xA5u	Parameters of the encoder manufacturer 3
R_ENDAT_MRS_OPE_PARAM	0xA7u	Operating parameters
R_ENDAT_MRS_OEM1	0xA9u	Parameters of the OEM 1
R_ENDAT_MRS_OEM2	0xABu	Parameters of the OEM 2
R_ENDAT_MRS_OEM3	0xADu	Parameters of the OEM 3
R_ENDAT_MRS_OEM4	0xAFu	Parameters of the OEM 4
R_ENDAT_MRS_COMP_VAL1	0xB1u	Compensation Values of the encoder manufacturer 1
R_ENDAT_MRS_COMP_VAL2	0xB3u	Compensation Values of the encoder manufacturer 2
R_ENDAT_MRS_COMP_VAL3	0xB5u	Compensation Values of the encoder manufacturer 3
R_ENDAT_MRS_COMP_VAL4	0xB7u	Compensation Values of the encoder manufacturer 4
R_ENDAT_MRS_PARAM_ENDAT22	0xBDu	Parameters of the encoder manufacturer for EnDat 2.2
R_ENDAT_MRS_PARAM_SEC2	0xBFu	Parameters of the section 2 memory area
R_ENDAT_MRS_OPE_PARAM2	0xBBu	Operating parameters 2

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

Table 4.9 Parameters for FIFO data reception settings

Constant	Value	Content
R_ENDAT_FIFO_POS_PAETH	1u	FIFO receive threshold for POS commands
R_ENDAT_FIFO_POS_NORM	2u	Number of FIFO data read words for POS command
R_ENDAT_FIFO_PAETH	3u	FIFO receive threshold (other than POS command)
R_ENDAT_FIFO_NORM	4u	Number of words of FIFO data read (other than POS command)
R_ENDAT_FIFO_CLR	22u	Number of reads when clearing FIFO data

4.9 Fixed-Width Integer Types

Table 4.10 lists the fixed-width integers for the sample code. These fixed-width integers are defined in the standard libraries.

Table 4.10 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_endat_info_t

Initialization information of the EnDat control unit

```
typedef struct
{
    uint8_t             ftclk;                  Transmission clock frequency setting
                                                See "Table 4.5 Transmission Clock Frequencies". This
                                                setting is reflected in the FTCLK bit of the CFG1 register.
    bool                filter;                 Noise filter settings (true: enabled, false: disabled)
                                                This setting is reflected in the INF bit, NFINF bit, and
                                                NFSCNT bit of the NF register.
    bool                delay_comp;             Propagation delay correction (true: valid, false: invalid)
                                                This setting is reflected in the DLY bit of the CFG1
                                                register.
    uint8_t             tst;                    Set the Low period at the start of data transmission
                                                See "Table 4.7  Low-level Period at the Start of Data
                                                Transmission". This setting is reflected in the TST bit of
                                                the CFG2 register.
    endat_wait_cb_t     penc_init_reset_wait    A pointer to a callback function that generates the wait
                                                time after an encoder reset
                                                See "4.5.1 enc_init_reset_wait_callback" for details.
                                                Do not set NULL.
    endat_wait_cb_t     penc_init_mem_wait;     Pointer to callback function that generates wait time for
                                                encoder memory area selection timeout error detection.
                                                See "4.5.2 enc_init_mem_wait_callback" for details.
                                                Do not set NULL.
    endat_wait_cb_t     penc_init_pram_wait;    Encoder parameter Send / receive timeout Error
                                                detection function pointer to generate wait time
                                                See "4.5.3 enc_init_pram_wait_callback" for details.
                                                Do not set NULL.
    endat_wait_cb_t     penc_init_cable_wait;   A pointer to a function that produces a wait time for time-
                                                out error detection for cable propagation delay
                                                measurements. If propagation delay compensation is
                                                disabled (delay_comp = false), the setting can be
                                                omitted.
                                                See "4.5.4 enc_init_cable_wait_callback" for details.
                                                Do not set NULL when propagation delay compensation
                                                is enabled.
} r_endat_info_t
```

(2) r_endat_watchdog_t

Watchdog Timer setting time

```
typedef struct
{
    uint8_t     range;    Set the unit of time for the Watchdog Timer
                          See "Table 4.6  Watchdog Timer Time Units\"
    uint8_t     time;     Set the Watchdog Timer time
                          See "Table 4.11 Watchdog Timer Table "
} r_endat_watchdog_t
```

Table 4.11 Watchdog Timer Table

time	Time of the Watchdog Timer	
	range = R_ENDAT_WD_RANGE_US	range = R_ENDAT_WD_RANGE_MS
0	Stop	Stop
1	2 us	0.2 ms
2	4 us	0.4 ms
3	6 us	0.6 ms
:	:	:
10	20 us	2.0 ms
:	:	:
127	254 us	25.4 ms

Note: Except for the stop time, there is a margin of error. Refer to the hardware manual for details.

(3) r_endat_req_t

Request information to be sent to the EnDat2.2 compliant encoder. The mode command, MRS code, address and port address are combined and sent to the encoder. The combinations are shown in "Table 4.12 Mode Command Combination Table ".

```
typedef struct
{
```

uint8_t	mode_cmd;	EnDat 2.2 Mode Command See “Table 4.4 EnDat 2.2 Mode Commands”.
bool	dt;	Continuous mode setting (true: enabled, false: disabled) This setting is valid only if mode_cmd = R_ENDAT_POS.
uint8_t	mrs;	MRS code See “Table 4.8 MRS Codes”. The setting is valid only if the mode command combined with the MRS code in the “Table 4.12 Mode Command Combination Table” is designated as the mode_cmd.
uint8_t	addr;	Address (0x00 to 0xFF) The setting is valid only if the mode command combined with the address in the “Table 4.12 Mode Command Combination Table” is designated as the mode_cmd.
uint16_t	param_instruction;	Parameters to be written to memory area of the encoder The setting is valid only if the mode command combined with the parameters or block address in the “Table 4.12 Mode Command Combination Table” is designated as the mode_cmd.
r_endat_watchdog_t	watchdog;	Setting time of watchdog timer See 4.10.1(2) r_endat_watchdog_t. When sending a request for the following settings, set it to disabled (time =0). mode_cmd=R_ENDAT_POS and dt=true mode_cmd=R_ENDAT_RESET mode_cmd=R_ENDAT_RX_PARAM mode_cmd=R_ENDAT_PARAM This setting is reflected in the CFG2 register WDG bit.
bool	elc;	ELC mode setting (true: enabled, false: disabled) This setting is valid only if mode_cmd=R_ENDAT_POS and dt=false.
bool	fifo_enable	FIFO reception setting (true: enable, false: disable)
bool	rtcnt_enable	Recovery time measurement setting (true: enable, false: disable)
r_endat_isr_result_cb_t	pisr_result;	Pointer to a callback function that conveys the result of the request See “4.5.5 endat_callback” and “4.5.6 endat_poscon_callback” for details. Do not set NULL.
r_endat_isr_fifodt_cb_t	pisr_fifodt	Pointer to callback function to notify FIFO reception data See "4.5.7 endat_fifodt_callback.
r_endat_isr_rdst_cb_t	pisr_rdst;	Pointer to a callback function that conveys that the next data communication is ready. See “4.5.8 endat_rdst_callback” for detail. Do not set NULL.

```
} r_endat_req_t
```

Table 4.12 Mode Command Combination Table

mode_cmd	Command value	mrs / addr	param_instruction
R_ENDAT_POS	0x07u	--	--
R_ENDAT_MEM	0x0Eu	MRS Code	--
R_ENDAT_RX_PARAM	0x1Cu	Address	Parameters *1
R_ENDAT_PARAM	0x23u	Address	--
R_ENDAT_RESET	0x2Au	Address	--
R_ENDAT_POS_ADD_DATA	0x38u	--	--
R_ENDAT_POS_MEM	0x09u	MRS Code	Block address *2
R_ENDAT_POS_RX_PARAM	0x1Bu	Address	Parameters *1
R_ENDAT_POS_PARAM	0x24u	Address	--
R_ENDAT_POS_RX_ERR_RESET	0x2Du	Address	--

Note 1. Consider the setting value according to the address.
2. Use only when the MRS code is R_ENDAT_MRS_PARAM_SEC2

(4) r_endat_result_t

Send/receive results

```
typedef struct
{
    r_endat_req_err_t       result;     Results of sending and receiving requests
                                        See “4.10.3(3) r_endat_req_err_t”.
    r_endat_data_t          data;       Received data
                                        See “4.10.1(5) r_endat_data_t”.
    r_endat_status_t        status;     Encoder Status
                                        See “4.10.1(7) r_endat_status_t”.
} r_endat_result_t
```

(5) r_endat_data_t

Received data

```
typedef struct
{
    uint64_t        pos;        Received positional value or test value
                                The RXD1 bit in the RECV1L register is stored in the
                                lower 32 bits. The RXD1 bit in the RECV1U register is
                                stored in the upper 32 bits.
    uint32_t        add_datum1; Additional data 1
                                Stores the RXD3-bit value of the RECV3 register.
    uint32_t        add_datum2; Additional data 2
                                Stores the RXD2 bit value of the RECV2 register.
} r_endat_data_t
```

(6) r_endat_fifodt_t

FIFO Received data

```
typedef struct
{
    uint8_t     mode_cmd;                   EnDat 2.2 mode command
                                            See “Table 4.4 EnDat 2.2 Mode Commands” for
                                            details.
    uint16_t    rtcnt_pre;                  Recovery Time Count (before execution)
                                            The value of the RTCNT register is stored
                                            immediately before execution of the mode command.
    uint16_t    rtcnt;                      Recovery Time Count (when FIFO data is read)
                                            The value of the RTCNT register is stored
                                            immediately after FIFO data read. At the time this
                                            count value is stored, for continuing the recovery time,
                                            the count-up of the RTCNT register may not be
                                            completed. For the value after the count-up is
                                            completed, check the count value before mode
                                            command execution (rtcnt_pre) in the next FIFO
                                            received data.
                                            If the recovery time measurement is disabled
                                            (rtcnt_enable is false) or continuous mode is selected
                                            (dt is true) in the request information, count-up is not
                                            performed.
    uint32_t    fdat [R_ENDAT_FIFO_NORM]; FIFO reception data array Received data stored in the
                                            FIFO is read and stored; for the R_ENDAT_POS
                                            command, position information is stored, and for other
                                            position data acquisition commands, additional
                                            information is stored along with the position
                                            information.
} r_endat_fifodt_t
```

(7) r_endat_status_t

Encoder Status

```
typedef struct
{
    bool        busy;       Encoder internal memory status
                            (true: accessing, false: accessible)
    bool        rm;         Increment encoder origin status
                            (true: origin detection, false: origin undetected)
    bool        wrn;        Warning status inside the encoder
                            (true: with warning, false: no warning)
} r_endat_status_t
```

(8) r_endat_protocol_err_t

EnDat I/F and encoder error information

```
typedef struct
{
    bool        err1;       Error1 bit status (true: occurred, false: not occurred)
    bool        crc1;       CRC check error for positional value (true: occurred, false: not
                            occurred)
    bool        ftype1;     EnDat TYPE1 error (true: occurred, false: not occurred)
    bool        ftype2;     EnDat TYPE2 error (true: occurred, false: not occurred)
    bool        mrsadr;     Address error in EnDat TYPE2 error (true: occurred, false: not
                            occurred)
    bool        fifoerr;    FIFO overflow error (true: occurred, false: not occurred)
    bool        err2;       Error2 bit status (true: occurred, false: not occurred)
    bool        crc3;       CRC check error for Additional data 1 (true: occurred, false: not
                            occurred)
    bool        crc2;       CRC check error for Additional data 2 (true: occurred, false: not
                            occurred)
    bool        wdg;        Watchdog error (true: occurred, false: not occurred)
    bool        ftype3;     EnDat TYPE3 error (true: occurred, false: not occurred)
    bool        modeerr;    Mode command transmission error (true: occurred, false: not
                            occurred)
} r_endat_protocol_err_t
```

4.10.2 Unions

No unions are used.

4.10.3 Enumerated Types

(1) r_endat_err_t

Error codes of the encoder I/F

```
typedef enum
{
    ENDAT_SUCCESS           =0,   Normal termination
    ENDAT_ERR_INVALID_ARG         Argument error
    ENDAT_ERR_BUSY                API is not executable
    ENDAT_ERR_ACCESS              Error in the execution order of APIs
    ENDAT_ERR_DRV                 Internal error in driver
} r_endat_err_t
```

(2) r_endat_cmd_t

Command settings when the R_ENDAT_Control function is used.

```
typedef enum
{
    ENDAT_CMD_REQ                 Send command to the encoder
    ENDAT_CMD_POS_STOP            Stop continuous reception of positional values from the
                                  encoder
} r_endat_cmd_t
```

(3) r_endat_req_err_t

Result of sending and receiving requests

```
typedef enum
{
    ENDAT_REQ_SUCCESS   =0,       Normal completion of data transmission and reception
    ENDAT_REQ_ERR                 Data transmission/reception control error occurs
} r_endat_req_err_t
```

(4) r_endat_e2prom_dir_t

Direction of E2PROM read and write

```
typedef enum
{
    ENDAT_E2PROM_READ   ,         Read from memory
    ENDAT_E2PROM_WRITE  ,         Write to memory
} r_endat_e2prom_dir_t
```

4.11 Description of the Sample Program

4.11.1 Operation Outline

This sample program supports the EnDat 2.2 compliant functional safety-relevant encoder "ECN1123 FS".
This sample program performs the following processing.

1) Send requests input from the debugger's terminal I/O to the EnDat Safety encoder (ECN1123).
2) Display the data received from the EnDat Safety encoder (ECN1123).
3) Send and receive commands using the ELC event input trigger function of the EnDat I/F. (GPT events are linked as input events.)

