# RZ/T2M 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™ the communications protocol specification (“A-format™ specification”) by using the Encoder I/F Configuration Library (“EC-Lib”) of the RZ/T2M.

This sample program supports RZ/T2M Encoder I/F Configuration Data (A-format) Ver.2.0 and later.

The major features of the program are listed below.

- Supports A-format™ command codes.
- Capable of acquiring angle information, etc. from an encoder conforming to the A-format™ specification (MAR-M50A or SAR-HL700A from Nikon).

### Target Device

RZ/T2M

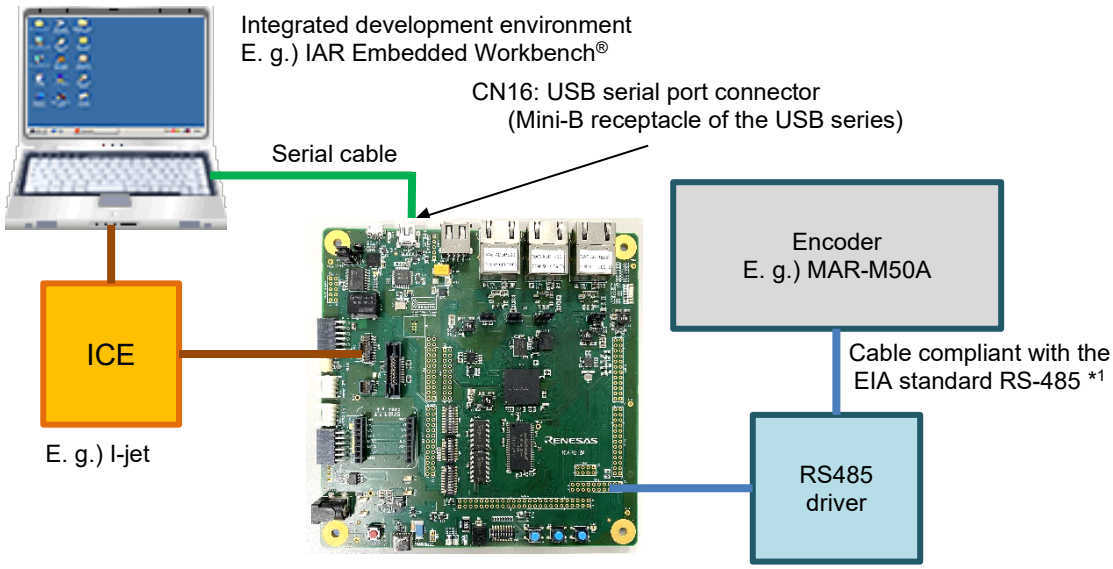
## Table of Contents

## 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 (A_AS) | Communications with the A-format compliant encoder. |
| Interrupt controller (ICU) | Controls the A_AS interrupts. |
| General PWM Timer (GPT) Channel 0 | Generates events at fixed intervals for input to the ELC. |
| Event link controller (ELC) | Makes the link between events output from channel 0 of the GPT and the A_AS 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/T2M group |
| 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®<br>RENESAS e² studio |
| Board | RSK+RZT2M (RTK9RZT2M0C00000BE) |
| Devices (function to be used on the board) | None |

Note: 1. Refer to the release note for the RZ/T2M 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/T2M 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 (Pin Function Name) | I/O Port | Input/Output | Description |
|---|---|---|---|---|
| A_AS0 | SD0 (ENCIF0) | P01_6 | Input | Data input |
| | CMND0 (ENCIF2) | P02_0 | Output | Data output |
| | D_R0 (ENCIF3) | P02_2 | Output | Drive/receive control |
| A_AS1 | SD1 (ENCIF5) | P17_3 | Input | Data input |
| | CMND1 (ENCIF7) | P17_5 | Output | Data output |
| | D_R1 (ENCIF8) | P03_0 | 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/T2M 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 | 7 |
| | R_A_AS_Close | 7 |
| | R_A_AS_GetVersion | 8 |
| | R_A_AS_Control | 8 |
| User-defined functions | a_as_txerr_callback | 12 |
| | a_as_rxset_callback | 13 |
| | a_as_rxend_callback | 14 |
| | a_as_elctimer_callback | 15 |
| Interrupt handlers | a_as0_int_isr | 17 |
| | a_as1_int_isr | 17 |

## 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 A_AS module.<br>1. Setting of the noise filter<br>2. Setting of the communication parameters (for T2,T3,T4,T5,T9,IFMG,BR registers)<br>3. Clearing of state information (SS register)<br>4. Setting for enabling interrupts by using the INTE register. |
| Argument | id : Designates the ID code to be used. (It is defined in r_a_as_rzt2_dat.h.)<br>R_A_AS0_ID : Specifies channel 0<br>R_A_AS1_ID : Specifies channel 1<br>Others : Setting is not allowed<br>p_info : Sets information about the encoder.<br>Designate the address of the structure r_a_as_info_t where encoder information is stored. |
| Return value | R_A_AS_SUCCESS: Normal termination<br>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.)<br>R_A_AS_ERR_ACCESS: Abnormal termination (already opened.) |
| Note | Before calling this function, be sure to configure and activate the multi-protocol encoder interface by using the EC-Lib.<br>This function configures the encoder interface.<br>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.)<br>R_A_AS0_ID : Specifies Channel 0<br>R_A_AS1_ID : Specifies Channel 1<br>Others : Setting is not allowed |
| Return Value | R_A_AS_SUCCESS: Normal termination<br>R_A_AS_ERR_INVALID_ARG: Abnormal termination (the id is not the stipulated value)<br>R_A_AS_ERR_ACCESS: Abnormal termination (a request is being transmitted.) |
| Note | In case completion of data transmission and reception (RXEND) interrupts are disabled, operation to control the encoder terminates even if the transmission of a request is in progress. Refer to the part of “R_A_AS_INTE” in “Table 4.6, User-defined Constants to be Used in the A-format Driver (r_a_as_rzt2_config.h)”.<br>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.<br>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<br>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<br>R_A_AS_ERR_INVALID_ARG: Abnormal termination (the id or cmd is not a stipulated value.)<br>See “4.4.4(1) Control Commands” for other returned values. | | |
| Note | Be sure to call R_A_AS_Open before calling this function.<br>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<br>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<br>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.)<br>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.<br>・R_A_AS_CMD_TX_TRG<br>・R_A_AS_CMD_TX_ELC<br>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. | |
| Arguments | 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.<br>R_A_AS_ERR_INVALID_ARG: Abnormal termination (the id is not a stipulated value.)<br>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.<br>In normal reception, 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_txerr_callback” to “4.5.3 a_as_rxend_callback” |
| Arguments | id : Designates the ID code to be used. (It is defined in r_a_as_rzt2_dat.h.)<br>R_A_AS0_ID : Specifies channel 0<br>R_A_AS1_ID : Specifies channel 1<br>Others : Setting is not allowed<br>cmd : Designates R_A_AS_CMD_TX_TRG<br>p_buf : Not used. (Designate NULL.) |
| Return value | R_A_AS_SUCCESS: Normal termination.<br>R_A_AS_ERR_INVALID_ARG: Abnormal termination (the id is not the stipulated value)<br>R_A_AS_ERR_BUSY: Abnormal termination (transmission is in progress)<br>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.<br>Use this command after setting request information by using control command R_A_AS_CMD_SET_PARAM.<br>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.<br>In normal reception, 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_txerr_callback” to “4.5.3 a_as_rxend_callback”<br>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.)<br>R_A_AS0_ID : Specifies channel 0.<br>R_A_AS1_ID : Specifies channel 1.<br>Others : Setting is not allowed.<br>cmd : Designates R_A_AS_CMD_TX_ELC.<br>p_buf : Not used. (Designate NULL.) |
| Return value | R_A_AS_SUCCESS: Normal termination<br>R_A_AS_ERR_INVALID_ARG: Abnormal termination (the id is not the stipulated value)<br>R_A_AS_ERR_BUSY: Abnormal termination (transmission or the ELC event trigger operation is in progress)<br>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.<br>Calling this control command from within a callback function is not allowed. |

### (e) R_A_AS_CMD_SETDF

| R_A_AS_CMD_SETDF | |
|---|---|
| Synopsis | Set type of data frame |
| 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 data frame type of the designated encoder. |
| Arguments | id : Designates the ID code to be used. (It is defined in r_a_as_rzt2_dat.h.)<br>R_A_AS0_ID : Specifies channel 0<br>R_A_AS1_ID : Specifies channel 1<br>Others : Setting is not allowed<br>cmd : Designates R_A_AS_CMD_SETDF<br>p_buf : Data frame information<br>Designate the pointer to the structure where the data frame information is stored. See section “4.10.1(3), r_a_as_setdf_t” for details. |
| Return value | R_A_AS_SUCCESS: Normal termination<br>R_A_AS_ERR_INVALID_ARG: Abnormal termination (the id is not a stipulated value.)<br>R_A_AS_ERR_ACCESS: Abnormal termination (transmission or the ELC event trigger operation is in progress) |
| Note | This control command is for A-format version 3.0 encoders or later.<br>Calling this control command from within a callback function is not allowed. |

## 4.5 Specification of User-defined Functions

### 4.5.1 a_as_txerr_callback

| a_as_txerr_callback | |
|---|---|
| Synopsis | Callback function that conveys the result of transmission and reception when a transmission error is generated (TXERR) in normal reception. |
| Header | r_a_as_rzt2_if.h |
| Declaration | void a_as_txerr_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 in normal reception. This function is called when an interrupt request is generated in response to a transmission error.<br>Note that this function is in the context of the interrupt handler. To secure responsiveness to interrupts, be sure to return from this function quickly. The given function name is an example, and the user can freely select the name. |
| Argument | p_result : Result of transmission and reception<br>This is the pointer to the array where the result of transmission and reception declared by the structure r_a_as_result_t is stored.<br>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.<br>See “Table 4.3, Results of Transmission and Reception” for details on the result of transmission and reception. |
| 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.<br>Depending on the timing of response from the encoder, this callback function may be called even if the function R_A_AS_Close, or R_A_AS_Control (R_A_AS_CMD_ELC_DISABLE) have been executed. |

### 4.5.2 a_as_rxset_callback

| a_as_rxset_callback | | |
|---|---|---|
| Synopsis | Callback function that conveys the result of transmission and reception when the setting of received data is complete (RXSET) in normal reception. | |
| Header | r_a_as_rzt2_if.h | |
| 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 in normal reception. This function is called when an interrupt request is generated in response to completion of setting of the received data.<br>Note that this function is in the context of the interrupt handler. To secure responsiveness to interrupts, be sure to return from this function quickly. The given function name is an example, and the user can freely select the name. | |
| Argument | p_result | : Result of transmission and reception<br>This is the pointer to the array where the result of transmission and reception declared by the structure r_a_as_result_t is stored.<br>See “Table 4.2, Result of Transmission and Reception Stored in Each Array Element” for the contents of 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.<br>Commands for multiple encoder addresses update the results of transmission and reception for encoder category ENC1.<br>See “Table 4.3, Results of Transmission and Reception” for details on the result of transmission and reception. |
| 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.<br>Depending on the timing of response from the encoder, this callback may be called even if the function R_A_AS_Close, or R_A_AS_Control (R_A_AS_CMD_ELC_DISABLE) have been executed. | |

### 4.5.3 a_as_rxend_callback

| a_as_rxend_callback | |
|---|---|
| Synopsis | Callback function that conveys the result of transmission and reception when the reception of data has been completed (RXEND) in normal reception. |
| Header | r_a_as_rzt2_if.h |
| 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 in normal reception. This function is called when an interrupt request is generated in response to the completion of data reception.<br>Note that this function is in the context of the interrupt handler. To secure responsiveness to interrupts, be sure to return from this function quickly. The given function name is an example, and the user can freely select the name. |
| Argument | p_result : Result of transmission and reception<br>This is the pointer to the array where the result of transmission and reception declared by the structure r_a_as_result_t is stored.<br>See “Table 4.2, Result of Transmission and Reception Stored in Each Array Element” for the contents of 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.<br>See “Table 4.3, Results of Transmission and Reception” for details on the result of transmission and reception. |
| 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.<br>Depending on the timing of response from the encoder, this callback may be called even if the function R_A_AS_Close or R_A_AS_Control (R_A_AS_CMD_ELC_DISABLE) have been executed. |

### 4.5.4 a_as_elctimer_callback

| a_as_elctimer_callback | |
|---|---|
| Synopsis | Callback function that conveys the result of transmission and reception when the data reception is complete in continuous operation of the ELC event trigger. |
| Header | r_a_as_rzt2_if.h |
| 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 in normal reception. This function is called every time when an interrupt request is generated in response to completion of the data reception.<br>Note that this function is in the context of the interrupt handler. To secure responsiveness to interrupts, be sure to return from this function quickly. The given function name is an example, and the user can freely select the name. |
| Argument | p_result : Result of transmission and reception<br>This is the pointer to the array where the result of transmission and reception declared by the structure r_a_as_result_t is stored.<br>See “Table 4.2, Result of Transmission and Reception Stored in Each Array Element” for the contents of 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.<br>See “Table 4.3, Results of Transmission and Reception” for details on the result of transmission and reception. |
| 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.<br>Depending on the timing of response from the encoder, this callback may be called even if the function R_A_AS_Close or R_A_AS_Control (R_A_AS_CMD_ELC_DISABLE) have 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 |
| Occurrence of a transmission error (TXERR) | Only valid when used in the callback functions | Invalid | Only valid when used in the callback functions. |
| Completion of the setting of received data (RXSET) | Only valid when used in the callback functions. | Valid *1 | Only valid when used in the callback functions. |
| Completion of the transmission and reception of data (RXEND) | Only valid when used in the callback functions. | Valid *1 *2 | Only valid when used in the callback functions. |

Note: 1. The result of data reception is valid until the transmission of the next request if the ELC event trigger is disabled. Otherwise, the result of data reception is valid until generation of the next ENCIF_INT0, ENCIF_INT4 interrupt.
2. IF TXERR has occurred, it will be invalid.

