RZ/T2L Group

A-format sample program

Introduction

This application note describes a sample program for acquiring and displaying information from an encoder conforming to the A-format™ Version 2.0 communications protocol specification (“A-format™ V2 specification”) by using the Encoder Interface of the RZ/T2L.

The major features of the program are listed below.

- Supports A-format™ V2 command codes.
- Capable of acquiring angle information, etc. from an encoder conforming to the A-format™ V2 specification (MAR-M50A from Nikon).

Target Device

RZ/T2L

Table of Contents

1. Specifications 4
2. Operating Environment 5
3. Peripheral Functions 6
3.1 Pins 6
4. Software 7
4.1 A-format Driver Function 7
4.2 File Structure 7
4.3 Functions 7
4.4 Specifications of API Functions 8
4.4.1 R_A_AS_Open 8
4.4.2 R_A_AS_Close 8
4.4.3 R_A_AS_GetVersion 9
4.4.4 R_A_AS_Control 9
4.5 Specification of User-defined Functions 13
4.5.1 a_as_rxset_callback 13
4.5.2 a_as_rxend_callback 13
4.5.3 a_as_elctimer_callback 14
4.6 Interrupt Handlers 15
4.6.1 a_as0_int_isr 15
4.6.2 a_as1_int_isr 15
4.7 Interrupts 15
4.8 Constants / Error Codes 16
4.9 Fixed-width Integers 19
4.10 Structures / Unions / Enumerated Types 19
4.10.1 Structures 19
4.10.2 Unions 22
4.10.3 Enumerated Types 22
4.11 Sample Program 23
4.11.1 Operation Overview 23
4.11.2 Functions Used in the Sample Program 25
4.11.3 Specifications of Sample Program Functions 26
4.11.4 Variables Used in the Sample Program 29
4.11.5 Constants Used in the Sample Program 29
4.11.6 Flowchart of Main Process 30
4.11.7 Operation Sequence 37
4.11.8 Console Commands 41
5. Sample Code 43

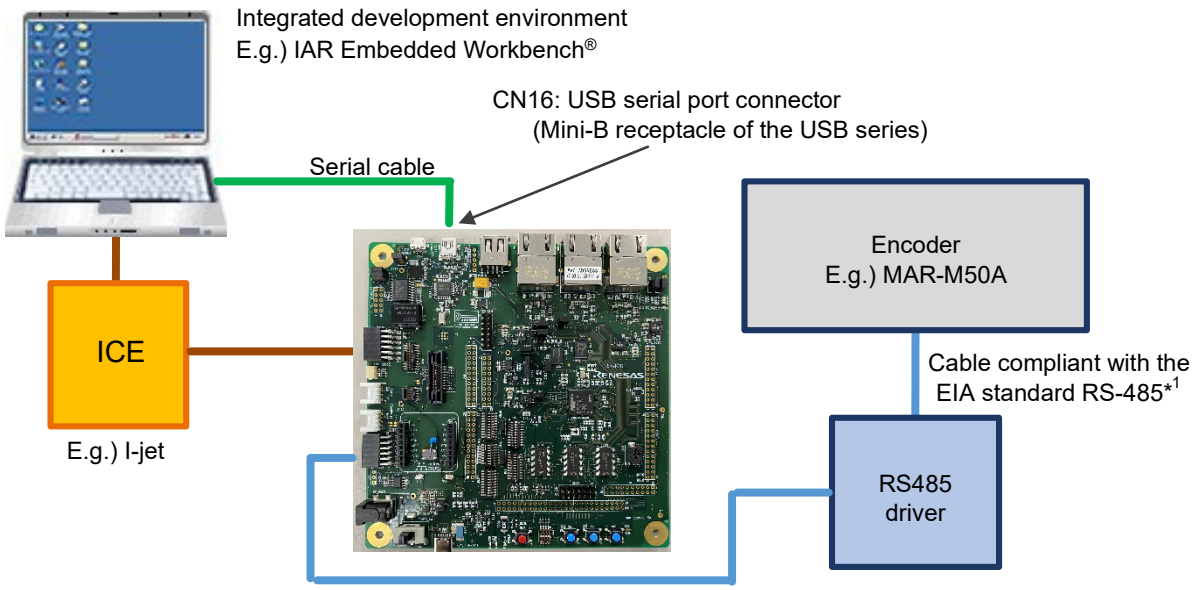
Revision History 44

1. Specifications

Table 1.1 lists the peripheral functions to be used and their applications. Figure 1.1 shows the operating environment when the sample code is being executed.

Table 1.1 Peripheral Functions and Applications

Peripheral Module	Application
A-format communications controller (AFMT)	Communications with the A-format V2 compliant encoder.
Interrupt controller (ICU)	Controls the AFMT interrupts.
General PWM Timer (GPT) Unit 0 Channel 0	Generates events at fixed intervals for input to the ELC.
Event link controller (ELC)	Makes the link between events output from unit 0 channel 0 of the GPT and the AFMT module.
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.



Note 1. For allowable cable length for transmission and reception, contact the manufacturer of the encoder you are using.

Figure 1.1 Operating Environment

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/T2L
Operating frequency	CPUCLK = 800MHz
Operating voltage	1.1V(Core) / 1.8V(PLL, etc.) / 3.3V(I/O)
Integrated development environment *1	IAR Systems IAR Embedded Workbench® for Arm® RENESAS e² studio
Board	RSK+RZT2L (RTK9RZT2L0C00000BJ)
Devices (function to be used on the board)	None

Note: 1. Refer to the release note for the RZ/T2L Group Encoder I/F A-format sample program for the version information of each integrated development environment.

3. Peripheral Functions

The basics of the peripheral modules, operating modes, and registers are described in the “RZ/T2L Group User’s Manual: Hardware”

3.1 Pins

The pins used and their functions are listed in the table below.

Table 3.1 Pins Used and Their Functions

Channel	Port Name	I/O Port	Input/Output	Description
AFMT0	ENCIFDI0 (SDAT)	P02_2	Input	Data input
	ENCIFDO0 (REQ)	P02_3	Output	Data output
	ENCIFOE0 (D/R)	P01_7	Output	Drive/receive control
AFMT1	ENCIFDI1 (SDAT)	P10_1	Input	Data input
	ENCIFDO1 (REQ)	P10_0	Output	Data output
	ENCIFOE1 (D/R)	P09_7	Output	Drive/receive control

4. Software

4.1 A-format Driver Function

The functions of the A-format driver are listed below.

1 Initial settings
2 Transmission of command codes
3 Acquisition of reception data

4.2 File Structure

For the file structure, refer to the release note for the RZ/T2L Group Encoder I/F A-format sample program.

4.3 Functions

The functions to be used are listed in the table below.

Table 4.1 List of Functions

Category	Function Name	Page
A-format driver API functions	R_A_AS_Open	8
	R_A_AS_Close	8
	R_A_AS_GetVersion	9
	R_A_AS_Control	9
User-defined functions	a_as_rxset_callback	13
	a_as_rxend_callback	13
	a_as_elctimer_callback	14
Interrupt handlers	a_as0_int_isr	15
	a_as1_int_isr	15

4.4 Specifications of API Functions

4.4.1 R_A_AS_Open

R_A_AS_Open	
Synopsis	Starts controlling operation of the encoder.
Header	r_a_as_rzt2_if.h
Declaration	int32_t R_A_AS_Open(const int32_t id, r_a_as_info_t* p_info);
Description	This function is used for the following initial settings of the AFMT module. 1. Release A-format Interface from the module-stop state 2. Setting of the encoder interface 3. Setting of the communication parameters (for BRSEL register) 4. Setting for enabling interrupts.
Argument	id : Designates the ID code to be used. (It is defined in r_a_as_rzt2_dat.h.) R_A_AS0_ID : Specifies channel 0. R_A_AS1_ID : Specifies channel 1. Others : Setting is not allowed. p_info : Sets information about the encoder Designate the address of the structure r_a_as_info_t where encoder information is stored.
Return value	R_A_AS_SUCCESS: Normal termination R_A_AS_ERR_INVALID_ARG: Abnormal termination (the id or p_info member variable of the structure r_a_as_info_t is unspecified.) R_A_AS_ERR_ACCESS: Abnormal termination (already opened.)
Note	This function configures the encoder interface. If this function is executed after power for the encoder has been switched on, send CDF8 eight consecutive times to clear the status flag after executing this function. Calling this API function from within a callback function is not allowed.

4.4.2 R_A_AS_Close

R_A_AS_Close	
Synopsis	Stops controlling operation of the encoder
Header	r_a_as_rzt2_if.h
Declaration	int32_t R_A_AS_Close(const int32_t id);
Description	This function stops controlling operation of the encoder on the designated channel.
Argument	id : Designates the ID code to be used. (It is defined in r_a_as_rzt2_dat.h.) R_A_AS0_ID : Specifies Channel 0. R_A_AS1_ID : Specifies Channel 1. Others : Setting is not allowed.
Return Value	R_A_AS_SUCCESS: Normal termination R_A_AS_ERR_INVALID_ARG: Abnormal termination (the id is not the stipulated value) R_A_AS_ERR_ACCESS: Abnormal termination (a request is being transmitted.)
Note	Calling this API function from within a callback function is not allowed.

4.4.3 R_A_AS_GetVersion

R_A_AS_GetVersion	
Synopsis	Acquire the version number of the encoder interface driver.
Header	r_a_as_rzt2_if.h
Declaration	uint32_t R_A_AS_GetVersion(const r_a_as_type_t type);
Description	This function acquires the version number of the A-format driver.
Argument	type : Designates R_A_AS_A_FORMAT.
Return value	A major part of the version number is stored in the sixteen MSBs and the minor part of the version number is stored in the sixteen LSBs.
Supplement	If a type other than those listed above is stipulated, the returned value will be 0xFFFFFFFF.
Note	Calling this API function from within a callback function is not allowed.

4.4.4 R_A_AS_Control

R_A_AS_Control	
Synopsis	Controlling operation of the encoder.
Header	r_a_as_rzt2_if.h
Declaration	int32_t R_A_AS_Control(const int32_t id, const r_a_as_cmd_t cmd, void *const p_buf);
Description	This function controls operations of the encoder by using the cmd argument. See “4.4.4(1) Control Commands” for the operation of the control commands.
Argument	id : Designates the ID code to be used. (It is defined in r_a_as_rzt2_dat.h.) R_A_AS0_ID : Specifies channel 0. R_A_AS1_ID : Specifies channel 1. Others : Setting is not allowed. cmd : Command For the list of the commands, see “Table 4.9, Control Commands of the R_A_AS_Control Function”. p_buf : Arguments corresponding to each cmd.
Return value	R_A_AS_SUCCESS: Normal termination R_A_AS_ERR_INVALID_ARG: Abnormal termination (the id or cmd is not a stipulated value.) See “4.4.4(1) Control Commands” for other returned values.
Note	Be sure to call R_A_AS_Open before calling this function. Calling this API function from within a callback function is not allowed.

(1) Control Commands

(a) R_A_AS_CMD_SET_PARAM

R_A_AS_CMD_SET_PARAM		
Synopsis	Sets request information	
Header	r_a_as_rzt2_if.h	
Declaration	int32_t R_A_AS_Control(const int32_t id, const r_a_as_cmd_t cmd, void *const p_buf);	
Description	This function sets request information	
Argument	id	: Designates the ID code to be used. (It is defined in r_a_as_rzt2_dat.h.)
		R_A_AS0_ID : Specifies channel 0.
		R_A_AS1_ID : Specifies channel 1.
		Others : Setting is not allowed.
	cmd	: Designates R_A_AS_CMD_SET_PARAM.
	p_buf	: Request information Designate the pointer to the structure r_a_as_req_t where the request information is stored. See section “4.10.1(2) r_a_as_req_t” for details.
Return value	R_A_AS_SUCCESS: Normal termination R_A_AS_ERR_INVALID_ARG: Abnormal termination (the id is not a stipulated value, p_buf is null, or the p_buf member variable of the structure r_a_as_req_t is not a stipulated value.) R_A_AS_ERR_ACCESS: Abnormal termination (the applicable channel has not been started.)	
Note	This control command is utilized only for setting request information. Transmission of commands to the encoder proceeds by using the following control commands. ・R_A_AS_CMD_TX_TRG ・R_A_AS_CMD_TX_ELC Calling this control command from within a callback function is not allowed.	