(1) System Block Diagram

Figure 4.1 shows a block diagram of the system.

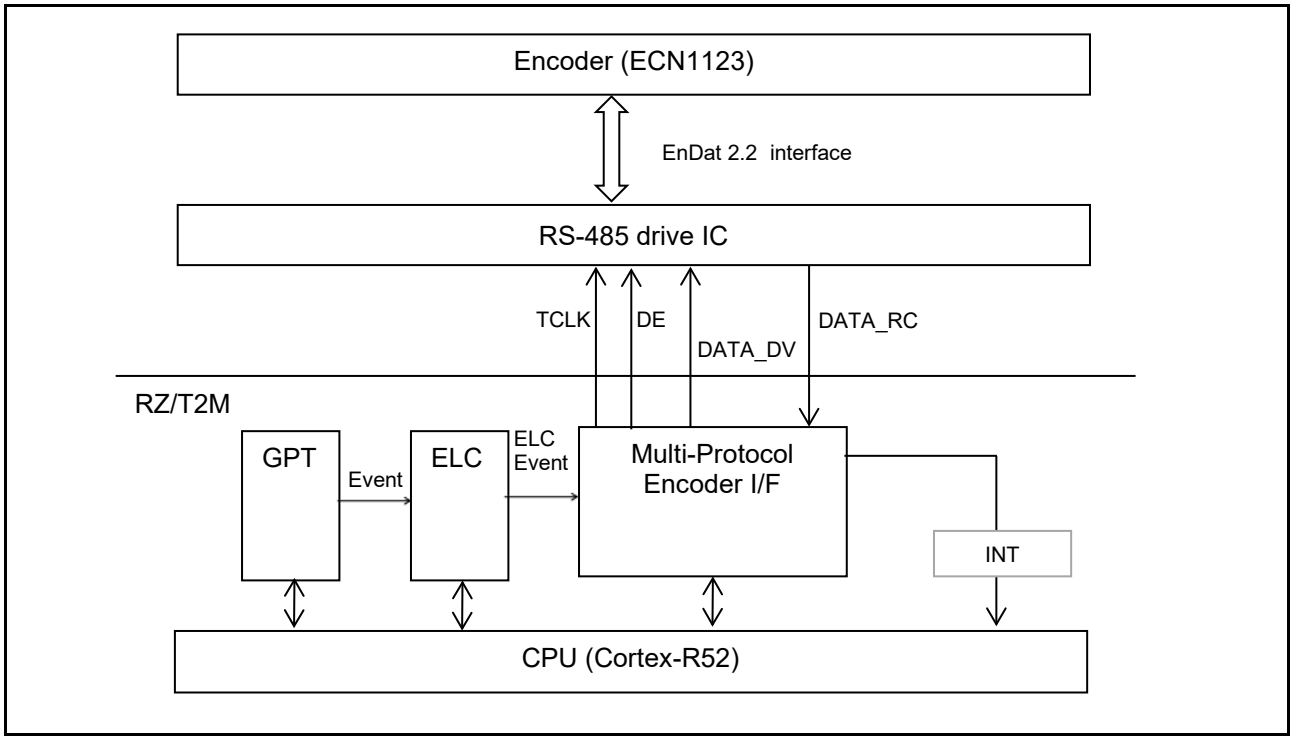

Figure 4.1 System Block Diagram

(2) Software Structure

Figure 4.2 shows the structure of the software.

The EnDat driver has a start processing part composed of the R_ENDAT_Open function, an end processing part composed of the R_ENDAT_Close function, a request sending part composed of the R_ENDAT_Control function, and a data receiving part (interrupt handler) composed of callback functions.

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

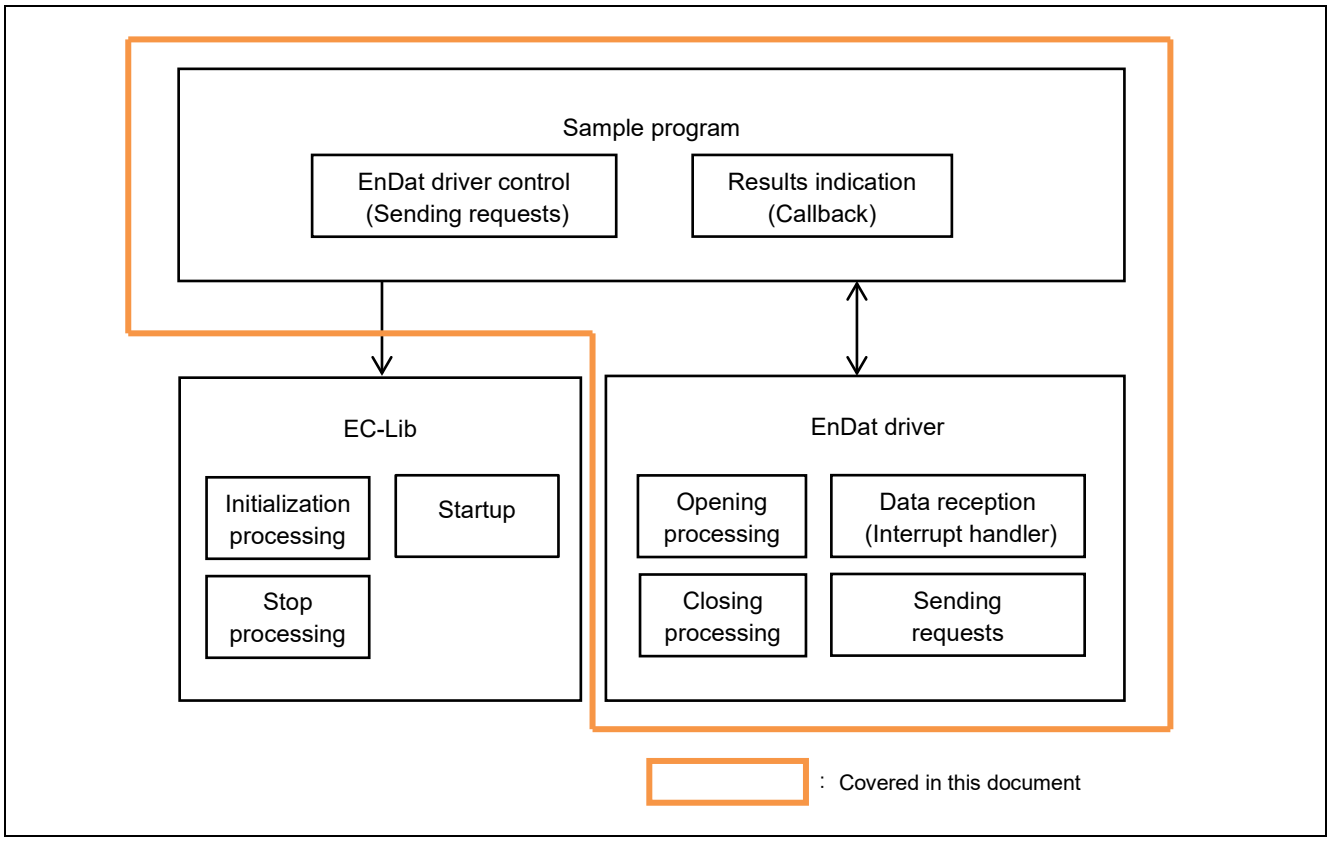

Figure 4.2 Software Structure

4.11.2 Sample Program Functions

Table 4.13 lists the major functions of the sample program.

Table 4.13 Major Functions of the Sample Program

Function Name	Page Number	
	Specification	Flowchart
hal_entry	31	-
enc_main	31	41
endat_cmd_control	31	42
endat_power_on_wait	31	-
enc_init_reset_wait_callback	32	-
enc_init_mem_wait_callback	32	-
enc_init_pram_wait_callback	32	-
enc_init_cable_wait_callback	32	-
endat_pos	33	43
endat_poscon	33	44
endat_elctimer	33	45
endat_stop	33	46
endat_temp	34	47
endat_read	34	48
endat_write	34	49
endat_spos	35	50
endat_pos_safe	35	51
endat_sel_info	36	52
endat_callback	36	53
endat_poscon_callback	36	54
endat_fifodt_callback	36	54
endat_rdst_callback	37	55
get_cmd	37	-
cmd_exit	37	-
result_display	37	-
result_display_param	38	
result_fifo_display	38	
timer_start	38	-
timer_stop	38	-

4.11.3 Specifications of Sample Program Functions

(1) hal_entry

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

(2) enc_main

enc_main		
Synopsis	Main function of the EnDat sample program	
Header	-	
Declaration	int32_t enc_main(uint8_t ch);	
Description	This is the main function of the EnDat 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
	Other than 0	: Abnormal termination (error code of the encoder IF driver)

(3) endat_cmd_control

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

(4) endat_power_on_wait

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

(5) enc_init_reset_wait_callback

enc_init_reset_wait_callback	
Synopsis	Waiting time generation function after encoder reset
Header	-
Declaration	static void enc_init_reset_wait_callback(void);
Description	This callback function generates a waiting time of 60 ms after the encoder reset process in the initialization process of the connected encoder. See “4.5.1 enc_init_reset_wait_callback”
Arguments	None
Return value	None

(6) enc_init_mem_wait_callback

enc_init_mem_wait_callback	
Synopsis	Waiting time generation function for encoder memory area selection process
Header	-
Declaration	static void enc_init_mem_wait_callback(void);
Description	This callback function generates a waiting time of 743 us for detecting a timeout error in the process of selecting a memory area in the initialization process of the connected encoder. See “4.5.2 enc_init_mem_wait_callback”
Arguments	None
Return value	None

(7) enc_init_pram_wait_callback

enc_init_pram_wait_callback	
Synopsis	Waiting time generation function for encoder parameter sending/receiving process
Header	-
Declaration	static void enc_init_pram_wait_callback(void);
Description	This callback function generates a waiting time of 13 ms for detecting a timeout error in the initialization process of the connected encoder, during which the encoder sends and receives parameters. See “4.5.3 enc_init_pram_wait_callback”
Arguments	None
Return value	None

(8) enc_init_cable_wait_callback

enc_init_cable_wait_callback	
Synopsis	Wait time generation function for encoder cable propagation delay d measurement process
Header	-
Declaration	static void enc_init_cable_wait_callback(void);
Description	Callback function to generate a waiting time of 588 us for detecting a timeout error in the process of measuring the cable propagation delay in the initialization process of the connected encoder. See “4.5.4 enc_init_cable_wait_callback”
Arguments	None
Return value	None

(9) endat_pos

endat_pos	
Synopsis	Function to get a positional value from the encoder
Header	-
Declaration	static void endat_pos(uint32_t arg_num, char_t *parg[]);
Description	This function is executed when the console command pos is entered. It acquires a positional value from the encoder.
Arguments	arg_num : Number of strings entered from the console (Not used) *parg[] : First address of string entered from console (Not used)
Return value	None

(10) endat_poscon

endat_poscon	
Synopsis	Function to get positional values continuously from the encoder
Header	-
Declaration	static void endat_poscon(uint32_t arg_num, char_t *parg[]);
Description	This function is executed when the console command poscon is entered to continuously acquire position values from the encoder using Continuous mode.
Argumetns	arg_num : Number of strings entered from the console (Not used) *parg[] : First address of string entered from console (Not used)
Return value	None

(11) endat_elctimer

endat_elctimer	
Synopsis	Function to get positional values continuously from the encoder synchronously with the ELC events
Header	-
Declaration	static void endat_elctimer(uint32_t arg_num, char_t *parg[]);
Description	This function is executed when the console command elctimer is entered. It acquires positional values continuously from the encoder synchronously with the ELC events.
Arguments	arg_num : Number of strings entered from the console *parg[] : First address of string entered from console
Return value	None

(12) endat_stop

endat_stop	
Synopsis	Function to stop continuous acquisition positional values from the encoder
Header	-
Declaration	static void endat_stop(uint32_t arg_num, char_t *parg[]);
Description	This function is executed when the console command stop is entered. It stops the continuous transmission of positional values from the encoder when it is operating in Continuous mode. While operating in ELC mode, this function cancels ELC mode operation and stops issuing continuous positional value acquisition commands. After the continuous positional value transmission from the encoder is stopped, the last 10 positional values are displayed.
Arguments	arg_num : Number of strings entered from the console (Not used) *parg[] : First address of string entered from console (Not used)
Return value	None

(13) endat_temp

endat_temp		
Synopsis	Function to get temperature information from the encoder	
Header	-	
Declaration	static void endat_temp(uint32_t arg_num, char_t *parg[]);	
Description	This function is executed when the console command temp is entered. It acquires temperature information from the encoder	
Arguments	arg_num	: Number of strings entered from the console (Not used)
	*parg[]	: First address of string entered from console (Not used)
Return value	None	

(14) endat_read

endat_read		
Synopsis	Function to get parameters from the encoder	
Header	-	
Declaration	static void endat_read(uint32_t arg_num, char_t *parg[]);	
Description	This function is executed when the console command read is entered. It reads parameters from the specified address of the encoder	
Arguments	arg_num	Number of strings entered from the console
	*parg[]	First address of the string entered from the console
Return value	None	

(15) endat_write

endat_write		
Synopsis	Function to write parameters to the encoder	
Header	-	
Declaration	static void endat_write(uint32_t arg_num, char_t *parg[]);	
Description	This function is executed when the console command write is input. It writes parameters to the specified address of the encoder.	
Arguments	arg_num	Number of strings entered from the console
	*parg[]	First address of the string entered from the console
Return value	None	