## 4.6 Interrupt Handler

### 4.6.1 a_as0_int_isr

| a_as0_int_isr | |
|---|---|
| Synopsis | Interrupt handler for ENCIF_INT0 |
| Header | - |
| Declaration | static void a_as0_int_isr(void); |
| Description | This is the interrupt handler for the following A_AS interrupt sources.<br>1. Completion of data transmission and reception.<br>2. Generation of a transmission error (due to transmission of an undefined command).<br>3. Completion of setting of received data. |
| Argument | None |
| Return value | None |

### 4.6.2 a_as1_int_isr

| a_as1_int_isr | |
|---|---|
| Synopsis | Interrupt handler for ENCIF_INT4 |
| Header | - |
| Declaration | static void a_as1_int_isr(void); |
| Description | This is the interrupt handler for the following A_AS interrupt sources.<br>1. Completion of data transmission and reception.<br>2. Generation of a transmission error (due to transmission of an undefined command).<br>3. Completion of setting of received data. |
| Argument | None |
| Return value | None |

## 4.7 Interrupt

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 |
|---|---|---|
| ENCIF_INT0 | 372 | This interrupt request is generated in response to the following source conditions in ch0.<br>1. Transmission and reception of data has been completed.<br>2. A transmission error is generated (due to the transmission of an undefined command)<br>3. Setting of received data is complete. |
| ENCIF_INT4 | 376 | This interrupt request is generated in response to the following source condition in ch1.<br>1. Transmission and reception of data has been completed.<br>2. A transmission error is generated (due to the transmission of an undefined command)<br>3. Setting of received data is complete. |

## 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 | Command Data Frame Selection |
| Table 4.14 | 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_A_AS_INTE | 0xC0 | The value of the interrupt enable register (setting this register enables interrupt sources RXEND, TXERR) |
| R_A_AS_NFINTV_2500KBPS | 0x00 | The NFINTV value when the bit rate is 2.5 Mbps. * |
| R_A_AS_NFINTV_4MBPS | 0x00 | The NFINTV value when the bit rate is 4 Mbps. * |
| R_A_AS_NFINTV_6670KBPS | 0x00 | The NFINTV value when the bit rate is 6.67 Mbps. * |
| R_A_AS_NFINTV_8MBPS | 0x00 | The NFINTV value when the bit rate is 8 Mbps. * |
| R_A_AS_NFSCNT_2500KBPS | 0x07 | The NFSCNT value when the bit rate is 2.5 Mbps. * |
| R_A_AS_NFSCNT_4MBPS | 0x04 | The NFSCNT value when the bit rate is 4 Mbps. * |
| R_A_AS_NFSCNT_6670KBPS | 0x02 | The NFSCNT value when the bit rate is 6.67 Mbps. * |
| R_A_AS_NFSCNT_8MBPS | 0x01 | The NFSCNT value when the bit rate is 8 Mbps. * |
| R_A_AS_T2_ONE_2500KBPS | 0x0014 | The value set in the T2 register when the connection is one-to-one, and the bit rate is 2.5 Mbps. * |
| R_A_AS_T2_ONE_4MBPS | 0x0014 | The value set in the T2 register when the connection is one-to-one, and the bit rate is 4 Mbps. * |
| R_A_AS_T2_ONE_6670KBPS | 0x0014 | The value set in the T2 register when the connection is one-to-one, and the bit rate is 6.67 Mbps. * |
| R_A_AS_T2_ONE_8MBPS | 0x0014 | The value set in the T2 register when the connection is one-to-one, and the bit rate is 8 Mbps. * |
| R_A_AS_T2_BUS_2500KBPS | 0x0096 | The value set in the T2 register when the connection is a bus connection, and the bit rate is 2.5 Mbps. * |
| R_A_AS_T2_BUS_4MBPS | 0x0064 | The value set in the T2 register when the connection is a bus connection, and the bit rate is 4 Mbps. * |
| R_A_AS_T2_BUS_6670KBPS | 0x0040 | The value set in the T2 register when the connection is a bus connection, and the bit rate is 6.67 Mbps. * |
| R_A_AS_T2_BUS_8MBPS | 0x003C | The value set in the T2 register when the connection is a bus connection, and the bit rate is 8 Mbps. * |
| R_A_AS_T3_2500KBPS | 0x001E | The value set in the T3 register when the bit rate is 2.5 Mbps. * |
| R_A_AS_T3_4MBPS | 0x0014 | The value set in the T3 register when the bit rate is 4 Mbps. * |
| R_A_AS_T3_6670KBPS | 0x0014 | The value set in the T3 register when the bit rate is 6.67 Mbps.* |
| R_A_AS_T3_8MBPS | 0x000E | The value set in the T3 register when the bit rate is 8 Mbps. * |
| R_A_AS_T4 | 0x00C8 | The value set in the T4 register |
| R_A_AS_T5_2500KBPS | 0x003C | The value set in the T5 register when the bit rate is 2.5 Mbps. * |
| R_A_AS_T5_4MBPS | 0x0028 | The value set in the T5 register when the bit rate is 4 Mbps. * |
| R_A_AS_T5_6670KBPS | 0x0028 | The value set in the T5 register when the bit rate is 6.67 Mbps.* |
| R_A_AS_T5_8MBPS | 0x001E | The value set in the T5 register when the bit rate is 8 Mbps. * |
| R_A_AS_T9_ONE | 0x005E | The value set in the T9 register when the connection is one-to-one. |
| R_A_AS_T9_BUS | 0x00C2 | The value set in the T9 register when the connection is a bus connection. |

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 |
| R_A_AS_CMD_SETDF | 0xAF000006 | Sets data frame types for A-format version 3.0 encoders or later. |

**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 | Defined for the command data frame CDF0. |
| R_A_AS_CDF1 | 1 | Defined for the command data frame CDF1. |
| R_A_AS_CDF2 | 2 | Defined for the command data frame CDF2. |
| R_A_AS_CDF3 | 3 | Defined for the command data frame CDF3. |
| R_A_AS_CDF4 | 4 | Defined for the command data frame CDF4. |
| R_A_AS_CDF5 | 5 | Defined for the command data frame CDF5. |
| R_A_AS_CDF6 | 6 | Defined for the command data frame CDF6. |
| R_A_AS_CDF7 | 7 | Defined for the command data frame CDF7. |
| R_A_AS_CDF8 | 8 | Defined for the command data frame CDF8. |
| R_A_AS_CDF9 | 9 | Defined for the command data frame CDF9. |
| R_A_AS_CDF10 | 10 | Defined for the command data frame CDF10. |
| R_A_AS_CDF11 | 11 | Defined for the command data frame CDF11. |
| R_A_AS_CDF12 | 12 | Defined for the command data frame CDF12. |
| R_A_AS_CDF13 | 13 | Defined for the command data frame CDF13. |
| R_A_AS_CDF14 | 14 | Defined for the command data frame CDF14. |
| R_A_AS_CDF15 | 15 | Defined for the command data frame CDF15. |
| R_A_AS_CDF16 | 16 | Defined for the command data frame CDF16. |
| R_A_AS_CDF17 | 17 | Defined for the command data frame CDF17. |
| R_A_AS_CDF18 | 18 | Defined for the command data frame CDF18. |
| R_A_AS_CDF19 | 19 | Defined for the command data frame CDF19. |
| R_A_AS_CDF20 | 20 | Defined for the command data frame CDF20. |
| R_A_AS_CDF21 | 21 | Defined for the command data frame CDF21. |
| R_A_AS_CDF22 | 22 | Defined for the command data frame CDF22. |
| R_A_AS_CDF23 | 23 | Defined for the command data frame CDF23. *1 |
| R_A_AS_CDF24 | 24 | Defined for the command data frame CDF24. *1 |
| R_A_AS_CDF25 | 25 | Defined for the command data frame CDF25. *1 |
| R_A_AS_CDF26 | 26 | Defined for the command data frame CDF26. *1 |
| R_A_AS_CDF27 | 27 | Defined for the command data frame CDF27. |
| R_A_AS_CDF28 | 28 | Defined for the command data frame CDF28. |
| R_A_AS_CDF29 | 29 | Defined for the command data frame CDF29. |
| R_A_AS_CDF30 | 30 | Defined for the command data frame CDF30. |
| R_A_AS_CDF13x | 113 | Defined for the command data frame CDF13 (FC=11). *1 |
| R_A_AS_CDF14x | 114 | Defined for the command data frame CDF14 (FC=11). *1 |
| R_A_AS_CDF16x | 116 | Defined for the command data frame CDF16 (FC=11). *1 |
| R_A_AS_CDF18x | 118 | Defined for the command data frame CDF18 (FC=11). *1 |
| R_A_AS_CDFx_OFFSET | 100 | Defined for offset of FC=11 commands against FC=00. |

Note: 1. It is command data frame for A-format version 3.0 encoders or later.

**Table 4.13 Command Data Frame Selection**

| Constant Name | Setting | Description *1 |
|---|---|---|
| R_A_AS_SET_CDF1 | 0 | Type of the command data frames CDF1 and CDF5. |
| R_A_AS_SET_CDF8 | 1 | Type of the command data frames CDF8 to CDF12. |

Note: 1. Setting of the data frame types is available for A-format version 3.0 encoders or later.