(b) R_A_AS_CMD_ELC_DISABLE

R_A_AS_CMD_ELC_DISABLE		
Synopsis	Disables input of the ELC event triggers	
Header	r_a_as_rzt2_if.h	
Declaration	int32_t R_A_AS_Control(const int32_t id, const r_a_as_cmd_t cmd, void *const p_buf);	
Description	This function disables event input triggers from the ELC.	
Argument	id	: Designates the ID code to be used. (It is defined in r_a_as_rzt2_dat.h.)
		R_A_AS0_ID : Specifies channel 0.
		R_A_AS1_ID : Specifies channel 1.
		Others : Setting is not allowed.
	cmd	: Designates R_A_AS_CMD_ELC_DISABLE.
	p_buf	: Not used. (Designate NULL.)
Return value	R_A_AS_SUCCESS: Input of the ELC event trigger is disabled. R_A_AS_ERR_INVALID_ARG: Abnormal termination (the id is not a stipulated value.) R_A_AS_ERR_ACCESS: Abnormal termination (the ELC event trigger operation is not in progress or the applicable channel has not been started.)	
Note	Calling this control command from within a callback function is not allowed.	

(c) R_A_AS_CMD_TX_TRG

R_A_AS_CMD_TX_TRG	
Synopsis	Starts transmission of requests to the encoder
Header	r_a_as_rzt2_if.h
Declaration	int32_t R_A_AS_Control(const int32_t id, const r_a_as_cmd_t cmd, void *const p_buf);
Description	This function starts transmission of requests to the encoder. A callback function which corresponds to the interrupt source being enabled is called every time a transmission of request is made. See section “4.5.1 a_as_rxset_callback” and “4.5.2 a_as_rxend_callback” for details.
Argument	id : Designates the ID code to be used. (It is defined in r_a_as_rzt2_dat.h.) R_A_AS0_ID : Specifies channel 0. R_A_AS1_ID : Specifies channel 1. Others : Setting is not allowed. cmd : Designates R_A_AS_CMD_TX_TRG p_buf : Not used. (Designate NULL.)
Return value	R_A_AS_SUCCESS: Normal termination. R_A_AS_ERR_INVALID_ARG: Abnormal termination (the id is not the stipulated value) R_A_AS_ERR_BUSY: Abnormal termination (transmission is in progress) R_A_AS_ERR_ACCESS: Abnormal termination (the applicable channel has not been started)
Note	This control command is utilized only for starting transmission of requests to the encoder. Use this command after setting request information by using control command R_A_AS_CMD_SET_PARAM. Calling this control command from within a callback function is not allowed.

(d) R_A_AS_CMD_TX_ELC

R_A_AS_CMD_TX_ELC	
Synopsis	Starts transmission of requests to the encoders in response to the input of an ELC event trigger.
Header	r_a_as_rzt2_if.h
Declaration	int32_t R_A_AS_Control(const int32_t id, const r_a_as_cmd_t cmd, void *const p_buf);
Description	This function enables input of the ELC event trigger and starts the transmission of requests to the encoder. A callback function which corresponds to the interrupt source being enabled is called every time a transmission of request is made. See section “4.5.1 a_as_rxset_callback” and “4.5.2 a_as_rxend_callback” for details. An error code R_A_AS_ERR_BUSY is returned if this control command is executed while the ELC event trigger is in progress.
Argument	id : Designates the ID code to be used. (It is defined in r_a_as_rzt2_dat.h.) R_A_AS0_ID : Specifies channel 0. R_A_AS1_ID : Specifies channel 1. Others : Setting is not allowed. cmd : Designates R_A_AS_CMD_TX_ELC. p_buf : Not used. (Designate NULL.)
Return value	R_A_AS_SUCCESS: Normal termination R_A_AS_ERR_INVALID_ARG: Abnormal termination (the id is not the stipulated value) R_A_AS_ERR_BUSY: Abnormal termination (transmission or the ELC event trigger operation is in progress) R_A_AS_ERR_ACCESS: Abnormal termination (the applicable channel has not been started.)
Note	This control command is utilized only for starting transmission of requests to the encoder. Use this command after setting request information by using control command R_A_AS_CMD_SET_PARAM. Calling this control command from within a callback function is not allowed.

4.5 Specification of User-defined Functions

4.5.1 a_as_rxset_callback

a_as_rxset_callback	
Synopsis	Notification of transmission and reception results in setting completion of reception.
Header	-
Declaration	void a_as_rxset_callback(r_a_as_result_t * p_result);
Description	This is a callback function to be registered by the function R_A_AS_Control (R_A_AS_CMD_SET_PARAM). This function conveys the result of transmission and reception. It is called in response to completion of the data reception (AFMTi_EOF). After this function, the function a_as_rxend_callback() is called, too. This function is called in interrupt context. To ensure interrupt responsiveness, return promptly. The function name is an example, and can be selected freely.
Argument	p_result : Result of transmission and reception Pointer to the array where the result of transmission and reception declared by the structure r_a_as_result_t is stored. See “Table 4.2 Result of Transmission and Reception Stored in Each Array Element” for each element. The result of transmission and reception for the encoder address specified by the function R_A_AS_Control (R_A_AS_CMD_SET_PARAM) is updated. See “Table 4.3 Results of Transmission and Reception” for details.
Return value	None
Note	If the ELC event trigger is enabled, the result of transmission and reception should be acquired within the set timer interval. Depending on the response timing of the encoder, this function may be called even if the function R_A_AS_Close, or R_A_AS_Control (R_A_AS_CMD_ELC_DISABLE) has been executed.

4.5.2 a_as_rxend_callback

a_as_rxend_callback	
Synopsis	Notification of transmission and reception results in completion of reception.
Header	-
Declaration	void a_as_rxend_callback (r_a_as_result_t * p_result);
Description	This is a callback function to be registered by the function R_A_AS_Control (R_A_AS_CMD_SET_PARAM). This function conveys the result of transmission and reception. It is called by a completion of the data reception interrupt (AFMTi_EOF) following the function a_as_rxset_callback() call. This function is called in interrupt context. To ensure interrupt responsiveness, return promptly. The function name is an example, and can be selected freely.
Argument	p_result : Result of transmission and reception Pointer to the array where the result of transmission and reception declared by the structure r_a_as_result_t is stored. See “Table 4.2 Result of Transmission and Reception Stored in Each Array Element” for each element. The result of transmission and reception for the encoder address specified by the function R_A_AS_Control (R_A_AS_CMD_SET_PARAM) is updated. See “Table 4.3 Results of Transmission and Reception” for details.
Return value	None
Note	If the ELC event trigger is enabled, the result of transmission and reception should be acquired within the set timer interval. Depending on the response timing of the encoder, this function may be called even if the function R_A_AS_Close or R_A_AS_Control (R_A_AS_CMD_ELC_DISABLE) has been executed.

4.5.3 a_as_elctimer_callback

a_as_elctimer_callback	
Synopsis	Notification of transmission and reception results in continuous operation of the ELC event.
Header	-
Declaration	void a_as_elctimer_callback(r_a_as_result_t * p_result);
Description	This is a callback function to be registered by the function R_A_AS_Control (R_A_AS_CMD_SET_PARAM). This function conveys the result of transmission and reception. It is called every time when an interrupt request is generated in response to completion of the data reception (AFMTi_EOF). This function is called in interrupt context. To ensure interrupt responsiveness, return promptly. The function name is an example, and can be selected freely.
Argument	p_result : Result of transmission and reception Pointer to the array where the result of transmission and reception declared by the structure r_a_as_result_t is stored. See “Table 4.2 Result of Transmission and Reception Stored in Each Array Element” for each element. The result of transmission and reception for the encoder address specified by the function R_A_AS_Control (R_A_AS_CMD_SET_PARAM) is updated. See “Table 4.3 Results of Transmission and Reception” for details.
Return value	None
Note	If the ELC event trigger is enabled, the result of transmission and reception should be acquired within the set timer interval. Depending on the response timing of the encoder, this function may be called even if the function R_A_AS_Close or R_A_AS_Control (R_A_AS_CMD_ELC_DISABLE) has been executed.

Table 4.2 Result of Transmission and Reception Stored in Each Array Element

Array Number	Content
p_result[0]	Result of transmission and reception for the encoder section ENC1.
p_result[1]	Result of transmission and reception for the encoder section ENC2.
p_result[2]	Result of transmission and reception for the encoder section ENC3.
p_result[3]	Result of transmission and reception for the encoder section ENC4.
p_result[4]	Result of transmission and reception for the encoder section ENC5.
p_result[5]	Result of transmission and reception for the encoder section ENC6.
p_result[6]	Result of transmission and reception for the encoder section ENC7.
p_result[7]	Result of transmission and reception for the encoder section ENC8.

Table 4.3 Results of Transmission and Reception

Interrupt Source	Results of Transmission and Reception (Member variables of p_result)		
	Result	data	Status
Completion of the data reception (AFMTi_EOF) *1	Only valid when used in the callback functions.	Valid *2	Only valid when used in the callback functions.

Note: 1. i = 0, 1
2. If the ELC event trigger is disabled, the data reception results are valid until next request.
If the ELC event trigger is enabled, the data reception results are valid until next AFMTi_EOF interrupt is generated.

4.6 Interrupt Handlers

4.6.1 a_as0_int_isr

a_as0_int_isr	
Synopsis	Interrupt handler for AFMT0_EOF
Header	-
Declaration	static void a_as0_int_isr(void);
Description	This is the interrupt handler for the completion of A_AS channel 0 data reception.
Argument	None
Return value	None

4.6.2 a_as1_int_isr

a_as1_int_isr	
Synopsis	Interrupt handler for AFMT1_EOF
Header	-
Declaration	static void a_as1_int_isr(void);
Description	This is the interrupt handler for the completion of A_AS channel 1 data reception.
Argument	None
Return value	None

4.7 Interrupts

Interrupt requests used with the A-format driver are listed below.

Table 4.4 Interrupt for the A-format Driver

Interrupt Request Signal	ID	Description
AFMT0_EOF	267	This interrupt request is generated in response to the completion of channel 0 data reception.
AFMT1_EOF	268	This interrupt request is generated in response to the completion of channel 1 data reception.

4.8 Constants / Error Codes

See the individual tables below for settings and descriptions related to the constants and error codes.

Table 4.5 List of Tables for Definitions of Constants and Error Codes

Table number	Contents
Table 4.6	User-defined Constants to be Used in the A-format Driver (r_a_as_rzt2_config.h)
Table 4.7	Driver Type
Table 4.8	Methods of Connection between A_AS and Encoders
Table 4.9	Control Commands of the R_A_AS_Control Function
Table 4.10	Bitrate
Table 4.11	Encoder Addresses
Table 4.12	Commands
Table 4.13	Error Codes

Table 4.6 User-defined Constants to be Used in the A-format Driver (r_a_as_rzt2_config.h)

Constant Name	Setting	Description
R_AFMT_T2_ONE_2500KBPS	0x0000	The value set in the BRSEL register TM bit when the connection is one-to-one, and the bit rate is 2.5 Mbps. *
R_AFMT_T2_ONE_4MBPS	0x0000	The value set in the BRSEL register TM bit when the connection is one-to-one, and the bit rate is 4 Mbps. *
R_AFMT_T2_ONE_6670KBPS	0x0000	The value set in the BRSEL register TM bit when the connection is one-to-one, and the bit rate is 6.67 Mbps. *
R_AFMT_T2_ONE_8MBPS	0x0000	The value set in the BRSEL register TM bit when the connection is one-to-one, and the bit rate is 8 Mbps. *
R_AFMT_T2_BUS_2500KBPS	0x001A	The value set in the BRSEL register TM bit when the connection is a bus connection, and the bit rate is 2.5 Mbps. *
R_AFMT_T2_BUS_4MBPS	0x0010	The value set in the BRSEL register TM bit when the connection is a bus connection, and the bit rate is 4 Mbps. *
R_AFMT_T2_BUS_6670KBPS	0x0009	The value set in the BRSEL register TM bit when the connection is a bus connection, and the bit rate is 6.67 Mbps. *
R_AFMT_T2_BUS_8MBPS	0x0008	The value set in the BRSEL register TM bit when the connection is a bus connection, and the bit rate is 8 Mbps. *

Note: In this sample program, the values set in each register are the recommended values.