(16) endat_spos

endat_spos		
Synopsis	Function to obtain position values from the encoder along with Safety specification data	
Header	-	
Declaration	static void endat_spos(uint32_t arg_num, char_t *parg[]);	
Description	This function is executed when the console command spos is entered. The following processing is performed. 1. Select Position 2 word 1 for Additional Information 1 2. Acquire additional information with position value (recovery time measurement) 3. Select Position 2 word 2 for Additional Information 1 4. Obtain additional information with position value 5. Select Position 2 word 3 for Additional Information 1 6. Obtain additional information with position value 7. Select NOP (no data) for Additional Information 1 8. Obtain additional information with position value	
Arguments	arg_num	Number of strings entered from console (unused)
	*parg[]	First address of string entered from console (not used)
Return value	None	

(17) endat_pos_safe

endat_pos_safe		
Synopsis	Function to obtain additional information with position value from the encoder	
Header	-	
Declaration	static void endat_pos_safe(uint32_t arg_num, char_t *parg[]);	
Description	This function is executed when the console command pos_safe is entered. It obtains the selected additional information with position value.	
Arguments	arg_num	Number of strings entered from the console
	*parg[]	First address of the string entered from the console
Return value	None	

(18) endat_sel_info

endat_sel_info	
Synopsis	Function to select additional information of the encoder
Header	-
Declaration	static void endat_sel_info(uint32_t arg_num, char_t *parg[]);
Description	This function is executed when the console command sel_info is input. It selects information to be acquired as Additional Information 1 or Additional Information 2.
Arguments	arg_num Number of strings entered from the console *parg[] First address of the string entered from the console
Return value	None

(19) endat_callback

endat_callback	
Synopsis	Callback function that conveys the result of the request transmission to the encoder
Header	-
Declaration	static void endat_callback(r_endat_result_t *presult, r_endat_protocol_err_t *36er);
Description	This function stores the result in memory.
Arguments	presult : result of the request transmission 36er : EnDat I/F and encoder error information
Return value	None

(20) endat_poscon_callback

endat_poscon_callback	
Synopsis	Callback function that conveys the result of the request transmission to the encoder
Header	-
Declaration	static void endat_poscon_callback (r_endat_result_t *presult, r_endat_protocol_err_t *36er);
Description	This function stores the continuously acquired results in memory
Arguments	presult : result of the request transmission perr : EnDat I/F and encoder error information
Return value	None

(21) endat_fifodt_callback

endat_fifodt_callback	
Synopsis	Callback function for FIFO data reception result notification
Header	-
Declaration	static void endat_fifodt_callback(r_endat_fifodt_t *pfdat);
Description	This function stores the pointer to the acquired FIFO data.
Arguments	None
Return value	None

(22) endat_rdst_callback

enc_init_cable_wait_callback	
Synopsis	Callback function to notify that the next data communication is ready to start
Header	-
Declaration	static void endat_rdst_callback(void);
Description	Notifies that data reception is complete and the next data communication is ready; called each time data reception is completed while operating in Continuous or ELC mode.
Arguments	
Return value	None

(23) get_cmd

get_cmd	
Synopsis	Function to get commands from the console
Header	-
Declaration	static uint32_t get_cmd(char_t *parg[], const uint32_t arg_max);
Description	This function gets the command from the console
Arguments	parg : 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

(24) cmd_exit

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

(25) result_display

result_display	
Synopsis	Function to display the result of data reception
Header	-
Declaration	static void result_display(r_endat_result_t *presult, r_endat_protocol_err_t *perr);
Description	This function indicates the result of data reception in response to a request sent to the EnDat.
Arguments	presult : result of the request transmission perr : EnDat I/F and encoder error information
Return value	None

(26) result_display_param

result_display_param	
Synopsis	Function to display result of parameter transmission or reception
Header	-
Declaration	static void result_display_param(r_endat_result_t *presult, r_endat_protocol_err_t *perr);
Description	This function displays the result of parameter transmission or reception.
Arguments	presult : result of parameter transmission or reception perr : EnDat I/F and encoder error information
Return value	None

(27) result_fifo_display

result_fifo_display	
Synopsis	Function to display the result of FIFO data reception
Header	-
Declaration	static void result_fifo_display(r_endat_result_t *presult);
Description	This function displays the received FIFO data when a request is successfully sent and received to the encoder.
Arguments	presult : Result of received FIFO data
Return value	None

(28) 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

(29) 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 4.14 lists the major static type variables. Const type variables are not used.

Table 4.14 Major Static Variables

Type	Variable Name	Description	Function to be used
bool	endat_flag	Transmission completion flag (true: transmission completed, false: transmission is in progress)	endat_pos endat_poscon endat_elctimer endat_stop endat_temp endat_read endat_write endat_spos endat_pos_safe endat_sel_info endat_callback endat_rdst_callback
bool	endat_elc_flg	ELC mode operating flag (true: operating in ELC mode, false: not operating in ELC mode)	endat_pos endat_poscon endat_elctimer endat_stop
r_endat_result_t	*pendat_result	Address containing data acquisition results	endat_pos endat_temp endat_read endat_write endat_spos endat_pos_safe endat_sel_info endat_callback
r_endat_protocol_err_t	*pendat_err	Address containing error information	endat_pos endat_temp endat_read endat_write endat_spos endat_pos_safe endat_sel_info endat_callback
r_endat_fifodt_t	*pendat_result_fifo	Address containing FIFO data acquisition results	endat_fifodt_callback result_fifo_display
r_endat_req_err_t	poscon_err[ENDAT_POS_NUM]	Errors in continuously acquired positional values An array with 10 elements is used as a ring buffer to store the results of the latest 10 acquisitions.	endat_poscon endat_elctimer endat_stop endat_poscon_callback
uint64_t	poscon[ENDAT_POS_NUM]	Continuously acquired positional values An array with 10 elements is used as a ring buffer to store the results of the latest 10 acquisitions.	endat_poscon endat_elctimer endat_stop endat_poscon_callback
uint8_t	poscon_valid	Number of valid elements in poscon, poscon_err array Indicates the number of valid elements of positional values stored in the array.	endat_poscon endat_elctimer endat_stop endat_poscon_callback

Type	Variable Name	Description	Function to be used
uint8_t	poscon_num	Index of update position for poscon and poscon_err arrays The indexed elements are replaced by the next acquired position values.	endat_poscon endat_elctimer endat_stop endat_poscon_callback
bool	poscon_empty	Space information in poscon and poscon_err arrays (true: has space, false: has no space)	endat_poscon endat_elctimer endat_poscon_callback
int32_t	cur_id	Used EnDat I/F driver ID	enc_main endat_cmd_control endat_pos endat_pocson endat_elctimer endat_stop endat_temp endat_read endat_write endat_spos endat_pos_safe endat_sel_info

4.11.5 Constants of Sample Program

Table 4.15 Major constants lists the major constants used in the sample program.

Table 4.15 Major constants

Constants	Value	Content
ENDAT_ENC_TSAT_WAIT	1300u	Standby time after power-on (1.3 s)
ENDAT_ENC_100US_WAIT	100u	Waiting time after EC-Lib startup (100 us)
ENDAT_ENC_INIT_RESET_WAIT	60u	Time to wait after encoder reset process (60 ms)
ENDAT_ENC_INIT_MEM_WAIT	743u	Waiting time for detection of timeout errors in the process of selecting memory area in encoder initialization (743 us)
ENDAT_ENC_INIT_PRAM_WAIT	13u	Waiting time for detection of timeout errors in the process of sending and receiving parameters in encoder initialization. (13 ms)
ENDAT_ENC_INIT_CABLE_WAIT	588u	Waiting time for detection of timeout errors in the process of measuring cable propagation delay in encoder initialization. (588 us)
ENDAT_WDG_MAX	127u	Maximum watchdog timer setting value
ENDAT_POS_NUM	10u	Number of elements of the array for storing continuously received position values
ENDAT_TEMP_SCA_FAC	0.1	Number of elements of the array for storing continuously received position values
ENDAT_TEMP_ABS_ZERO	273.2	Constant for temperature data unit conversion