**Table 4.14 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 Integer

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

**Table 4.15 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 Structure/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 T2, T3, T5, and T9 registers.
    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 T2, T3, T5, T9, and BR registers.
    uint16_t    ifmg;       Margin value
                            Designate the timer margin for use in monitoring by the watchdog timer
                            of the time to start receiving the encoder data (IF).
                            Note: This setting is reflected in the IFMG register.
} 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<br>Designate the encoder address. See “Table 4.11, Encoder Addresses” for the value to be designated.<br>This setting is reflected in the EA bit of the TXC register. |
| uint8_t | cmd; | Command<br>Designate the command code to be sent to the encoder. See “Table 4.12, Commands” for the value to be designated.<br>If a value other than those listed in the table is designated, a transmission error interrupt is generated upon sending of the value. Additionally, 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.<br>Some commands are not available depending on the connection method. |
| uint8_t | memadr; | Memory address<br>Designate the address of the memory in the encoder. *1<br>Set this value only in the following cases:<br>cmd = R_A_AS_CDF13<br>cmd = R_A_AS_CDF14<br>cmd = R_A_AS_CDF13x<br>cmd = R_A_AS_CDF14x |
| uint8_t | membank; | Memory bank<br>Designate the memory bank of the encoder.<br>Set this value only in the following cases:<br>cmd = R_A_AS_CDF13x<br>cmd = R_A_AS_CDF14x |
| uint16_t | memdat; | Data to be written to the memory<br>Designate the data to be written to the memory.<br>Set this value only in the following cases:<br>cmd = R_A_AS_CDF14<br>cmd = R_A_AS_CDF14x |
| uint32_t | encid; | Recognition code or velocity coefficient<br>Designate the 24-bit recognition code.<br>Set this value only in the following cases:<br>cmd = R_A_AS_CDF18<br>cmd = R_A_AS_CDF19<br>cmd = R_A_AS_CDF20<br>Designate the 19-bit velocity coefficient.<br>Set this value only in the following case:<br>cmd = R_A_AS_CDF18x |
| r_a_as_result_cb_t | cbadr_txerr | The pointer to the callback function *2 to be called in response to an ENCIF_INT interrupt request issued due to TXERR. See “4.5.1, a_as_txerr_callback” for details. |
| r_a_as_result_cb_t | cbadr_rxset; | The pointer to the callback function *2 to be called in response to an ENCIF_INT interrupt request issued due to RXSET. See “4.5.2, a_as_rxset_callback” for details. |
| r_a_as_result_cb_t | cbadr_rxend; | The pointer to the callback function *2 to be called in response to an ENCIF_INT interrupt request issued due to RXSET. See “4.5.3, 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, R_A_AS_CDF13x or R_A_AS_CDF14x, 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_setdf_t

Information of the data frame type settings

typedef struct
{

| | | |
|---|---|---|
| uint8_t | encadr; | Encoder address<br>Designate the encoder address. See “Table 4.11, Encoder Addresses” for the value to be designated. |
| bool | encmod; | Data frame type extension for the CDF1 and CDF5<br>false: Data frame for the CDF1 and CDF5 is received as ABS LSB 24-bit.<br>true: Data frame for the CDF1 and CDF5 is received as ABS full 40-bit and velocity. |

} r_a_as_result_t

### (4) r_a_as_result_t

Result of transmission and reception in normal reception

typedef struct
{

| | | |
|---|---|---|
| r_a_as_req_err_t | result; | Result of transmission and reception<br>See the enumerated type r_a_as_req_err_t for details. |
| r_a_as_data_t | data; | Reception data<br>See the structure type r_a_as_data_t for details. |
| r_a_as_status_t | status; | State of the A_AS<br>See the structure type r_a_as_status_t for details. |

} r_a_as_result_t

### (5) r_a_as_data_t

Data received in normal reception

```
typedef struct
{
    uint32_t        rxi;        RXI register value
                                The value of the RXI register is stored here.
    uint32_t        rxd0;       RXD0 register value
                                The value of the RXD0 register is stored here.
    uint32_t        rxd1;       RXD1 register value
                                The value of the RXD1 register is stored here.
    uint32_t        rxd2;       RXD1 register value
                                The value of the RXD2 register is stored here.
} r_a_as_data_t
```

### (6) r_a_as_status_t

State of the A_AS on transmission or reception

```
typedef struct
{
    bool        iwdgerr;    IF watchdog errors. (true: occurred, false: not occurred)
    bool        dwdgerr;    DF watchdog errors. (true: occurred, false: not occurred)
    bool        starterr;   Start bit errors. (true: occurred, false: not occurred)
    bool        stoperr;    Stop bit errors. (true: occurred, false: not occurred)
    bool        syncerr;    Sync code errors. (true: occurred, false: not occurred)
    bool        rxeaerr;    Received encoder address errors. (true: occurred, false: not occurred)
    bool        crcerr;     CRC errors. (true: occurred, false: not occurred)
    bool        rxccerr;    Received command code errors. (true: occurred, false: not occurred)
    bool        mdaterr;    EEPROM data errors. (true: occurred, false: not occurred)
    bool        madrerr;    EEPROM address errors. (true: occurred, false: not occurred)
    bool        rxdzerr;    Recognition code errors. (true: occurred, false: not occurred)
    bool        fd1err;     Fixed-data (1) errors. (true: occurred, false: not occurred)
    bool        fd2err;     Fixed-data (2) errors. (true: occurred, false: not occurred)
    bool        fd3err;     Fixed-data (3) errors. (true: occurred, false: not occurred)
    bool        fd5err;     Fixed-data (5) errors. (true: occurred, false: not occurred)
    bool        fd6err;     Fixed-data (6) errors. (true: occurred, false: not occurred)
    bool        fd7err;     Fixed-data (7) errors. (true: occurred, false: not occurred)
    bool        bankerr;    Bank errors. (true: occurred, false: not occurred)
    bool        elcin;      ELCIN input information is stored.
                            (true: Data transfer has started by ELC event trigger input.)
    uint8_t     txcc;       Transmission command code is stored.
                            (0: CDF0 to 20 and 23 to 30, 1: CDF21, 2: CDF22)
    bool        rxset;      Whether the received data is ready. (true: ready, false: not ready)
    bool        timer;      Timer status information. (true: operating, false: stopped)
    bool        txerr;      Transmission errors. (true: occurred, false: not occurred)
    bool        rxend;      Completion of reception. (true: completed, false: not completed)
} r_a_as_status_t
```

### 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 or SAR-HL700A). 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 TRG 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 A_AS module. The ELC links input events for the A_AS module to events output by the GPT. See “Figure 4.7, Flowchart of a_as_elctimer” for an example setting of input events.

#### (1) System Block Diagram

A system block diagram is shown below.

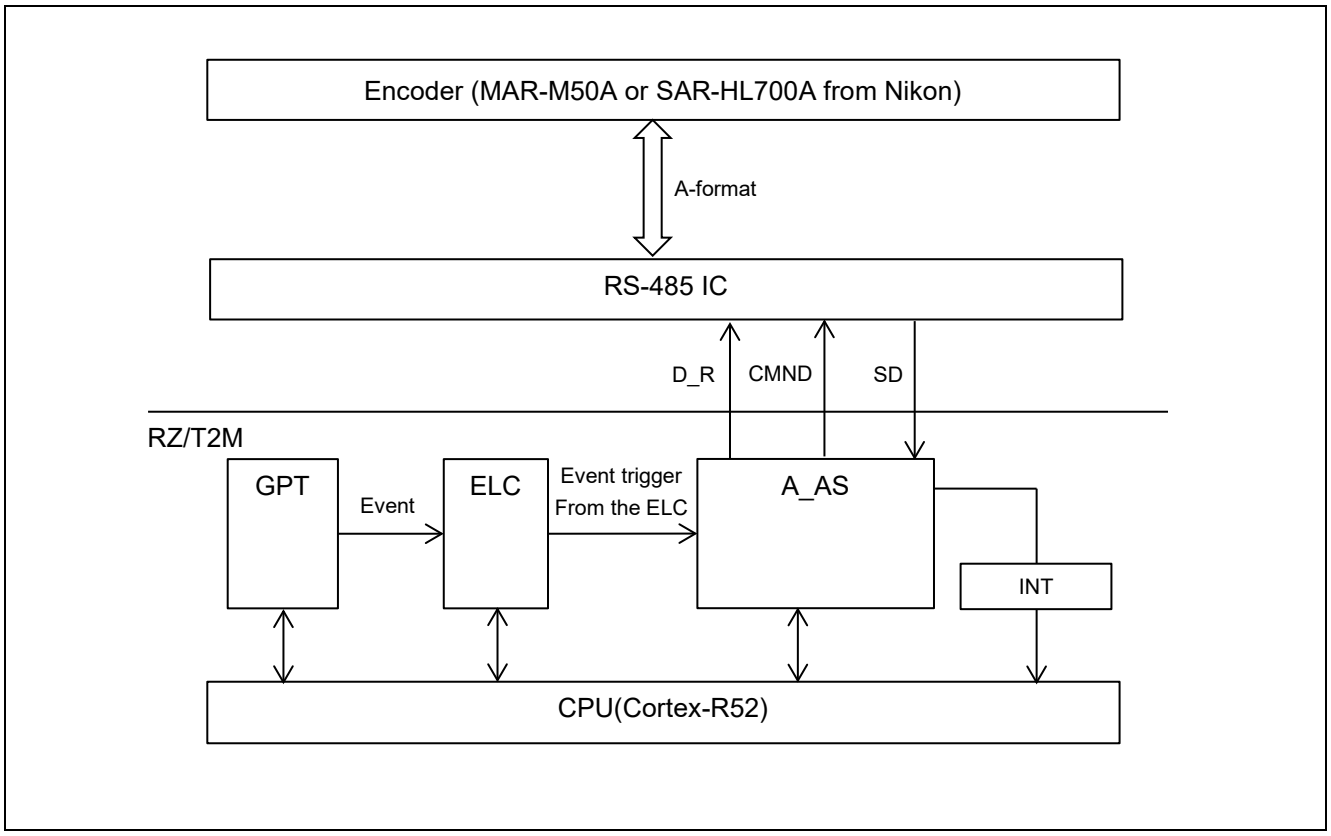



**Figure 4.1 System Block Diagram**

### (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.

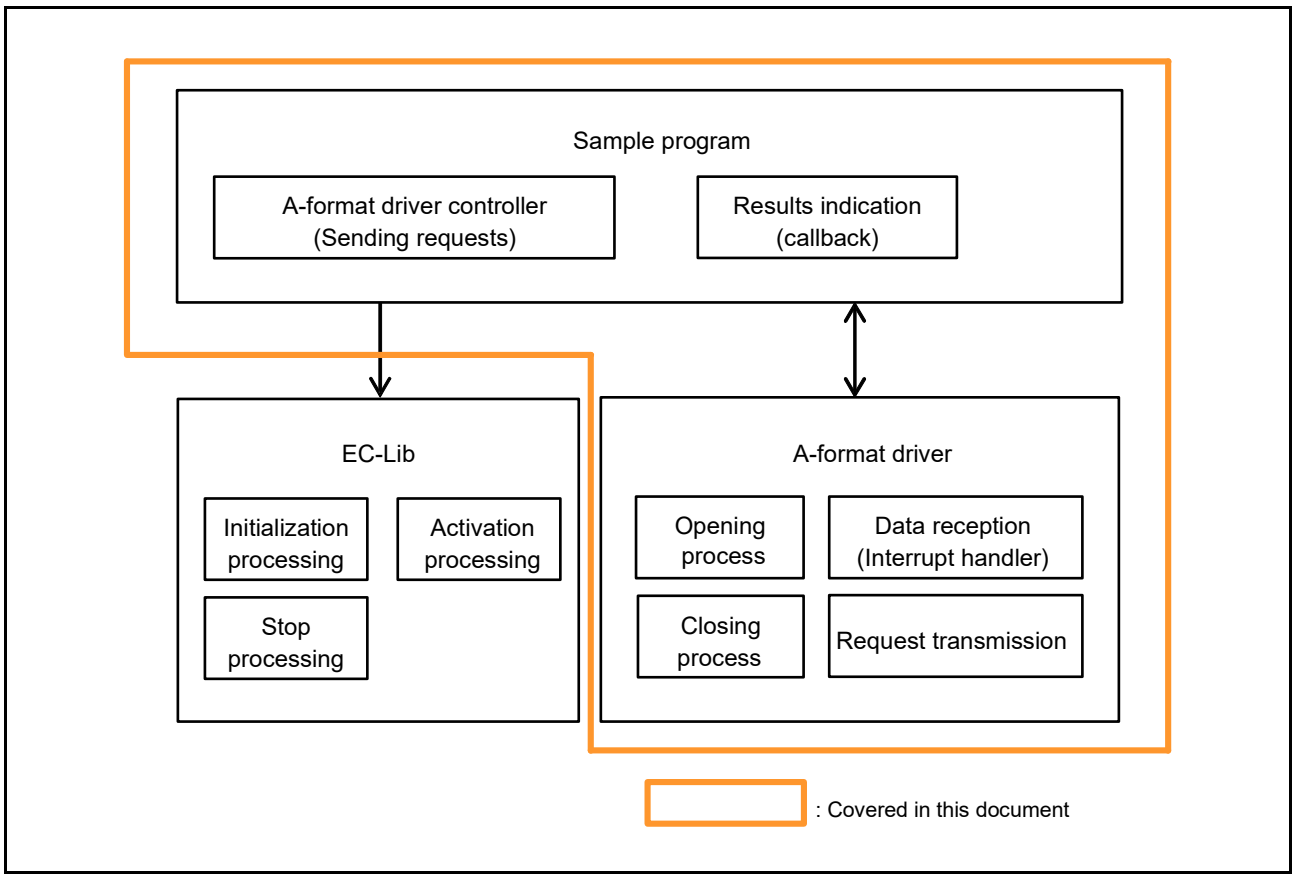



**Figure 4.2 Software Structure**

### 4.11.2 Functions Used in the Sample Program

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

**Table 4.16 Major Functions Used in the Sample Program**

| Function Name | Page |
|---|---|
| hal_entry | 31 |
| enc_main | 31 |
| a_as_cmd_control | 31 |
| a_as_enc_init | 31 |
| a_as_req | 32 |
| a_as_setdf | 32 |
| a_as_elctimer | 32 |
| a_as_elcstop | 32 |
| a_as_exit | 33 |
| a_as_txerr_callback | 33 |
| a_as_rxset_callback | 33 |
| a_as_rxend_callback | 33 |
| a_as_elctimer_callback | 34 |