Table 4.7 Driver Type

Constant Name	Setting	Description
R_A_AS_A_FORMAT	0	Designates the A-format driver.

Table 4.8 Methods of Connection between A_AS and Encoders

Constant Name	Setting	Description
R_A_AS_ONE_FOR_ONE	0	One-to-one connection
R_A_AS_BUS	1	Bus connection

Table 4.9 Control Commands of the R_A_AS_Control Function

Constant Name	Setting	Description
R_A_AS_CMD_SET_PARAM	0xAF000000	Sets requests information.
R_A_AS_CMD_ELC_DISABLE	0xAF000002	Disable input of the ELC event trigger.
R_A_AS_CMD_TX_TRG	0xAF000003	Starts transmission of a command.
R_A_AS_CMD_TX_ELC	0xAF000005	Starts transmission of a command by input of the ELC event trigger.

Table 4.10 Bitrate

Constant Name	Setting	Description
R_A_AS_2500KBPS	0	2.5 Mbps
R_A_AS_4MBPS	1	4 Mbps
R_A_AS_6670KBPS	2	6.67 Mbps
R_A_AS_8MBPS	3	8 Mbps

Table 4.11 Encoder Addresses

Constant Name	Setting	Description
R_A_AS_ECN1	0	The address of the encoder section ENC1.
R_A_AS_ECN2	1	The address of the encoder section ENC2.
R_A_AS_ECN3	2	The address of the encoder section ENC3.
R_A_AS_ECN4	3	The address of the encoder section ENC4.
R_A_AS_ECN5	4	The address of the encoder section ENC5.
R_A_AS_ECN6	5	The address of the encoder section ENC6.
R_A_AS_ECN7	6	The address of the encoder section ENC7.
R_A_AS_ECN8	7	The address of the encoder section ENC8.

Table 4.12 Commands

Constant Name	Setting	Description
R_A_AS_CDF0	0	The value defined for the command data frame CDF0.
R_A_AS_CDF1	1	The value defined for the command data frame CDF1.
R_A_AS_CDF2	2	The value defined for the command data frame CDF2.
R_A_AS_CDF3	3	The value defined for the command data frame CDF3.
R_A_AS_CDF4	4	The value defined for the command data frame CDF4.
R_A_AS_CDF5	5	The value defined for the command data frame CDF5.
R_A_AS_CDF6	6	The value defined for the command data frame CDF6.
R_A_AS_CDF7	7	The value defined for the command data frame CDF7.
R_A_AS_CDF8	8	The value defined for the command data frame CDF8.
R_A_AS_CDF9	9	The value defined for the command data frame CDF9.
R_A_AS_CDF10	10	The value defined for the command data frame CDF10.
R_A_AS_CDF11	11	The value defined for the command data frame CDF11.
R_A_AS_CDF12	12	The value defined for the command data frame CDF12.
R_A_AS_CDF13	13	The value defined for the command data frame CDF13.
R_A_AS_CDF14	14	The value defined for the command data frame CDF14.
R_A_AS_CDF15	15	The value defined for the command data frame CDF15.
R_A_AS_CDF16	16	The value defined for the command data frame CDF16.
R_A_AS_CDF17	17	The value defined for the command data frame CDF17.
R_A_AS_CDF18	18	The value defined for the command data frame CDF18.
R_A_AS_CDF19	19	The value defined for the command data frame CDF19.
R_A_AS_CDF21	21	The value defined for the command data frame CDF21.
R_A_AS_CDF22	22	The value defined for the command data frame CDF22.
R_A_AS_CDF27	27	The value defined for the command data frame CDF27.
R_A_AS_CDF28	28	The value defined for the command data frame CDF28.
R_A_AS_CDF29	29	The value defined for the command data frame CDF29.
R_A_AS_CDF30	30	The value defined for the command data frame CDF30.

Table 4.13 Error Codes

Constant Name	Setting	Description
R_A_AS_SUCCESS	0	Normal termination
R_A_AS_ERR_INVALID_ARG	-1	Argument error
R_A_AS_ERR_BUSY	-2	The API cannot be executed.
R_A_AS_ERR_ACCESS	-3	API execution order error.

4.9 Fixed-width Integers

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

Table 4.14 Fixed-width Integers for the Sample Code

Symbol	Description
int8_t	8-bit signed integer
int16_t	16-bit signed integer
int32_t	32-bit signed integer
int64_t	64-bit signed integer
uint8_t	8-bit unsigned integer
uint16_t	16-bit unsigned integer
uint32_t	32-bit unsigned integer
uint64_t	64-bit unsigned integer

4.10 Structures / Unions / Enumerated Types

The major structures, unions, and enumerated types are listed below.

4.10.1 Structures

(1) r_a_as_info_t

Initialization information of the A_AS control unit,

```
typedef struct
{
    uint8_t     connect;    Connection method
                            Designate the method of connection between A_AS and the
                            encoders. See “Table 4.8, Methods of Connection between A_AS
                            and Encoders” for the values to be designated.
                            Note: This setting is reflected in the BRSEL register.
    uint8_t     bitrate;    Bit rate
                            Designate the bit rate for communications with the encoder. See
                            “Table 4.10, Bitrate” for the values to be designated.
                            Note: This setting is reflected in the BRSEL register.
    uint16_t    ifmg;       Margin value
                            Note: This is reserved for driver interface compatibility with RZ/T2M
                            group A-format driver. This setting value is not used for RZ/T2L.
} r_a_as_info_t
```

(2) r_a_as_req_t

Information of requests to be sent to the encoders.

```
typedef struct
{
```

uint8_t	encadr;	Encoder address Designate the encoder address. See “Table 4.11 Encoder Addresses” for the value to be designated. This setting is reflected in the XEA bit of the COMMAND register.
uint8_t	cmd;	Command Designate the command code to be sent to the encoder. See “Table 4.12 Commands” for the value to be designated. If the value is not listed in the said table and exceeds 0x20, an error with the error code R_A_AS_ERR_INVALID_ARG is generated. Some commands are not available depending on the connection method.
uint8_t	memadr;	Memory address Designate the address of the memory in the encoder. Set this value only in the following cases: cmd = R_A_AS_CDF13 cmd = R_A_AS_CDF14
uint16_t	memdat;	Data to be written to the memory Designate the data to be written to the memory. Set this value only in the following case: cmd = R_A_AS_CDF14
uint32_t	encid;	Recognition code Designate the 24-bit recognition code. Set this value only in the following cases: cmd = R_A_AS_CDF18 cmd = R_A_AS_CDF19
r_a_as_result_cb_t	cbadr_txerr	The pointer to the callback function for TXERR Note: This is reserved for driver interface compatibility with RZ/T2M group A-format driver. This setting value is not used for RZ/T2L. Set NULL to this pointer.
r_a_as_result_cb_t	cbadr_rxset;	The pointer to the callback function *2 to be called in response to AFMTi_EOF (i = 0,1) interrupt request. See “4.5.1, a_as_rxset_callback” for details.
r_a_as_result_cb_t	cbadr_rxend;	The pointer to the callback function *2 to be called following the cbadr_rxset() function in response to AFMTi_EOF (i = 0,1) interrupt request. See “4.5.2, a_as_rxend_callback” for details.
bool	pre;	Set true in this variable to set request information while transmission and reception of data by the ELC event trigger is in progress. Set false to set request information in other cases. (true: Set request information while transmission and reception of data by the ELC event trigger is in progress)

```
} r_a_as_req_t
```

Note: 1. For the command R_A_AS_CDF13, the accessible address range is between 0x00 and 0xFF. For the command R_A_AS_CDF14, the accessible address range is between 0x00 and 0xEF.
2. Callback function will not be run if the pointer is NULL.

(3) r_a_as_result_t

Result of transmission and reception

```
typedef struct
{
    r_a_as_req_err_t    result;     Result of transmission and reception
                                    See the enumerated type r_a_as_req_err_t for details.
    r_a_as_data_t       data;       Reception data
                                    See the structure type r_a_as_data_t for details.
    r_a_as_status_t     status;     State of the A_AS
                                    See the structure type r_a_as_status_t for details.
} r_a_as_result_t
```

(4) r_a_as_data_t

Received data

```
typedef struct
{
    uint32_t            rxi;        ENCnRXDATA0 register value
                                    The value of the ENCnRXDATA0 register is stored here.
    uint32_t            rxd0;       ENCnRXDATA1 register value
                                    The value of the ENCnRXDATA1 register is stored here.
    uint32_t            rxd1;       ENCnRXDATA2 register value
                                    The value of the ENCnRXDATA2 register is stored here.
} r_a_as_data_t
```

(5) r_a_as_status_t

State of the A_AS on transmission or reception

```
typedef struct
{
    bool     iwdgerr;    IF watchdog errors. Corresponds to CA[2] SYNC status flag. *1
    bool     dwdgerr;    DF watchdog errors. Corresponds to CA[2] SYNC status flag. *1
    bool     starterr;   Start bit errors. Corresponds to CA[1] FORM status flag.
    bool     stoperr;    Stop bit errors. Corresponds to CA[1] FORM status flag.
    bool     syncerr;    Sync code errors. Corresponds to CA[2] SYNC status flag. *1
    bool     rxeaerr;    Received encoder address error. Corresponds to CA[0] CMD status flag.
    bool     crcerr;     CRC error. Corresponds to CA[3] CRC status flag.
    bool     rxccerr;    Received command code error. Corresponds to CA[0] CMD status flag.
    bool     mdaterr;    Indicates EEPROM data error. *2
    bool     madrerr;    Indicates EEPROM address error. *2
    bool     rxdzerr;    Indicates ID code error. *2
    bool     fd1err;     Indicates fixed-data (1) error. *2
    bool     fd2err;     Indicates fixed-data (2) error. *2
    bool     fd3err;     Indicates fixed-data (3) error. *2
    bool     fd5err;     Indicates fixed-data (5) error. *2
    bool     elcin;      Information in response to ELCIN input is being stored. *2
    uint8_t  txcc;       Transmission command code is stored.
                         (0: CDF0 to 19 and 23 to 30, 1: CDF21, 2: CDF22)
    bool     rxset;      This value indicates whether the setting of received data is complete.
                         (true: Ready to read; false: Not ready to read)
    bool     timer;      Indicates whether the timer is operating or not. *2
    bool     txerr;      Indicates transmission error. *2
    bool     rxend;      Indicates completion of reception. (true: received; false: not received)
} r_a_as_status_t
```

Note: 1. CA[2] SYNC status flag shows timeout error.
2. This is reserved for driver interface compatibility with RZ/T2M group A-format encoder driver. This is not used in the driver for RZ/T2L. It is always false.

4.10.2 Unions

Unions are not used in this sample program.

4.10.3 Enumerated Types

(1) r_a_as_req_err_t

Result of reception from the encoder

```
typedef enum
{
    R_A_AS_REQ_SUCCESS = 0,   Normal termination of data transmission and reception
    R_A_AS_REQ_ERR            An error occurred in data transmission or reception.
                              This error code is generated even if one of the error
                              indicators (excluding timer, rxend, rxset, elcin, and txcc) of
                              the structure r_a_as_status_t is "true".
    R_A_AS_REQ_BP_ERR         The reception FIFO buffer is full.
                              This error code is generated only in bypass reception.
} r_a_as_req_err_t
```

4.11 Sample Program

4.11.1 Operation Overview

The sample program described in this application note supports bus connection of up to eight A-format compliant encoders (Nikon MAR-M50A). This sample program handles the following processes.