4.11.6 Flowchart of Main Processing

(1) Flowchart of enc_main

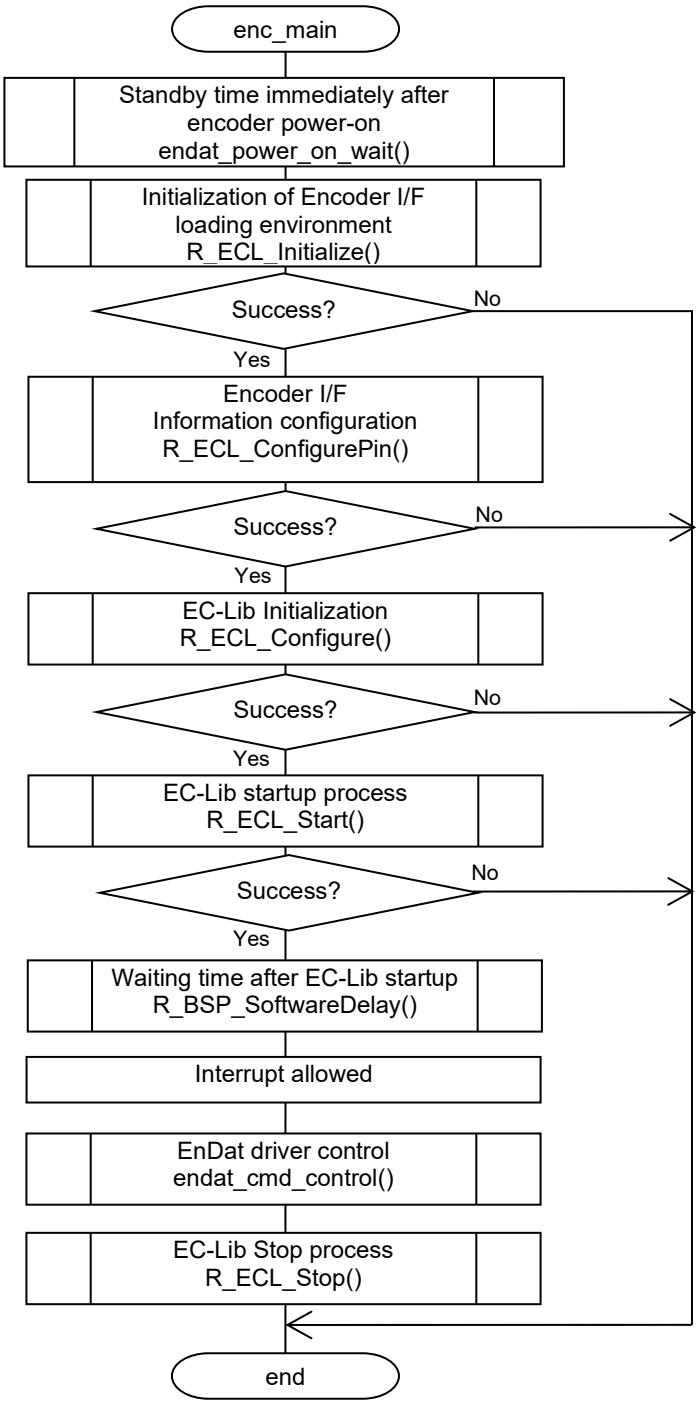

Figure 4.3 Flowchart of enc_main Function

(2) Flowchart of endat_cmd_control

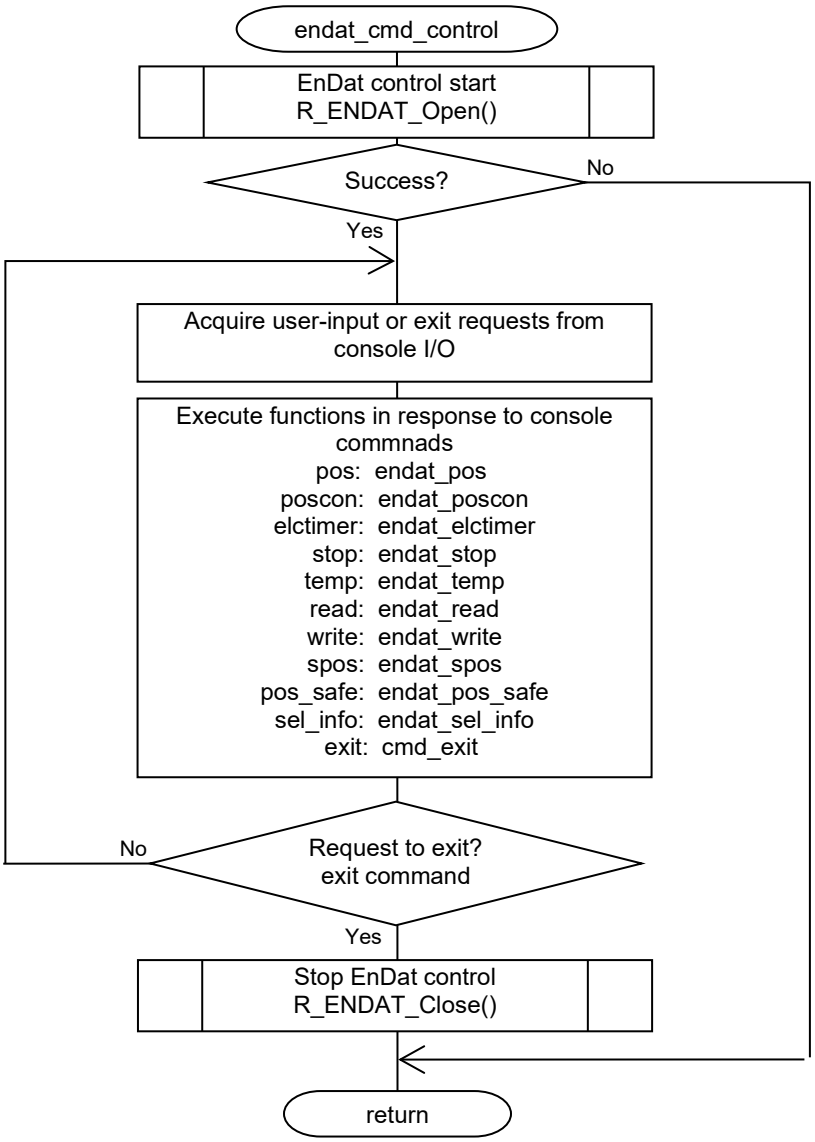

Figure 4.4 Flowchart of endat_cmd_control Function

(3) Flowchart of endat_pos

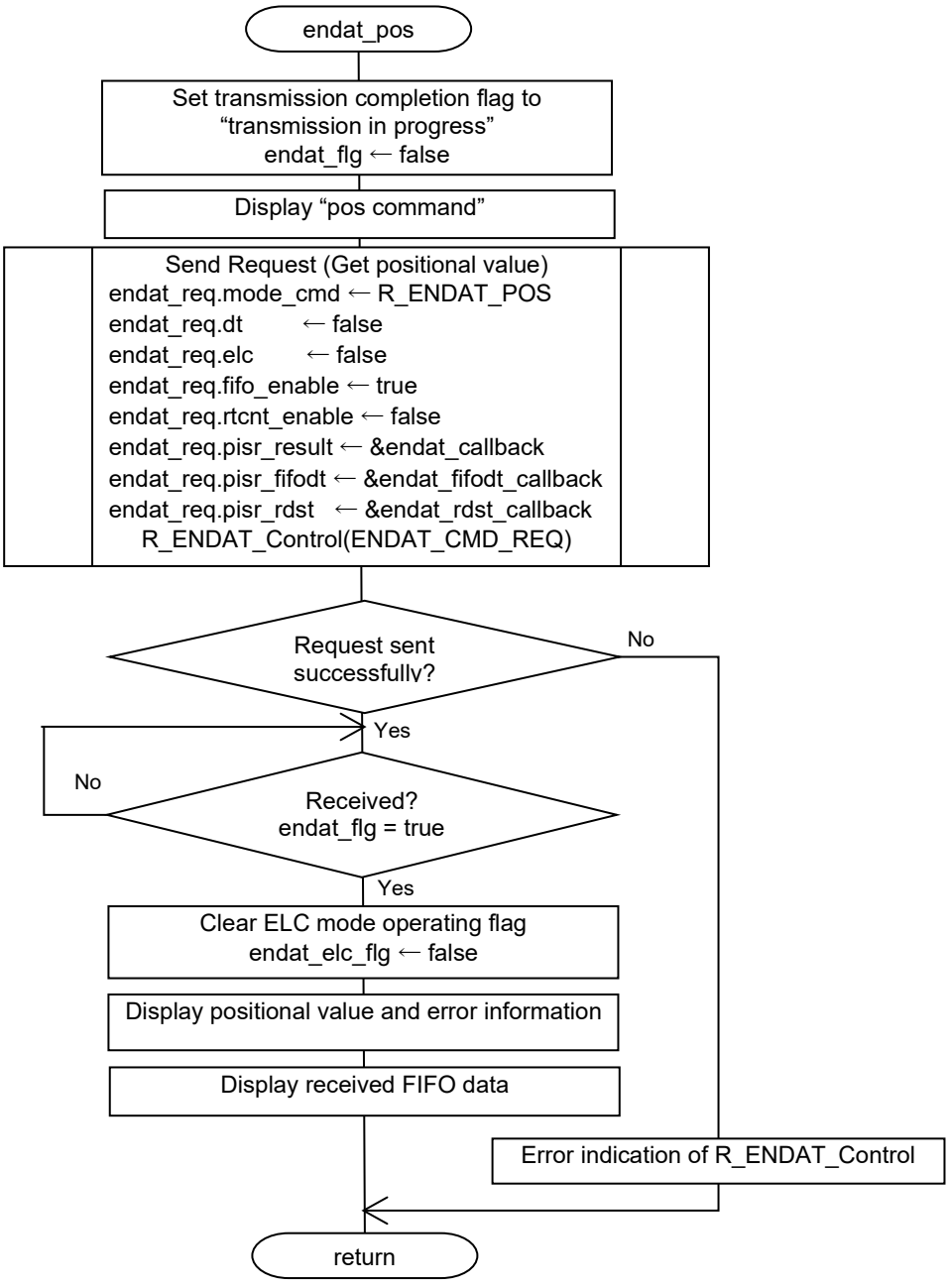

Figure 4.5 Flowchart of endat_pos Function

(4) Flowchart of endat_poscon

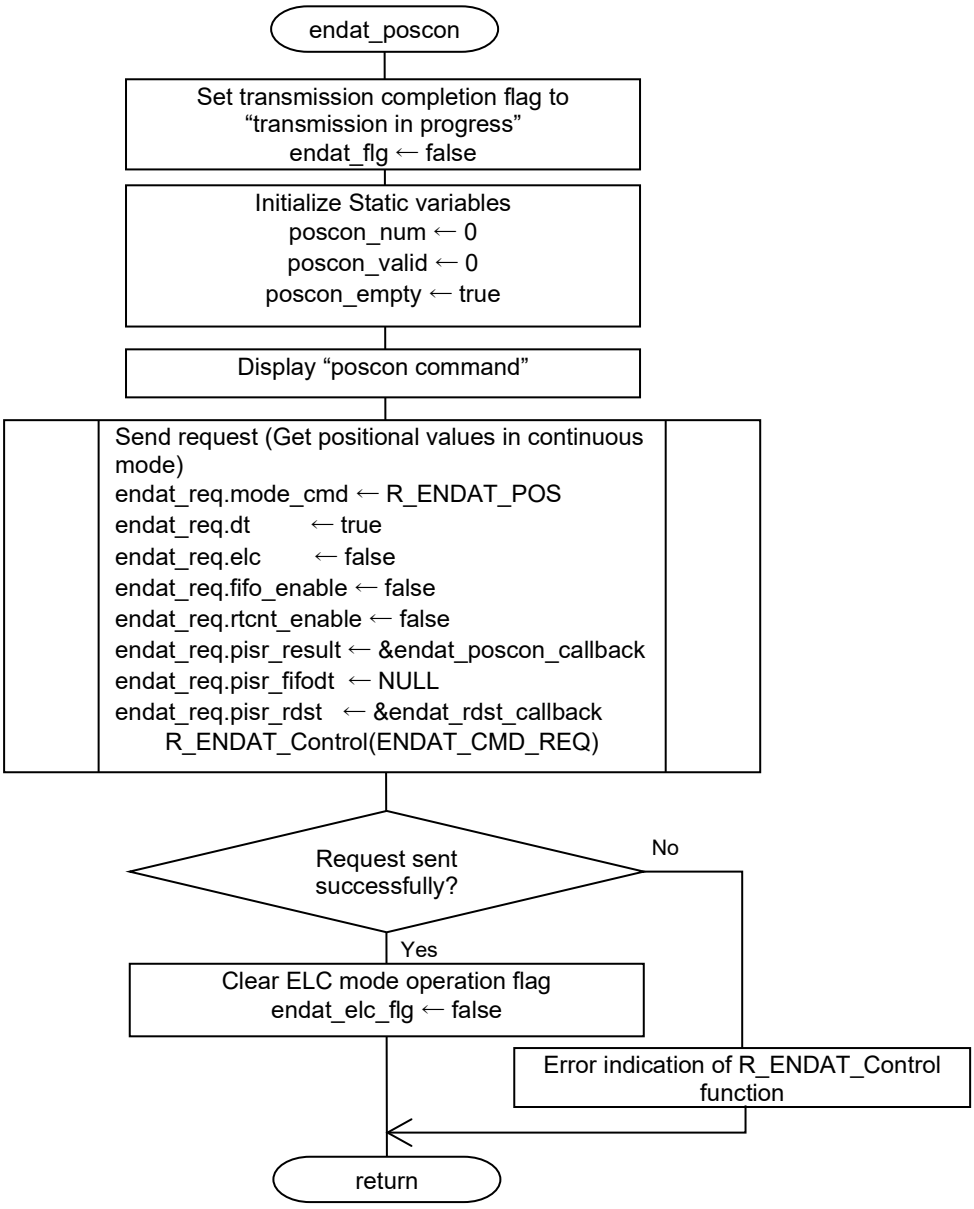

Figure 4.6 Flowchart of endat_poscon Function

(5) Flowchart of endat_elctimer

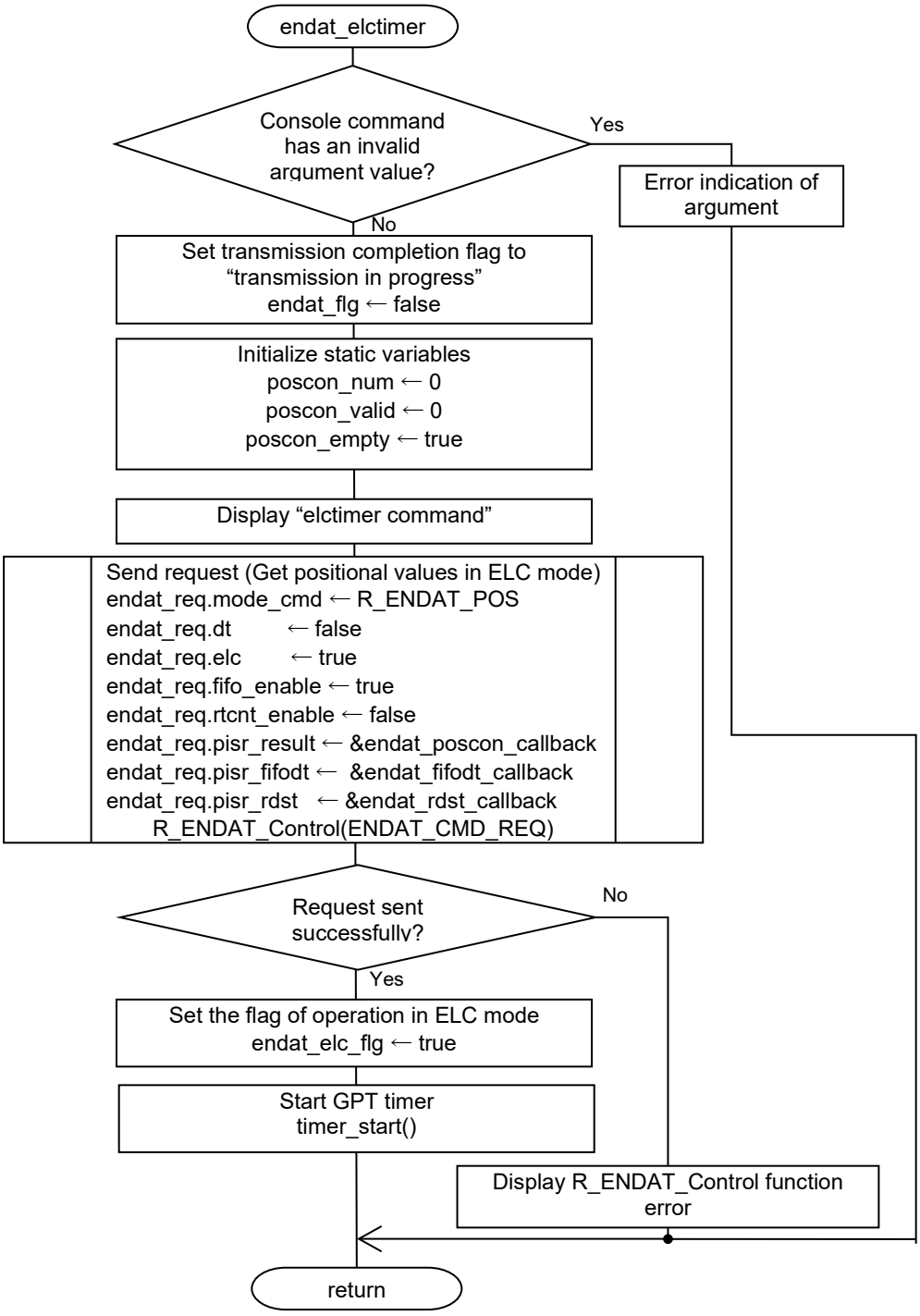

Figure 4.7 Flowchart of endat_elctimer Function

(6) Flowchart of endat_stop

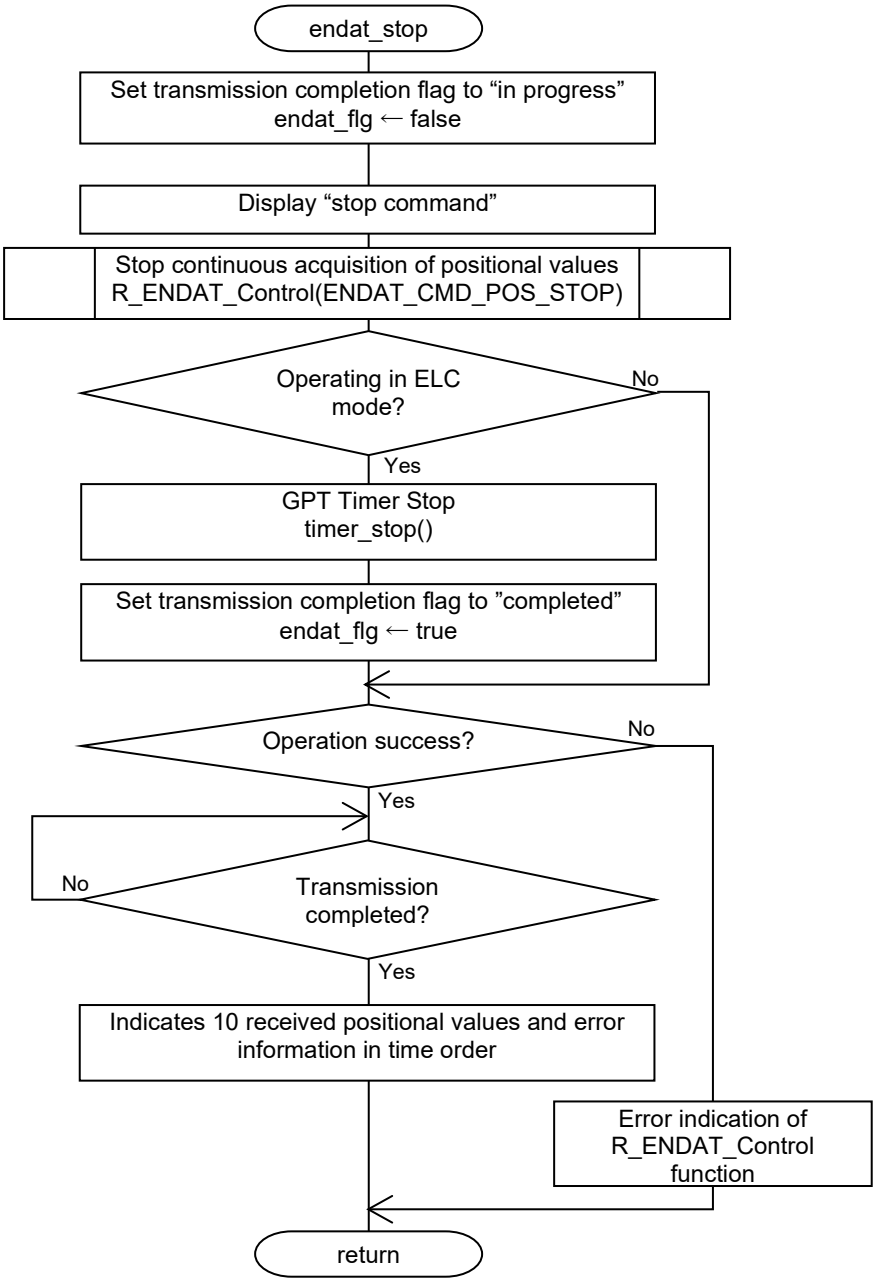

Figure 4.8 Flowchart of endat_stop Function

(7) Flowchart of endat_temp

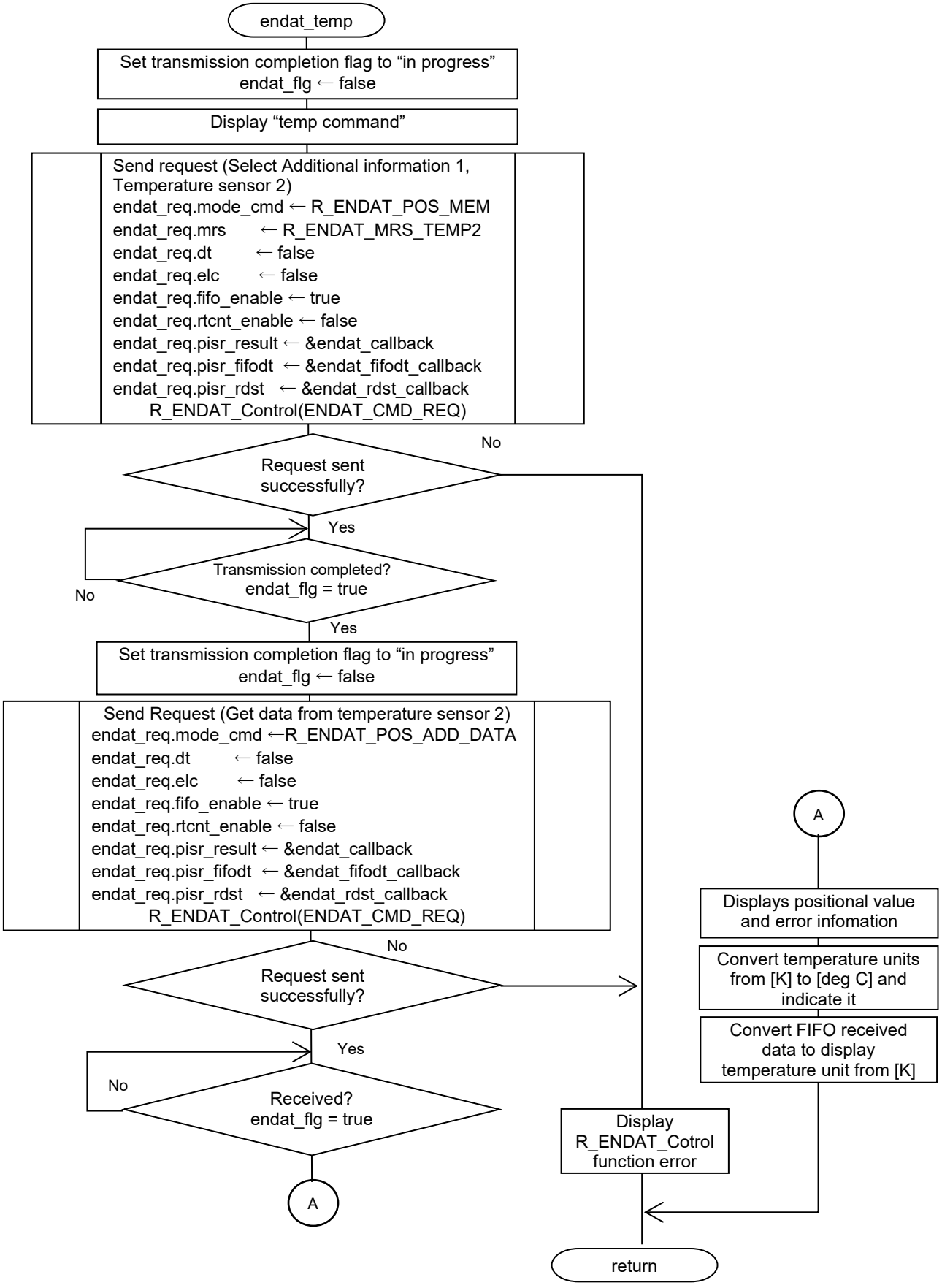

Figure 4.9 Flowchart of endat_temp Function

(8) Flowchart of endat_read

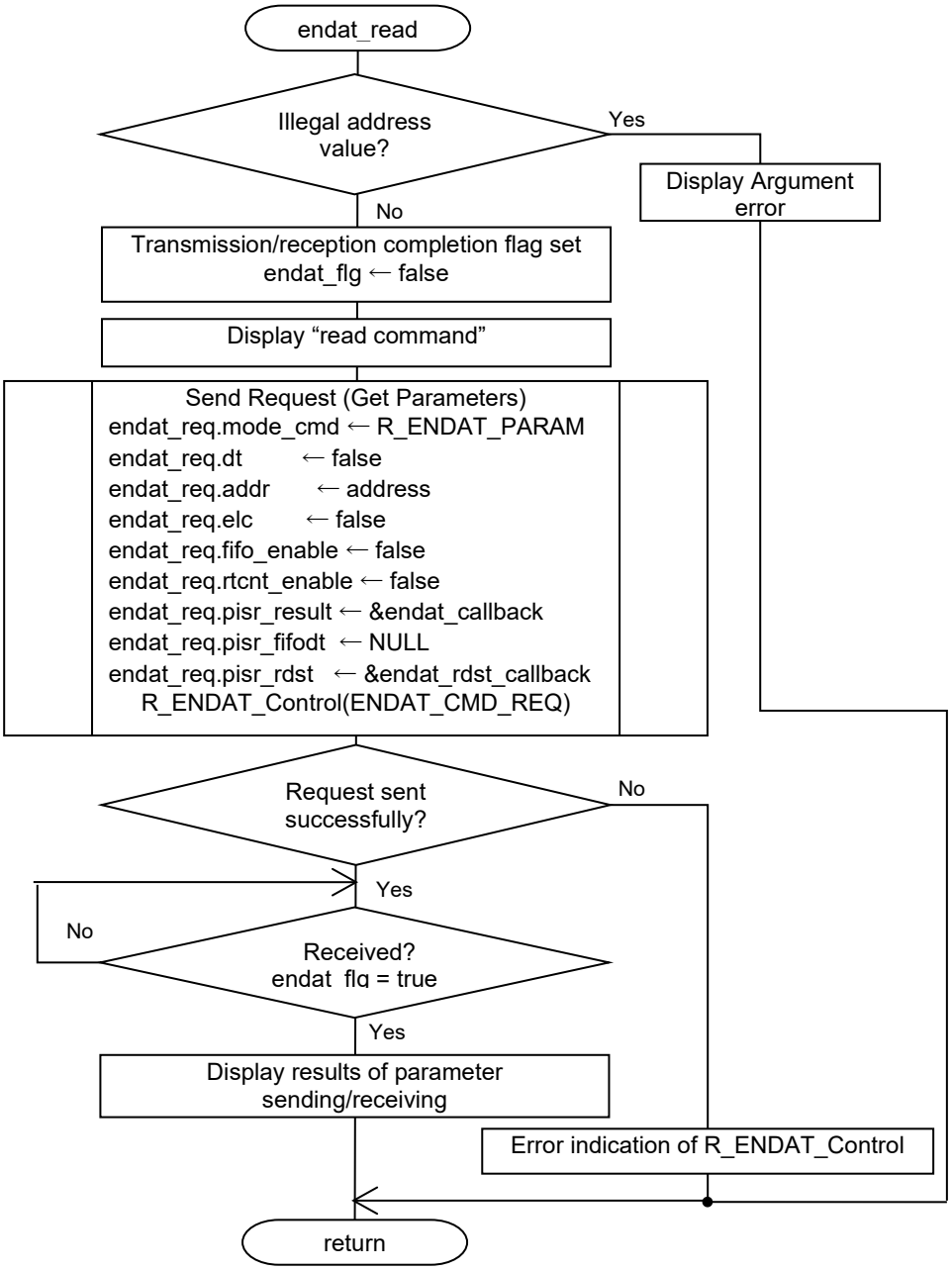

Figure 4.10 Flowchart of endat_read

(9) Flowchart endat_write