### 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. |
| 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<br>0: specify channel 0, 1: specify channel 1 |
| Return value | 0: Normal termination<br>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.<br>Starting control of the encoder<br>Input processing of the console command<br>Ending control of the encoder | |
| Argument | id | Encoder ID<br>R_A_AS0_ID: specify Encoder ID ch0<br>R_A_AS1_ID: specify Encoder ID ch1 |
| 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<br>R_A_AS0_ID: Specify Encoder ID ch0<br>R_A_AS1_ID: Specify Encoder ID ch1 |
| Return value | 0: Normal termination<br>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_setdf

| a_as_setdf | | |
|---|---|---|
| Synopsis | Console command “setdf” function | |
| Header | - | |
| Declaration | static void a_as_setdf(uint32_t arg_num, char_t *p_arg[]); | |
| Description | This function is executed when the console command "setdf" is input. See section “4.11.6(3), Flowchart of a_as_req” and “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_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(5), Flowchart of a_as_elctimer” and section “4.11.8, Console Commands” for details. | |
| Arguments | 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_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(6), Flowchart of a_as_elcstop” and “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 calue | None | |

### (9) a_as_exit

| a_as_exit | | |
|---|---|---|
| Sunopsis | 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 | |

### (10) a_as_txerr_callback

| a_as_txerr_callback | |
|---|---|
| Synopsis | Callback for the console command “req” when a transmission error is generated |
| Header | - |
| Declaration | static void a_as_txerr_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 transmitted A-format request in normal reception in the variable a_as_result and indicates that a transmission error is generated. See section “4.11.6(7), Flowchart of a_as_txerr_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_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 normal reception in the variable a_as_result and indicates completion of the setting of received data. See section “4.11.6(8), 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 | |

### (12) 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 in normal reception. See section “4.11.6(9), 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 |

**(13) 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(10), 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 4.17 Static Variables Used in the Sample Program**

| Type | Variable Name | Description |
|---|---|---|
| bool | a_as_flg | Transmission and reception completion flag.<br>(true: transmission and reception completed;<br>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<br>(true: ELC event input trigger operation is in progress;<br>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.<br>(true: Transmission and reception in progress,<br>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. |
| uint8_t | enc_df_type[A_AS_ENC_NUM] | Data frame types for CDF1, CDF5, and CDF8 to CDF12 of the A-format version 3.0 encoders or later are stored. |

### 4.11.5 Constants Used in the Sample Program

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

**Table 4.18 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 Processing

The flowcharts for the major processes are given below.