1) Transmitting requests input through the terminal I/O for the debugger to the encoder, by writing to the TXTRG register.
2) Displaying the data received from the encoder in the console.
3) Transmitting commands and receiving results in response by using the ELC event trigger function of the AFMT module. The ELC links input events for the AFMT module to events output by the GPT. See “Figure 4.6 Flowchart of a_as_elctimer” for an example setting of input events.

(1) System Block Diagram

A system block diagram is shown below.

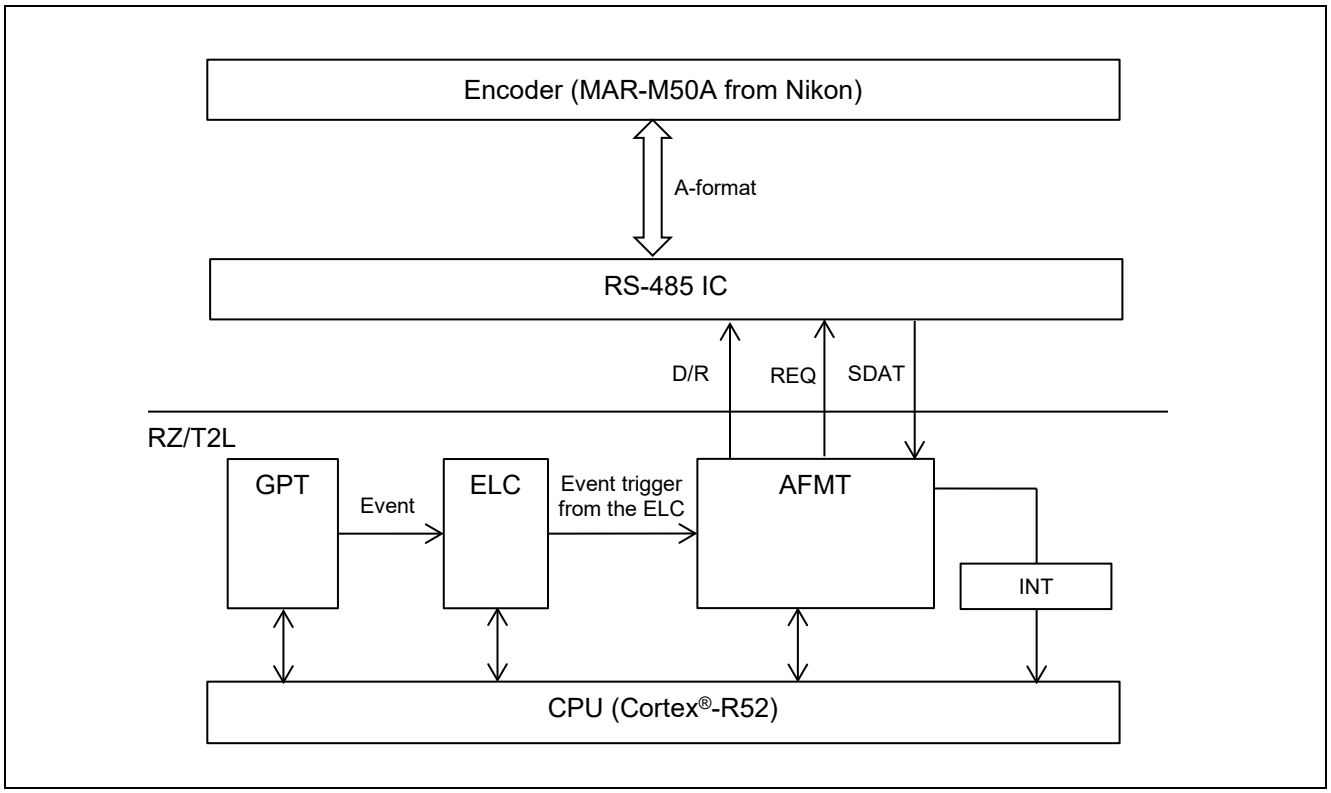


Figure 4.1 System Block Diagram

(2) Software Structure

Figure 4.2 shows a system block diagram.

The A-format driver has four sections: the opening process part configured of the function R_A_AS_Open, the closing process part configured of the function R_A_AS_Close, the request transmission part configured of the function R_A_AS_Control, and the data reception part (interrupt handler) configured of the callback functions.

The A-format driver controller section of the sample program controls the A-format driver and sends requests, and the results indication section indicates the result of data reception.

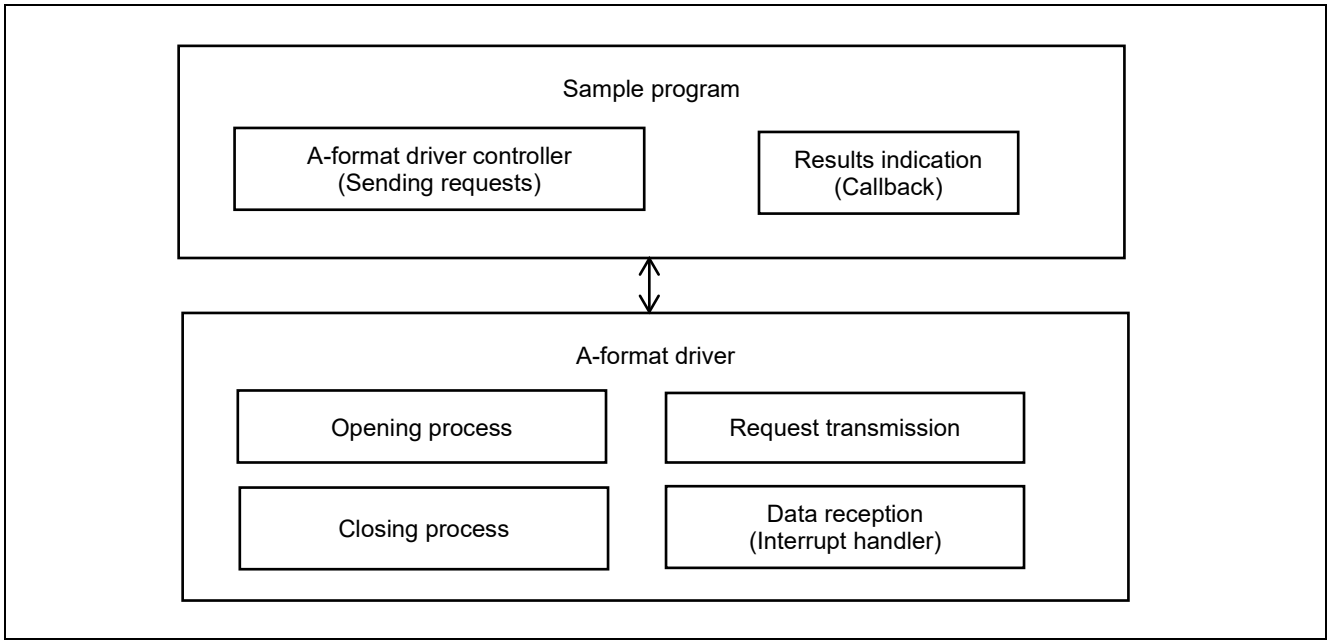


Figure 4.2 Software Structure

4.11.2 Functions Used in the Sample Program

The major functions used in the sample program are listed below.

Table 4.15 Major Functions Used in the Sample Program

Function Name	Page
hal_entry	26
enc_main	26
a_as_cmd_control	26
a_as_enc_init	27
a_as_req	27
a_as_elctimer	27
a_as_elcstop	27
a_as_exit	28
a_as_rxset_callback	28
a_as_rxend_callback	28
a_as_elctimer_callback	28

4.11.3 Specifications of Sample Program Functions

(1) hal_entry

hal_entry	
Synopsis	Entry function of the A-format sample program
Header	-
Declaration	void hal_entry(void);
Description	This is the entry function of the A-format sample program. The function enc_main() is called from here.
Argument	None
Return value	None

(2) enc_main

enc_main		
Synopsis	Main function of the A-format sample program.	
Header	-	
Declaration	int32_t enc_main(uint8_t ch);	
Description	This is the main function of the A-format sample program. See section “4.11.6(1), Flowchart of enc_main” for details.	
Argument	ch	Encoder channel 0: specify channel 0, 1: specify channel 1
Return value	0: Normal termination Others: Abnormal termination (an error code in the encoder interface)	

(3) a_as_cmd_control

a_as_cmd_control		
Synopsis	Controls the A-format driver.	
Header	-	
Declaration	static void a_as_cmd_control(int32_t id);	
Description	This function performs the following processes. Starting control of the encoder Input processing of the console command Ending control of the encoder	
Argument	id	Encoder ID R_A_AS0_ID: Specify channel 0 encoder ID R_A_AS1_ID: Specify channel 1 encoder ID
Return value	None	

(4) a_as_enc_init

a_as_enc_init		
Synopsis	Initializing the encoder	
Header	-	
Declaration	static int32_t a_as_enc_init(int32_t id);	
Description	This function is for initializing the encoder. The command data frame CDF8 is transmitted eight times consecutively to clear the status flag.	
Argument	id	Encoder ID R_A_AS0_ID: Specify channel 0 encoder ID R_A_AS1_ID: Specify channel 1 encoder ID
Return value	0: Normal termination Others: Abnormal termination (an error code in the encoder interface)	

(5) a_as_req

a_as_req		
Synopsis	Console command “req” function	
Header	-	
Declaration	static void a_as_req(uint32_t arg_num, char_t *p_arg[]);	
Description	This function is executed when the console command "req" is input. See section “4.11.6(3), Flowchart of a_as_req” and section “4.11.8, Console Commands” for details.	
Argument	arg_num	The number of character strings input through the console
	*p_arg[]	The starting address where the character strings are stored
Return value	None	

(6) a_as_elctimer

a_as_elctimer		
Synopsis	Console command “elctimer” function	
Header	-	
Declaration	static void a_as_elctimer(uint32_t arg_num, char_t *p_arg[]);	
Description	This function is executed when the console command “elctimer” is input. See section “4.11.6(4), Flowchart of a_as_elctimer” and section “4.11.8, Console Commands” for details.	
Argument	arg_num	The number of character strings input through the console.
	*p_arg[]	The starting address where the character strings are stored.
Return value	None	

(7) a_as_elcstop

a_as_elcstop		
Synopsis	Console command “elcstop” function	
Header	-	
Declaration	static void a_as_elcstop(uint32_t arg_num, char_t *p_arg[]);	
Description	This function is executed when the console command "elcstop" is input. See section “4.11.6(5), Flowchart of a_as_elcstop” and section “4.11.8, Console Commands” for details.	
Argument	arg_num	The number of character strings input through the console.
	*p_arg[]	The starting address where the character strings are stored.
Return value	None	

(8) a_as_exit

a_as_exit		
Synopsis	Console command “exit” function	
Header	-	
Declaration	static void a_as_exit(uint32_t arg_num, char_t *p_arg[]);	
Description	This function is executed when the console command “exit”	
Argument	arg_num	The number of character strings input through the console
	*p_arg[]	The starting address where the character strings are stored.
Return value	None	

(9) a_as_rxset_callback

a_as_rxset_callback		
Synopsis	Callback for the console command “req” when the setting of received data is complete.	
Header	-	
Declaration	static void a_as_rxset_callback(r_a_as_result_t *p_result);	
Description	This is a callback function that is executed when the console command "req" is input. It holds a pointer to the result of the transmitted an A-format request and received data in response in the variable a_as_result and indicates completion of the setting of received data. See section “4.11.6(6), Flowchart of a_as_rxset_callback” for details.	
Argument	*p_result	The address in the RAM where the result of transmission of the request and reception starts.
Return value		

(10) a_as_rxend_callback

a_as_rxend_callback		
Synopsis	Callback for the console command “req” when reception of the data has been completed.	
Header	-	
Declaration	static void a_as_rxend_callback(r_a_as_result_t *p_result);	
Description	This is a callback function that is executed when the console command "req" is input. It indicates completion of the reception of data in response to the A-format request. See section “4.11.6(7), Flowchart of a_as_rxend_callback” for details.	
Argument	*p_result	The address in the RAM where the result of transmission of the request and reception starts.
Return value	None	

(11) a_as_elctimer_callback

a_as_elctimer_callback		
Synopsis	Callback for the console command "elctimer".	
Header	-	
Declaration	static void a_as_elctimer_callback(r_a_as_result_t * p_result);	
Description	This is a callback function that is executed when the console command "elctimer" is input. It holds the result of the transmitted request and received data in the variables a_as_ti_result and a_as_ti_data. See section “4.11.6(8), Flowchart of a_as_elctimer_callback” for details.	
Argument	*p_result	The address in the RAM where the result of transmission of the request and reception starts.
Return value	None	

4.11.4 Variables Used in the Sample Program

The major static variables used in the sample program are listed below.