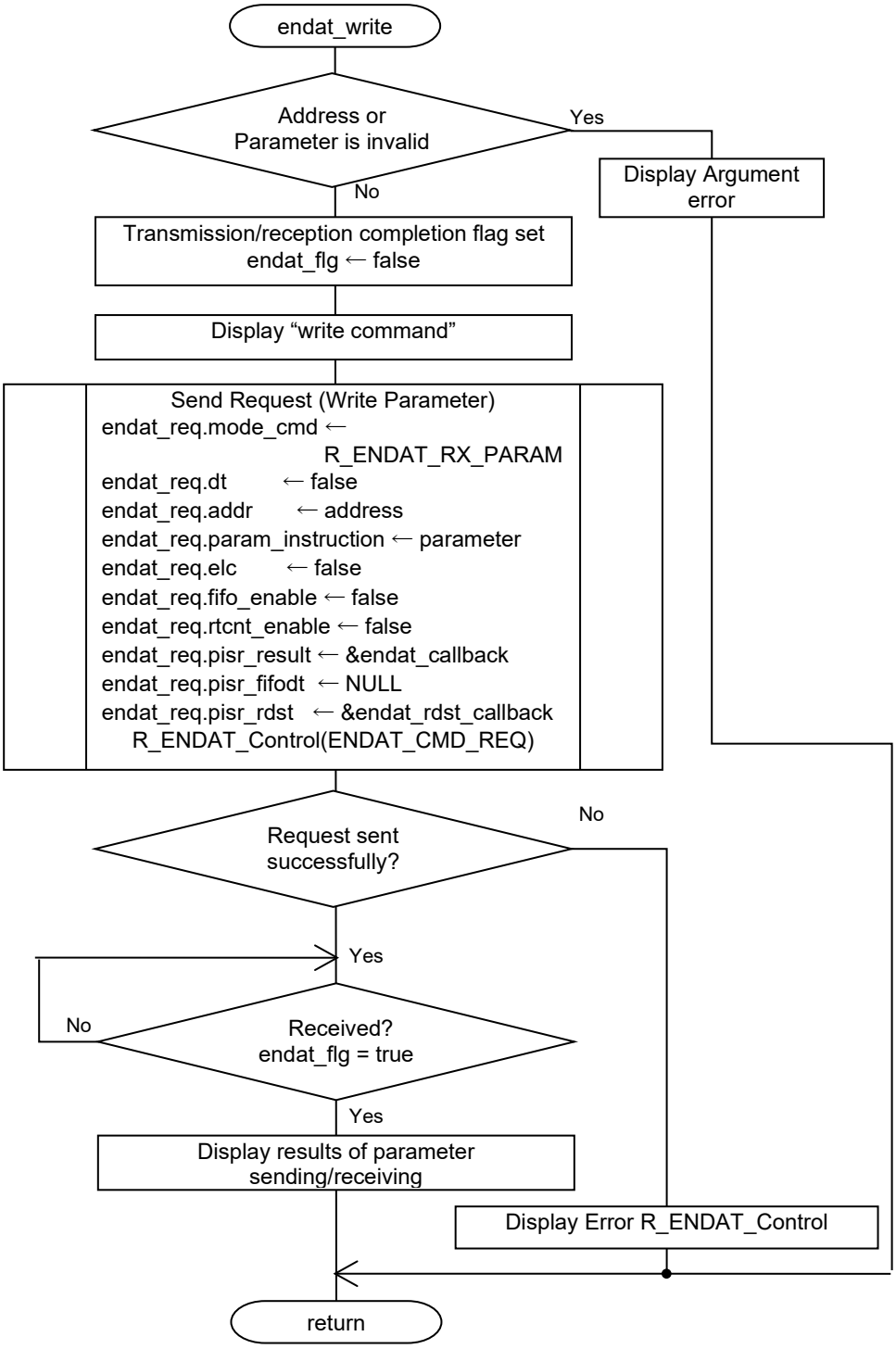

Figure 4.11 Flowchart of endat_write

(10) Flowchart endat_spos

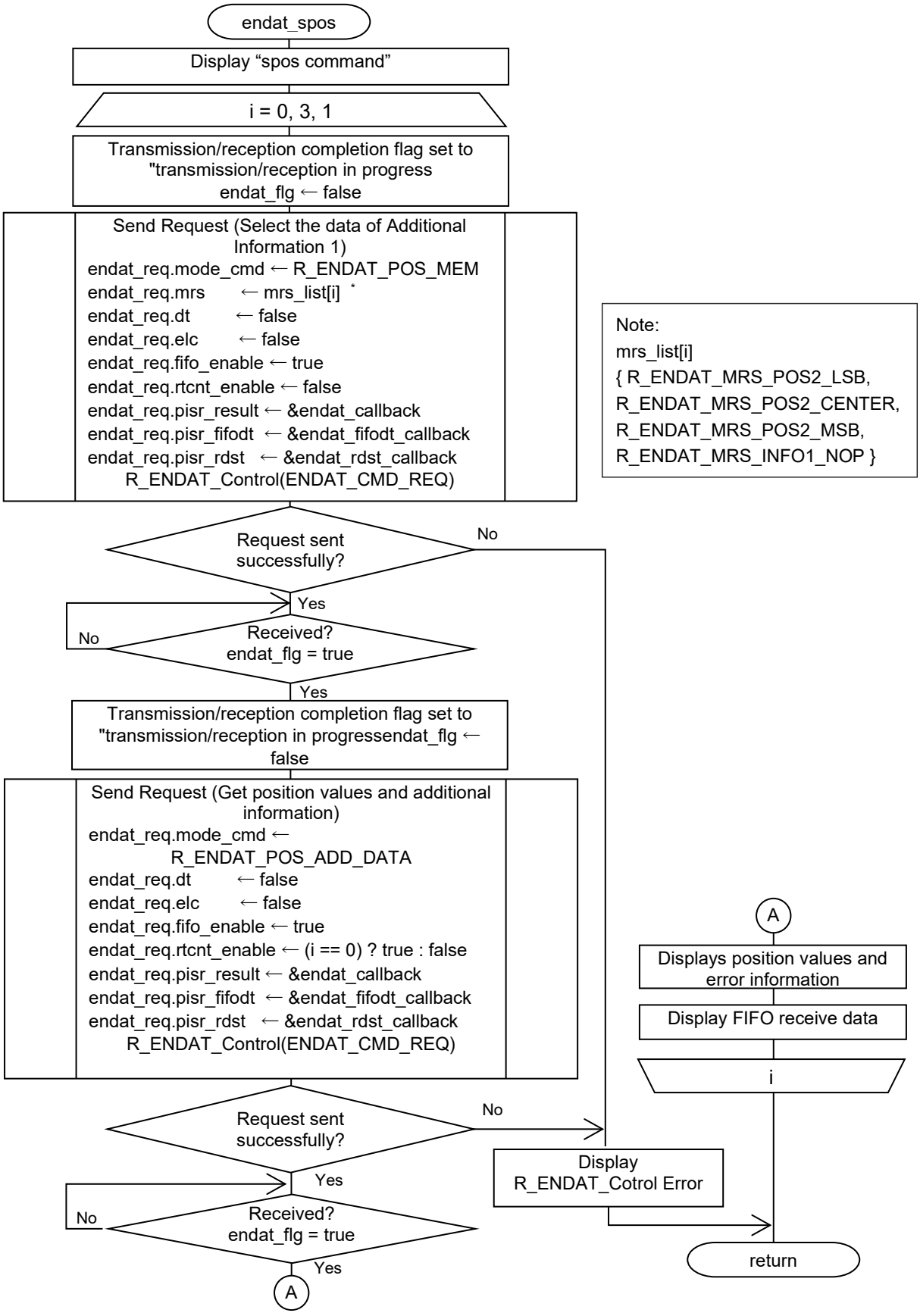

Figure 4.12 Flowchart endat_spos

(11) Flowchart endat_pos_safe

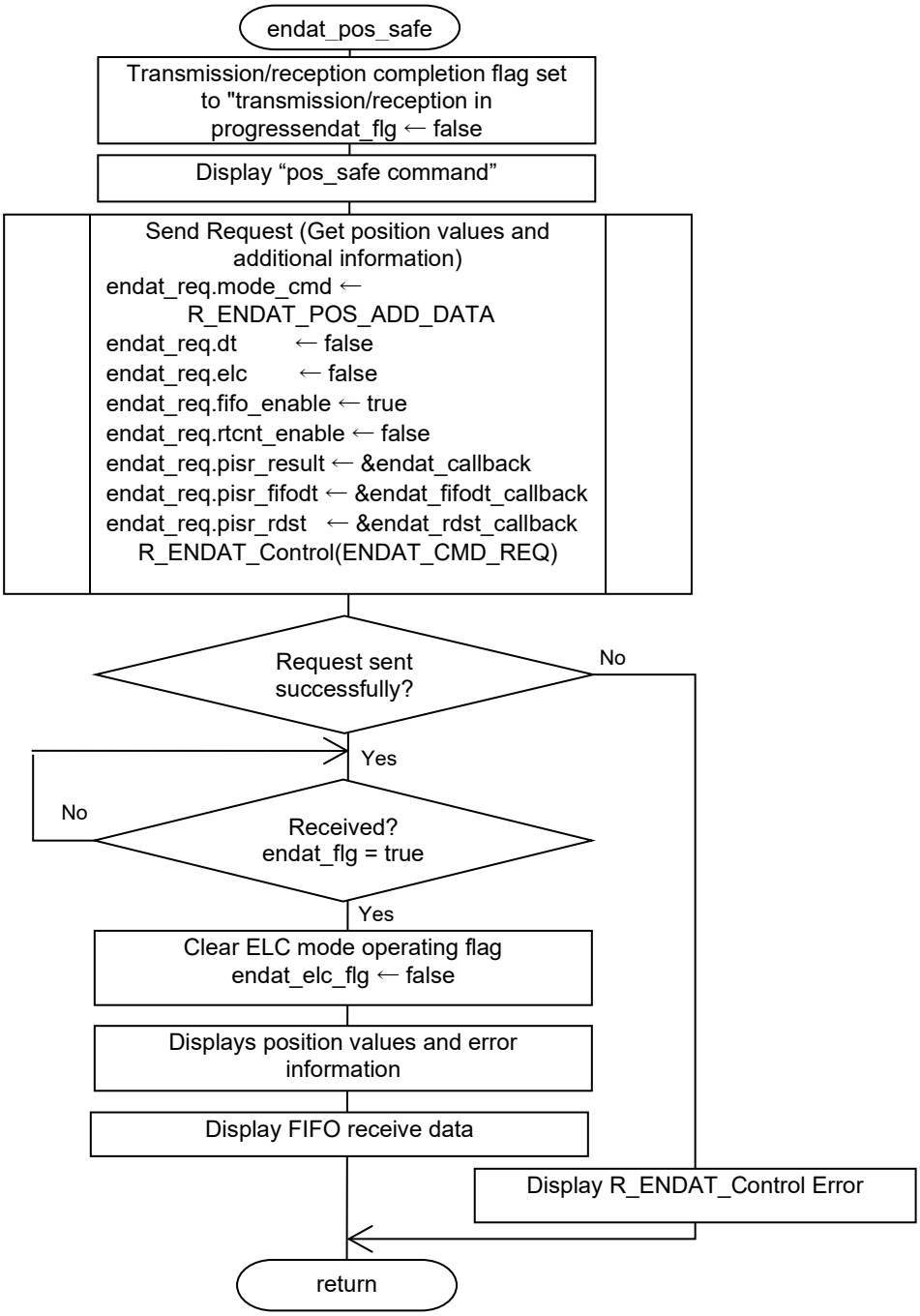

Figure 4.13 Flowchart endat_pos_safe

(12) Flowchart endat_sel_info

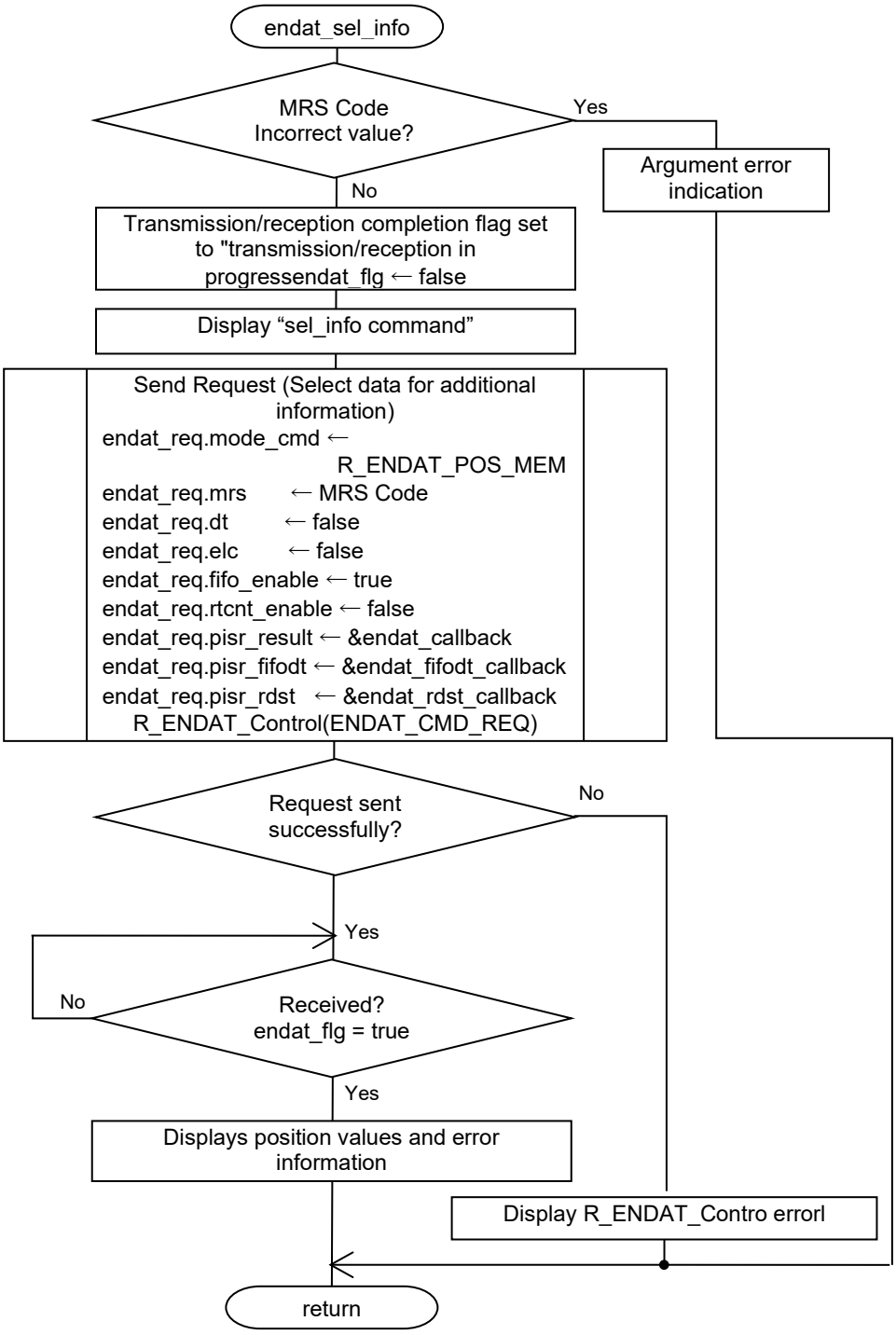

Figure 4.14 Flowchart endat_sel_info

(13) Flowchart of endat_callback

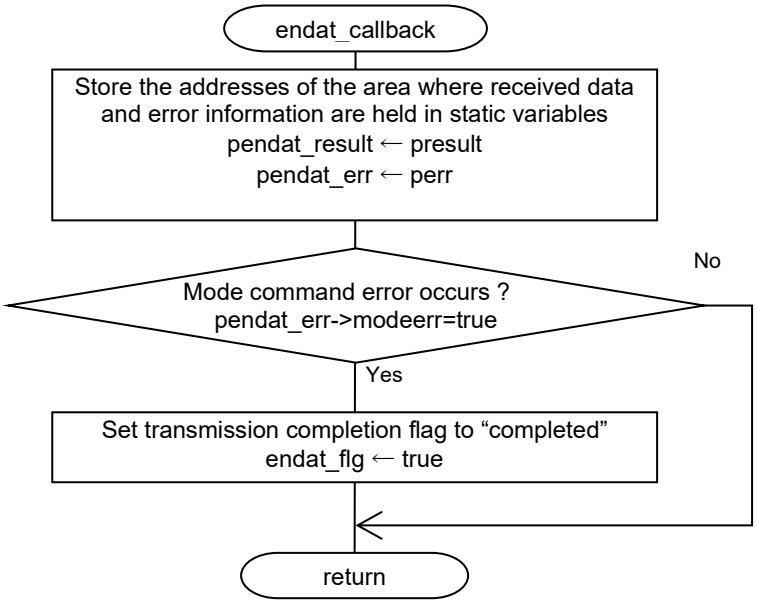

Figure 4.15 Flowchart endat_callback Function

(14) Flowchart of endat_poscon_callback

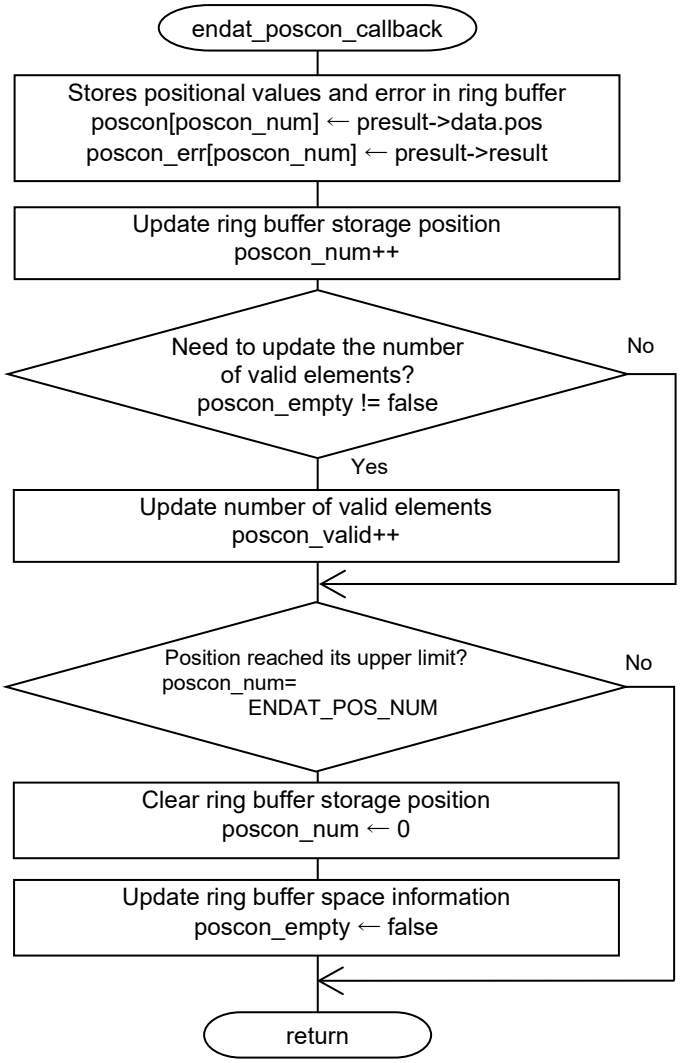

Figure 4.16 Flowchart endat_poscon_callback Function

(15) Flowchart endat_fifodt_callback

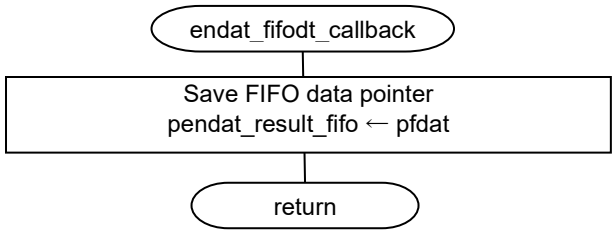

Figure 4.17 Flowchart endat_fifodt_callback

(16) Flowchart of endat_rdst_callback

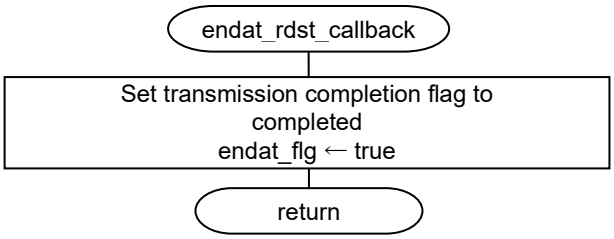

Figure 4.18 Flowchart of endat_rdst_callback Function

4.11.7 Operation Sequence

(1) Startup Sequence

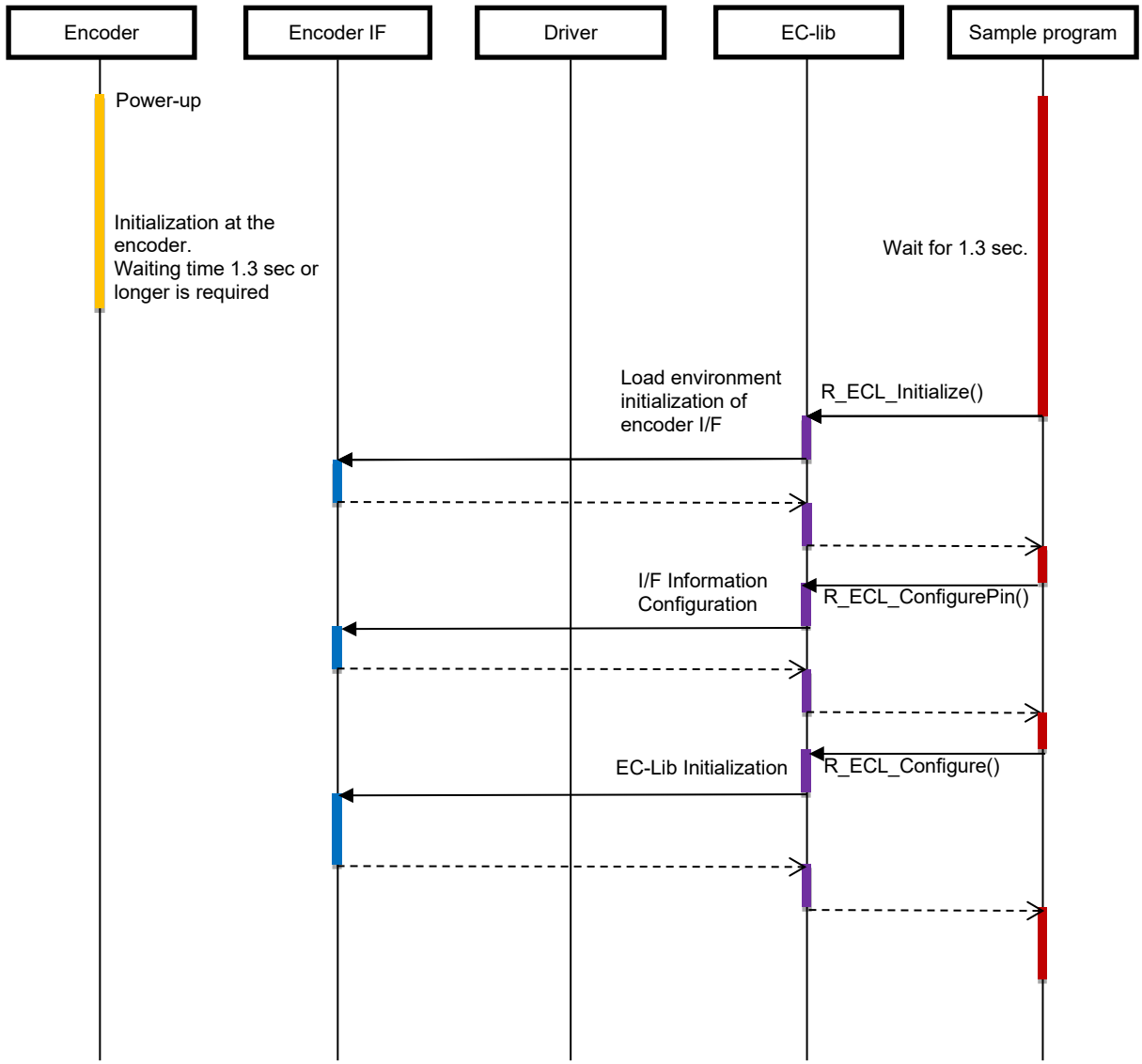