#### (1) Flowchart of enc_main

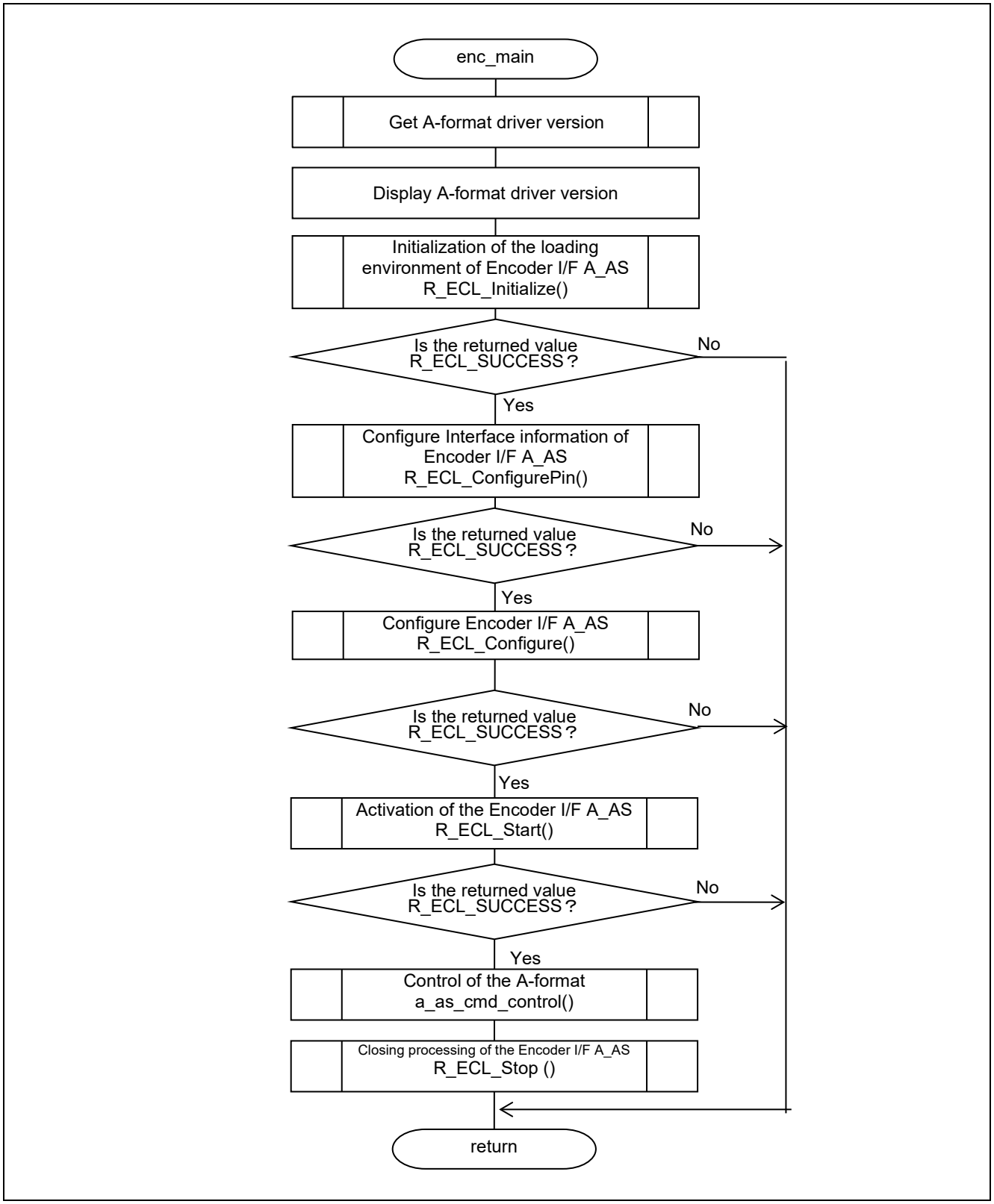



**Figure 4.3 Flowchart of enc_main**

### (2) Flowchart of a_as_cmd_control

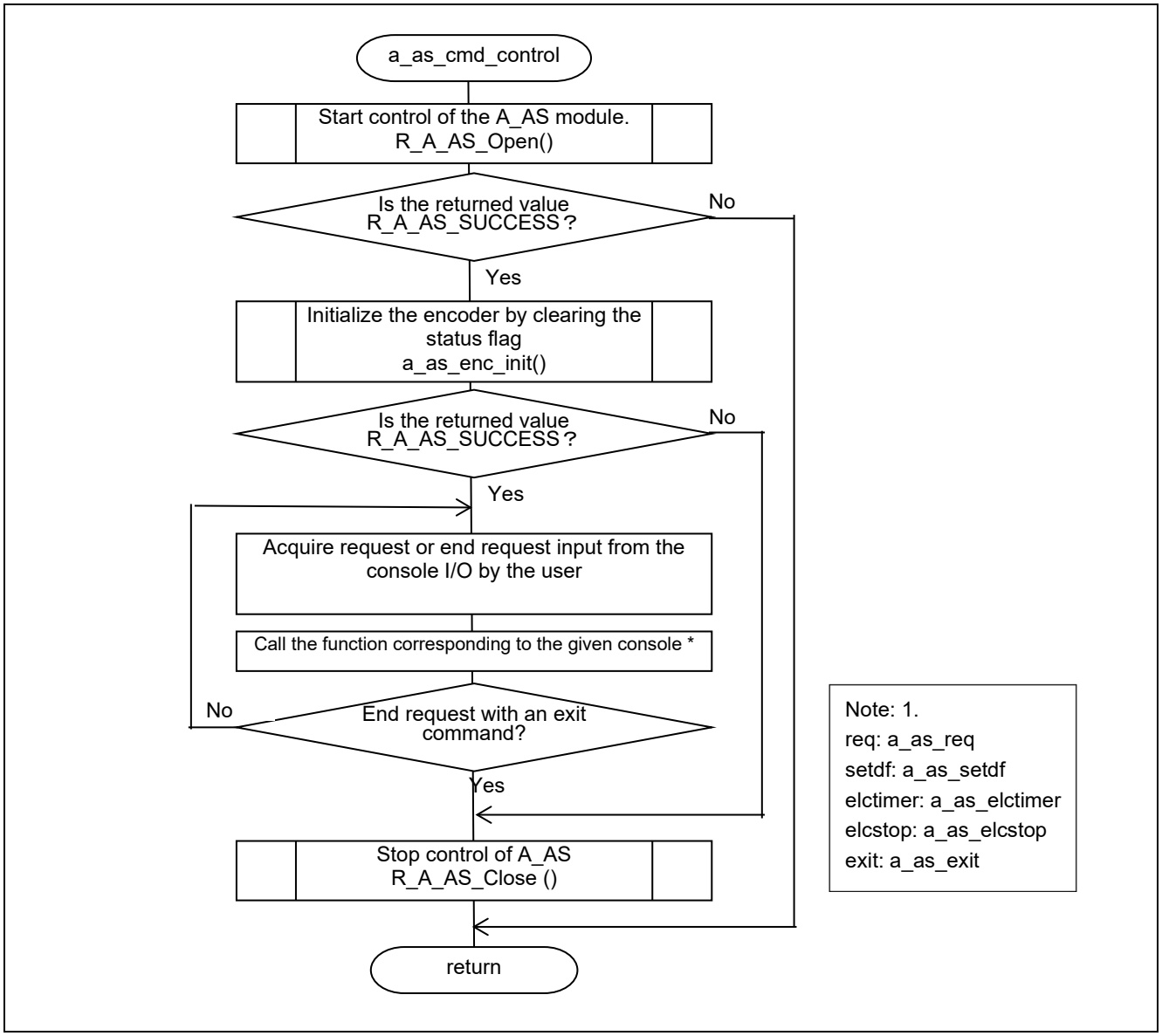



**Figure 4.4 Flowchart of a_as_cmd_control**

### (3) Flowchart of a_as_req

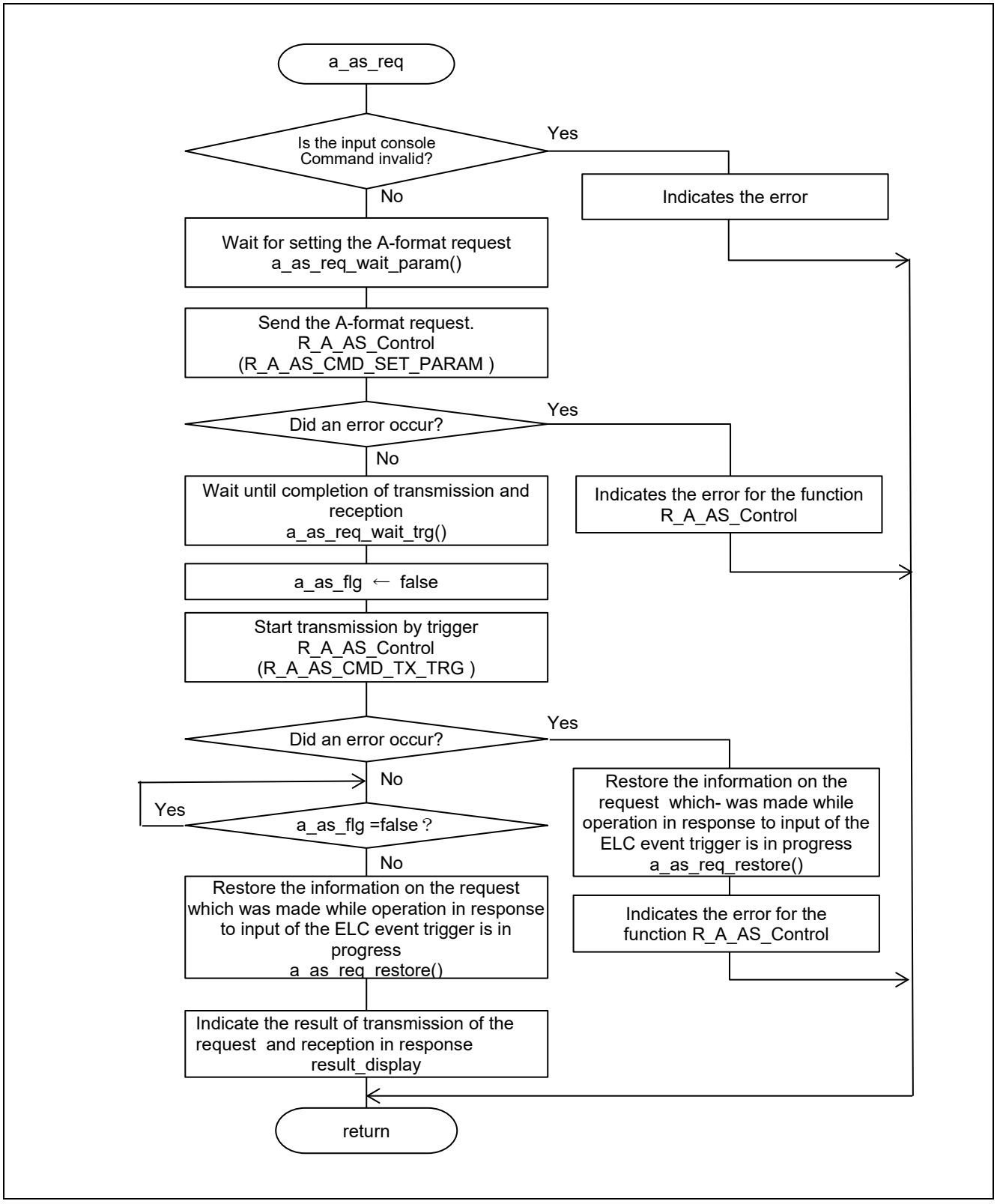



**Figure 4.5 Flowchart of a_as_req**

### (4) Flowchart of a_as_setdf

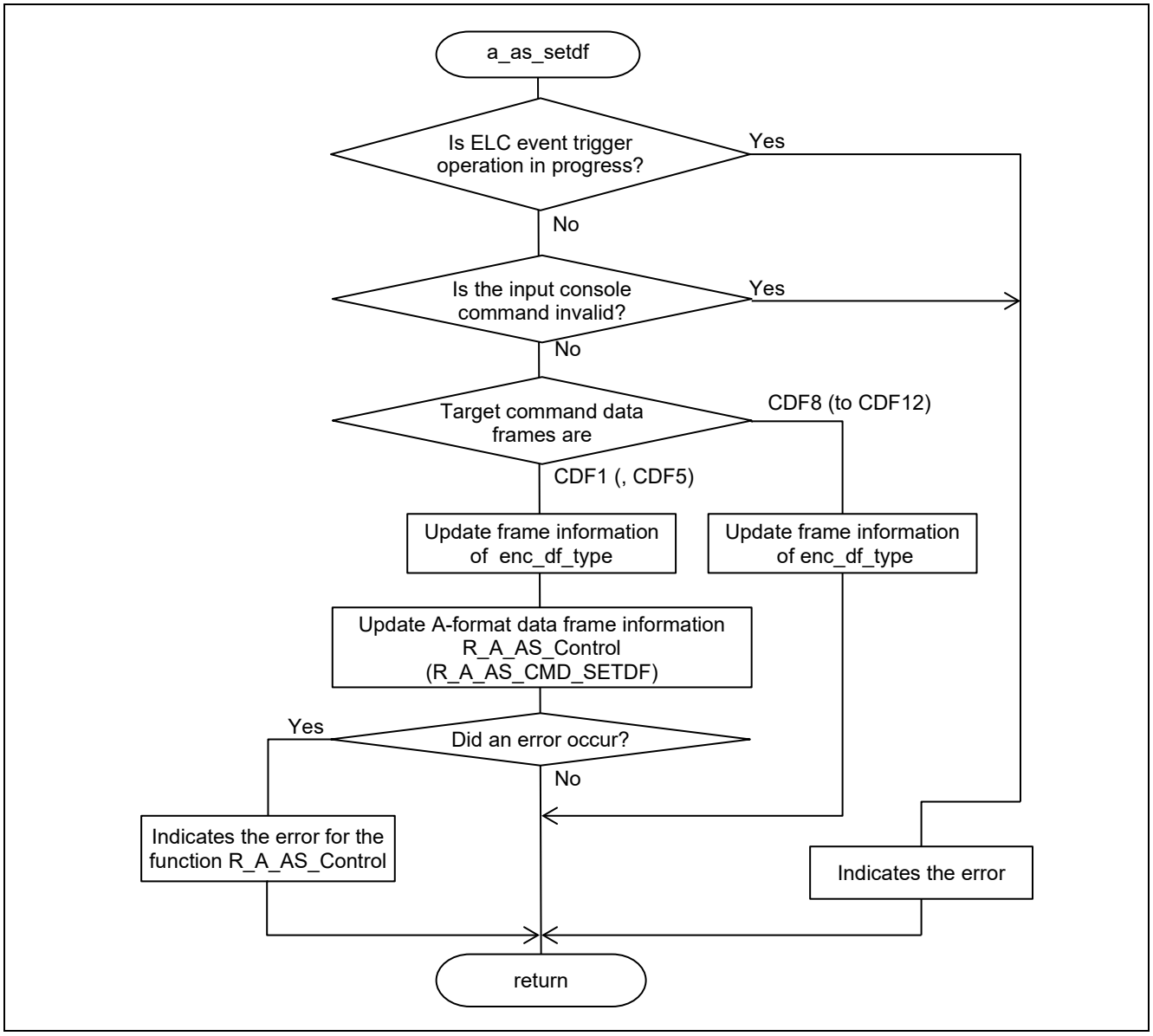



**Figure 4.6 Flowchart of a_as_setdf**

### (5) Flowchart of a_as_elctimer

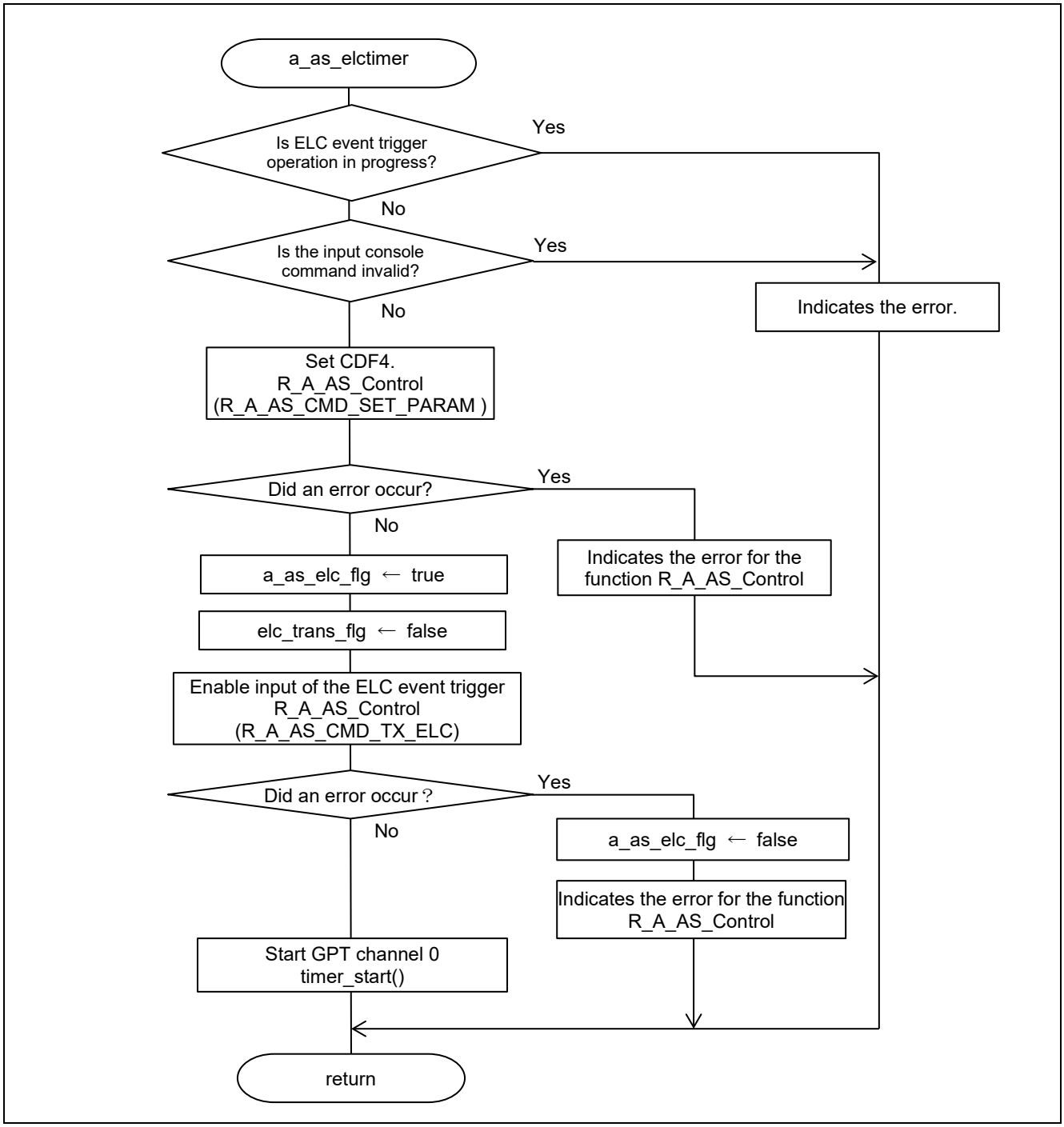



**Figure 4.7 Flowchart of a_as_elctimer**

### (6) Flowchart of a_as_elcstop

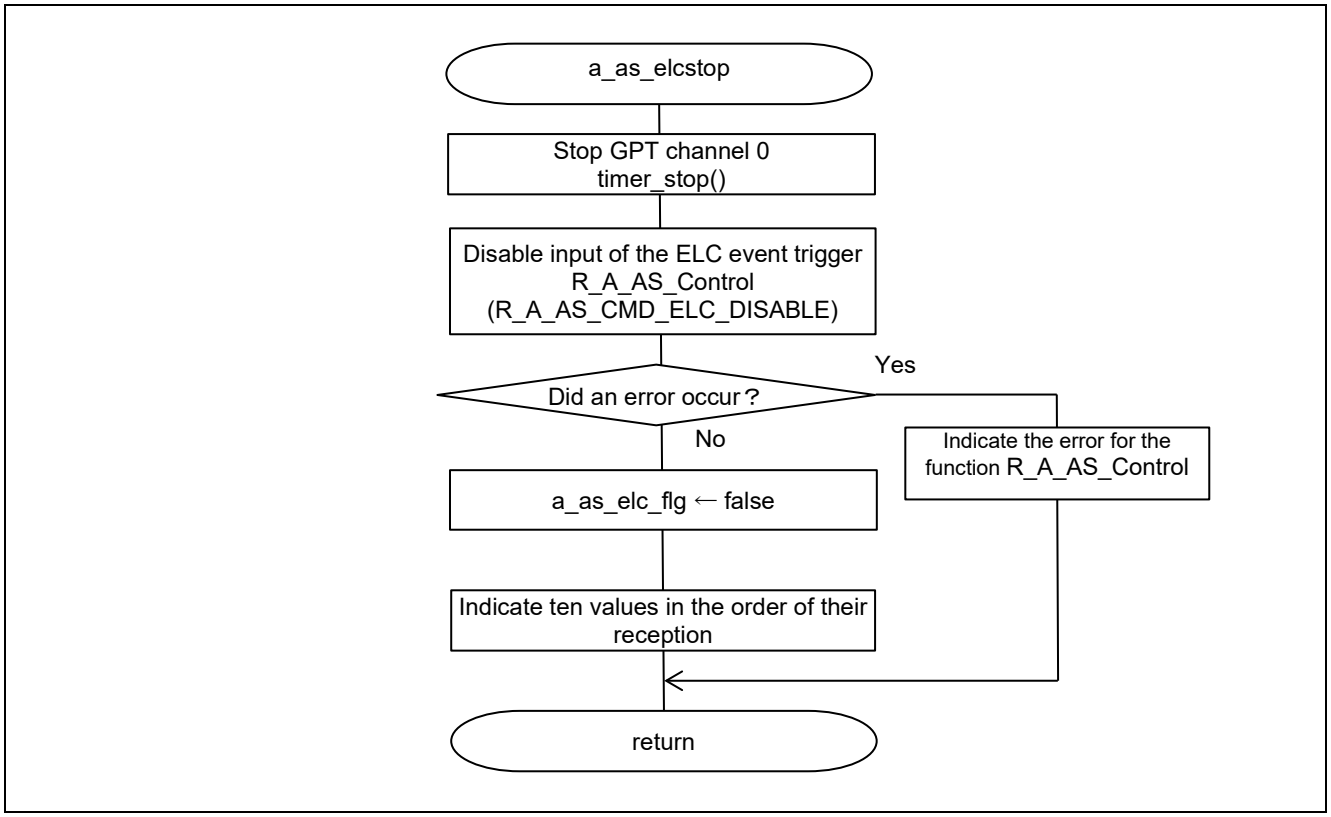



**Figure 4.8 Flowchart of a_as_elcstop**

### (7) Flowchart of a_as_txerr_callback

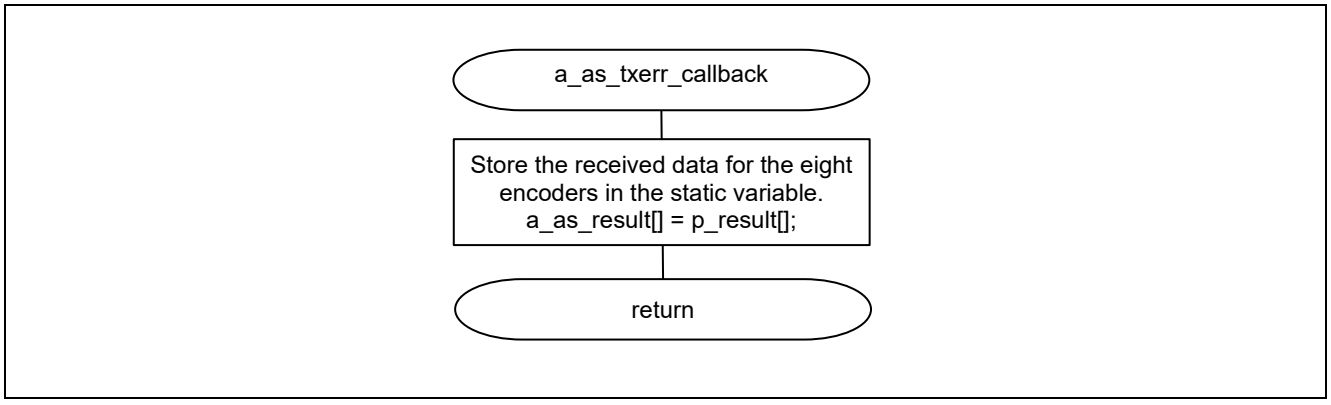



**Figure 4.9 Flowchart of a_as_txerr_callback**

### (8) Flowchart of a_as_rxset_callback

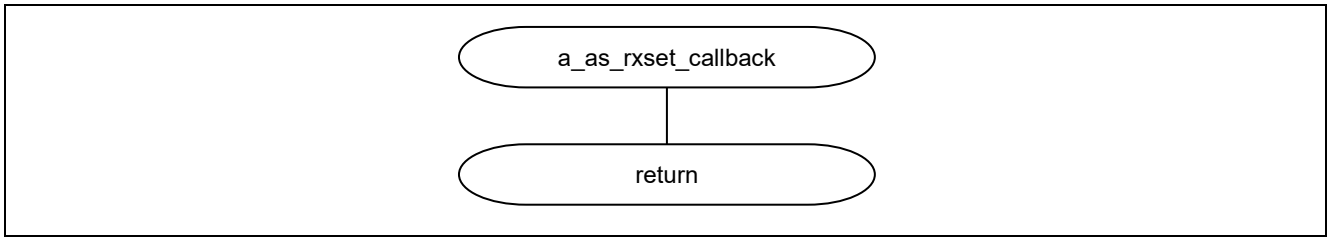



**Figure 4.10 Flowchart of a_as_rxset_callback**

### (9) Flowchart of a_as_rxend_callback

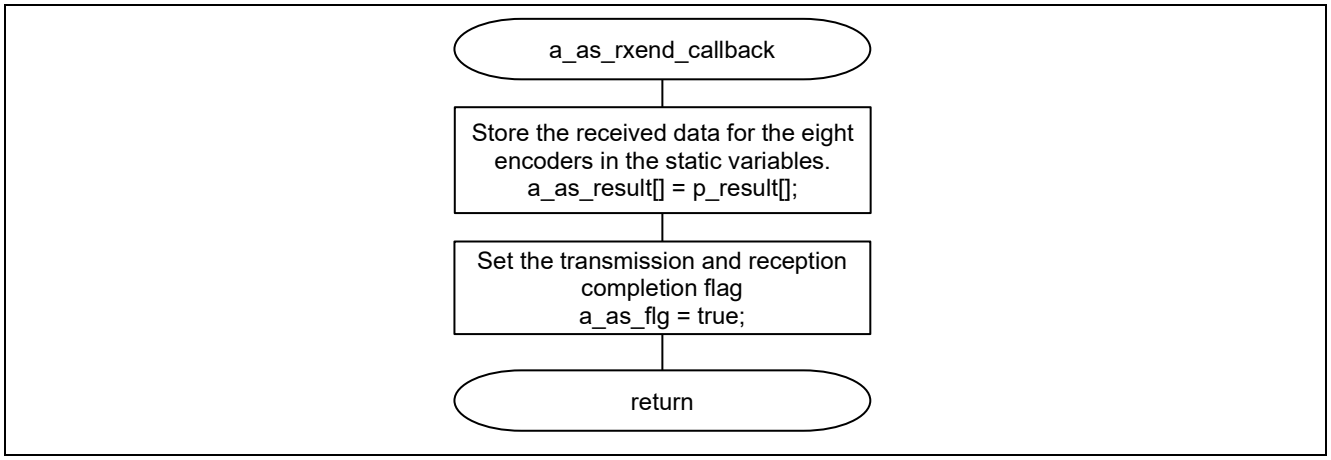



**Figure 4.11 Flowchart of a_as_rxend_callback**

### (10) Flowchart of a_as_elctimer_callback

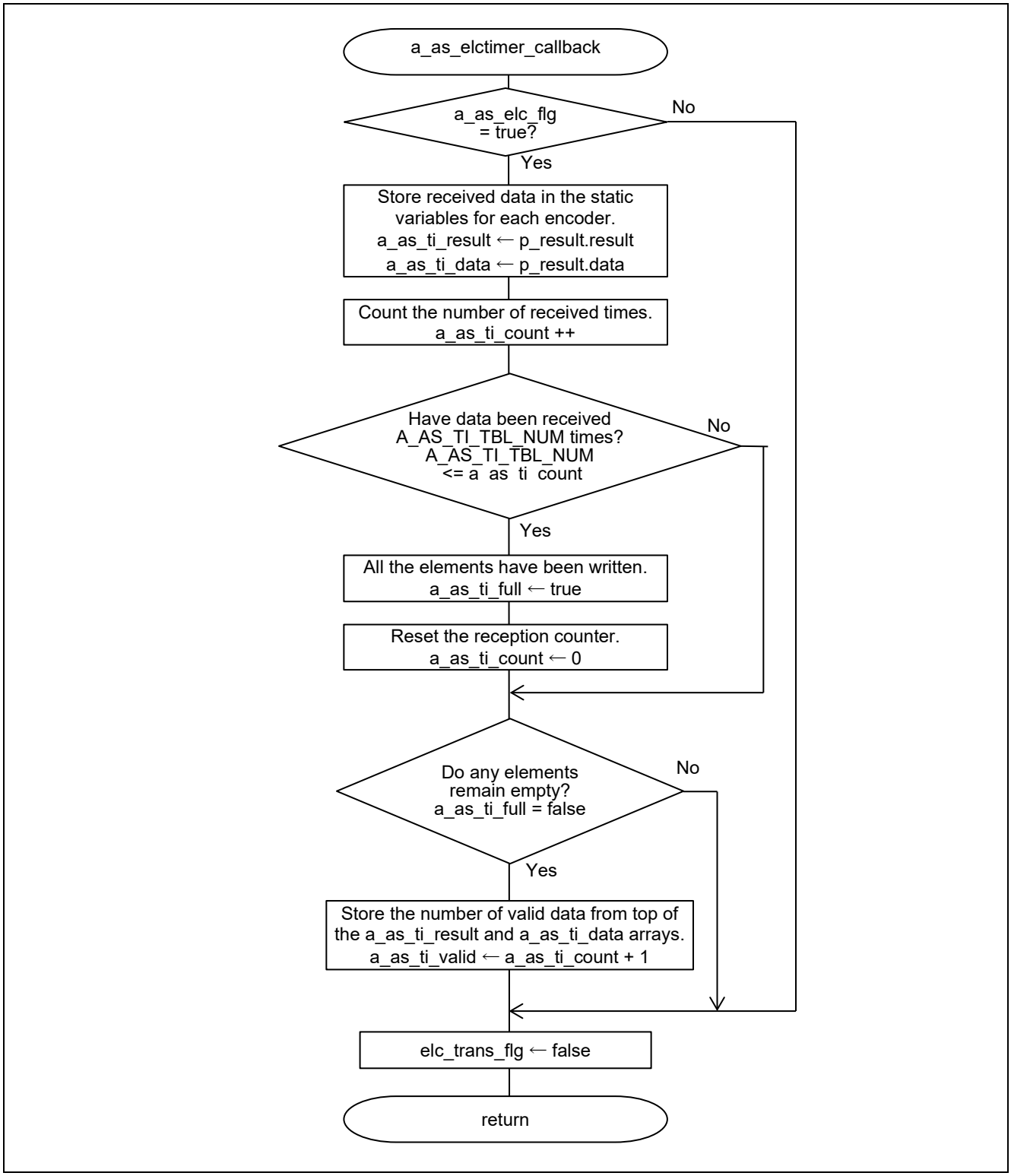



**Figure 4.12 Flowchart of a_as_elctimer_callback**

### 4.11.7 Operation Sequence

#### (1) Startup Sequence

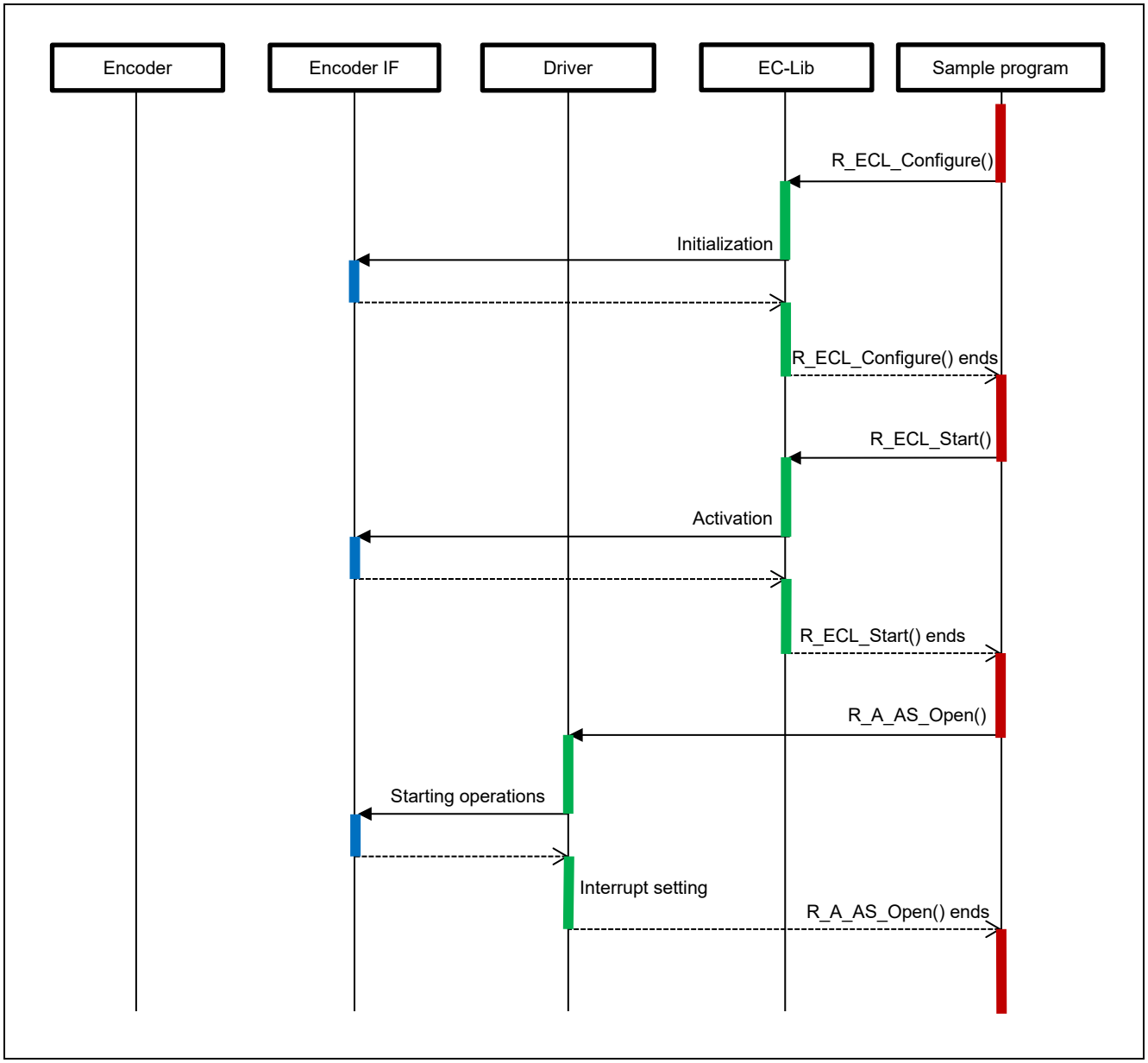



**Figure 4.13 Startup Sequence Diagram**

### (2) Request Transmission and Data Reception Sequence

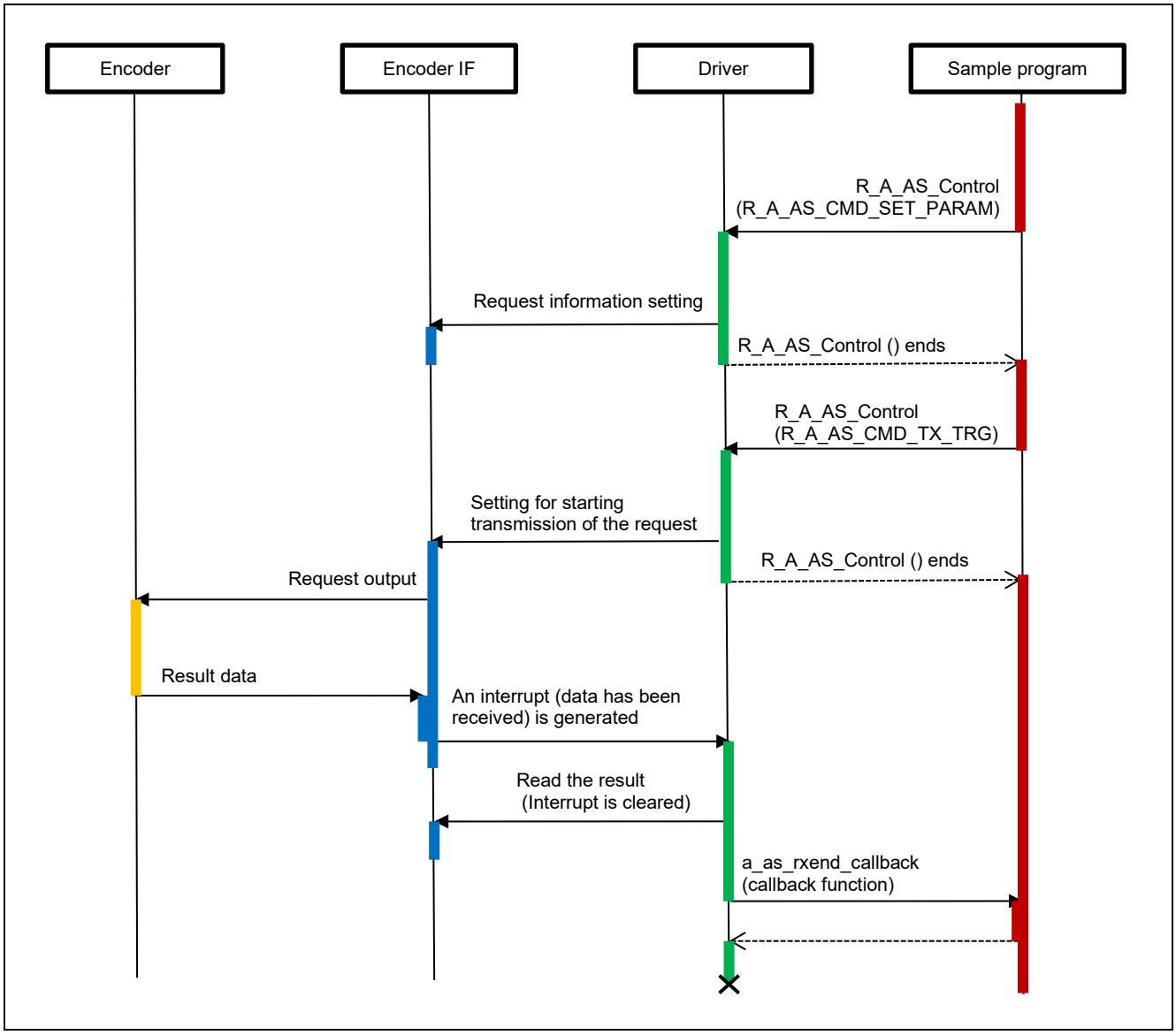



**Figure 4.14 Request Transmission and Data Reception Sequence Diagram**

### (3) ELC Event Trigger Operation Sequence

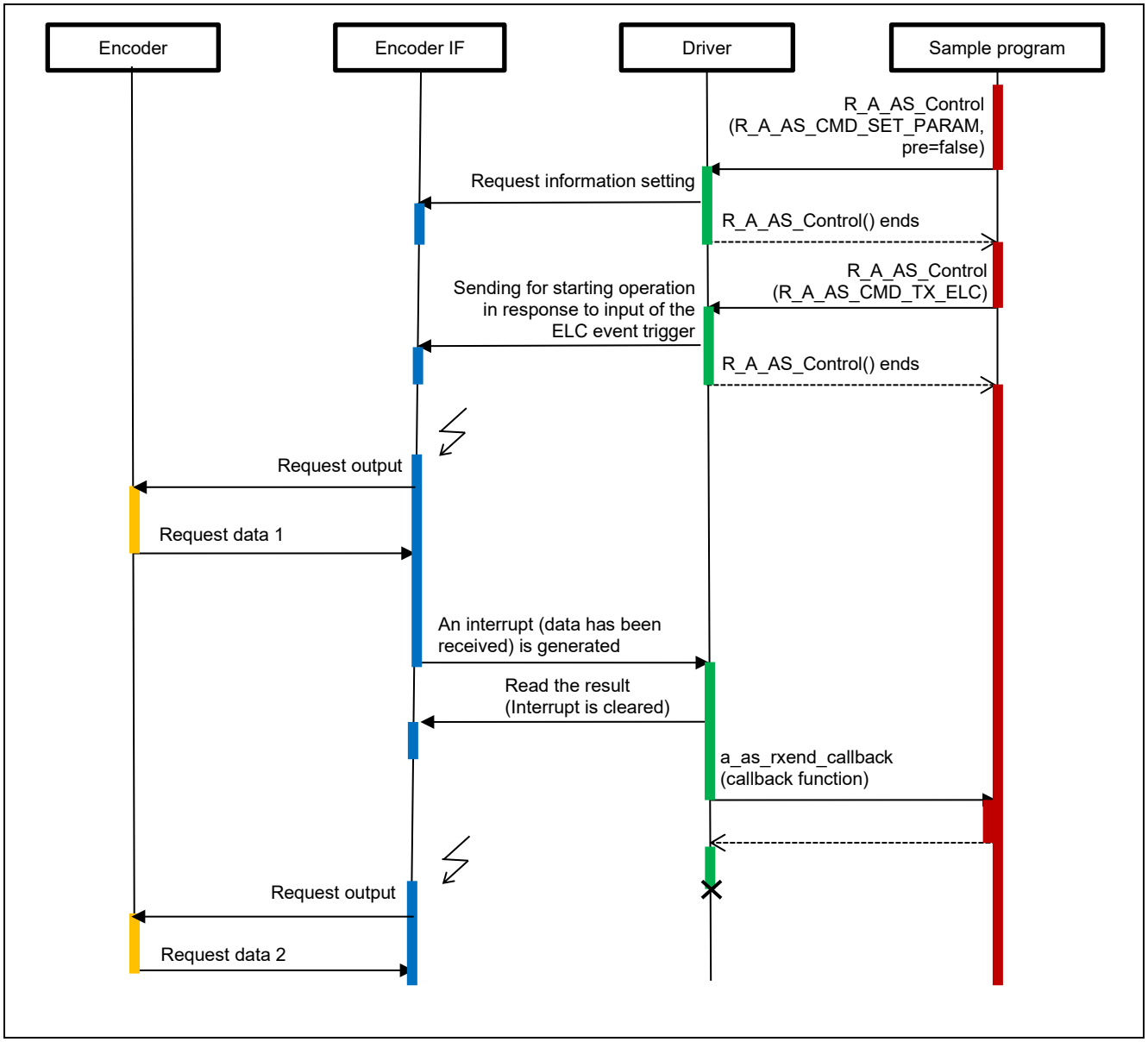



**Figure 4.15 ELC Event Trigger Operation Sequence Diagram (1/2)**

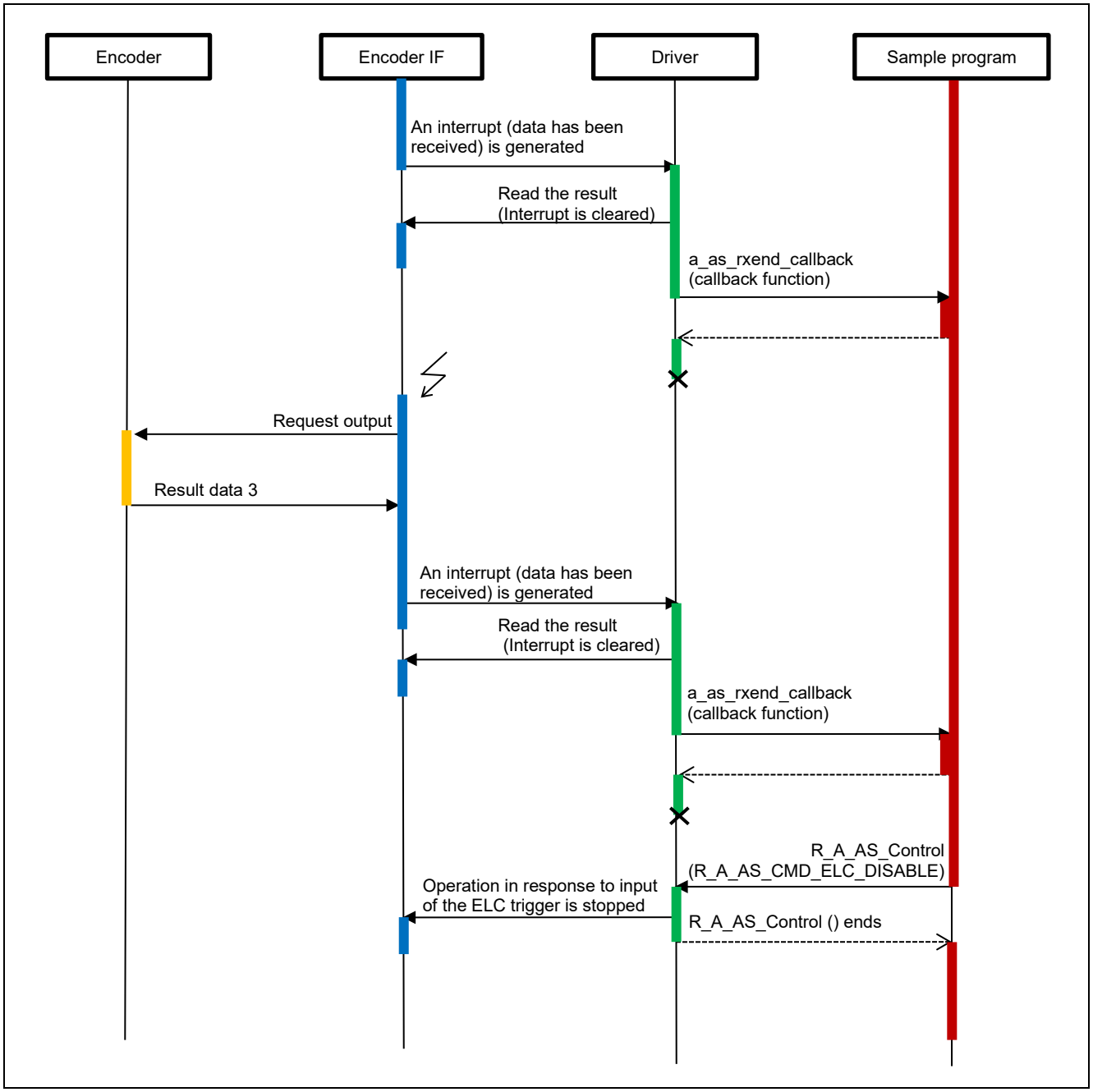