Table 4.16 Static Variables Used in the Sample Program

Type	Variable Name	Description
bool	a_as_flg	Transmission and reception completion flag. (true: transmission and reception completed; false: transmission and reception in progress)
r_a_as_result_t	a_as_result[A_AS_ENC_NUM]	The results of acquisition are stored.
bool	a_as_elc_flg	ELC event input trigger flag (true: ELC event input trigger operation is in progress; false: ELC event input trigger operation is stopped)
bool	elc_trans_flg	The flag that indicates the state of transmission and reception of data from the ELC event input trigger. (true: Transmission and reception in progress, false: Transmission and reception completed)
r_a_as_req_t	a_as_req_elc	Holds the request information while ELC event input trigger operation is in progress.

4.11.5 Constants Used in the Sample Program

The major constants used in the sample program are listed below.

Table 4.17 Major Constants Used in the Sample Program

Constant	Setting	Description
A_AS_ENC_NUM	8	The number of connected encoders.

4.11.6 Flowchart of Main Process

The flowcharts for the major processes are given below.

(1) Flowchart of enc_main

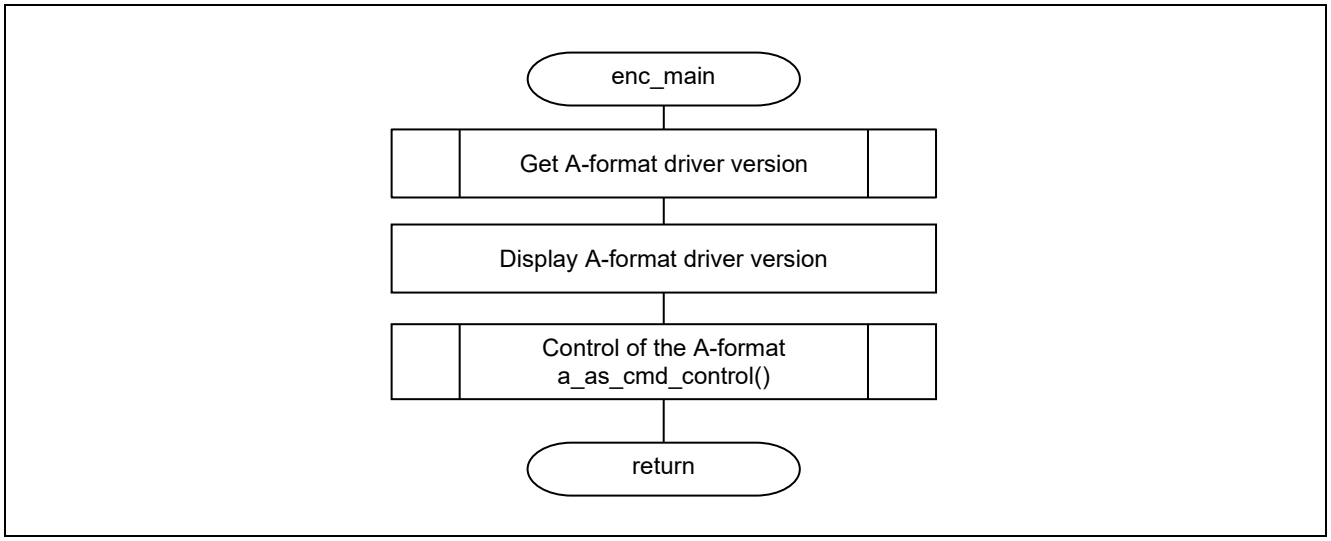


Figure 4.3 Flowchart of enc_main

(2) Flowchart of a_as_cmd_control

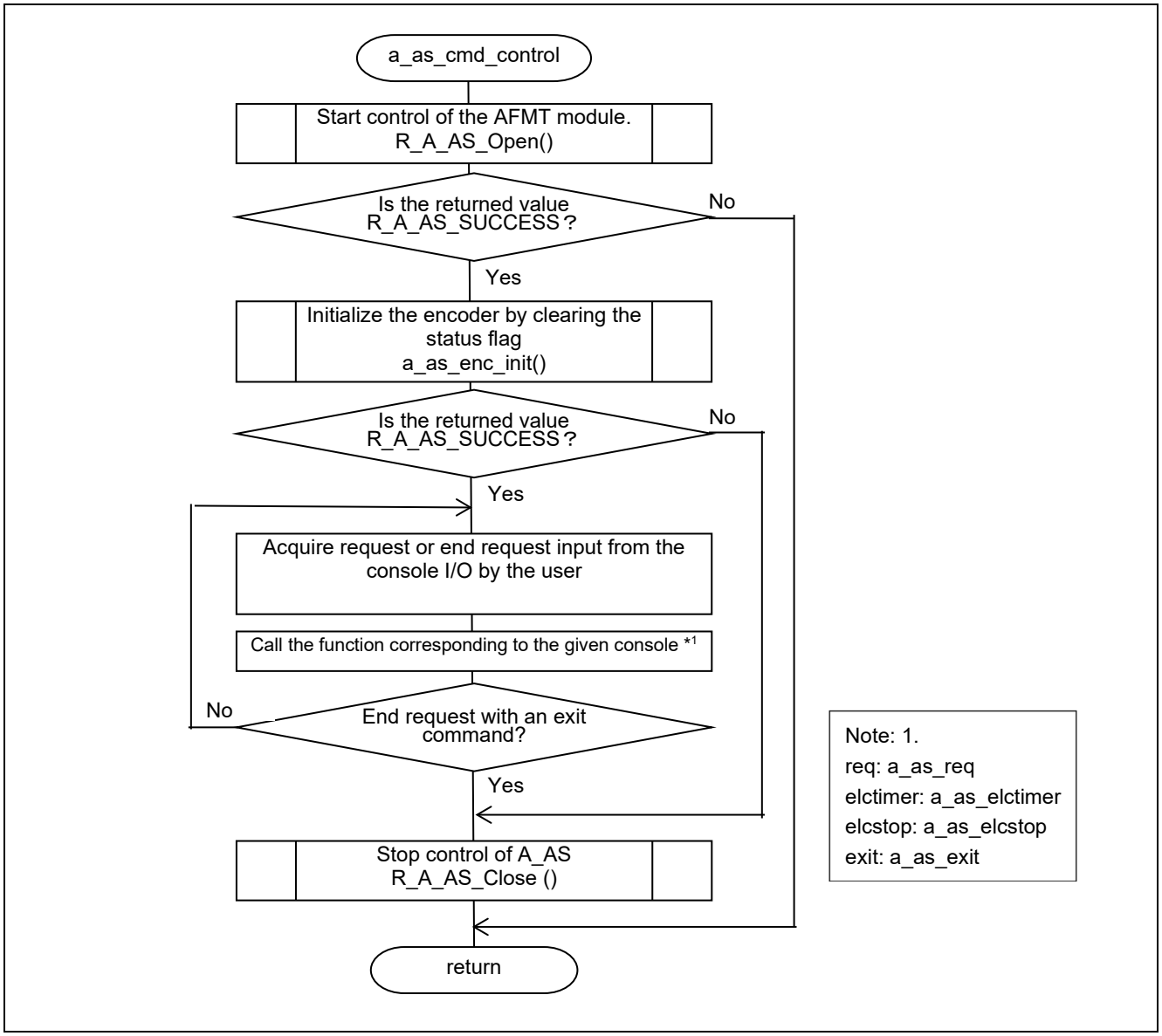


Figure 4.4 Flowchart of a_as_cmd_control

(3) Flowchart of a_as_req

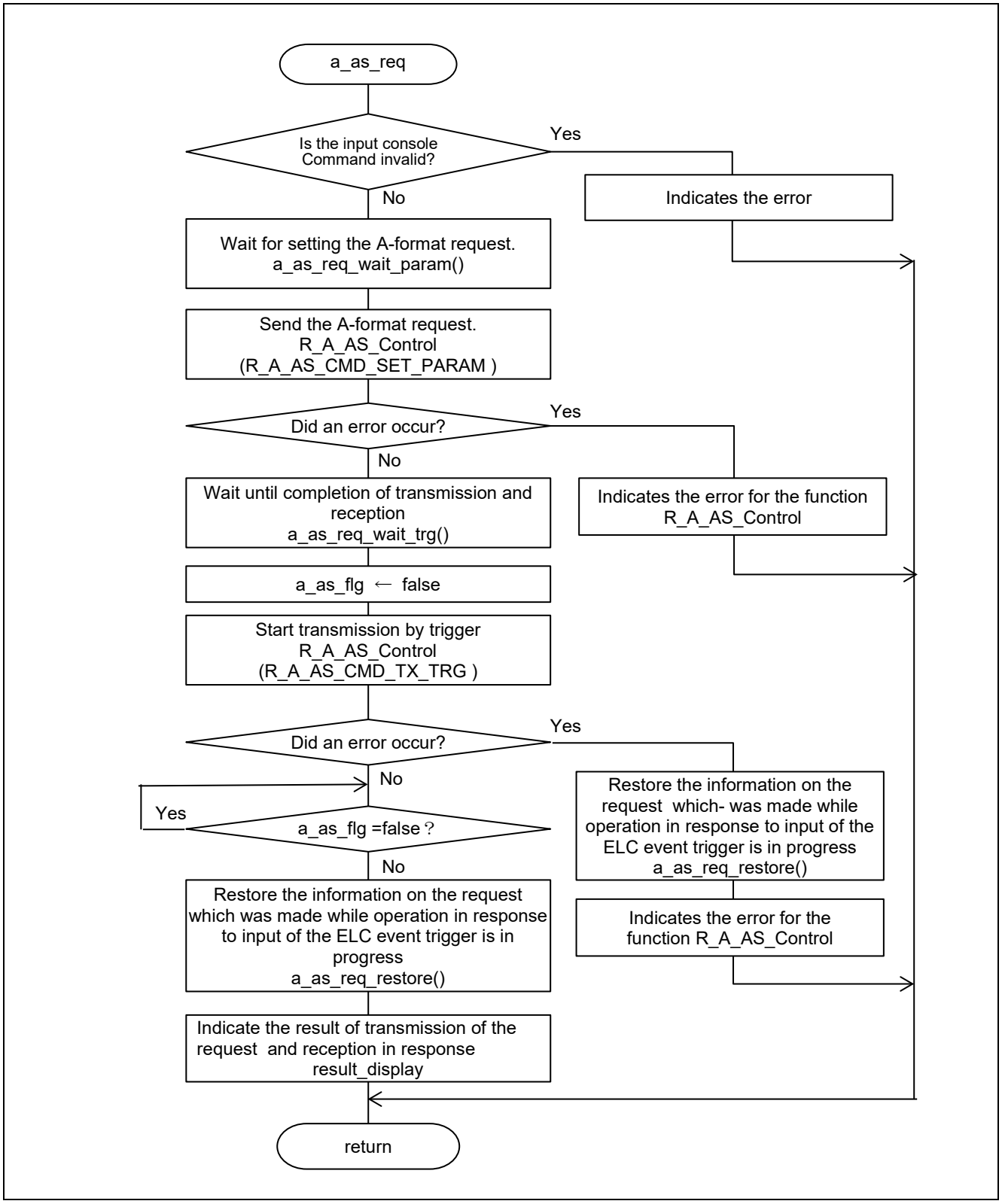


Figure 4.5 Flowchart of a_as_req

(4) Flowchart of a_as_elctimer

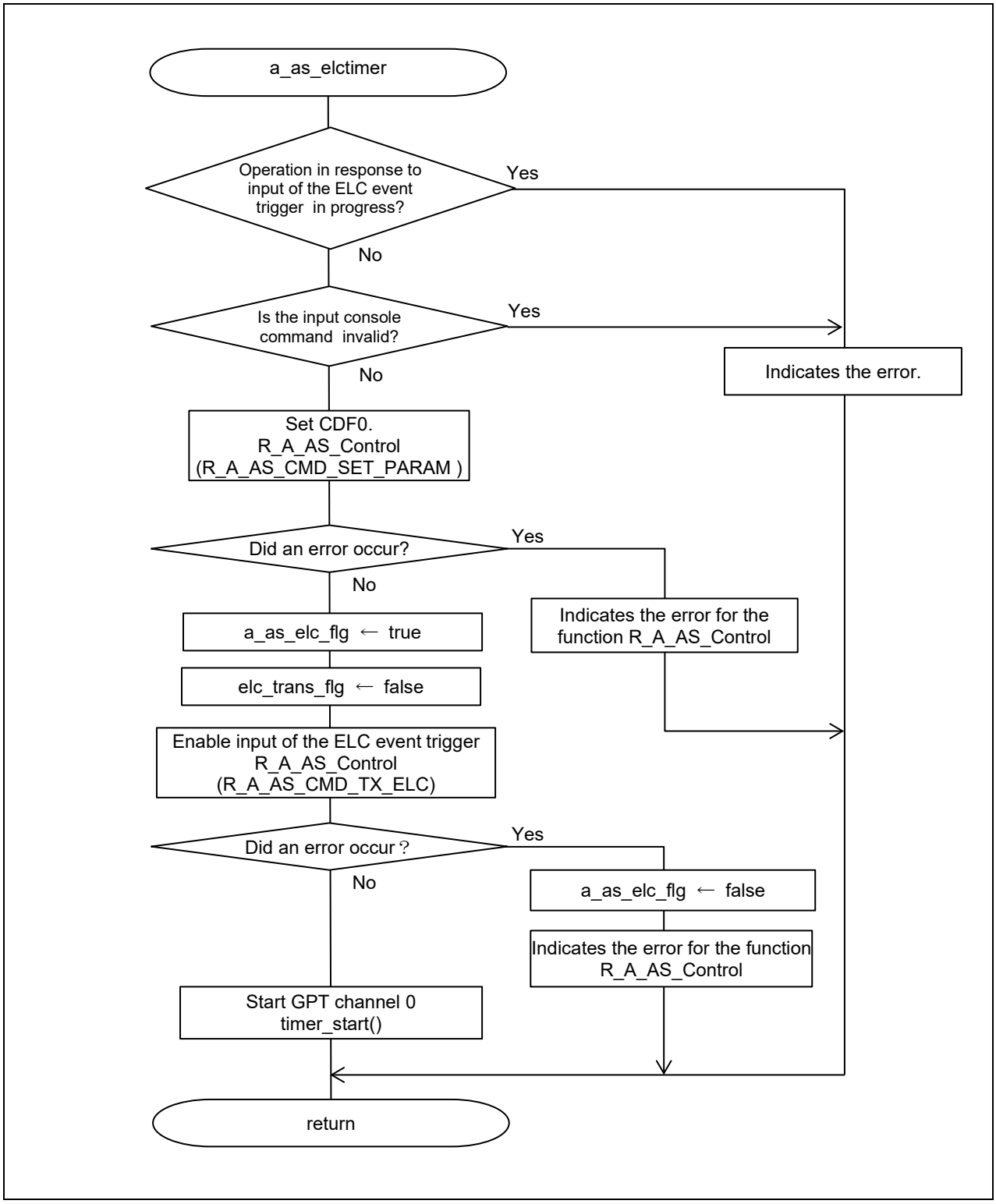


Figure 4.6 Flowchart of a_as_elctimer

(5) Flowchart of a_as_elcstop

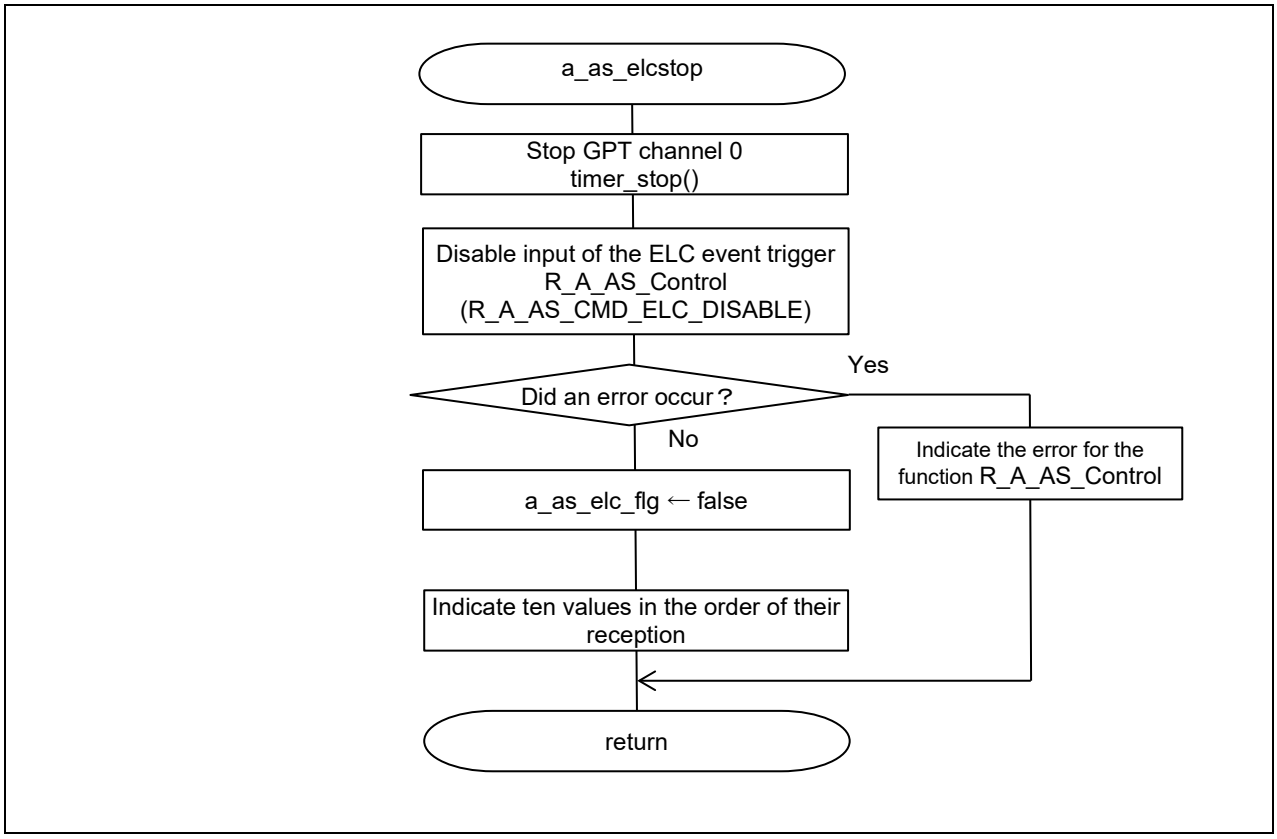


Figure 4.7 Flowchart of a_as_elcstop