Figure 4.19 Startup Sequence Diagram

(2) Start Sequence

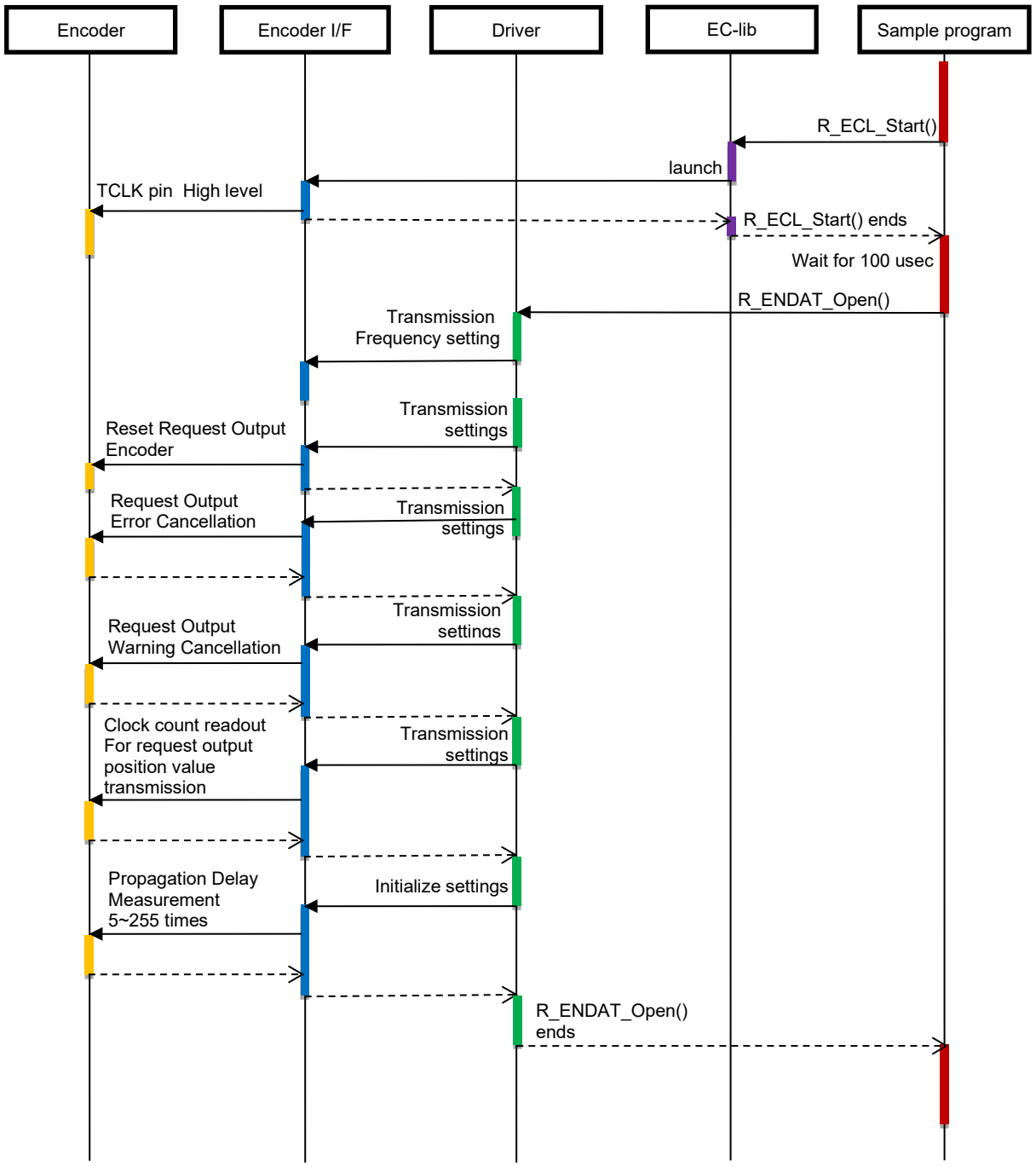

Figure 4.20 Start Sequence Diagram

(3) Sequences of Request Transmission and Data Reception

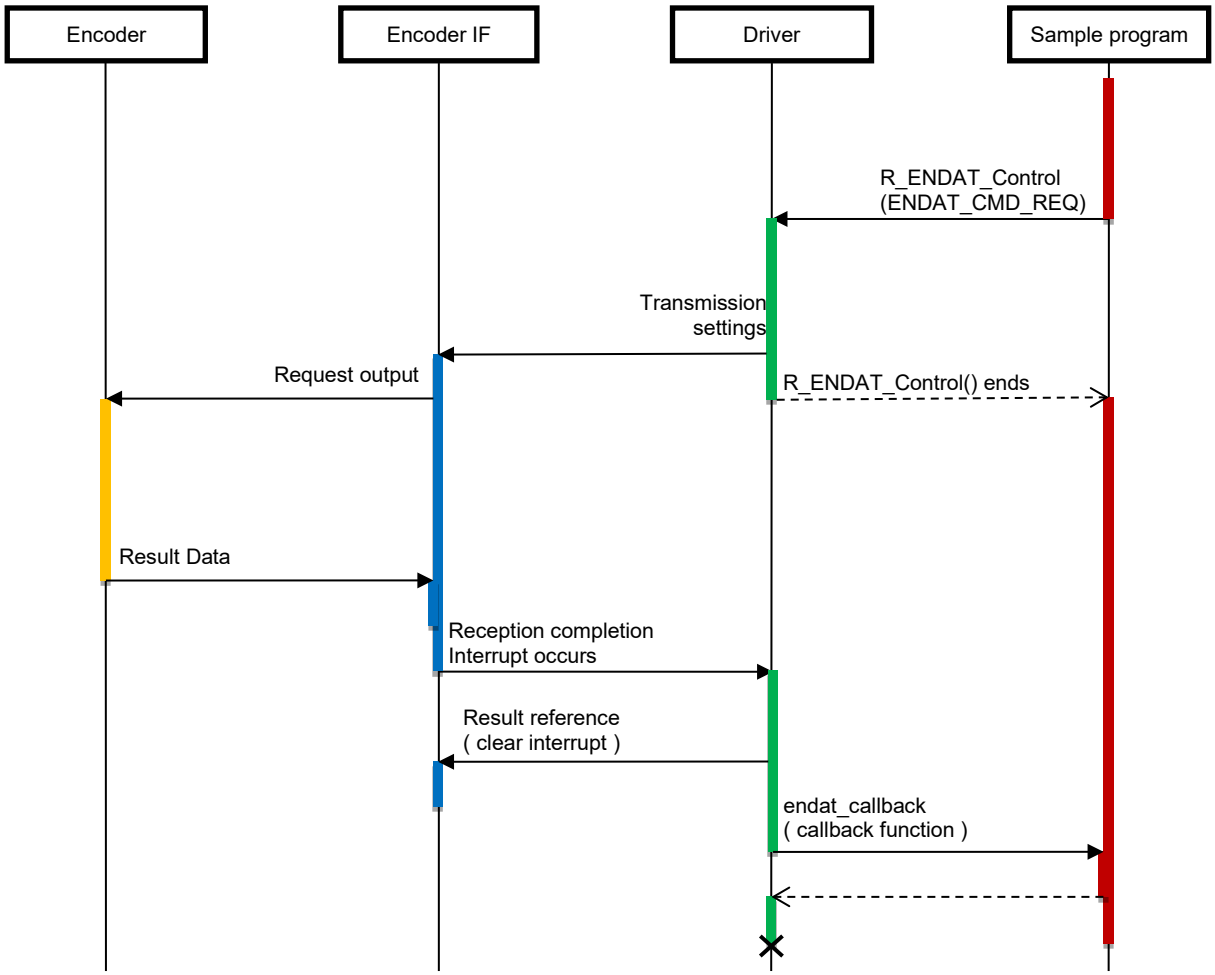

Figure 4.21 Sequences of Request Transmission and Data Reception Diagram

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

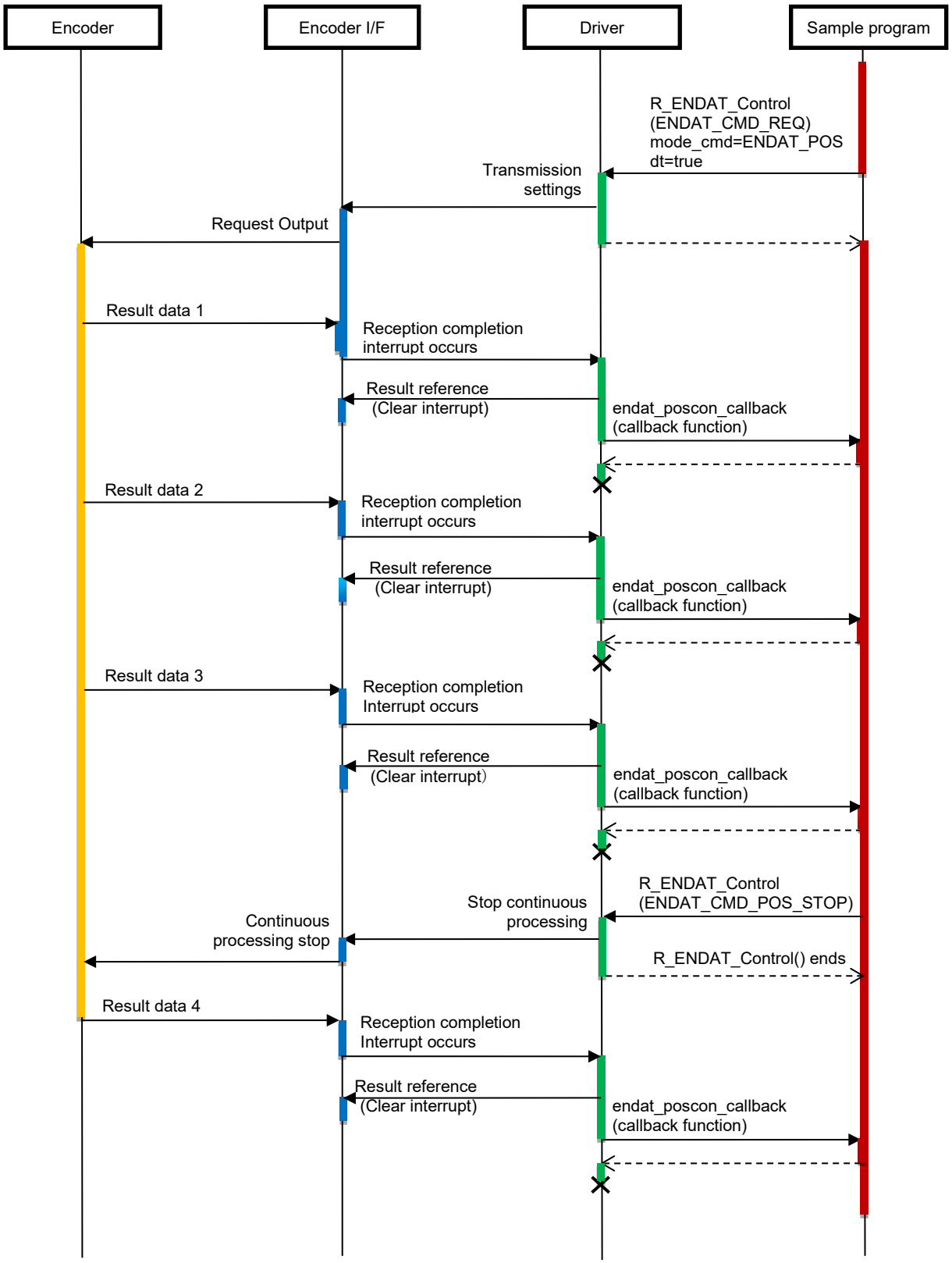

Figure 4.22 Sequence of Request Transmission (Continuous Mode) and Continuous Data Reception

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

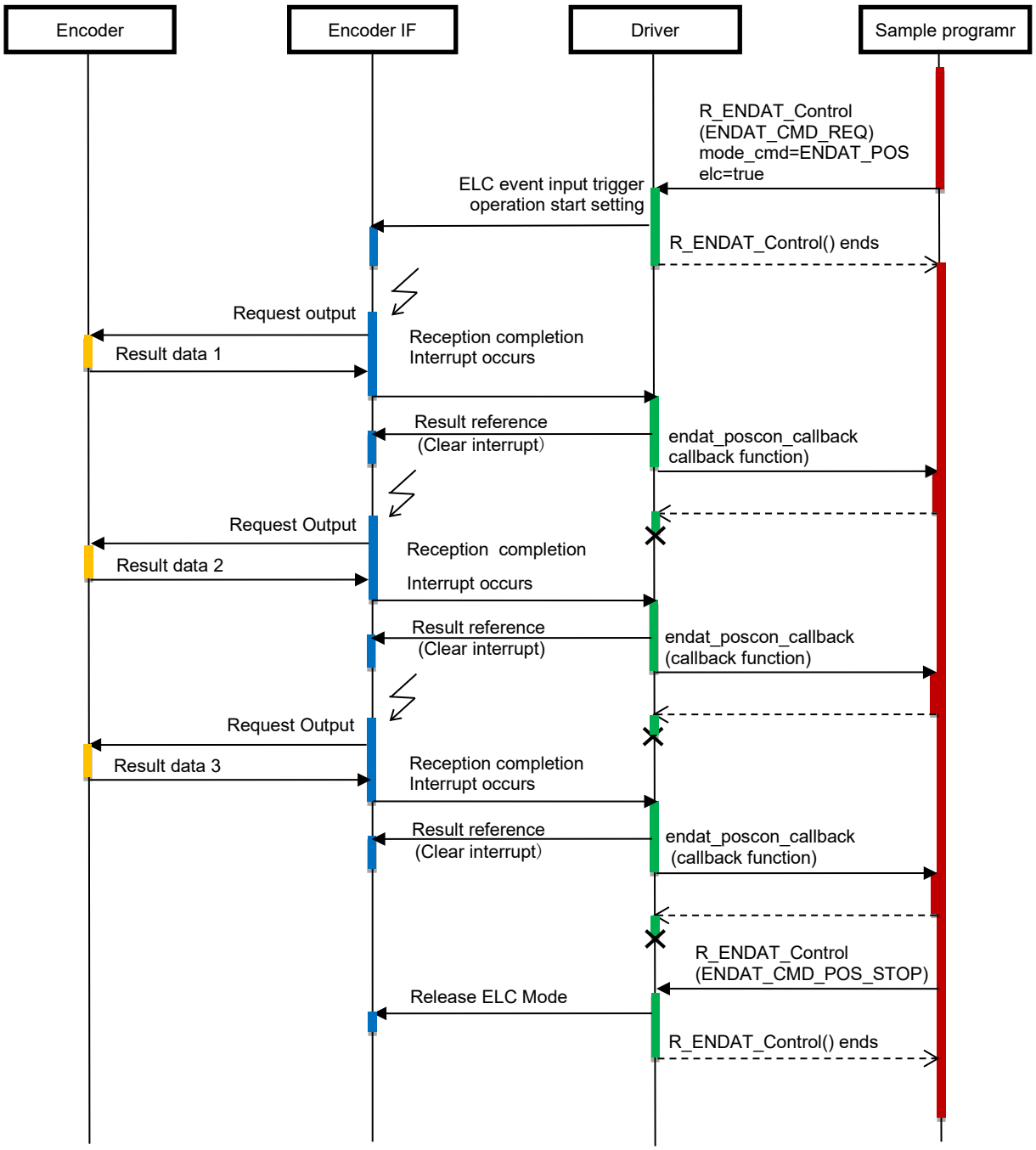

Figure 4.23 Sequence of Request Transmission (ELC Mode) and Continuous Data Reception

(6) Stop Sequence

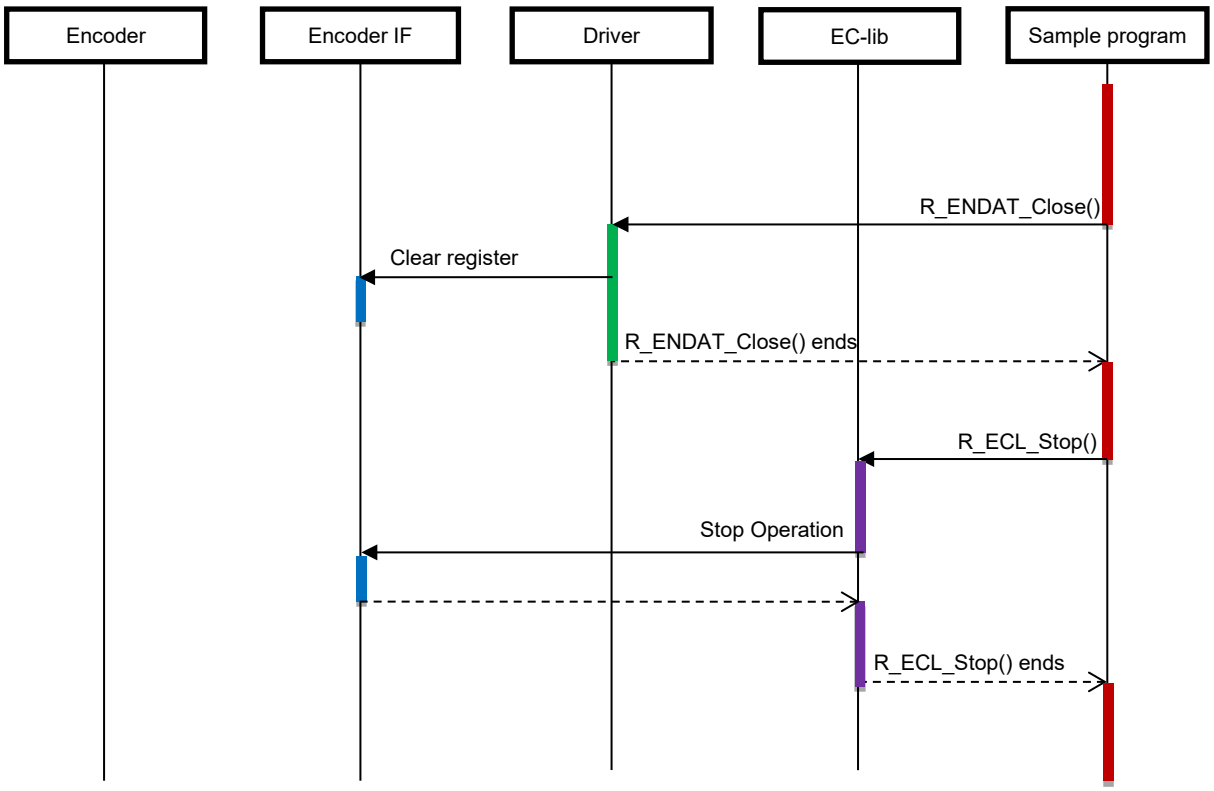

Figure 4.24 Stop Sequence Diagram

4.11.8 Console Commands

This sample program corresponds to the EnDat 2.2 compliant encoder “ECN1123”. The command available for input from the console are listed below.

Table 4.16 Console Commands

command	content