**Figure 4.16 ELC Event Trigger Operation Sequence Diagram (2/2)**

### (4) Stop Sequence

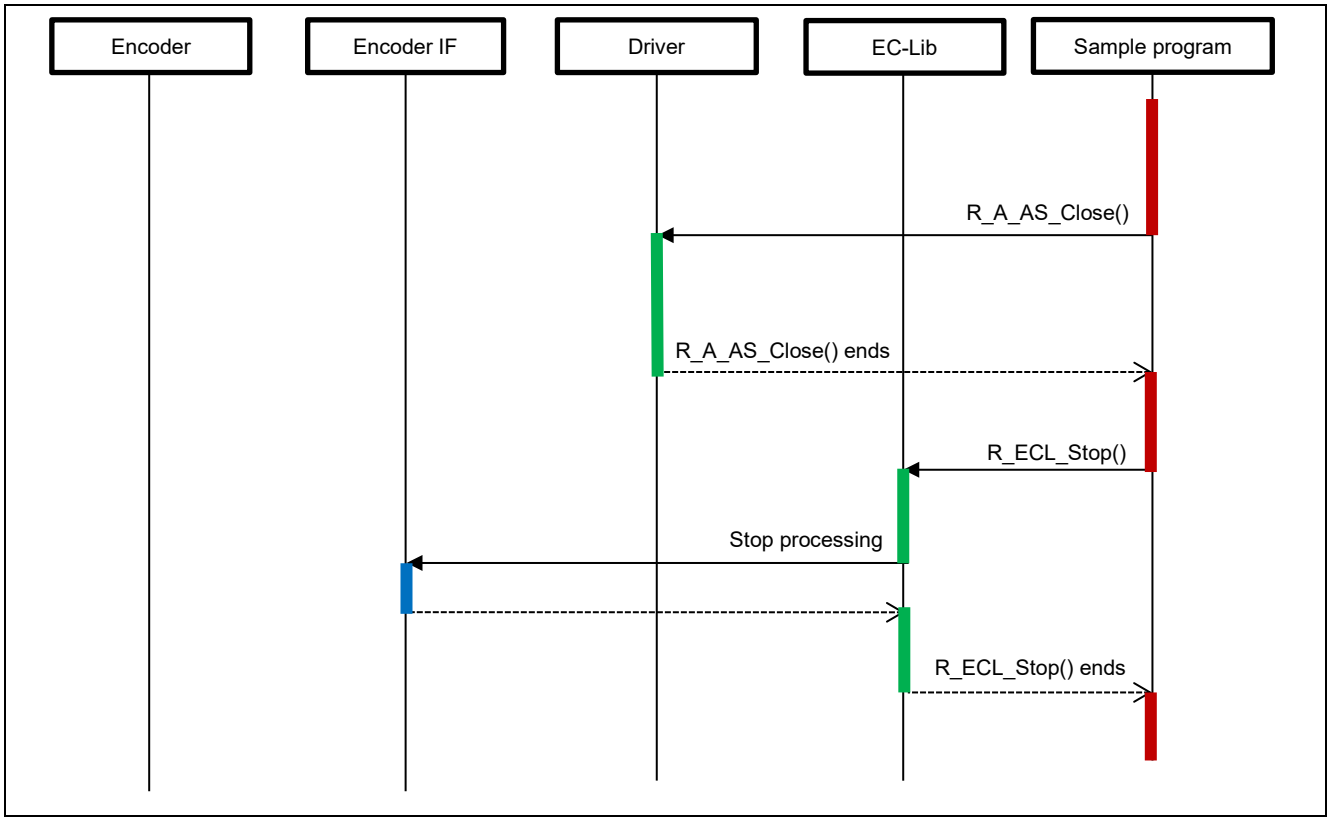



**Figure 4.17 Stop Sequence**

### 4.11.8 Console Commands

This sample program supports the MAR-M50A or SAR-HL700A A-format-compatible encoders. The commands which can be input through the console are as follows.

**Table 4.19 List of Console Commands**

| Command | Content |
|---|---|
| req *ea cmd param1 param2* | Sends the command frame input in the *cmd* to the encoder.<br>*ea*: Encoder section number (decimal notation)<br>*cmd*: Command frame<br>*param1* and *param2*: The values to be input depends on the *cmd* value. See “Table 4.20, req Commands and Corresponding Parameters”. |
| setdf *ea cmd encmod* | Sets data frame types of the encoder. *<br>*ea*: Encoder section number (decimal notation)<br>*cmd*: Target command data frame (CDF1: Data frame type of CDF1 and CDF5, CDF8: Data frame type of CDF8 to CDF12)<br>*encmod*: Extended data frame is used or not (0: Data frame is not extended, Others: Data frame is extended. ABS full 40-bit + velocity is selected for CDF1 and CDF5. ABS LSB 24-bit is selected for CDF8 to CDF12.)<br>Usage example (For the section number 2 encoder, data frame types of CDF1 and CDF5 are treated as ABS full 40-bit + velocity)<br>setdf 2 CDF1 1 |
| elctimer *val* | Sends the command frame CDF4 for eight encoders at 4 Mbps by the ELC event trigger.<br>*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. |

Note: 1. This command is used for A-format version 3.0 encoders or later. Data frame types for the CDF1, CDF5 and CDF8 to CDF12 are predetermined at the shipping of the encoders. Set data frame types to suit each encoder.

**Table 4.20 req Commands and Corresponding Parameters**

| Cmd | param1 | param2 |
|---|---|---|
| CDF0 to CDF12 *1 | None | None |
| CDF13 | An address in EEPROM (in hexadecimal)<br>Specify a value between 0x00 to 0xFF. | None |