(6) Flowchart of a_as_rxset_callback

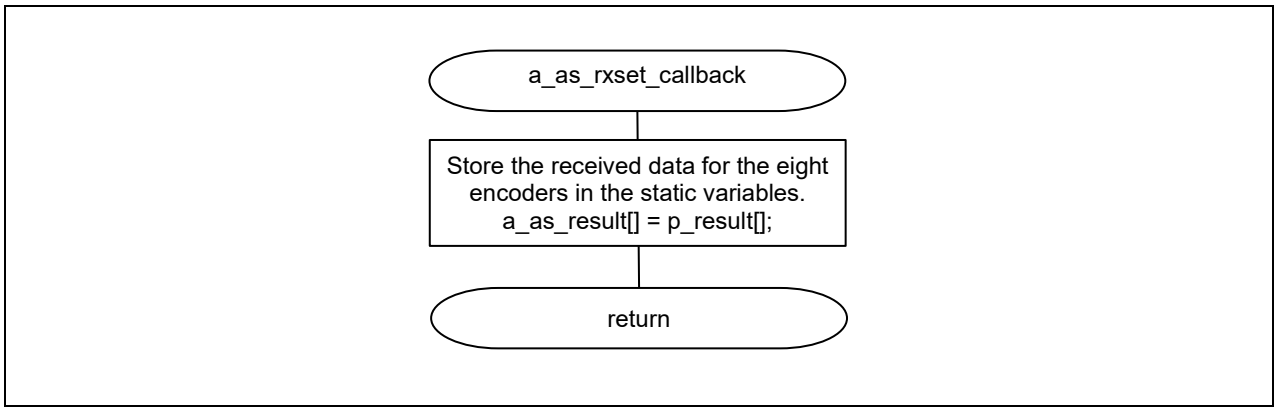


Figure 4.8 Flowchart of a_as_rxset_callback

(7) Flowchart of a_as_rxend_callback

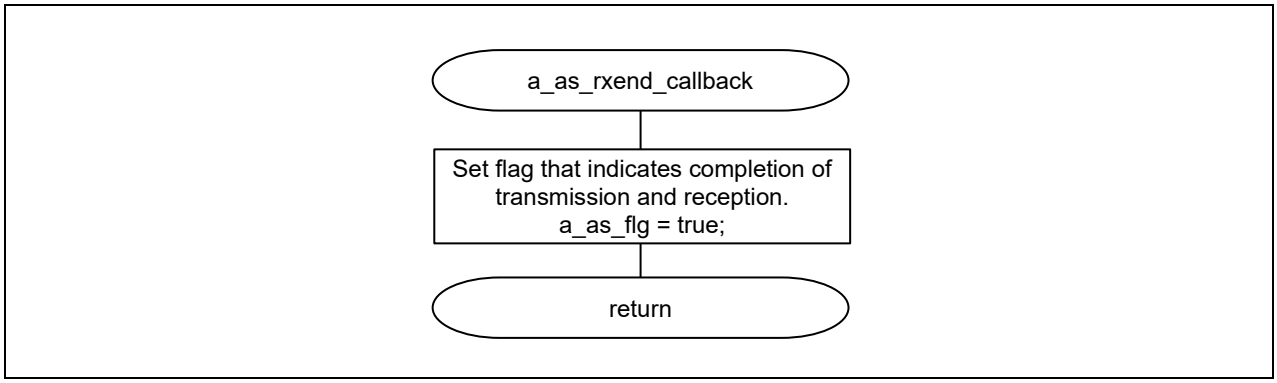


Figure 4.9 Flowchart of a_as_rxend_callback

(8) Flowchart of a_as_elctimer_callback

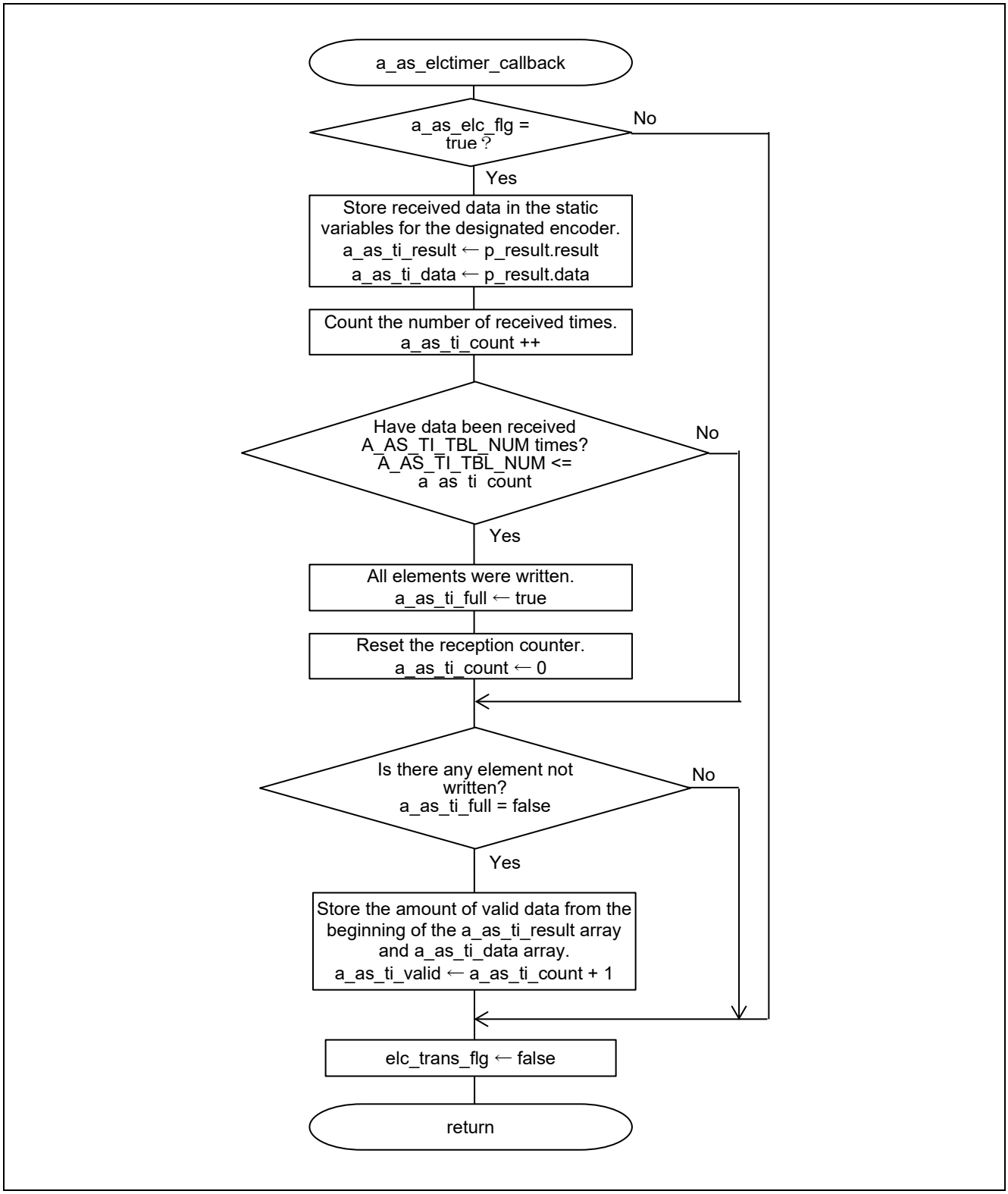


Figure 4.10 Flowchart of a_as_elctimer_callback

4.11.7 Operation Sequence

(1) Startup Sequence

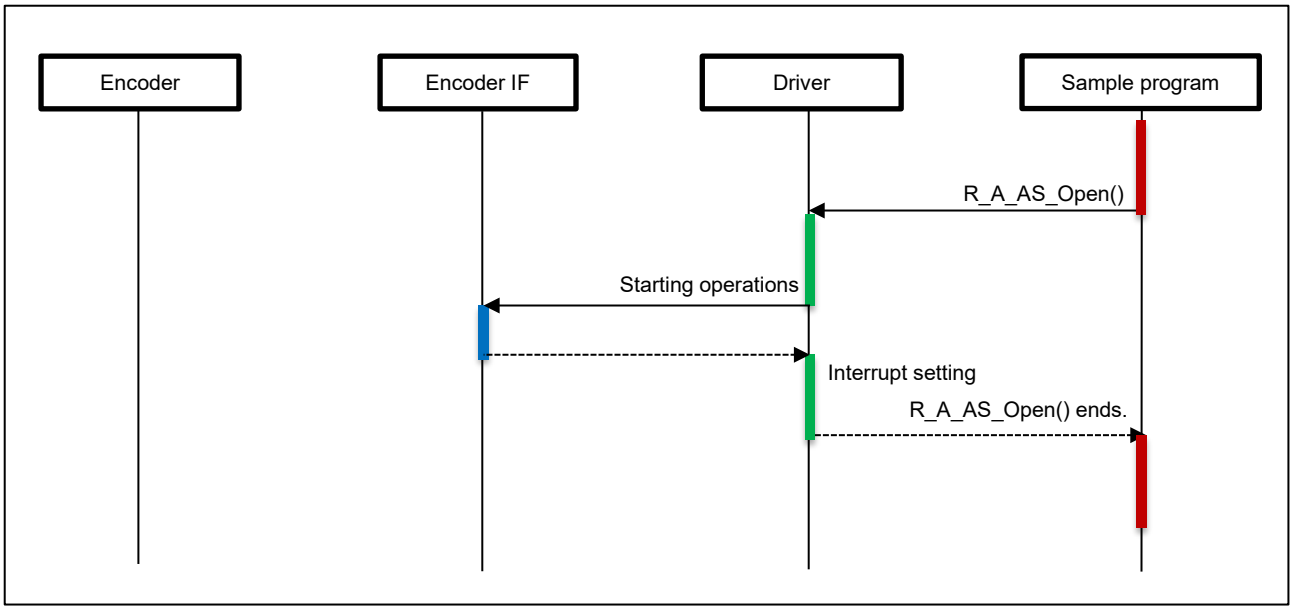


Figure 4.11 Startup Sequence Diagram

(2) Request Transmission and Data Reception Sequence

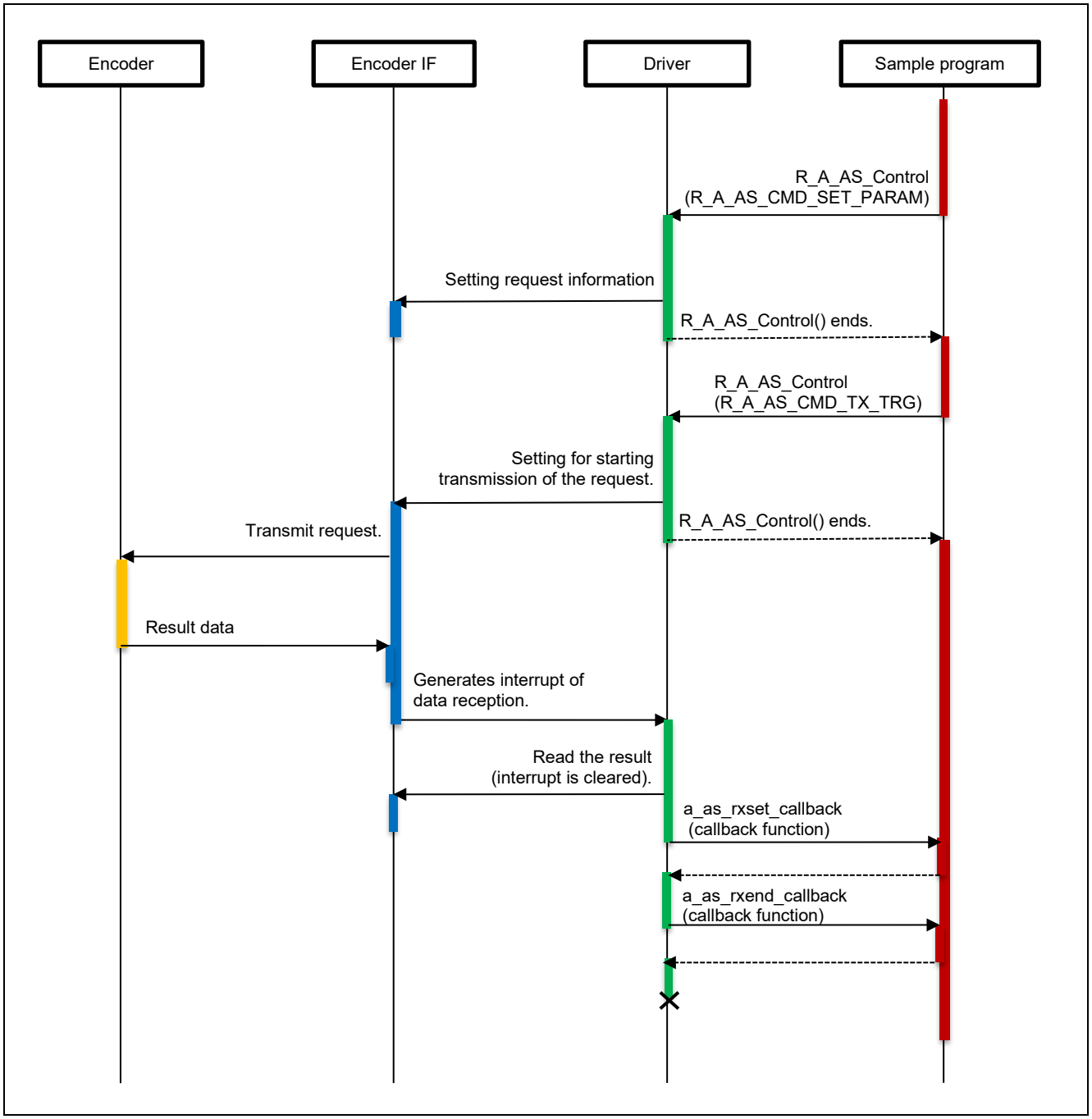


Figure 4.12 Request Transmission and Data Reception Sequence Diagram

(3) ELC Event Trigger Operation Sequence

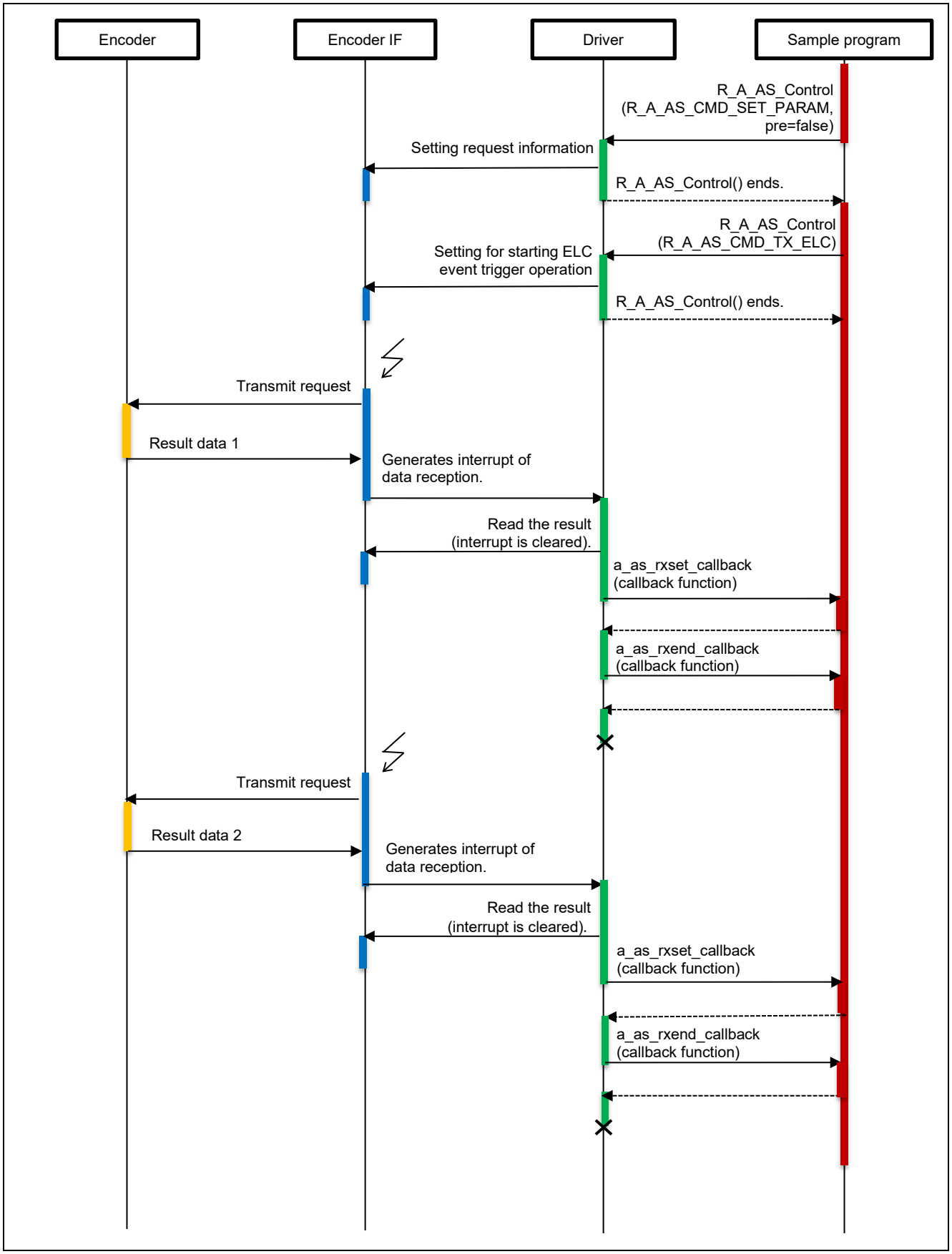


Figure 4.13 ELC Event Trigger Operation Sequence Diagram (1/2)