pos	Get positional value only once.
spos	Switching the contents of the additional information 1 to word 1 of Position 2, word 2 of Position 2, word 3 of Position 2, and NOP, the position value and the additional information 1 are acquired. Recovery time measurement is enabled when word 1 of Position 2 is acquired. Memory area selection, position value, and additional information acquisition are performed in the command. Recovery time measurement is not yet completed when word 1 data of Position 2 is retrieved; the value shown as RTCNT after word 2 data is the value after the measurement is completed.
poscon	The positional values are acquired continuously. To stop continuous acquisition, enter the "stop" command.
elctimer *val*	The positional value is continuously acquired in a timer cycle as an ELC event input trigger operation. The timer cycle *val* is specified in units of us (maximum 6990us). To stop continuous acquisition, enter the "stop" command.
stop	Stops continuous acquisition of positional values.
temp	Obtains temperature measurements along with positional values from the encoder.
set_info *MRS*	Specify the memory region selection code (*MRS*) in hexadecimal.
pos_safe	The contents of the additional information specified by the MRS code along with the position value are retrieved only once.
read *addr*	Reads parameters from the address in hexadecimal *addr* in the memory area specified by the MRS code.
write *addr param*	Writes a hexadecimal *param* parameter to the address of hexadecimal *addr* in the memory area specified by the MRS code.
exit	Exit the program.

5. Sample Code

The sample code is available from the Renesas Electronics website.

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Feb 28.23		First Edition issued
2.00	Jul 05.24	5 20 37	Update description of the board name. Remove description about location of integer type definition. Correct description of the result_display function.

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/.