| | Usage example (reading the data from address 0 of the section 2 encoder)<br>req 2 CDF13 00 | |
| CDF14 | An address in EEPROM (in hexadecimal)<br>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)<br>req 2 CDF14 00 1234 | |
| CDF15 to CDF17 *1 | None | None |
| CDF18 to CDF20 *1 | Recognition code (in hexadecimal)<br>Specify a value between 0x00 0000 to 0xFF FFFF. | None |
| | Usage example: (writing 0x12 3456 to the recognition code of the section 2 encoder).<br>req 2 CDF18 123456 | |
| CDF21, CDF22, CDF27 to CDF30 | None | None |
| CDF23 to CDF26 *2 | None | None |
| CDF13x *2 | A Bank and an address of EEPROM (in hexadecimal)<br>Specify a value between 0x0000 to 0xFFFF.<br>MSB 8 bits of the 16 bits are used as the bank number. LSB 8 bits are an address. | None |
| | Usage example (reading the data from bank 1, address 0 of the section 2 encoder)<br>req 2 CDF13x 100 | |
| CDF14x *2 | A Bank and an address of EEPROM (in hexadecimal)<br>Specify a value between 0x0000 to 0xFFFF.<br>MSB 8 bits of the 16 bits are used as the bank number. LSB 8 bits are an address. | The data to be written to EEPROM (in hexadecimal) Specify a value between 0x0000 to 0xFFFF. |
| | Usage example (writing 0x1234 to bank 1, address 0 of the section 2 encoder)<br>req 2 CDF14x 100 1234 | |
| CDF16x *2 | None | None |
| CDF18x *2 | Velocity coefficient (in hexadecimal)<br>Specify a value between 0x0 0000 to 0x7 FFFF. | None |
| | Usage example (writing 0x1 2345 to velocity coefficient of the section 2 encoder)<br>req 2 CDF18x 12345 | |

Note: 1. Though input of CDF11, CDF17, and CDF19 are possible, they cannot be used because the sample program is for operation with a bus connection.
2. This is a command data frame available for A-format version 3.0 encoders or later.