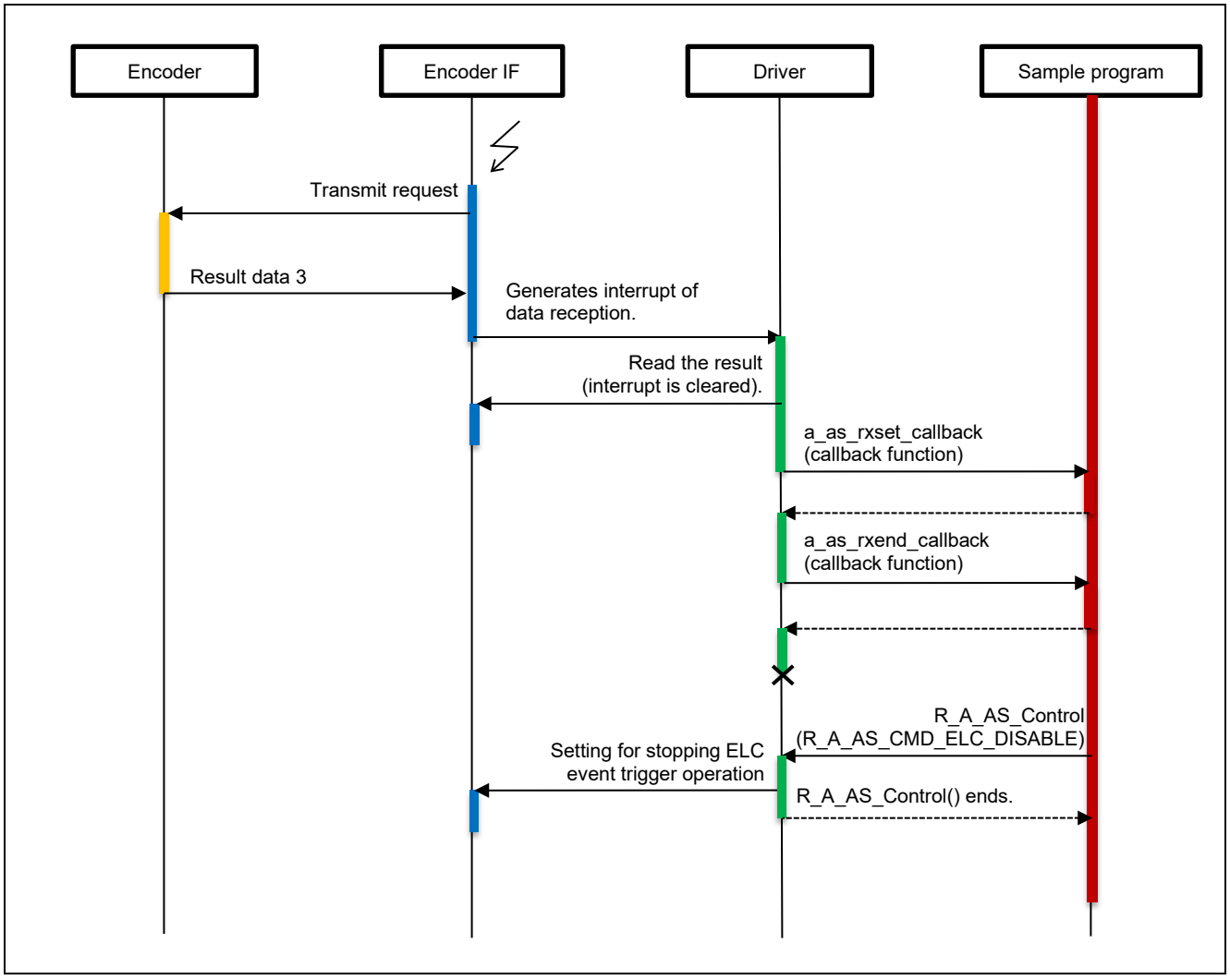


Figure 4.14 ELC Event Trigger Operation Sequence Diagram (2/2)

(4) Stop Sequence

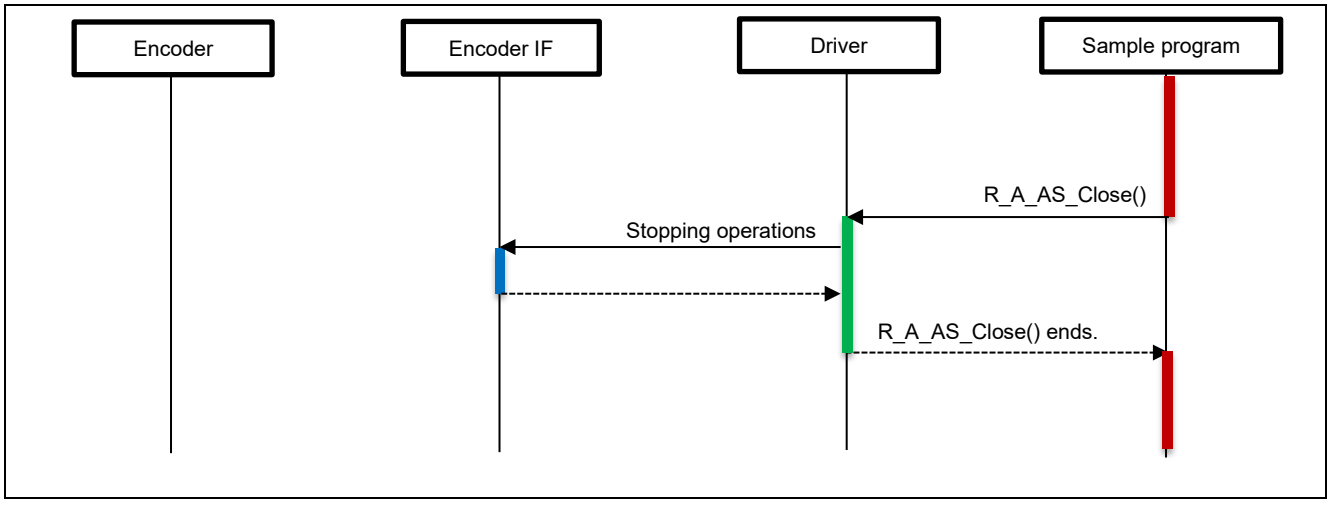


Figure 4.15 Stop Sequence Diagram

4.11.8 Console Commands

This sample program supports the MAR-M50A A-format compliant encoders. The commands which can be input through the console are as follows.

Table 4.18 List of Console Commands

Command	Content
req *ea cmd param1 param2*	Sends the command data frame input in the cmd to the encoder. *ea*: Encoder section number (decimal notation) *cmd*: Command data frame *param1* and *param2*: The values to be input depends on the *cmd* value. See “Table 4.19, req Commands and Corresponding Parameters”.
elctimer *ea val*	Sends the command data frame CDF0 to the designated encoder at 2.5 Mbps by the ELC event trigger. *ea*: Encoder section number (decimal notation) *val*: Timing of the trigger in microseconds. The maximum value allowed is 6990. Operation in response to input of the ELC event trigger is stopped by executing the console command "elcstop".
elcstop	Stops the operation in response to input of the ELC event trigger.
exit	Exits the program.

Table 4.19 req Commands and Corresponding Parameters

cmd	param1	param2
CDF0 to CDF12	None	None
CDF13	An address in EEPROM (in hexadecimal) Specify a value between 0x00 to 0xFF.	None
	Usage example (reading the data from address 0 of the section 2 encoder) req 2 CDF13 00	
CDF14	An address in EEPROM (in hexadecimal) Specify a value between 0x00 to 0xEF	The data to be written to EEPROM (in hexadecimal) Specify a value between 0x0000 to 0xFFFF.
	Usage example (writing 0x1234 to address 0 of the section 2 encoder) req 2 CDF14 00 1234	
CDF15 to CDF17	None	None
CDF18, CDF19	Recognition code (in hexadecimal) Specify a value between 0x00 0000 to 0xFF FFFF.	None
	Usage example (writing 0x12 3456 to the recognition code of the section 2 encoder) req 2 CDF18 123456	
CDF21, CDF27, CDF29	None	None

Note: Though input of CDF11, CDF17, and CDF19 are possible, they cannot be used because the sample program is for operation with a bus connection.
Multiple transmission commands (CDF4 to 7, CDF22, CDF28, CDF30) and CDF20 command are not supported.

(1) Result of Running

After running, it will display the command prompt following the version. Enter the command after “a_as >”.

```
A-format sample program start
R_A_AS_GetVersion = 4.0


a_as >
```

(2) Example of Command Execution

It is an example of executing console command that sends command data frame CDF0 to a connecting encoder ENC1. Encoder address, status and angular information are reported as the response from the encoder.

```
a_as >req 1 CDF0
req command
------------------------------
ENC1
R_A_AS_REQ_SUCCESS
EA : 0
ES : 0
CC : 0
ABS 40bit [39:32] : 0x00000005
ABS 40bit  [31:0] : 0x00560E11
```

5. Sample Code

The sample program is available on the Renesas Electronics website.

Revision History

Rev.	Date	Description	
		Page	Summary
0.80	Oct.04.22	-	First Edition issued.
1.00	Mar.31.23	5	Update board number of the operating environment.
2.00	May 17.24	5 21 38 41 45 46	Update description of the board name. Remove description about location of integer type definition. Change setting command of the R_A_AS_Control from CDF4 to CDF0 in the elctimer command. Change description of the storing data in the flowchart. Correct stop sequence diagram having Japanese characters. Change elctimer command usage of the table 4-18. Add notes for describing that multiple transmission commands are not supported in table 4-19.
3.00	Sep 26.25	1, 4, 5 11 - 16 8 - 16 25 29, 33 47	Change description for trademarks. Remove description of unsupported timer operation. Change description of headers for functions. Revise description of structure r_a_as_status_t. Add description of a_as_elctimer_callback in user-defined functions. Add example of command execution.
4.00	Apr 10.26	8 to 16, 31 to 33 13 to 16	Change prefix of pointer variables to “p_”. Revise header column of the specifications of user-defined functions.
4.01	Jul 03.26	7, 13 to 15, 20, 22, 25, 28, 34	Correct the processing to use data reception interrupts instead of a timeout interrupt. (Functions a_as_txerr_callback() and a_as_err_isr() are removed. ENCIF_ERR0 is removed from list of interrupts. Description of timeout error is corrected.)

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.

- A-format is a trademark of Nikon Corporation.
- IAR Embedded Workbench is a registered trademark of IAR Systems.
- Arm and Cortex are registered trademarks of Arm Limited (or its subsidiaries) in the EU and/or elsewhere. All rights reserved.
- Additionally all product names and service names in this document are a trademark or a registered trademark which belongs to the respective owners.

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/.

© 2026 Renesas Electronics Corporation. All rights reserved.