### (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 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 |
| 1.00 | Apr.08.22 | - | First Edition issued. |
| 1.10 | Jun.28.22 | 5,6,49 | Corrected the description of operating voltage.<br>Corrected the description of the device used.<br>Corrected the list of used pins.<br>Deleted memory size description. |
| 1.20 | Nov.30.22 | 13,18,<br>41,44,<br>45,46 | Corrected the description of interrupt register.<br>Corrected the description of flowchart.<br>Corrected the description of sequence. |
| 2.00 | Jun 7.24 | 5<br>22 | Update description of the board name.<br>Remove description about location of integer type definition. |
| 2.10 | Mar 28.25 | 1<br>1, 27, 48<br>11<br>17, 19, 20<br>23, 24<br><br>25<br>25<br>29, 31<br>34<br>36, 38<br>39<br>42<br>48<br>49 | Update configuration data version number.<br>Add SAR-HL700A for the supported encoder name.<br>Add control command R_A_AS_CMD_SETDF.<br>Update Table 4.5, Table 4.9 and Table 4.12. Add Table 4.13.<br>Add mambank in the request information structure. Add descriptions for new commands.<br>Add rxd2 in the received data structure.<br>Add fd6err, fd7err and bankerr in A_AS status structure.<br>Add description for the a_as_setdf function.<br>Add enc_df_type in Table 4.18.<br>Add a_as_setdf function in flowcharts.<br>Correct GPT channel in the flowchart.<br>Correct a_as_elctimer_callback function flowchart.<br>Add setdf in console commands.<br>Add new commands in req commands and parameters table. |
| 3.00 | Oct 10.25 | 30, 34<br><br>51 | Add description of a_as_elctimer_callback in user-defined functions.<br>Add example of command execution. |
| 4.00 | Feb 27.26 | 6, 17<br>7 to 34 | Change interrupt handler name to a_as0_int_isr, a_as1_int_isr.<br>Change prefix of pointer variables to “p_”. |

# 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/.